Chapitre 5
Les objets et collections en VBA

1. Notion d'objet

Le VBA est un langage qui permet de faire de la programmation orientée objet (POO) : un objet représente une idée, un concept ou toute entité du monde réel, comme un avion, un individu ou encore un film. Il possède une structure interne et un comportement, et peut communiquer avec ses pairs. Les éléments qui permettent de décrire un objet forment ce qu'on appelle une classe. Chaque objet issu d'une classe est une instance de classe. Les classes comportent des propriétés, des méthodes et des événements.

1.1 Propriétés

L'objet est une entité que l'on peut distinguer grâce à ses propriétés (sa couleur, ses dimensions, par exemple). Si l'on prend par exemple un livre, il est caractérisé par ses propriétés : nombre de pages, titre, nombre de chapitres, éditeur, contenu, etc. Chacune de ses propriétés peut être spécifique à chaque livre, mais tous les livres possèdent fondamentalement les mêmes propriétés. Certaines propriétés des objets peuvent être modifiées (vitesse d'une voiture par exemple) et d'autres non (une marque de voiture).

En programmation VBA, la syntaxe générale d'accès aux propriétés d'un objet est la suivante :

```
MonObjet.SaPropriete
```

On accède ici à la propriété `SaPropriete` de l'objet `MonObjet`. La combinaison `MonObjet.SaPropriete` aura le même comportement qu'une variable classique, pouvant prendre une valeur ou retourner une valeur avec l'utilisation de l'opérateur =.

Par exemple, il est possible de lire le contenu SQL d'une requête Access et de l'afficher avec le code suivant :

```
Sub LireSQLRequete()
    Dim UneRequete As QueryDef
    UneRequete.SQL = "SELECT * FROM MaTable"
    ...
    MsgBox UneRequete.SQL
End Sub
```

Nous reviendrons plus en détail sur l'objet `QueryDef` dans le chapitre Les objets d'accès aux données DAO et ADO.

Une propriété d'un objet peut être elle-même un objet, ce qui pourra être utilisé par la suite dans le code.

1.2 Méthodes

En plus des éléments qui caractérisent un objet, il est également possible de réaliser des actions avec ces objets, comme par exemple "décoller", "accélérer", "tourner la page", etc. Ces actions sont appelées des méthodes. Ces actions sont représentées sous la forme de procédures et de fonctions en VBA selon qu'elles peuvent ou non retourner des valeurs. Certaines actions peuvent nécessiter des paramètres pour fonctionner (nombre de caractères dans un livre, avec ou sans espace).

En programmation VBA, la syntaxe générale d'accès aux méthodes d'un objet est la suivante :

```
MonObjet.SaMethode [Monparametre]
```

Dans l'exemple suivant, il s'agit de simplement rafraîchir le lien entre une table liée et l'application Access :

```
Sub RafraichirLienTable()
    Dim UneVariableTemporaire As TableDef
    ...
    UneVariableTemporaire.RefreshLink
End Sub
```

Et dans le cas où la méthode est une fonction, on peut stocker le résultat dans une variable comme dans l'exemple suivant :

```
Sub Nombre_Champs()
    Dim Nb_Champs As Integer
    Dim MaTable As TableDef
    ...
Nb = MaTable.Fields.Count
End Sub
```

De la même façon que dans les appels de fonctions ou de procédures, les paramètres sont séparés par des virgules dans les appels de méthodes d'objet.

1.3 Événements

Pour chaque objet, il est possible de détecter et de prendre en considération le résultat d'actions particulières externes, que l'on appelle événements. Tous les événements qui ont lieu au cours de l'exécution d'un programme sont détectés et gérés par la machine, mais selon le choix du développeur, seuls certains peuvent faire l'objet d'un traitement particulier, comme par exemple un clic sur un bouton, une case que l'on coche, une ouverture ou fermeture de fenêtre, etc. Ces événements sont gérés au travers de procédures ayant tantôt des paramètres, tantôt non.

En VBA, la syntaxe la plus fréquente des événements est la suivante :

```
Sub MonObjet_MonEvenementDetecte()
    'Le code qui s'exécutera lorsque l'événement sera détecté
End Sub
```

Par exemple, lorsqu'il s'agit d'un clic sur un bouton `MonBouton`, on aura le code suivant :

```
Sub MonBouton_Click()
    'code qui s'exécutera
End Sub
```

Les événements sous Access seront traités plus en détail dans le chapitre Les événements Access.

