Description de l'objet Feuille de calcul dans Excel

par SilkyRoad (silkyroad.developpez.com)

Date de publication : 14.11.2006

Dernière mise à jour : 15.04.2007

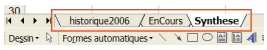
Ce tutoriel décrit l'objet Feuille de calcul dans un classeur Excel.

Le document montre comment manipuler les feuilles par macro et propose une description des différentes méthodes et propriétés.

Tous les exemples de cette page ont été testés avec Excel2002.

I - Introduction

Un classeur Excel est composé de feuilles de calcul et de feuilles graphiques, présentées sous formes d'onglets.



Les outils disponibles dans chaque feuille permettent la transformation des données brutes en informations pour calculer, analyser, mettre en forme et partager les résultats.

Chaque feuille de calcul est constituée de cellules: sur 65536 lignes par 256 colonnes (jusqu'à la version 2003 d'Excel), 1048576 lignes par 16384 colonnes dans Excel 2007. Les cellules permettent de stocker, puis de manipuler les données et sont principalement utilisées pour recevoir **les fonctions** (Formules) du tableur. Une feuille peut aussi contenir d'autres objets: Des images, contrôles, graphiques incorporés, formes automatiques, fichiers incorporés...

Le nombre d'onglets est uniquement limité par la quantité de mémoire disponible dans un classeur.

Il existe quatre possibilités pour spécifier une feuille par macro:

* **Par le nom d'onglet**:

```
'Cet exemple active la feuille nommée "historique2006"

Worksheets("historique2006").Activate
```

* **Par l'index** (L'index correspond à la position de l'onglet dans le classeur. La feuille de gauche porte l'index 1):

```
'Active la 1ere feuille du classeur

Worksheets(1).Activate
```

* **Par la feuille active** (Visible au premier plan):

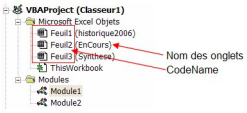
```
'Renvoie le nom de la feuille placée au premier plan

ActiveSheet.Name
```

* **Par le CodeName**:

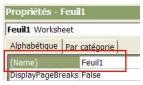
'Active la feuille dont le nom de code objet est "Feuil1"

Feuil1.Activate

Le CodeName représente le nom de l'objet Feuille dans l'éditeur de macros. Il ne faut pas le confondre avec le nom de l'onglet. Les 2 sont identiques par défaut, mais le CodeName ne change pas lorsque vous modifiez le nom d'un onglet.



Le CodeName peut être modifié dans la fenêtre propriété du module objet.



La propriété Sheets renvoie tous les types de feuilles contenues dans le classeur: Les feuilles graphiques et feuilles de calcul.

La propriété Worksheets renvoie uniquement la collection de feuilles de calcul.

Lorsque vous souhaitez placer la feuille de calcul dans une variable, utilisez l'objet Worksheet:

Vba

```
'Déclare la variable objet Worksheet
Dim Ws As Worksheet

'Attribue la référence objet à la variable
'La feuille est un Objet: la variable doit donc être précédée de l'instruction Set lors de
 l'attibution.
Set Ws = Worksheets("Feuil2")
'Manipulation de l'objet:
'Exemple pour renvoyer l'index (position) de la feuille dans le classeur
MsgBox "Index de la feuille nommée '" & Ws.Name & "': " & Ws.Index
```

Si vous souhaitez créer une boucle sur l'ensemble des feuilles de calcul, utilisez:

Vba

```
'Déclare la variable objet Worksheet
```

Vba
```
Dim Ws As Worksheet

'Boucle sur toutes les feuille de calcul du classeur. Les onglets graphiques ne sont pas pris
'en compte.
'ThisWorkbook correspond à l'objet classeur contenant la macro
For Each Ws In ThisWorkbook.Worksheets
    'Renvoie le nom de chaque feuille
    MsgBox Ws.Name
Next Ws
```

Si le classeur contient un onglet Graphique et que vous souhaitez l'intégrer dans la boucle, utilisez:

Vba
```
'Déclare la variable objet
Dim Ws As Object

'Boucle sur tous les onglets du classeur
'ThisWorkbook correspond à l'objet classeur contenant la macro
For Each Ws In ThisWorkbook.Sheets
    'Renvoie le nom de chaque onglet
    MsgBox Ws.Name
Next Ws


'Nota:
'La clause TypeOf permet de contrôler vers quel objet de l'application pointe la variable.
'Pour retrouver le nom de toutes les feuilles graphiques dans le classeur actif, vous pouvez
 utiliser:
   'Dim Sh As Object
   'For Each Sh In ActiveWorkbook.Sheets
   'If TypeOf Sh Is Chart Then MsgBox Sh.Name
   'Next
```

II - Les méthodes et propriétés dans La feuille de calcul

II-A - Les méthodes

II-A-1 - Activate

Permet d'activer un onglet.

Cet exemple active la feuille nommée "Feuil1".

Vba
```
Worksheets("Feuil1").Activate
```

II-A-2 - Add

Permet d'ajouter une feuille dans le classeur. Add(Before, After, Count, Type).

L'argument Before spécifie la feuille avant laquelle la nouvelle feuille est ajoutée.

L'argument After spécifie la feuille après laquelle la nouvelle feuille est ajoutée.

L'argument Count spécifie le nombre de feuilles à ajouter. La valeur par défaut est 1 si l'argument n'est pas précisé.

L'argument Type spécifie le type de feuille ajouté (xlWorksheet, xlChart).

Vba
```
'Ajoute une feuille et la positionne à la fin du classeur.
ThisWorkbook.Sheets.Add After:=Sheets(ThisWorkbook.Sheets.Count)

'Exemple pour ajouter 3 feuilles:
ThisWorkbook.Sheets.Add After:=Sheets(ThisWorkbook.Sheets.Count), Count:=3
```

II-A-3 - Calculate

Permet de déclencher le calcul des fonctions dans la feuille indiquée.

Vba
```
'Lance le calcul dans la 1ere feuille du classeur
Worksheets(1).Calculate
```

II-A-4 - ChartObjects

Renvoie la collection de graphiques placés dans la feuille de calcul. Ces graphiques sont aussi dits "incorporés".

La procédure suivante boucle sur les graphiques de la feuille active afin de les redimensionner et les repositionner verticalement.

Vba

```
Dim Ch As ChartObject

Application.ScreenUpdating = False

'Boucle sur les graphiques de la feuille active
For Each Ch In ActiveSheet.ChartObjects
    'Récupère le nom de chaque graphique
    'MsgBox Ch.Name

    With Ch
        .Left = 50 'position horizontale
        .Top = 20 + (230 * (Ch.Index - 1)) 'position verticale
        .Height = 200 'hauteur graphique
        .Width = 350 'largeur graphique
    End With
Next Ch

Application.ScreenUpdating = True
```

Un graphique spécifique peut être manipulé par son index ou son nom:

Vba

```
'Permet de renommer le 1er graphique de la feuille active
ActiveSheet.ChartObjects(1).Name = "Le Graphique"

'Vérifie le nom attribué
MsgBox ActiveSheet.ChartObjects(1).Name

'Permet de renommer un graphique nommé "Le Graphique"
ActiveSheet.ChartObjects("Le Graphique").Name = "Nouveau Nom"

'Vérifie le nom attribué
MsgBox ActiveSheet.ChartObjects(1).Name
```

II-A-5 - CheckSpelling

Permet de lancer la correction orthographique dans la feuille spécifiée.

Cet exemple vérifie l'orthographe des mots saisis dans le Feuil1. La boîte de dialogue s'affiche si des corrections sont nécéssaires.

