PROGRAMMATION ORIENTÉE OBJET

Le VBA est un langage de Programmation Orientée Objet (POO), c'est-à-dire qu'il utilise différents objets (classeur, feuille, cellules, etc.) auxquels il associe des attributs (taille, couleurs, nom, etc.), des valeurs ("Bonjour", 357, etc.), des événements (activer, fermer, etc.)… Voici donc une présentation de quelques objets avec la programmation de leurs attributs.

DÉFINITIONS

OBJET

Un objet est un élément (classeur, feuille, etc.) dont on peut définir ou modifier les propriétés.

PROCÉDURE

La procédure va accueillir le code de la macro. Elle commence par l'instruction **Sub** et se termine par l'instruction **End Sub**.

MODULE STANDARD

Le module standard est le support permettant d'écrire sa procédure.

PROJET

Un projet VBA est l'ensemble des modules de code VB associé à un classeur.

COLLECTION

La collection permet de travailler sur plusieurs objets.

PRINCIPE DES OBJETS

PROPRIÉTÉS DES OBJETS

Chaque objet a ses propres attributs.

On y fait référence par *Objet.Propriété*

Exemple avec les propriétés de paramétrage des évènements

La plupart de ces commandes peuvent avoir pour attributs *True* (Activée) ou *False* (désactivée)

Par défaut, les attributs de ces messages sont souvent à *True*. Exemple d'événements :

```
Application.ScreenUpdating = True ' Mise à jour de l'écran
Application.DisplayAlerts = False
' Désactive les messages d'alertes, de confirmation, etc.
Application.AskToUpdateLinks = True ' Confirmation des liens entre classeurs
Application.StatusBar = True ' Mise à jour des messages sur la barre d'état
Application.EnableEvents = False
' Désactive les événements qui peuvent se déclencher
' à l'arrivée sur un classeur, une feuille, une cellule etc.
Application.Calculation = xlManual
' Calcul manuel (ou xlAutomatic pour automatique)
```

MÉTHODES

Les méthodes sont des procédures ou des fonctions attachées aux objets.

Leur structure est : *objet.méthode argument1,argument2,…*

Exemple :

```
Range("B2:D12").Select ' sélection des cellules de B2 à D12
```

ÉVÉNEMENTS

Les objets peuvent répondre à des événements (ouverture d'un classeur, ajout d'une feuille, sélection de cellules, etc.)

PROCÉDURES ÉVÈNEMENTIELLES

Des procédures événementielles permettent la gestion de ces événements :

```
With <objet>
   code de la méthode ou les propriétés de l'objet
End With
```

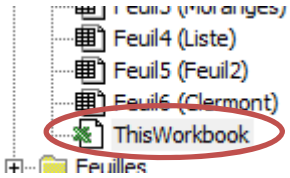
L'instruction *Set* permet de nommer un objet, pour le réutiliser ensuite, exemple :

```
Sub NouveauClasseur()
   Dim Classeur As Workbook
   ' Création d'un nouveau classeur :
   Set Classeur = Application.Workbooks.Add
   ' Affectation des noms des feuilles :
   With Classeur
      .Worksheets(1).Name = "Janvier"
      .Worksheets(2).Name = "Février"
      .Worksheets(3).Name = "Mars"
   End With
End Sub
```

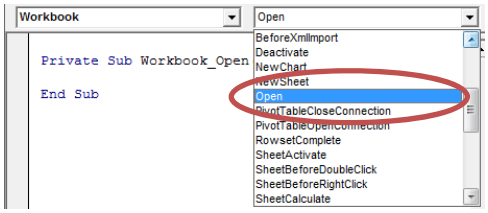
LANCEMENT AUTOMATIQUE D'UNE PROCÉDURE ÉVÉNEMENTIELLE

À l'ouverture d'un classeur

1. Accéder au code VBA (exemple : **Alt | F11**)
2. Dans le volet d'explorateur de projets, à gauche, cliquer sur « ThisWorkbook »



