

Programmation événementielle

Dans la programmation conventionnelle, la séquence des opérations d'un programme est déterminée par le programme principal (main procedure). Cependant, dans la programmation événementielle (event-driven), la séquence des opérations à effectuer est déterminée par les actions que posera l'utilisateur lorsqu'il utilise l'interface de la base de données. Par exemple, lorsqu'un utilisateur ouvre un formulaire, fait la saisie des données, clique sur des boutons de commandes il déclenche des événements. La programmation événementielle répond en s'exécutant lors du déclenchement de ces événements. Ainsi, des petites portions de code, ou des macros, s'exécutent lorsque les événements se déclenchent. Le programmeur doit donc créer de petites parties de code ou des macros plus petites qui s'exécuteront lors de ces événements.

L'exemple le plus simple est celui d'un bouton de commande dans un formulaire. Si ce bouton doit imprimer un rapport lorsque l'utilisateur clique sur celui-ci, l'événement « On Click » a lieu. Une macro (ou du code VBA mais puisqu'il est question de macros dans ce chapitre, on parlera plutôt de macros), attachée à cet événement, peut alors s'exécuter. Cette dernière ouvre le rapport et l'imprime. Les macros ainsi exécutées interagissent alors avec les objets de la base de données.

Introduction à la hiérarchie de l'application

Lorsqu'on automatise les opérations d'une base de données, on crée des instructions qui seront exécutés lorsqu'un objet reconnaît un événement. Il est donc important de savoir pour quels objets on peut écrire ces instructions, les événements que ces objets reconnaissent et aussi savoir comment écrire ces instructions.

Comme on a travaillé depuis le début de façon interactive dans *Access*, on est déjà familier avec les objets qui apparaissent dans le **Volet de navigation**. Jusqu'ici, on ne s'est jamais préoccupé de la façon dont ces objets étaient reliés entre eux. Cependant, lorsqu'on veut écrire des macros, il est important de comprendre la relation qui existe entre ces objets afin de pouvoir faire référence à un objet en particulier.

Les objets de la base de données sont, comme on le sait, les tables contenant les données, les formulaires et les états ainsi que les requêtes. Ces objets peuvent, à leur tour, contenir d'autres objets. On a qu'à penser à un formulaire qui contient des contrôles tels les zones de texte, les étiquettes, les zones de liste déroulante, les boutons de commande etc.

Tous ces objets ainsi regroupés sont des éléments qui font partie de la hiérarchie de l'application *Access*. Cette hiérarchie est très élaborée et contient beaucoup plus d'objets que ceux qu'on peut voir dans le **Volet de navigation**. Cependant, afin de nous faire un peu la main, on ne verra, dans le présent ouvrage, qu'une partie de cette hiérarchie. Ce n'est qu'après la compréhension de cette hiérarchie qu'on pourra, en toute connaissance de cause, faire référence aux différents objets lors de l'écriture des macros.

Collection et objet

Dans *Access*, la majorité des objets font partie d'une collection. En effet, dans une base de données, il existe une collection de formulaires **Forms** (hé oui, en anglais) qui représente tous les formulaires ouverts dans la base de données. Le mot OUVERT est important ici car un formulaire qui n'est pas ouvert ne fait pas partie de la collection **Forms**, ce n'est qu'un formulaire fermé et ce n'est pas encore un objet. Quand on fait référence à un formulaire faisant partie de la collection, on parle d'un objet **Form**.

Les noms des collections dans Access débutent par une majuscule et sont tous au pluriel, c'est-à-dire qu'ils ont un « s » à la fin du nom de la collection.

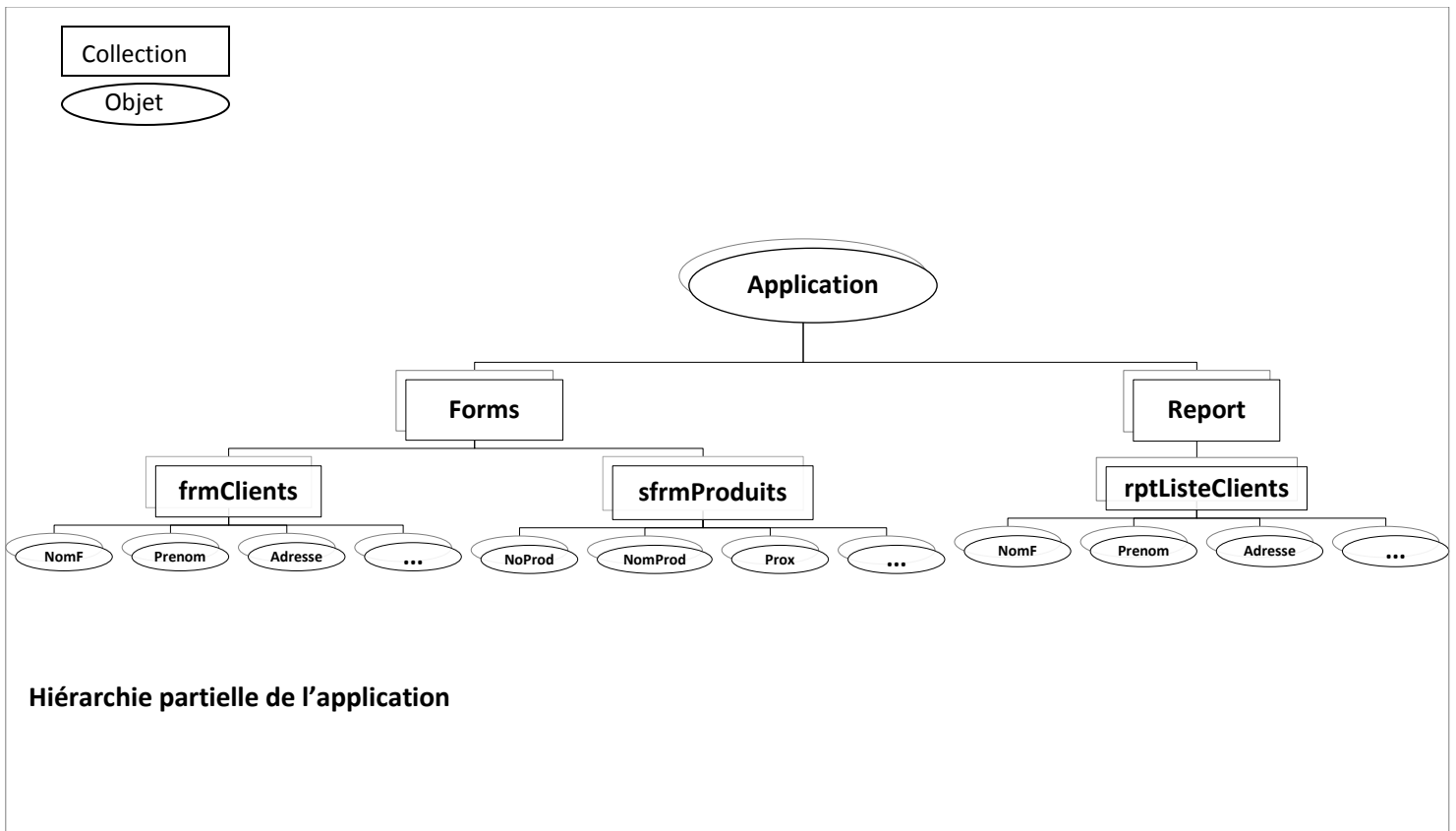
Également, chaque objet formulaire ouvert possède sa collection **Controls** qui contient, à son tour, tous les contrôles ayant été ajoutés dans ce formulaire.