Vba

```
Dim IgnoreMajuscule As Boolean
Dim Suggestion As Boolean
```

Vba
```
Dim MonDico As String

'indique le nom de fichier du dictionnaire personnalisé à examiner si le mot
'n'est pas trouvé dans le dictionnaire principal.(Facultatif)
MonDico = ""

'Affiche une liste de suggestions lorsqu'un mot est mal orthographié.(Facultatif)
Suggestion = True
'Permet d'ignorer les mots en majuscules.(Facultatif)
IgnoreMajuscule = True

Worksheets("Feuil1").CheckSpelling MonDico, IgnoreMajuscule, Suggestion
```

II-A-6 - ClearArrows

Permet d'effacer les flèches d'audit dans la feuille de calcul.

Ces flèches ont été ajoutées à l'aide de la fonction d'audit (Menu Outils / Audit de formules / ...).

Vba
```
Worksheets("Feuil1").ClearArrows
```

II-A-7 - Copy

Effectue une copie de la feuille.

Les arguments Before et After permettent de spécifier l'onglet avant ou après lequel la feuille copiée sera placée (Il n'est pas possible d'utiliser ensemble les deux arguments).

Vba
```
'Crée une copie de la Feuil1 et la positionne à la fin du classeur.
Worksheets("Feuil1").Copy After:=Sheets(Sheets.Count)
```

Si vous ne spécifiez pas les arguments Before ou After, la procédure crée un nouveau classeur contenant la feuille copiée.

Vba
```
'Crée une copie de la Feuil1 dans un nouveau classeur
Worksheets("Feuil1").Copy
'Sauvegarde la copie
ActiveWorkbook.SaveAs "C:\LaFeuilleDupliquee.xls"
'Ferme le classeur sauvegardé
ActiveWorkbook.Close True
```

II-A-8 - CircleInvalid

Encercle les entrées incorrectes dans la feuille de calcul.

Vba
```
Worksheets("Feuil2").CircleInvalid
```

II-A-9 - ClearCircles

Supprime les cercles qui indiquent des entrées incorrectes de la feuille de calcul.

Vba
```
Worksheets("Feuil2").ClearCircles
```

II-A-10 - Delete

Permet de supprimer la feuille de calcul.

Nota:

Il n'est pas possible de supprimer une feuille si elle est unique dans le classeur.

Vba
```
'DisplayAlerts = False permet de ne pas afficher le message d'alerte qui survient
'lorsque l'on supprime un onglet.
Application.DisplayAlerts = False
'Suppression de la feuille nommée "Feuil2"
Worksheets("Feuil2").Delete
'Ne pas oublier de réinitialiser DisplayAlerts à True
Application.DisplayAlerts = True
```

II-A-11 - Evaluate

Convertit le nom (conforme aux conventions Microsoft Office Excel) spécifié en objet ou en valeur.

Vba
```
Dim Cellule As Range

'--- Utilise Evaluate pour récupérer le contenu de la cellule A1
MsgBox Worksheets("Feuil1").Evaluate("A1")
'Equivalent:
MsgBox Worksheets("Feuil1").[A1]


'--- Utilise Evaluate pour définir un objet Range
Set Cellule = Worksheets("Feuil1").[A1]
'Applique la couleur verte dans la cellule
Cellule.Interior.ColorIndex = 4
```

II-A-12 - FillAcrossSheets

Copie la plage de cellules d'un onglet au même emplacement dans toutes les autres feuilles de calcul spécifiées.

Vba

```
Dim Tableau As Variant
Dim ArgType As Long

'--- Précise comment doit être copiée la plage de cellules
'xlFillWithAll est la valeur par défaut si l'agument n'est pas spécifié
ArgType = xlFillWithAll 'copie tout
'ArgType = xlFillWithContents 'contenu des cellules
'ArgType = xlFillWithFormats 'Le format des cellules
'---

Tableau = Array("Base", "Feuil1", "Feuil3", "Feuil6")
'Copie la plage A1:A10 de la feuille "Base" vers les feuilles spécifiées
'dans le tableau Array("Feuil1", "Feuil3", "Feuil6"):
    'Nota:
    'La feuille source doit être aussi indiquée dans le tableau
Sheets(Tableau).FillAcrossSheets Worksheets("Base").Range("A1:A10"), ArgType
```

Pour recopier la plage A1:E5 de la Feuil1 dans toutes autres les feuilles du classeur, utilisez:

Vba

```
Worksheets.FillAcrossSheets Worksheets("Feuil1").Range("A1:E5"), xlFillWithAll
```

II-A-13 - Move

Permet de déplacer une feuille dans le classeur.

Les arguments Before et After permettent de spécifier l'onglet avant ou après lequel la feuille sera déplacée. (Il n'est pas possible d'utiliser ensemble les deux arguments).

Cet exemple trie les feuilles par ordre alphabétique.

Vba

```
Sub TrierFeuilles()
    Dim WS As Object
    Dim I As Byte

    Application.ScreenUpdating = False
        For Each WS In ActiveWorkbook.Sheets
            For I = 2 To ActiveWorkbook.Sheets.Count
                If Sheets(I - 1).Name > Sheets(I).Name Then _
                    Sheets(I - 1).Move After:=Sheets(I)

            Next I
        Next WS
```

```Vba
        Application.ScreenUpdating = True
    End Sub
```

Si vous n'indiquez pas les arguments Before ou After, la propriété déplace la feuille du classeur d'origine vers un nouveau classeur.

II-A-14 - OLEObjects

Représente les contrôles ActiveX, les objets OLE incorporés ou liés dans une feuille de calcul.

Cette procédure crée un objet CommandButton dans la feuille et y associe une procédure évènementielle.

```Vba
    Dim Ws As Worksheet
    Dim Obj As OLEObject
    Dim laMacro As String
    Dim x As Integer

    Set Ws = Sheets.Add 'Ajoute une nouvelle feuille

    'ajoute un CommandButton dans la nouvelle feuille
    Set Obj = Ws.OLEObjects.Add("Forms.CommandButton.1")
    With Obj
        .Name = "monBouton" 'renomme le bouton
        .Left = 50 'position horizontale par rapport au bord gauche de la feuille
        .Top = 50 'position verticale par rapport au bord haut de la feuille
        .Width = 150 'largeur
        .Height = 30 'hauteur
        .Object.Caption = "Supprimer données feuille"
    End With

    'Ajoute la procédure dans la feuille
    With ThisWorkbook.VBProject.VBComponents(ActiveSheet.Name).CodeModule
        .CreateEventProc "Click", "monBouton"
        x = .ProcStartLine("monBouton_Click", vbext_pk_Proc)
        .InsertLines x + 2, "Cells.Clear"
    End With
```

Cette procédure montre comment boucler sur la collection de contrôles dans la feuille.

```Vba
    Dim Obj As OLEObject

    For Each Obj In Feuil1.OLEObjects
        'Nom de l'objet.
        Debug.Print Obj.Name

        'Type de contrôle.
        '(Renvoie une erreur si la feuille contient des objets type fichiers insérés)
```

```Vba
        Debug.Print TypeName(Obj.Object)
    Next Obj
```

Il est possible de boucler sur un type d'objets spécifique.

Cet exemple utilise la fonction TypeName:

```Vba
    Dim Obj As OLEObject

    'Boucle sur les objets contenus dans la feuille.
    For Each Obj In Feuil1.OLEObjects
        'Vérifie s'il s'agit d'un contrôle ActiveX
        If Obj.ShapeRange.Type = msoOLEControlObject Then
            'Vérifie s'il s'agit d'un TextBox et en récupère le contenu si c'est le cas.
            If TypeName(Obj.Object) = "TextBox" Then MsgBox Obj.Object.Value
        End If
    Next Obj
```

Une autre méthode pour identifier les TextBox. Utilisation de l'instruction TypeOf:

```Vba
    Dim Obj As OLEObject

    'Boucle sur les objets contenus dans la feuille.
    For Each Obj In Feuil1.OLEObjects
        'Vérifie s'il s'agit d'un contrôle ActiveX
        If Obj.ShapeRange.Type = msoOLEControlObject Then
            'Vérifie s'il s'agit d'un TextBox et en récupère le contenu si c'est le cas.
            If TypeOf Obj.Object Is MSForms.TextBox Then MsgBox Obj.Object.Value
        End If
    Next Obj
```

Et pour boucler sur des objets à partir de leur nom, utilisez l'exemple ci dessous qui boucle sur des TextBox nommés "TextBox1" à "TextBox4":

```Vba
    Dim i As Byte
    For i = 1 To 4
        'Boucle sur 4 Textbox nommés "Textbox1" à "Textbox4" pour y afficher
        'les données contenues dans les cellules A1 à A4, dans la Feuil2.
        Feuil1.OLEObjects("TextBox" & i).Object.Text = Feuil2.Cells(i, 1)
    Next
```

Cet exemple montre comment insérer un document Word, sous forme d'icône, dans la feuille.