3. Dans la liste déroulante de gauche, choisir l'élément concerné (exemple : *Workbook* pour un classeur)



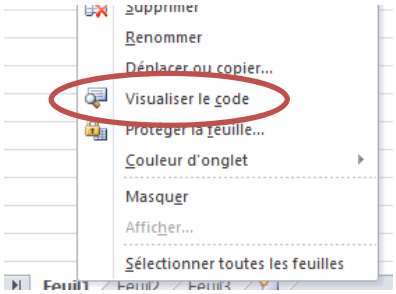
4. Dans la liste de droite, choisir l'évènement voulu (exemple : *Open* pour l'ouverture d'un classeur)
5. Éventuellement, effacer les procédures inutiles créées à chaque choix d'un élément dans la liste
6. Saisir le code voulu, exemple :

```
Private Sub Workbook_Open()
    Call MessageAccueil ' Lance la macro MessageAccueil
End Sub
```

- ✓ Voici quelques exemples de procédures gérant les événements à mettre à la fin du nom de la macro :
  - Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range) : déclenchement d'une macro lors de la modification du contenu d'une cellule
    Sh contiendra le nom de l'onglet modifié
    Target sera la cellule modifiée
  - _Open (comme Workbook_Open) ou _Beforeclose : à l'ouverture ou juste avant la fermeture
  - _Activate ou _Deactivate : lors de l'activation ou désactivation
- ✓ Lien vers les différents évènements des classeurs :
  silkyroad.developpez.com/VBA/EvenementsClasseur

Lors de la modification d'une zone de cellules dans une feuille

1. Clic droit sur l'onglet de la feuille voulue
   et cliquer sur *Visualiser le code*



   Il est également possible d'accéder aux procédures d'une feuille en cliquant deux fois sur la feuille voulue dans le volet des projets de la fenêtre du VBA.
2. Dans la liste déroulante de gauche, choisir l'élément concerné
   (exemple : *Worksheet* pour un classeur)
3. Dans la liste de droite, choisir l'évènement voulu (exemple : Change pour être lancer lors du changement du contenu de cellules).
4. Saisir le code voulu, exemple :

```
Private Private Sub Worksheet_Change(ByVal Target As Range)
    If Target.Address = Range("A1").Address Then MAJ_Nom
    If Target.Address = Range("A2").Address Then MAJ_Adresse
End Sub
```

Si le contenu de la cellule A1 change, la macro MAJ_Nom est exécutée, si c'est la cellule A2, c'est la macro MAJ_Adresse qui sera exécutée.

- ✓ Site présentant les différents évènements des feuilles de calcul :
  silkyroad.developpez.com/VBA/EvenementsFeuille

Désactiver et réactiver les évènements :

Comme un évènement qui se déclenche lors du changement d'une cellule peut lui-même modifier une cellule, il est parfois conseillé de désactiver le lancement automatique, exemple :

```
Private Sub Workbook_SheetChange(ByVal Sh As Object, ByVal Target As Range)
    Application.EnableEvents = False ' Désactive les évènements
    Call Date_MAJ
    Application.EnableEvents = True ' Réactive les évènements
End Sub
```

LES OBJETS D'EXCEL

WORKBOOKS

Les objets de la classe *Workbook* sont des classeurs Excel

EXEMPLE D'APPEL D'OBJETS DE LA CLASSE WORKBOOK :

Ouverture du classeur Compta.xlsx situé dans le même dossier que l'actuel classeur

```
Workbooks.Open Filename:=ThisWorkbook.Path & "\Compta.xlsx"
```

Sauvegarde le classeur actif sous le nom de *Facture.xlsx* dans le dossier *Clients* :

```
ActiveWorkBook.SaveAs ThisWorkbook.Path & "\Clients\Facture.xlsm"
```

- Mettre *ThisWorkbook* pour sauvegarder le classeur contenant le code

Active le classeur Ventes.xlsx (s'il est déjà ouvert)

```
WorkBooks("Ventes.xlsx").Activate
```

Ferme le classeur Gestion.xlsx

```
WorkBooks("Suivi.xlsx").Close
```

WORKSHEET

Les objets de la classe *WorkSheet* concernent les feuilles de calcul

EXEMPLE D'UTILISATIONS D'OBJETS DE LA CLASSE WORKSHEET

Active la feuille appelée janvier :

```
WorkSheets("Janvier").Activate
```

Ajoute trois feuilles après la dernière feuille du classeur :

```
Worksheets.Add after:=Worksheets(Worksheets.Count), Count:=3
```

Supprime la troisième feuille du classeur :

```
WorkSheets(3).Delete
```

Renomme la feuille active « Secteur Nord » :

```
ActiveSheet.Name = "Secteur Nord"
```

✓ *Worksheets* désigne les feuilles de calcul
Sheets désigne n'importe quelle feuille (calcul, graphique, etc.)

