# Chapitre 3
# La programmation objet sous Excel

## 1. Présentation

VBA Excel est un langage de programmation **orienté objet**, même s'il ne dispose pas de toutes les fonctionnalités des langages de ce type.

La plupart des éléments manipulés dans Excel sont des objets : les classeurs, les feuilles de calcul, les plages de cellules, les cellules...

Les objets sont organisés selon un **modèle hiérarchique** : certains objets contiennent d'autres objets qui peuvent eux-mêmes en contenir d'autres... Ces objets sont appelés **conteneur** ou **objet parent**. Par exemple, l'objet **Application** est le conteneur des objets **Workbook** (classeurs ouverts dans Excel), qui sont eux-mêmes les conteneurs des objets **Worksheet** (feuilles de calcul d'un classeur). Le conteneur le plus vaste est l'objet **Application**.

Un ensemble d'objets de même nature constitue une **collection** (collection Workbooks : ensemble des classeurs ouverts dans Excel, collection Worksheets : ensemble des feuilles de calcul d'un classeur).

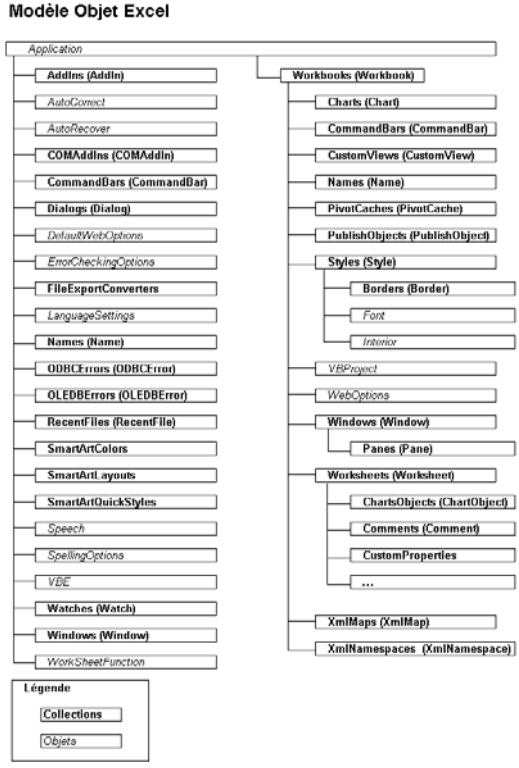
Un objet dispose d'un ensemble de caractéristiques appelées **propriétés** (par exemple pour l'objet **Application** : la propriété **UserName** représente le nom de l'utilisateur, la propriété **Version** renvoie le numéro de version de Microsoft Excel) et de comportements ou actions appelés **méthodes** (par exemple, toujours pour l'objet **Application**, la méthode **FindFile** affiche la boîte de dialogue **Ouvrir**, la méthode **Quit** quitte Excel...).

Un objet répond à des **événements** provoqués par l'utilisateur (ex : ouverture d'un classeur, clic sur un bouton de commande, changement de cellule active...) ou par le système.

Les **classes** sont des modèles permettant de créer des objets de même nature. Les objets issus d'une même classe héritent systématiquement de toutes les méthodes, propriétés et événements de leur classe d'origine. Il est possible de créer des classes d'objets avec VBA Excel en utilisant des modules de classe.

## 2. Le modèle objet Excel

### 2.1 Présentation



## 2.2 Principaux objets et collections

### Les objets

**Application**

Objet faisant référence à l'application Microsoft Excel active.

**AutoCorrect**

Objet représentant les attributs de correction automatique de Microsoft Excel.

**AutoRecover**

Objet représentant les options de récupération automatique d'un classeur. Ces macros sont accessibles depuis Excel à partir de l'onglet **Enregistrer** du menu **Outils** - **Options**.

**DefaultWebOptions**

Objet renvoyant les attributs utilisés par Excel lors de l'ouverture ou l'enregistrement d'une page web.