Vba

```
Dim Ws As Worksheet
Dim Fichier As String

'Indique la feuille qui va recevoir l'objet
Set Ws = ThisWorkbook.Worksheets("Feuil1")

'Définit le fichier à insérer
Fichier = "C:\Documents and Settings\michel\le document.doc"

'Insère le document Word, sous forme d'icone, dans la feuille
Ws.OLEObjects.Add Filename:=Fichier, _
    Link:=False, DisplayAsIcon:=True, IconIndex:=0, _
    IconLabel:=Fichier
```

II-A-15 - Paste

Colle le contenu du Presse-papiers dans la feuille spécifiée.

Vba

```
'Copie la plage de cellules A1:A10 dans la Feuil1
Worksheets("Feuil1").Range("A1:A10").Copy

'Effectue le collage dans la plage B1:B10 de la Feuil2
Worksheets("Feuil2").Paste Destination:=Worksheets("Feuil2").Range("B1:B10")
'ou
'Worksheets("Feuil2").Paste Destination:=Worksheets("Feuil2").Range("B1")

'Permet de désactiver le presse papier
Application.CutCopyMode = False
```

Le collage est effectué à l'emplacement de la sélection si vous ne spécifiez pas l'argument Destination.

L'argument Link permet de réaliser un collage avec liaison.

Vba

```
'
'Nécéssite d'activer la référence "Microsoft Forms 2.0 Object Library"
'
Dim Cible As dataObject

'-- Copie la plage de cellules A1:A10 dans la Feuil1 ---
Worksheets("Feuil1").Range("A1:A10").Copy

'Effectue un collage avec liaison dans la Feuil3
With Worksheets("Feuil3")
    .Activate
    .Range("E5").Select
    .Paste link:=True
End With
'-------------------------------------------------------
```

Vba
```
'--- permet de vider le presse papier ---
Set Cible = New dataObject
Cible.setText ""
Cible.putInClipboard
Set Cible = Nothing
'----------------------------------------
```

Les arguments Destination et Link ne peuvent pas être utilisés ensemble.

II-A-16 - PasteSpecial

Permet d'effectuer un collage en précisant le format à appliquer.

Cet exemple colle un graphique au format image dans une autre feuille du classeur.

Vba
```
'Effectue une copie du 1er graphique contenu dans la feuille nommée "Feuil1"
Worksheets("Feuil1").ChartObjects(1).Copy
'Active la Feuil2
Worksheets("Feuil2").Activate
'Colle le graphique au format Image MetaFichier dans la Feuil2
Worksheets("Feuil2").PasteSpecial Format:="Picture (Enhanced Metafile)", _
    Link:=False, DisplayAsIcon:=False
```

II-A-17 - PivotTables

Représente les tableaux croisés dynamiques contenus dans la feuille.

Vba
```
Dim Pvt As PivotTable

'Définit le TCD contenu dans la feuille
Set Pvt = Worksheets("Feuil4").PivotTables("Tableau croisé dynamique1")
'Met à jour le TCD
Pvt.RefreshTable
```

II-A-18 - PivotTableWizard

Permet de créer un objet PivotTable dans la feuille.

L'exemple suivant ajoute un TCD dans la feuille nommée "Feuil4", à partir des données de la Feuil1. Le tableau croisé dynamique est positionné dans la cellule B10.

Vba
```
Worksheets("Feuil4").PivotTableWizard _
```

Vba
```
        SourceType:=xlDatabase, _
        SourceData:=Worksheets("Feuil1").Range("A1:E100").Address(, , xlR1C1, True), _
        TableDestination:=Worksheets("Feuil4").Range("B10"), _
        TableName:="TCD_1"

    With Worksheets("Feuil4").PivotTables("TCD_1")
        .AddFields RowFields:="Ville"
        .PivotFields("ChiffreAffaire").Orientation = xlDataField
    End With
```

II-A-19 - PrintOut

Imprime la feuille. PrintOut(From, To, Copies, Preview, ActivePrinter, PrintToFile, Collate, PrToFileName)

L'argument From indique le numéro de la première page à imprimer. L'impression est effectuée à partir de la première page si aucune valeur n'est précisée. Chaque page est délimitée par les sauts de pages placés dans la feuille.

L'argument To indique le numéro de la dernière page à imprimer. L'impression est effectuée jusqu'à la dernière page si aucune valeur n'est précisée.

L'argument Copies Précise le nombre de copies. La valeur par défaut est 1 si vous ne spécifiez l'argument.

L'argument Preview affiche un aperçu avant impression si vous indiquez la valeur True.

L'argument ActivePrinter définit le nom de l'imprimante active.

L'argument Collate permet d'assembler plusieurs copies si vous indiquez la valeur True.

L'argument PrintToFile permet d'imprimer dans un fichier si vous indiquez la valeur True.

L'argument PrToFileName précise le chemin lorsque vous souhaitez imprimer dans un fichier. PrintToFile doit être ègal à True.

Vba
```
    'Imprime la totalité de la feuille
    Worksheets("Feuil1").PrintOut

    'Imprime les pages 2 à 3
    Worksheets("Feuil1").PrintOut From:=2, To:=3

    'Imprime la 1ere page en 2 exemplaires
    Worksheets("Feuil1").PrintOut From:=1, To:=1, Copies:=2
```

II-A-20 - PrintPreview

Permet d'afficher l'aperçu avant impression pour la feuille spécifiée.

Vba

Vba
```
Worksheets("Feuil1").PrintPreview
```

II-A-21 - Protect

Permet de protéger la feuille: L'équivalent du Menu Outils / Protection / Protéger la feuille.

Vba
```
'Protège la feuille.
Worksheets("Feuil1").Protect

'Protège la feuille et autorise l'utilisation des tableaux croisés dynamiques.
Worksheets("Feuil1").Protect AllowUsingPivotTables:=True

'Protège la feuille en spécifiant un mot de passe
Worksheets("Feuil1").Protect Password:="MotDePasse"

'Protège la feuille et autorise toutes les modifications effectuées par macro.
'/!\ L'option UserInterfaceOnly doit être réinitialisée à chaque ouverture du classeur.
Worksheets("Feuil1").Protect UserInterfaceOnly:=True
```

Ce chapitre ne décrit pas de façon exhaustive les arguments qui permettent de rendre accessibles certains objets lorsque la feuille est protégée: (DrawingObjects, Contents, Scenarios, UserInterfaceOnly, AllowFormattingCells, AllowFormattingColumns, AllowFormattingRows, AllowInsertingColumns, AllowInsertingRows, AllowInsertingHyperlinks, AllowDeletingColumns, AllowDeletingRows, AllowSorting, AllowFiltering, AllowUsingPivotTables)

Consultez l'aide Excel pour plus de détails.

Si vous oubliez le mot de passe, vous ne pourrez plus enlever la protection de la feuille. Pensez à conserver la liste de vos mots de passe en lieu sûr.

Quelques conseils concernant le choix des mots de passe:

Il n'existe pas de solution fiable à 100% pour empêcher une personne mal intentionnée qui souhaiterait faire sauter une protection. Néanmoins, vous pouvez respecter certaines règles pour lui rendre la vie plus difficile (notamment s'il utilise la méthode dite "Force brute"):

Il faut 60 secondes pour boucler sur 26 caractères et obtenir toutes les combinaisons possibles de 4 caractères.

