# 1. APPRENDRE A PROGRAMMER EN VBA

L'objectif est de réaliser nos premiers pas dans le **développement d'applications de base de données** avec **VBA Access**. Il s'agit donc d'une découverte en douceur. **VBA** signifie **Visual Basic For Application**, l'application ici étant **Access**.

**Access** est un **SGBDR**, système de gestion de base de données relationnelles. Il s'agit d'un outil qui permet de créer de puissantes applications d'entreprise. Il est conseillé de bien connaître **Access** avant de se lancer dans le développement d'**applications en VBA**.

Dans cette progression pédagogique **VBA Access**, nous ne reprendrons pas en détail l'apprentissage sur la déclaration de variables, les instructions de branchement conditionnelles ou les boucles de traitements récurrents.

## 1.1. EDITEUR DE CODE VBA ACCESS

Pour réaliser ses premières lignes de code en **VBA Access**, il faut commencer par créer un **module**. C'est le même principe qu'en **VBA Excel**.

- Démarrer **Access** et cliquer sur **base de données vide** sur l'écran de démarrage,
- Dans la boîte de dialogue qui suit, nommer la **base de données**,
- Puis, définir un emplacement à l'aide du bouton Parcourir,
- Enfin, cliquer sur le bouton **Créer**,

Vous basculez en mode feuille de données d'une nouvelle table suggérée. En effet, l'ossature d'une **base de données** se crée sur les tables reliées entre elles. Ce n'est pas notre objectif ici.

- Cliquer sur la croix, en haut à droite de l'onglet de cette table, pour la fermer,

Comme vous l'avez constaté, contrairement aux autres logiciels de la gamme Office, **Access** impose de commencer par enregistrer la **base de données** avant de débuter le travail. En effet, toute saisie d'un nouvel enregistrement dans une table ou par le biais d'un formulaire se stocke en temps réel. Les données sont connectées en permanence avec le disque dur. Pas de travail en mémoire dans ce cas précis.

Notre base ne dispose d'aucune table, donc d'aucune donnée. Pour notre découverte de **VBA Access**, nous ne souhaitons pas interagir tout de suite avec les éléments de **base de données**. Il s'agit dans un premier temps de prendre nos marques et de comprendre comment créer une procédure de code exécutable.

## 1.2. MODULE VISUAL BASIC ACCESS

Contrairement à Excel ou Word, **Access** ne propose pas de ruban Développeur. **Access** est une application pour réaliser du **développement**, si bien que les outils de **code** sont suggérés naturellement.

- Cliquer sur l'**onglet Créer** en haut de la fenêtre **Access**, pour activer son ruban,
- Dans la section Macros et code, tout à fait à droite du ruban, cliquer sur le bouton **Visual Basic**,

Nous basculons ainsi dans l'**éditeur de code Visual Basic pour Access**, très proche de l'éditeur **VBA** pour Excel. D'ailleurs le même raccourci clavier est utilisé pour basculer de la fenêtre de l'application à celle de l'éditeur. Il s'agit de **Alt + F11**. Tout **code VBA** doit être écrit dans un **module**. Nous devons donc en ajouter un.

- Cliquer sur le **menu Insertion** en haut de l'éditeur et choisir **Module** dans la liste,

Une nouveau **Module** nommé Module1 par défaut, est ajouté. Il apparaît dans l'**explorateur de projet** situé, en haut à gauche de l'**éditeur de code**. Ce nom peut être modifié grâce à la fenêtre **Propriétés** qui s'affiche en bas à gauche de l'**éditeur de code**. D'une manière générale, et comme nous l'avons vu avec VBA Excel, cette fenêtre propriétés permet de régler les attributs des **objets VBA**. Pour un **module**, seule la **propriété (Name)** est concernée. **VBA** est en effet un **langage de d programmation orienté objets**, POO. Nous utiliserons des **objets de programmation**. Nous les personnaliserons avec leurs **propriétés**. Nous réaliserons des actions avec leurs **méthodes**. Si la fenêtre Propriétés n'est pas visible, cliquez sur le menu Affichage et choisissez Fenêtre Propriétés, dans la liste.

- Dans la **propriété (Name)** de la fenêtre Propriétés, remplacer le texte Module1 par **codeVBA** et valider par Entrée,

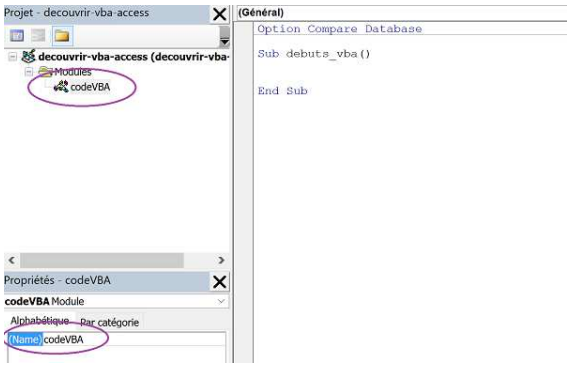
Le nouveau nom apparaît dans l'**explorateur de projet**. Au centre de l'écran figure la feuille de code, pour le module sélectionné dans l'explorateur.