**ErrorCheckingOptions**

Objet représentant les options de vérification d'erreurs de l'application Excel.

**LanguageSettings**

Objet contenant des informations sur les paramètres de langue d'Excel.

**Speech**

Objet contenant des méthodes et des propriétés qui se rapportent aux fonctions vocales.

**SpellingOptions**

Objet représentant les options d'orthographe de l'application.

**VBE**

Objet VBE représentant Visual Basic Editor.

**WorkSheetFunction**

Objet contenant toutes les fonctions disponibles dans Excel. Cet objet permet d'obtenir le résultat d'une fonction appliquée à une plage de cellules. Exemple : Moy = Application.WorksheetFunction.Average(Selection)

### Les collections

**AddIns**

Collection contenant toutes les macros complémentaires (objets AddIn).

**COMAddIns**

Représente les compléments COM actuellement installés dans Microsoft Excel.

**CommandBars**

Collection des barres de commandes d'Excel (objets CommandBar).

**Dialogs**

Collection des boîtes de dialogue intégrées d'Excel.

**FileExportConverters**

Collection de tous les convertisseurs de fichier pour l'enregistrement des fichiers dans Microsoft Excel.

**Names**

Collection de tous les noms (cellules nommées) contenus dans le classeur actif.

**ODBCErrors**

Collection de toutes les erreurs ODBC générées par la dernière opération effectuée dans un rapport de tableau croisé dynamique ou dans une table de requête.

**OLEDBErrors**

Collection représentant les informations concernant l'erreur renvoyée par la requête OLE DB la plus récente.

**RecentFiles**

Collection des fichiers récemment utilisés.

**SmartArtColors**

Collection des styles de couleurs SmartArt actuellement chargés dans l'application.

**SmartArtLayouts**

Collection des jeux de disposition SmartArt actuellement chargés dans l'application.

**SmartArtQuickStyles**

Collection des jeux de style SmartArt actuellement chargés dans l'application.

**Watches**

Collection d'objets représentant les plages qui sont suivies lorsque la feuille de calcul est recalculée.

**Windows**

Collection de toutes les fenêtres de l'application Excel ou d'un classeur.

**Workbooks**

Collection des classeurs (objet Workbook) ouverts.

**Worksheets**

Collection des feuilles de calcul (objet Worksheet) d'un classeur.

# 3. Principes d'utilisation des objets et collections

## 3.1 Les propriétés

Les propriétés servent à décrire un objet. Certaines propriétés sont en lecture seule et ne peuvent par conséquent pas être modifiées par du code VBA.

Syntaxe

```
{<objet> | <variable objet>}.<propriété>
```

Exemple

```
'   Modification du pointeur de la souris
Application.Cursor = xlWait

'   Affichage de la version de l'application Excel active
'   Cette propriété est en lecture seule
MsgBox Application.Version

'   Pointeur de la souris par défaut
Application.Cursor = xlDefault
```

## 3.2 Propriétés représentant des objets

Les objets globaux et les objets instanciés dans le code à partir des classes fournies par VBA possèdent des propriétés dont la valeur est mise à jour automatiquement par le système.

Ces **propriétés spécifiques** permettent d'accéder directement à certains objets : fenêtre active, classeur actif, cellules de la feuille active... Le tableau suivant présente les propriétés spécifiques les plus couramment utilisées.