Il faut moins de 3 secondes pour boucler sur 10 caractères et obtenir toutes les combinaisons possibles de 4 caractères, donc:

** Evitez les mots de passe trop courts.*

** Evitez d'utilisez vos prénoms, noms , dates ou autres mots en rapport avec le contenu de la feuille.*

** Utilisez un mot de passe qui comporte au moins 10 caractères, avec un assortiment de chiffres, des caractères spéciaux, des lettres minuscules et majuscules, parmi les possibilités suivantes:*

0123456789azertyuiopqsdfghjklmwxcvbnAZERTYUIOPQSDFGHJKLMWXCVBN!@#$%^&()+=*

II-A-22 - ResetAllPageBreaks

Réinitialise les sauts de page dans la feuille de calcul.

Vba

```
Worksheets("Feuil1").ResetAllPageBreaks
```

II-A-23 - Scenarios

Représente les scénarios de la feuille de calcul.

Le scénario est un outil qui permet de modifier les valeurs des cellules afin de voir de quelle manière elles affectent le résultat des formules.

Vba

```
Dim Scn As Scenario

'Définit le scénario à utiliser (nommé "TestScenario" dans l'exemple)
Set Scn = Worksheets("Feuil1").Scenarios("TestScenario")
'Le scénario peut aussi être spécifié par son index dans la feuille
'Set Scn = Worksheets("Feuil1").Scenarios(1)

'Lance le scénario
Scn.Show
```

II-A-24 - Select

Permet de sélectionner une feuille.

Vba

```
'Sélectionne le dernièr onglet du classeur.
Sheets(Sheets.Count).Select
```

II-A-25 - SetBackgroundPicture

Définit l'image d'arrière-plan dans la feuille.

Vba

Vba
```
Worksheets("Feuil1").SetBackgroundPicture _
    "C:\Documents and Settings\mimi\dossier Images\fourmiz.JPG"

'Pour enlever l'image de fond:
'Worksheets("Feuil1").SetBackgroundPicture ""
```

II-A-26 - ShowAllData

Permet de réafficher toutes les lignes suite à l'utilisation d'un filtre automatique: L'équivalent de l'élément "Tous" dans les colonnes filtrées.

Vba
```
Worksheets("Feuil1").ShowAllData
```

II-A-27 - ShowDataForm

Permet d'afficher le formulaire (grille de données) de la feuille: L'équivalent de Menu Outils / Données / Formulaire

La boîte de dialogue qui s'affiche permet de visualiser les enregistrements contenus dans la feuille. Les grilles de données permettent d'ajouter, modifier, atteindre et supprimer des enregistrements.

Privilégiez une structure classique pour votre base de données: La première ligne sert à indiquer le nom des champs à partir de la première colonne. N'utilisez pas de colonnes vides. Evitez les lignes vides en tête de feuille, sinon la procédure ne trouve pas les données et renvoie un message d'erreur 1004 "La methode ShowDataForm de la classe Worksheet a échoué".

Plus d'informations sur le site Microsoft.

Vba
```
Worksheets("Feuil1").ShowDataForm
'La suite de la procédure s'execute lorsque la boite de dialogue est refermée.
MsgBox "Fermeture du formulaire"
```

Si votre projet nécessite des formulaires complexes, Utilisez des **UserForm**.

II-A-28 - Unprotect

Supprime la protection de la feuille.

Vba

```
Worksheets("Feuil3").Unprotect
'Si la feuille est protégée par mot de passe:
'Worksheets("Feuil3").Unprotect "MotDePasse"
```

Faites attention à bien respecter les majuscules et minuscules lorsque vous spécifiez un mot de passe.

Si vous ne précisez pas l'argument "Password" dans un onglet protégé par un mot de passe, La boîte de dialogue "Ôter la protection de la feuille" s'affiche à l'écran.

II-B - Les propriétés

II-B-1 - Application

Représente l'application qui a créé l'objet spécifié. Cette propriété peut, par exemple, être utilisée pour retrouver le nom de l'application d'un objet précis lorsque que vous pilotez Excel par automation:

Vba

```
Dim appExcel As Excel.application
Dim wbExcel As Excel.Workbook
Dim wsExcel As Excel.Worksheet

'Ouverture de l'application
Set appExcel = CreateObject("Excel.Application")
appExcel.Visible = True

'Ajout d'un classeur
Set wbExcel = appExcel.Workbooks.Add

'Définit la feuille par défaut
Set wsExcel = wbExcel.ActiveSheet

Debug.Print wsExcel.application
```

II-B-2 - AutoFilter

Représente les filtres automatiques dans la feuille de calcul.

La macro suivante permet d'afficher les critères de filtres actifs, dans la Feuil1.

Vba

```
Dim Fltr As Filter
Dim Resultat As String

If Worksheets("Feuil1").AutoFilterMode Then
    For Each Fltr In Worksheets("Feuil1").AutoFilter.Filters
        If Fltr.On Then
```

Vba
```
                Resultat = "1er critère:" & Fltr.Criteria1 & vbCrLf

                If Fltr.Operator Then
                    Resultat = Resultat & "Opérateur:" & Fltr.Operator & vbCrLf
                    Resultat = Resultat & "2eme critère:" & Fltr.Criteria2
                End If

                MsgBox Resultat
            End If
        Next Fltr
    End If
```

Cet autre exemple nomme des cellules non adjacentes, issues d'un filtre automatique.

Vba
```
    ActiveWorkbook.Names.Add Name:="Zone1", _
        RefersTo:="=Feuil1!" & Feuil1.AutoFilter.Range.SpecialCells(xlCellTypeVisible).Address
```

II-B-3 - AutoFilterMode

Permet:

* De savoir si les flèches du filtre automatique sont affichées dans la feuille.

* D'enlever les flèches du filtre automatique qui sont affichées dans la feuille.

Vba
```
    'Indique s'il y a un filtre automatique placé dans la feuille
    MsgBox Worksheets("Feuil1").AutoFilterMode


    'Enlève le filtre automatique de la Feuil1
    Worksheets("Feuil1").AutoFilterMode = False
```

II-B-4 - Cells

Représente toutes les cellules contenues dans la feuille de calcul.

Vous pouvez manipuler l'ensemble des cellules, ou une seule en précisant son numéro de ligne et son numéro de colonne.

Vba
```
    Dim NumLigne As Integer, NumColonne As Integer

    NumLigne = 10
    NumColonne = 1
```

Vba
```
'Ecrit dans la cellule A10 de la Feuil1
Worksheets("Feuil1").Cells(NumLigne, NumColonne).Value = "essai"


'Affecte la taille de police 12 à toute les cellules de la feuille
Worksheets("Feuil1").Cells.Font.Size = 12
```

II-B-5 - CircularReference

Représente un objet Range contenant la première référence circulaire de la feuille.

Une référence circulaire est une formule qui fait référence à sa propre cellule. Cela bloque tous les calculs tant qu'un des classeurs ouverts contient une référence circulaire.

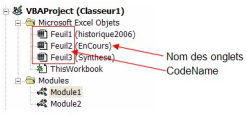
Cet exemple montre comment vérifier s'il y a une référence circulaire dans la feuille.

Vba
```
Dim oCirc As Range

On Error Resume Next
Set oCirc = Worksheets("Feuil1").CircularReference
On Error GoTo 0

If oCirc Is Nothing Then
    MsgBox "Il n'y a pas de référence circulaire dans la feuille."
    Else
    MsgBox "Il y a une référence circulaire dans cellule: " & oCirc.Address & _
      vbCrLf & oCirc.Formula
End If
```

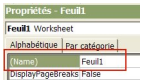
II-B-6 - CodeName

Représente le nom de code de l'objet Feuille.

Il ne faut pas le confondre avec le nom de l'onglet. Les 2 sont identiques par défaut, mais le CodeName ne change pas lorsque vous modifiez le nom d'un onglet.