Aussi, un objet qui ne fait pas partie d'une collection est un objet simple **Object**. On pense alors à l'objet nommé **Application** qui représente tout simplement l'application *Access*. L'objet **Application** se retrouve donc au sommet de la hiérarchie. De plus, l'objet **Forms** est un objet simple car il ne peut y avoir qu'une seule collection **Forms**.

Le même principe s'applique lorsqu'on pense aux états. Ainsi, il existe une collection **Reports** qui contient tous les objets **Report** ouvert dans la base de données. Aussi, chaque objet **Report** contient une collection **Controls** représentant tous les contrôles ayant été ajoutés dans chacun de ces états.

Hiérarchie partielle

Pour une application où deux formulaires sont ouverts « frmClients » et « frmProduits » et un rapport « rptListeClients », on peut représenter une partie de la hiérarchie de l'application comme suit :



Faire référence à un objet

Maintenant qu'on connaît un peu la hiérarchie de l'application Access, on peut voir comment on fait références, dans les macros ou dans le code VBA, aux objets qui font partie de la hiérarchie.

Il faut toutefois savoir qu'on met le point d'exclamation « ! » entre une collection et un de ses objets. Par contre, entre un objet et une collection, on doit mettre un point « . ». Il en va de même entre un objet et une de ses propriétés.

Référence explicite

Suite à ce qui a été mentionné au paragraphe précédent et en se référant aux objets de la hiérarchie illustrée précédemment, on veut faire référence au champ (contrôle) « NomF » du formulaire « frmClients ».

Lorsqu'on fait une référence explicite, on passe par toute la hiérarchie pour faire référence au champ « NomF » qui se trouve au bas de la hiérarchie :

Ainsi, on écrit :

Application.Forms!frmClients.Controls!NomF

De façon explicite, lorsqu'on veut faire référence à une des propriétés du champ « NomF », on écrit :

Application.Forms!frmClients.Controls!NomF.Propriété

De plus, si le champ « NomF » est à l'intérieur d'un sous-formulaire nommé « sfrmDetails » par exemple on y fait référence de cette manière :

Application.Forms!frmClients!sfrmDetails.form.Controls!NomF

C'est, on le constate, la méthode la plus longue.

Référence implicite

Il existe une méthode plus courte pour faire références à un objet qui se trouve au bas de la hiérarchie. On dit alors faire une référence implicite.

Hé oui, par défaut, *Access* prend pour acquis qu'on est dans l'application *Access*. Il n'est donc pas nécessaire d'inscrire **Application**. On débute alors par la collection **Forms** et on donne le nom du formulaire auquel on veut faire référence.

La collection par défaut d'un objet **Form** est la collection **Controls**. Il n'est donc pas nécessaire de l'inscrire. Ainsi, on inscrit ce qui suit pour faire la même référence :

Forms!frmClients!NomF

Dans le cas où le champ se trouve dans un sous-formulaire, on y fait référence comme suit :

Forms!frmClients!sfrmDetails!NomF

Si on veut faire référence à une des propriétés du champ « NomF », on ajoute un point et le nom de la propriété.

Forms!frmClients!NomF.Propriété

Sachant ceci, il sera plus facile d'écrire une macro qui synchronise par exemple un formulaire où un rapport avec un autre déjà affiché à l'écran.

Les Événements

Comme expliqué précédemment, les événements sont déclenchés par l'utilisateur par des actions qu'il fait par exemple dans un formulaire. Voici une liste de quelques événements qu'on peut utiliser pour écrire du code VBA ou une macro.

Les événements les plus courants se produisent lorsque :

Open (Sur ouverture) : le formulaire s'ouvre mais avant que le premier enregistrement ne soit affiché.

l'état s'ouvre mais avant son impression.