1.4 Les collections

Lorsque plusieurs objets d'une même classe sont regroupés, ils peuvent appartenir à une même collection d'objets. Pour se référer à un élément en particulier d'une collection d'objets, plusieurs syntaxes sont possibles, parmi les suivantes :

```
Nom_Collection!Nom_Objet
Nom_Collection![nomObjet]
Nom_Collection("NomObjet")
Nom_Collection(variable_Nom) 'variable_Nom est une chaîne de
caractères contenant le nom de l'objet
Nom_Collection(variable_Numero) 'variable_Numero est une valeur
numérique contenant le numéro de l'objet au sein de la collection.
```

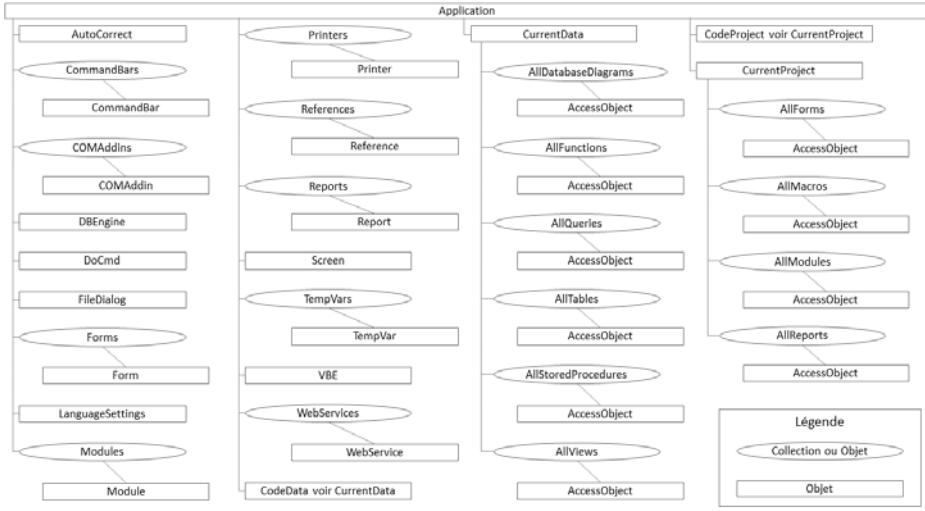
Les deux syntaxes les plus couramment utilisées étant les troisième et cinquième, car elles permettent l'usage de l'IntelliSense (voir chapitre VBE et la sécurité Access 2016).

Remarque

Il est d'usage dans les collections de commencer le décompte à 0, et non à 1. De plus, le numéro d'un objet au sein d'une collection dépendra de la présence d'autres éléments avant ou après lui, ce qui n'assure pas la certitude de tomber sur le bon objet par ce biais.

2. Modèle objet Access

L'objectif des quelques sous-chapitres qui vont suivre est de montrer le modèle de hiérarchie utilisé au sein de l'application Access, sous forme de collections d'objets, qui vont être expliqués par la suite.



3. Collections Access

Voici les principales collections que l'on peut manipuler en VBA sous Access.

Collection	Contient une collection de	Description
`COMAddins`	`COMAddin`	Collection des **compléments COM**
`CommandBars`	`CommandBar`	Collection des **barres de commande**
`Forms`	`Form`	Collection des **formulaires ouverts**. Voir également `CurrentProject.AllForms`.

Collection	Contient une collection de	Description
`Modules`	`Module`	Collection des **modules**
`Printers`	`Printer`	Collection des **imprimantes disponibles**
`References`	`Reference`	Collection des **références bibliothèques**. Voir **Outils - Références**
`Reports`	`Report`	Collection des **états**. Voir également `CurrentProject.AllReports`
`TempVars`	`TempVar`	Collection des **variables temporaires**
`WebServices`	`WebService`	Collection des connexions à des **services web**

4. Objets Access

Voici les principaux objets qu'il est possible de manipuler dans le modèle Access.

Objet	Description
`Application`	Représente l'application Microsoft Access active.
`AutoCorrect`	Représente les options de correction automatique d'Access.
`DBEngine`	Représente le moteur de base de données Microsoft Jet. Cet objet permet de contrôler tous les autres objets d'accès aux données.
`DoCmd`	Permet de convertir en VBA des actions Macros.
`FileDialog`	Permet d'accéder aux fonctionnalités des boîtes de dialogue (Ouvrir ou Enregistrer par exemple).