Vous pouvez aussi visualiser ce nom dans la fenêtre propriétés de la feuille.



Cet exemple boucle sur les feuilles de calcul, puis renvoie le nom des onglets ainsi que le CodeName correspondant.

Vba
```
Dim Ws As Worksheet

For Each Ws In ThisWorkbook.Worksheets
    MsgBox Ws.Name & " ---> " & Ws.CodeName
Next Ws
```

Il est possible de modifier le CodeName par macro, à condition que celui ci ne soit pas en cours d'utilisation. Le nom ne doit pas contenir d'espaces.

Vba
```
'---
'Nécessite dd'activer la référence "Microsoft Visual Basic For Application Extensibility 5.3"
'---
'
'Attention: Le CodeName n'accepte pas les espaces dans le nom
'
With ThisWorkbook
    .VBProject.VBComponents(.Worksheets("historique2006").CodeName).Name = "NouveauNom"
End With
```

Les CodeName sont aussi utilisés lorsque vous devez **manipuler les modules par macro**.

II-B-7 - Columns

Représente les colonnes de la feuille de calcul. Columns renvoie un objet Range.

Vba
```
'Applique la couleur jaune dans la 2eme colonne
Worksheets("Feuil1").Columns(2).Interior.ColorIndex = 6

'Applique la couleur jaune dans les colonnes E à G
Worksheets("Feuil1").Columns("E:G").Interior.ColorIndex = 6
```

II-B-8 - Comments

Représente les commentaires placés dans la feuille de calcul.

La procédure ci dessous boucle sur les commentaires de la Feuil1 pour en extraire le contenu et identifier leur emplacement.

Vba
```
Dim Cmt As Comment

For Each Cmt In Feuil1.Comments
    'Affiche le contenu des commentaires
    MsgBox Cmt.Text & vbCrLf & vbCrLf & "Dans la cellule " & Cmt.Parent.Address
Next Cmt
```

II-B-9 - ConsolidationFunction

Renvoie la fonction utilisée dans la consolidation en cours.

Une consolidation permet de combiner les valeurs de plusieurs plages de données. Vous pouvez y accéder en utilisant le menu Données / Consolider.

Vba
```
If Not IsEmpty(Worksheets("Feuil1").ConsolidationSources) Then
    Select Case Worksheets("Feuil1").ConsolidationFunction
        Case xlAverage: MsgBox "Moyenne"
        Case xlCount: MsgBox "Nombre"
        Case xlCountNums: MsgBox "Nb"
        Case xlMax: MsgBox "Max"
        Case xlMin: MsgBox "Min"
        Case xlProduct: MsgBox "Produit"
        Case xlStDev: MsgBox "EcartType"
        Case xlStDevP: MsgBox "EcartTypeP"
        Case xlSum: MsgBox "Somme"
        Case xlUnknown: MsgBox "Inconnu"
        Case xlVar: MsgBox "Var"
        Case xlVarP: MsgBox "VarP"
    End Select
End If
```

Nota:

La propriété ConsolidationSources permet de vérifier s'il y a une consolidation en cours, sinon la constante xlSum est renvoyée par défaut.

II-B-10 - ConsolidationOptions

Renvoie les options relatives à la consolidation.

Renvoie True pour chaque option cochée.

Vba

```
Dim Tableau() As Variant

Tableau = Worksheets("Feuil1").ConsolidationOptions

MsgBox "Etiquettes dans la ligne du haut: " & Tableau(1) & vbCrLf & _
    "Etiquettes dans la colonne de gauche: " & Tableau(2) & vbCrLf & _
    "Liaisons vers les données sources: " & Tableau(3)
```

II-B-11 - ConsolidationSources

Renvoie le nom des plages source, sous forme de tableau, pour la consolidation de la feuille.

Renvoie Empty s'il n'existe pas de consolidation dans la feuille de calcul (Voir aussi l'exemple proposé pour le chapitre de la propriété ConsolidationFunction).

Vba

```
Dim Tableau2() As Variant
Dim x As Byte

If Not IsEmpty(Worksheets("Feuil1").ConsolidationSources) Then

    Tableau2 = Worksheets("Feuil1").ConsolidationSources

    For x = 1 To UBound(Tableau2)
        MsgBox Tableau2(x)
    Next x

End If
```

II-B-12 - Count

Permet de compter le nombre de feuilles ou d'onglets dans le classeur.

Vba

```
'Compte le nombre de feuilles de calcul
'(les onglets graphiques ne sont pas pris en compte)
MsgBox ThisWorkbook.Worksheets.Count

'Compte le nombre d'onglets
MsgBox ThisWorkbook.Sheets.Count
```

II-B-13 - CustomProperties

Représente les propriétés personnelles ajoutées dans la feuille de calcul.

Vba
```
'Ajoute une propriété personnelle dans la feuille
Worksheets("Feuil1").CustomProperties.Add Name:="NomMetaData", Value:="Donnée MetaData"
```

Vba
```
'Lit la 1ere propriété personnelle de la feuille
With Worksheets("Feuil1").CustomProperties.Item(1)
    MsgBox .Name & ": " & .Value
End With
```

II-B-14 - DisplayPageBreaks

Cette propriété affiche les sauts de page dans la feuille.

Ne fonctionne pas si aucune imprimante n'est installée.

Vba
```
Worksheets("Feuil1").DisplayPageBreaks = True
MsgBox "Affichage des sauts de page: " & Worksheets("Feuil1").DisplayPageBreaks
```

II-B-15 - EnableAutoFilter

Permet de spécifier si les actions du filtre automatique sont accessibles quand la feuille est protégée.

Cette propriété prend la valeur True si le Filtre est utilisable, et doit être précisée avant la protection de la feuille.

Vba
```
'Vérifie le statut du filtre automatique (valeur Faux par défaut)
MsgBox Worksheets("Feuil1").EnableAutoFilter
'Active l'autorisation de filtre automatique
Worksheets("Feuil1").EnableAutoFilter = True
'Vérifie le statut du filtre automatique après la modification de la propriété
MsgBox Worksheets("Feuil1").EnableAutoFilter
'Protège la feuille
Worksheets("Feuil1").Protect
```

La propriété peut être modifiée par macro pendant que la feuille est protégée, uniquement si l'argument userInterfaceOnly a été spécifié lors de la mise en place de la protection (Voir aussi le chapitre II-A-21).

Vba
```
'Protège la feuille
Worksheets("Feuil1").Protect userInterfaceOnly:=True
'Autorise l'utilisation du filtre
Worksheets("Feuil1").EnableAutoFilter = True
```

II-B-16 - EnableCalculation

Permet d'indiquer si le recalcul doit se faire de façon automatique dans la feuille spécifiée. La valeur False empêche tous les calculs dans la feuille.

Vba
```
'Vérifie le statut du recalcul automatique
MsgBox Worksheets("Feuil1").EnableCalculation
'Empèche le recalcul automatique dans la Feuil1
Worksheets("Feuil1").EnableCalculation = False
'Vérifie le statut du recalcul automatique après la modification de la propriété
MsgBox Worksheets("Feuil1").EnableCalculation
```

Pour réactiver le calcul automatique dans la feuille.

Vba
```
'Tous les calculs sont immédiatement effectuées lorsque la propriété prend la valeur True
Worksheets("Feuil1").EnableCalculation = True
```

II-B-17 - EnableOutlining

Permet de spécifier si l'accès aux symboles du plan est possible quand la feuille est protégée.

Cette propriété prend la valeur True si le plan est utilisable. L'argument userInterfaceOnly doit être spécifié lors de la mise en place de la protection (Voir aussi le chapitre II-A-21).