RANGE

L'objet Range désigne une plage de cellules

EXEMPLE D'UTILISATION D'OBJETS DE LA CLASSE RANGE

Range

Stocke le mot « Bonjour » dans la cellule A1.

```
Range("A1").Value = "Bonjour" ' .Value est facultatif
```

Copie dans le presse-papiers le contenu des cellules A2 à A30

```
Range("A2:A30").Copy
```

Colle le contenu du presse-papiers dans la cellule D2 de la feuille « Résumé »

```
Worksheets("Résumé").Range("D2").Activate
ActiveSheet.Paste
```

Compte le nombre de cellules contenues dans la zone nommée « Liste »

```
Range("Liste").Count
```

Cells

La variable Tarif récupère le contenu de la cellule C5 :

```
Tarif = Cells(5,3).Value ' Ici aussi .Value est facultatif
```

Écrit 1000 dans la cellule sélectionnée :

```
ActiveCell = 1000
```

Stocke, dans la variable *Emplacement*, les coordonnées de la cellule située deux lignes au-dessus et sept colonnes à droite de la cellule active :

```
Emplacement = ActiveCell.Offset(-2, 7).Address
```

Efface le contenu — valeurs et mise en forme — des cellules de B5 à D20 :

```
Range(Cells(5, 2), Cells(20, 4)).Clear ' Équivalent à Range("B5:D20").Clear
```

✓ *ClearContents* n'efface que les valeurs
ClearFormats n'efface que la mise en forme

Columns et Row

Met le texte en rouge dans toutes les cellules de la cinquième colonne :

```
Columns(5).Font.Color = -16776961
```

Remplit le fond des cellules de la première ligne en bleu :

```
Rows(1).Interior.Color = 12611584
```

Pour la zone active (ActiveCell)

Affiche le contenu de la cellule de la septième colonne de la ligne contenant le curseur :

```
MsgBox Cells(ActiveCell.Row, 7)
```

Affiche le contenu de la deuxième cellule de la colonne contenant le curseur :

```
MsgBox Cells(2, ActiveCell.Column)
```

Collection d'objets

Une collection d'objets comprend plusieurs objets. Un classeur Excel contient une collection de feuilles, elles-mêmes contiennent une collection de plages qui contiennent une collection de cellule.

Grâce aux collections, il est possible de travailler sur plusieurs objets en même temps, exemple :

```
MsgBox Workbooks("Secteurs.xlsx").Worksheets("Velay").Range("B10")
```

Affiche le contenu de la cellule B10, de la feuille *Velay* du classeur *Secteurs.xlsx*

Définition d'une collection d'objet avec With et End With

Afin de ne pas répéter à chaque fois la collection d'objet, il est possible de la définir une fois pour toute. Au lieu de répéter à chaque fois le nom du classeur et de la feuille à utiliser :

```
Workbooks("Voyage.xlsm").Worksheets("Europe").Range("A1") = "France"
Workbooks("Voyage.xlsm").Worksheets("Europe").Range("A2") = "Danemark"
Workbooks("Voyage.xlsm").Worksheets("Europe").Range("A3") = "Espagne"
```

Il est plus simple de définir une seule fois le classeur et la feuille à utiliser :

```
With Workbooks("Voyage.xlsm").Worksheets("Europe")
    .Range("A1") = "France"
    .Range("A2") = "Danemark"
    .Range("A3") = "Espagne"
End With
```

✓ Le point devant chaque objet — ici .Range(…) —indique qu'il se situe dans la collection définie avec *With*.

WorkSheetFunction

Objet utilisant les fonctions de calculs prédéfinies d'Excel

Appel d'un objet de la classe WorkSheetFunction

WorksheetFunction.NomFonction(argument1,argument2,…)

Exemple

```
Range("A31").Value = Application.WorksheetFunction.Average(Range("A2:A30"))
```

Affiche la moyenne des cellules de A2 à A30 dans la cellule A31

Quelques fonctions

- Sum : somme
- Average : moyenne
- Min : minimum
- Max : maximum
- Median : valeur médiane
- Count : nombre de cellules contenant une valeur numérique
- CountA : nombre de cellules non vides. ■