| Propriété | Objet Parent | Objet renvoyé |
|---|---|---|
| ActiveCell | Application<br>Window | Objet **Range** représentant la première cellule active de la fenêtre active ou spécifiée. |

| Propriété | Objet Parent | Objet renvoyé |
|---|---|---|
| ActiveChart | Application<br>Window<br>Workbook | Objet **Chart** représentant le graphique actif. |
| ActiveControl | Frame Page<br>UserForm | Objet **Control** représentant le contrôle (ActiveX) actif. |
| ActivePane | Window | Objet **Pane** représentant le volet actif de la fenêtre active. |
| ActiveSheet | Application<br>Window<br>Workbook | Objet **Worksheet** représentant la feuille active du classeur actif ou du classeur spécifié. |
| ActiveWindow | Application | Objet **Window** représentant la fenêtre active. |
| ActiveWorkBook | Application | Objet **Workbook** représentant le classeur de la fenêtre active. |
| Parent | Objets multiples | Renvoie l'objet conteneur. |
| Selection | Application<br>Window | Objet **Range** représentant la ou les cellules sélectionnées. |
| ThisCell | Application | Renvoie la cellule à partir de laquelle la fonction définie par un utilisateur est appelée en tant qu'objet Range. |
| ThisWorkbook | Application | Objet **Workbook** représentant le classeur dans lequel s'exécute le code de la macro en cours. |

Remarque

*Les propriétés spécifiques renvoyant un objet Range (Cells, Offset, Columns, Rows...) sont décrites en détail dans le chapitre Les objets d'Excel.*

## 3.3 Les méthodes

Les méthodes permettent d'effectuer des actions liées aux objets.

Elles se présentent un peu comme des procédures :

– Elles peuvent ou non utiliser des arguments.
– Certaines méthodes peuvent renvoyer une valeur au même titre que les procédures **Function**, d'autres non au même titre que les procédures **Sub**.

Syntaxe de méthode ne renvoyant pas de valeur

```
{<objet> | <variable objet>}.<méthode> [<Liste d'arguments>]
```

Exemple

```
'   Active la cellule A1 de la 2ème feuille de calcul
'   du classeur actif
Application.Goto ActiveWorkbook.Worksheets(2).Range("A1")
'   Sélectionne une plage de cellules
Range("A1:C12").Select
'   Efface les cellules sélectionnées
Selection.Clear
'   Enregistre le classeur actif sous un nouveau nom
ActiveWorkbook.SaveAs "C:\devis\devis2.xlsm"
```

Remarque

*Comme pour les procédures, les différents arguments d'une méthode doivent être séparés par des virgules. Si un argument facultatif n'est pas défini explicitement, la méthode utilisera une valeur par défaut.*

Syntaxe de méthode renvoyant une valeur

```
<variable> = {<objet> | <variable objet>}.<méthode>
[<Liste d'arguments>]
```

Exemple

```
Dim strFileName As String
'   Affiche une boîte de dialogue Ouvrir
strFileName = Application.GetOpenFilename _
      (FileFilter:="Classeurs Excel (*.xlsm), *.xlsm", _
      Title:="Sélectionnez le fichier à ouvrir")
'   Si un fichier Excel a été sélectionné, celui-ci est ouvert
If InStr(strFileName, "xls") <> 0 Then
   Workbooks.Open strFileName
Else
   MsgBox "Vous devez sélectionner un fichier Excel"
End If
```

## 3.4 Les événements

Un événement est une **action spécifique** qui se produit sur ou avec un certain objet. Microsoft Excel est en mesure de répondre à plusieurs types d'événements : ouverture ou fermeture d'un classeur, sélection de cellules, ajout d'une feuille de calcul... Les événements résultent généralement d'une action de l'utilisateur.

L'utilisation d'une procédure événementielle vous permet d'associer votre propre code en réponse à un événement qui se produit dans un classeur, une feuille ou un formulaire.

Exemple

Lorsque l'utilisateur ajoute une nouvelle feuille de calcul au classeur, un message utilisateur est affiché.

```
Private Sub Workbook_NewSheet(ByVal Sh As Object)

MsgBox "La feuille " & Sh.Name & Chr(13) & _
       "vient d'être ajoutée au classeur " & _
       ActiveWorkbook.Name & Chr(13) & _
       "Le nombre de feuilles du classeur est maintenant de " & _
       ActiveWorkbook.Worksheets.Count

End Sub
```