Vba
```
'Autorise l'accès aux symboles du plan
Worksheets("Feuil1").EnableOutlining = True
'Protège la feuille
Worksheets("Feuil1").Protect userInterfaceOnly:=True
  'Nota:
  'La valeur de la propriété peut être modifiée après la mise en place
  'de la protection.
```

II-B-18 - EnablePivotTable

Permet de spécifier et vérifier si les manipulations des tableaux croisés dynamiques sont possibles lorsque la feuille est protégée.

Cette propriété prend la valeur True si les TCD sont utilisables. L'argument userInterfaceOnly doit être spécifié lors de la mise en place de la protection (Voir aussi le chapitre II-A-21).

Vba
```
'Autorise la manipulation des TCD dans la feuille protégée
```

```Vba
Worksheets("Feuil1").EnablePivotTable = True
'Protège la feuille
Worksheets("Feuil1").Protect contents:=True, userInterfaceOnly:=True
  'Nota:
  'La valeur de la propriété peut être modifiée après la mise en place
  'de la protection.
```

II-B-19 - EnableSelection

Permet de spécifier et vérifier si les cellules peuvent être sélectionnées dans une feuille protégée.

La propriété peut prendre une des trois constantes suivantes:

* xlNoSelection (-4142) : Empêche toute sélection dans la feuille.

* xlNoRestrictions (0 Valeur par défaut) : Permet de sélectionner n'importe quelle cellule.

* xlUnlockedCells (1) : Permet de sélectionner uniquement les cellules déverrouillées.

La propriété peut être modifiée par macro pendant que la feuille est protégée, uniquement si l'argument userInterfaceOnly a été spécifié lors de la mise en place de la protection (Voir aussi le chapitre II-A-21).

```Vba
'Définit le type de sélection autorisé avant de protéger la feuille:
'Permet de sélectionner uniquement les cellules déverrouillées.
Worksheets("Feuil1").EnableSelection = xlUnlockedCells
'Protège la feuille
Worksheets("Feuil1").Protect UserInterfaceOnly:=True
  'Nota:
  'La valeur de la propriété peut être modifiée après la mise en place
  'de la protection.
```

II-B-20 - FilterMode

Renvoie la valeur True si un filtre (Menu/Données/Filtrer) placé dans la feuille contient des lignes masquées.

Cette macro vérifie s'il y a des filtres actifs dans feuille et les enlève si c'est le cas.

```Vba
With Worksheets("Feuil1")
    If .FilterMode = True Then .ShowAllData
End With
```

II-B-21 - HPageBreaks

Représente les sauts de pages horizontaux dans la feuille de calcul.

Vba
```
Dim Hpb As HPageBreak

'Compte le nombre de pages dans la feuille
MsgBox "Cette feuille contient " & Worksheets("Feuil1").VPageBreaks.Count + _
    Worksheets("Feuil1").HPageBreaks.Count + 1 & " pages"

'Boucle sur les sauts de pages horizontaux
For Each Hpb In Worksheets("Feuil1").HPageBreaks
    'Renvoie l'adresse de chaque cellule suivant le saut de page
    MsgBox Hpb.Location.Address
Next Hpb
```

II-B-22 - Hyperlinks

Représente les liens hypertextes dans la feuille de calcul.

La procédure suivante boucle sur les liens hypertextes contenus dans la feuille.

Vba
```
Dim Hpk As Hyperlink

'Boucle sur les liens hypertextes de la feuille
For Each Hpk In Worksheets("Feuil1").Hyperlinks
    MsgBox Hpk.Address & vbCrLf & Hpk.Range.Address
Next Hpk
```

Cet exemple ajoute un lien hypertexte dans la cellule B5.

Vba
```
'Ajoute un lien hypertexte dans la cellule B5
Worksheets("Feuil1").Hyperlinks.Add Anchor:=Range("B5"), _
    Address:="http://www.developpez.com", TextToDisplay:="Mon lien hypertexte"
```

Cette dernière macro ajoute un onglet "Sommaire" qui se place en début de classeur et crée des liens vers chacune des autres feuilles.

Vba
```
Sub Test()
    CreeSommaire ThisWorkbook
End Sub
```

```Vba
Sub CreeSommaire(Wb As Workbook)
    Dim Ws As Worksheet
    Dim I As Byte
    Dim Cible As String

    'Ajoute une feuille (Sommaire) et la positionne en première position
    Set Ws = Wb.Worksheets.Add(before:=Wb.Sheets(1))
    Ws.Name = "SommaireFeuilles"

    'boucle sur les feuilles de calcul (hors feuille sommaire)
    For I = 2 To Wb.Worksheets.Count
            Cible = "'" & Wb.Worksheets(I).Name & "'!A1"
            'Crée le lien dans la feuille sommaire
            Ws.Hyperlinks.Add Anchor:=Ws.Cells(I, 1), Address:="", _
                SubAddress:=Cible, TextToDisplay:=Wb.Worksheets(I).Name
    Next I
End Sub
```

II-B-23 - Index

Renvoie la position de la feuille dans le classeur.

```Vba
MsgBox Worksheets("Nom Feuille").Index
```

II-B-24 - MailEnvelope

Représente l'en-tête de message électronique pour la feuille spécifiée.

Cette procédure envoie un mail en plaçant la plage de cellule A1:B5 dans le corps du message.

```Vba
Sub envoiPlageCellules_Excel2002()
'http://support.microsoft.com/default.aspx?scid=kb;en-us;816644
Worksheets("Feuil1").Range("A1:B5").Select ' la plage de cellules à envoyer

ActiveWorkbook.EnvelopeVisible = True

With Worksheets("Feuil1").MailEnvelope
    .Introduction = "bonjour , ci joint les données ..."
    .Item.To = "destinataire@mail.fr"
    .Item.Subject = "le sujet"
    .Item.attachments.Add "C:\dossier\ma piece jointe.txt"
    .Item.Send
End With
End Sub
```

II-B-25 - Name

Renvoie ou définit le nom de l'onglet.

Vba

```
'Modifie le nom de la 2eme feuille de calcul
Worksheets(2).Name = "Nouveau Nom"
'Récupère le nom de la 2eme feuille de calcul
MsgBox Worksheets(2).Name
```

II-B-26 - Outline

Représente le plan de la feuille de calcul.

Vba

```
Dim Ot As Outline
'Définit le plan dans la feuille
Set Ot = Worksheets("Feuil1").Outline
'Affiche le plan jusqu'au 3eme niveau de ligne
Ot.ShowLevels rowLevels:=3
```

II-B-27 - PageSetup

Représente les paramètres de mise en page de la feuille.

Vba

```
'Ajoute une info dans le pied de page gauche
Worksheets("Feuil1").PageSetup.LeftFooter = "La description perso"

'Ajoute le nom de l'utilisateur et la date dans le pied de page central
Worksheets("Feuil1").PageSetup.CenterFooter = Environ("username") & ", Le " & Date

'--- Ajoute une image dans l'entête gauche (à partir d'Excel202) ---
With Worksheets("Feuil1").PageSetup.LeftHeaderPicture
    .Filename = "http://www.developpez.net/forums/images/logo16.gif"
    .Height = 80 ' redéfinit la largeur de l'image
    .Width = 120 ' redéfinit la hauteur de l'image
End With

'Remarque: Il est nécessaire d'ajouter la chaîne "&G"
'de la propriété LeftHeader, afin que l'image s'affiche dans l'en-tête gauche?
'(info issue de l'aide en ligne Excel).
Worksheets("Feuil1").PageSetup.LeftHeader = "&G"
'-----------------------------------------------------------------

'Ajoute une info dans l'entête droit en personnalisant
'la police, écriture en gras , taille 16 et texte souligné
Worksheets("Feuil1").PageSetup.RightHeader = "&""Arial,Gras""&16&U" & "Developpez.com"
```

La macro suivante centre le contenu de la feuille pour l'impression.

Vba

Vba
```
With Worksheets("Feuil1")
    .PageSetup.CenterHorizontally = True
    .PageSetup.CenterVertically = True
    .PrintOut
End With
```

II-B-28 - Parent

Renvoie l'objet parent de la feuille: c'est à dire le classeur.

Cette procédure permet d'extraire les cellules et plages nommées contenues dans une feuille spécifique.

Vba
```
Dim Plage As Range
Dim Nm As Name

On Error Resume Next
'Boucle sur les noms du classeur
For Each Nm In Worksheets("Feuil1").Parent.Names
    Set Plage = Nm.RefersToRange

    If Not Plage Is Nothing Then
        'Vérifie si le nom appartient à la feuille
        If Worksheets("Feuil1").Name = Plage.Worksheet.Name Then _
            MsgBox Plage.Address
    End If

    Set Plage = Nothing
Next Nm
```

II-B-29 - ProtectContents

Vérifie si le contenu de la feuille est protégé: Renvoie Vrai ou Faux.

Vba
```
'Renvoie Vrai ou Faux
MsgBox Worksheets("Feuil1").ProtectContents
```

II-B-30 - ProtectDrawingObjects

Vérifie si les formes contenues dans la feuille sont protégées: Renvoie Vrai ou Faux.

Les formes peuvent être des images, des contrôles, des formes automatiques(Shapes).

Vba
```
'Renvoie Vrai ou Faux
MsgBox Worksheets("Feuil1").ProtectDrawingObjects
```

II-B-31 - Protection

Vérifie si les arguments de protection sont activés dans la feuille:

AllowFormattingCells, AllowFormattingColumns, AllowFormattingRows, AllowInsertingColumns, AllowInsertingRows, AllowInsertingHyperlinks,

AllowDeletingColumns, AllowDeletingRows, AllowSorting, AllowFiltering, AllowUsingPivotTables.

Consultez le chapitre II-A-21 pour voir comment appliquer ces arguments.

Vba

```
'Protège la feuille en autorisant l'utilisation des tris
'(les cellules de la plage à trier doivent être déverrouillées)
Worksheets("Feuil1").Protect AllowSorting:=True

'Contrôle si le tri est autorisé après la mise en place de la protection
MsgBox Worksheets("Feuil1").Protection.AllowSorting
'Contrôle si l'utilisation des tableaux croisés dynamiques est autorisée
MsgBox Worksheets("Feuil1").Protection.AllowUsingPivotTables
```

II-B-32 - ProtectionMode

Vérifie si l'argument UserInterfaceOnly est égal à True lorsque la feuille est protégée (Voir le chapitre II-A-21).

Vba

```
'Contrôle le statut de protection
MsgBox Worksheets("Feuil1").ProtectionMode
'Protège la feuille
Worksheets("Feuil1").Protect UserInterfaceOnly:=True
'Contrôle le statut de protection après la modification
MsgBox Worksheets("Feuil1").ProtectionMode
```

II-B-33 - ProtectScenarios

Vérifie si les scénarios de la feuille sont protégés: Renvoie Vrai ou Faux.

Vba

```
'Renvoie Vrai ou Faux
MsgBox Worksheets("Feuil1").ProtectScenarios
```

II-B-34 - QueryTables

Représente les tables de requête contenues dans la feuille de calcul.

Cet exemple permet de créer une requête dans une base Access et d'afficher le résultat à partir de de la cellule E3.

Vba

```
    With Worksheets("Feuil1").QueryTables.Add(Connection:= _
            "ODBC;DSN=MS Access Database;DBQ=C:\dataBase.mdb;DefaultDir=C:;DriverId=25;" & _
            "FIL=MS Access;MaxBufferSize=2048;PageTimeout=5;", Destination:=Range("E3"))

        .CommandText = "SELECT Table1.CodeClient, Table1.CityName, Table1.NomClient " & _
            "FROM `C:\dataBase`.Table1 Table1 WHERE Table1.CityName='condrieu'"

        .Name = "Ma requête dans Access"
        .FieldNames = True
        .RowNumbers = False
        .FillAdjacentFormulas = False
        .PreserveFormatting = True
        .RefreshOnFileOpen = False
        .BackgroundQuery = True
        .RefreshStyle = xlInsertDeleteCells
        .SavePassword = True
        .SaveData = True
        .AdjustColumnWidth = True
        .RefreshPeriod = 0
        .PreserveColumnInfo = True
        .Refresh BackgroundQuery:=False
    End With
```

II-B-35 - Range

Représente une cellule ou une plage de cellules dans la feuille.

Vba

```
'Récupère le contenu de la cellule A1
MsgBox Worksheets("Feuil1").Range("A1")
```

Vous pouvez spécifier les cellules d'une plage à partir de leur nom Range("A1:B10") ou en utilisant la référence aux lignes et colonnes Range(Cells(1, 1), Cells(10, 2)):

Cells(1, 1) correspond à la cellule du bord supérieur gauche dans la plage (A1).

Cells(10, 2) correspond à la cellule du bord inférieur droit dans la plage (B10).

Vba

```
'Dans cet exemple les deux lignes de macro spécifient la même plage de cellules:

  'Paramètre chaque cellule de la plage pour centrer le texte
  Worksheets("Feuil1").Range("A1:B10").HorizontalAlignment = xlCenter

  'Insère du texte dans chaque cellule de la plage
  Worksheets("Feuil1").Range(Cells(1, 1), Cells(10, 2)).Value = "x"
```

Cet exemple affiche le contenu de la dernière cellule non vide dans la colonne A.

Vba

```
Dim i As Long

'Renvoie le numéro de ligne de la dernière cellule non vide dans la colonne A.
i = Worksheets("Feuil1").Range("A65536").End(xlUp).Row

MsgBox Worksheets("Feuil1").Range("A" & i)
MsgBox Worksheets("Feuil1").Cells(i, 1)
```

II-B-36 - Rows

Définit les lignes de la feuille de calcul. Rows renvoie un objet Range.

Vba

```
'Applique la couleur jaune dans la 2eme ligne
Worksheets("Feuil1").Rows(2).Interior.ColorIndex = 6

'Applique la couleur jaune dans les lignes 5 à 8
Worksheets("Feuil1").Rows("5:8").Interior.ColorIndex = 6
```

La procédure suivante supprime les lignes si les cellules de la colonne A sont vide.

Vba

```
Dim i As Integer, j As Integer
Dim Cell As Range

Application.ScreenUpdating = False

    'Récupère le numero de ligne de la dernière cellule
    'non vide dans la colonne A.
    i = Worksheets("Feuil1").Range("A65536").End(xlUp).Row

    'boucle de la dernière à la premiere ligne
    For j = i To 1 Step -1
        'Vérifie si les cellules de la colonne A sont vides
        'et les supprime si c'est le cas.
        If Worksheets("Feuil1").Cells(j, 1) = "" Then _
            Worksheets("Feuil1").Rows(j).Delete

    Next j

Application.ScreenUpdating = True
```

II-B-37 - ScrollArea

Limite la possibilité de déplacement dans la feuille.

Il est alors impossible de sélectionner les cellules situées en hors de la plage spécifiée.

Vba
```
'Limite le déplacement à la plage A1:E50, dans la 2eme feuille.
Worksheets(2).ScrollArea = "A1:E50"
MsgBox "Le déplacement est limité à la plage " & Worksheets(2).ScrollArea
```

Et pour supprimer la limitation de déplacement:

Vba
```
Worksheets(2).ScrollArea = ""
```

II-B-38 - Shapes

Représente toutes les formes placées dans la feuille.

Une peut être un contrôle de la barre d'outils formulaire ou de la boîte à outils contrôles, les graphiques incorporés, les formes automatiques ou libres, les images...

Vba
```
'Ajoute une forme automatique (Rectangle) dans la feuille
With Worksheets("Feuil1").Shapes.AddShape(msoShapeRectangle, 40, 80, 140, 50)
    .Name = "NomForme"
    .TextFrame.Characters.Text = "Le texte dans la forme"
End With
```

Cet exemple boucle sur les formes contenues dans la feuille et renvoie leur nom et leur type.

Vba
```
Dim Obj As Shape

'Compte le nombre de formes dans la feuille
MsgBox "Nombre de formes dans la feuille: " & Worksheets("Feuil1").Shapes.Count

'Boucle sur les formes contenues dans la feuille
For Each Obj In Worksheets("Feuil1").Shapes
    Select Case Obj.Type
        '--- MsoShapeType peut être une des constantes suivantes ---
        '-----------------------------------------------------------

            'Consultez aussi les constantes AutoShapeType pour identifier les différents
            'types de formes automatiques msoAutoShape.
        Case msoAutoShape: MsgBox "Forme automatique: " & Obj.Name
        'Case msoCallout
        Case msoChart: MsgBox "Graphique: " & Obj.Name
```

Vba
```
            Case msoComment: MsgBox "Commentaire: " & Obj.Name
            Case msoDiagram: MsgBox "Diagramme/Organigramme: " & Obj.Name
            Case msoEmbeddedOLEObject: MsgBox "Objet incorporé: " & Obj.Name
                'Nota:
                'Chaque bouton des filtres automatiques est considéré
                'comme un objet formulaire (zone de liste/DropDown).
            Case msoFormControl: MsgBox "Objet formulaire: " & Obj.Name
            Case msoFreeform: MsgBox "Forme libre: " & Obj.Name
            Case msoGroup: MsgBox "Groupe: " & Obj.Name
            Case msoLine: MsgBox "Ligne: " & Obj.Name
            Case msoLinkedOLEObject: MsgBox "Objet incorporé lié au fichier: " & Obj.Name
            Case msoOLEControlObject: MsgBox "Objet de la boîte à outils Contrôle: " & Obj.Name
            Case msoPicture: MsgBox "Image: " & Obj.Name
                'msoShapeTypeMixed permet de vérifier si une plage de formes contient
                'des types d'objets différents:
                'Par exemple: If Selection.ShapeRange.Type = msoShapeTypeMixed Then
            'Case msoShapeTypeMixed: MsgBox "Combinaison de formes"
            Case msoTextBox: MsgBox "Zone de texte: " & Obj.Name
            Case msoTextEffect: MsgBox "WordArt: " & Obj.Name
        End Select
    Next Obj
```

II-B-39 - StandardHeight

Renvoie la hauteur par défaut (en points) pour les lignes de la feuille.

Un point est égal à 1/72ème de pouce.

Cet exemple réinitialise la hauteur standard de toutes les lignes:

Vba
```
    Dim Hauteur As Double

    Hauteur = Worksheets("Feuil1").StandardHeight
    Worksheets("Feuil1").Rows.RowHeight = Hauteur
```

II-B-40 - StandardWidth

Renvoie la largeur par défaut pour les colonnes de la feuille.

Cet exemple réinitialise la largeur standard de toutes les colonnes:

Vba
```
    Dim Largeur As Double

    Largeur = Worksheets("Feuil1").StandardWidth
    Worksheets("Feuil1").Columns.ColumnWidth = Largeur
```

II-B-41 - Tab

Représente l'onglet dans la feuille de calcul.

Cet exemple permet d'appliquer une couleur jaune à l'onglet de la Feuil1 (fonctionne uniquement à partir d'Excel2002).

Vba
```
Worksheets("Feuil1").Tab.ColorIndex = 6
```

II-B-42 - Type

Permet de récupérer le type de feuille contenu dans le classeur.

Vba
```
Dim Resultat As String
Dim Ws As Object

For Each Ws In ThisWorkbook.Sheets
    Select Case Ws.Type
        Case -4167: Resultat = Resultat & Ws.Name & " --->  xlWorksheet" & vbCr
        Case -4169: Resultat = Resultat & Ws.Name & " --->  xlChart" & vbCr
        Case 4: Resultat = Resultat & Ws.Name & " --->  xlExcel4IntlMacroSheet" & vbCr
        Case 3: Resultat = Resultat & Ws.Name & " --->  xlExcel4MacroSheet" & vbCr
    End Select
Next Ws

MsgBox Resultat
```

Nota:

Utilisez les valeurs plutot que les constantes afin de vérifier le type d'une feuille. Lors de mes tests (Excel2002), la constante xlChart renvoie -4109 alors que la propriété Type renvoie -4169 quand il s'agit d'un onglet graphique.

II-B-43 - UsedRange

Renvoie un objet Range qui définit la plage de cellules utilisée dans la feuille de calcul.

Vba
```
MsgBox Worksheets("Feuil3").UsedRange.Address
'Nota:
'Renvoie $A$1 si la feuille est vide
```

II-B-44 - Visible

Permet d'afficher ou de masquer la feuille de calcul.

Vba

Vba
```
'Masque la 2eme feuille:
'-----------------------
  'Worksheets(2).Visible = False
  'Equivalent de
  Worksheets(2).Visible = xlSheetHidden


'Réaffiche la 2eme feuille:
'-----------------------
  Worksheets(2).Visible = True
  'Equivalent de
  'Worksheets(2).Visible = xlSheetVisible
```

Si vous utilisez la constante xlSheetVeryHidden, la feuille est masquée et n'apparaît plus dans la boîte de dialogue "Afficher" (Menu Format / Feuille / Afficher). Dans ce cas, seule une macro permet de rendre la feuille visible. La feuille reste néanmoins visible dans l'explorateur de projets VBE.

Vba
```
'Masque la 2eme feuille:
'-----------------------
  Worksheets(2).Visible = xlSheetVeryHidden


'Réaffiche la 2eme feuille:
'-----------------------
  'Worksheets(2).Visible = True
```

II-B-45 - VPageBreaks

Représente les sauts de page verticaux dans la feuille de calcul.

Vba
```
Dim Vpb As VPageBreak

'Compte le nombre de pages dans la feuille
MsgBox "Cette feuille contient " & Worksheets("Feuil1").VPageBreaks.Count + _
    Worksheets("Feuil1").HPageBreaks.Count + 1 & " pages"

'Boucle sur les sauts de pages verticaux
For Each Vpb In Worksheets("Feuil1").VPageBreaks
    'Renvoie le numero de colonne suivant le saut de page
    MsgBox Vpb.Location.Column
Next Vpb
```

II-C - Les évènements

Consultez le tutoriel qui décrit les évènements disponibles dans la feuille de calcul:

Les évènements dans la feuille de calcul Excel.

III - Cas particuliers

III-A - L'élément masqué Pictures

Il représente les images placées dans la feuille de calcul.

Cet élément est pris en charge uniquement pour assurer une compatibilité descendante entre les différentes versions d'Excel. Il préférable d'utiliser la propriété Shapes (type msoPicture) dans vos nouveaux projets. Néanmoins comme l'élément Pictures fonctionne encore (du moins jusqu'à Excel 2003) il peut être intéressant de voir son fonctionnement.

Cet exemple insère une image dans la feuille:

```
'Insère une image dans la feuille nommée Feuil1
Worksheets("Feuil1").Pictures.Insert "C:\dossier\Image2.jpg"
```

Cet autre exemple boucle sur toutes les images contenues dans la feuille afin de les enregistrer sur le disque:

```
Sub ExtraireImagesFeuille()
    Dim Pict As Picture
    Dim Nb As Byte

    Application.ScreenUpdating = False

    ' boucle sur les images de la feuille
    For Each Pict In Worksheets("Feuil1").Pictures
        'copie image
        Pict.CopyPicture

        'crée un graphique
        With ActiveSheet.ChartObjects.Add(0, 0, Pict.Width, Pict.Height).Chart
            'colle l'image dans graphique
            .Paste
            'enregistre le graphique au format gif
            .Export ThisWorkbook.Path & "\" & Pict.Name & ".gif", "GIF"
        End With

        Nb = ActiveSheet.ChartObjects.Count
        'supprime le graphique
        ActiveSheet.ChartObjects(Nb).Delete
    Next Pict

    Application.ScreenUpdating = True
End Sub
```

IV - Téléchargement