- Sous la ligne Option Compare Database, saisir les bornes de la procédure suivante :

```
Sub debuts_vba()

End Sub
```

Nous créons ainsi une procédure nommée **debuts_vba**. Le **mot clé Sub** est utilisé pour déclarer une procédure de code. Toute procédure doit être bouclée par l'instruction **End Sub**. L'éditeur le fait d'ailleurs pour nous, lorsque nous enfonçons la touche Entrée, après la déclaration. Maintenant que la procédure est bornée, elle est prête à recevoir du **code VBA**, à saisir entre ses bornes.




## 1.3. BOITE DE DIALOGUE - INTERACTIONS AVEC L'UTILISATEUR

**VBA** met à disposition des **fonctions** permettant de déclencher des boîtes de dialogues. Elles permettent d'interagir avec l'utilisateur et peuvent aussi enregistrer ses choix et réponses. Auquel cas, ces valeurs doivent être stockées dans des **variables**, c'est-à-dire en mémoire, pour traitement ultérieur par le **code**.

**VBA** propose deux fonctions pour déclencher des boîtes de dialogues. Les **fonctions** sont des méthodes particulières puisqu'elles n'ont pas besoin d'objet de programmation pour s'exécuter. Il y a la **fonction InputBox** qui propose une boîte de dialogue permettant à l'utilisateur de saisir une réponse, dans une zone de texte. Et il y a la **fonction MsgBox** qui permet d'afficher un message à l'écran. De même, le bouton cliqué par l'utilisateur sur la boîte de dialogue, peut être mémorisé dans une variable, afin d'envisager la suite du traitement.

Pour que le code puisse traiter les actions ou réponses de l'utilisateur, il doit les mémoriser dans des **variables**. On dit qu'une **fonction** retourne une valeur. Un **InputBox** renvoie un texte (variable **String**), tandis qu'un clic sur un **MsgBox** renvoie une valeur numérique (un Entier, variable **Integer**), selon l'action. Nous devons donc déclarer ces deux variables afin d'enregistrer ces valeurs ultérieurement.

- Cliquer entre les bornes de la procédure (Après la ligne **Sub**) pour y placer le point d'insertion,
- Saisir les deux déclarations suivantes :

```
Dim reponse As String
Dim choix As Integer
```

C'est le **mot clé Dim** suivi du nom de la **variable** qui permet d'initialiser la **déclaration d'une variable**. Toute **variable** déclarée doit être **typée**. En fonction de la nature de la donnée qu'elle doit mémoriser, l'allocation mémoire n'est pas la même. **Typer les variables** avec précision est donc important afin qu'un code ne soit pas anormalement gourmand en ressources. C'est le **mot clé As** qui permet d'annoncer le typage. **String** permet de déclarer la variable comme un texte, une chaîne de caractères. **Integer** permet de déclarer la variable comme une valeur numérique entière, sans décimale donc. Ainsi nous déclarons la **variable reponse** comme une chaîne de caractères. Elle permettra de stocker la réponse saisie par l'utilisateur, depuis le **InputBox**. Nous déclarons la **variable choix** comme un entier. Elle permettra de stocker la valeur correspondant au bouton cliqué par l'utilisateur, depuis le **MsgBox**.

Il s'agit maintenant de les exploiter en les affectant aux résultats retournés par les boîtes de dialogues. La **fonction InputBox** requiert un argument obligatoire, le message adressé à l'utilisateur ou plutôt la question : **InputBox(Message, [Arguments_facultatifs])**. C'est identique pour la **fonction MsgBox**. Mais si nous souhaitons que cette dernière retourne la valeur du bouton cliqué par l'utilisateur, nous devons définir ces boutons : **MsgBox(Message, boutons_a_afficher, [Arguments_facultatifs])**.

- A la suite du code, après les déclarations, ajouter les affectations suivantes :

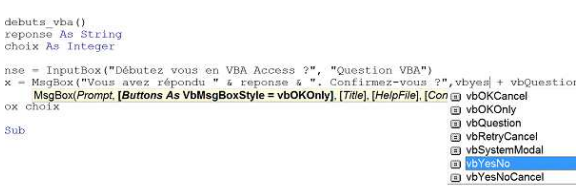
```
reponse = InputBox('Débutez vous en VBA Access ?', 'Question VBA')
choix = MsgBox('Vous avez répondu ' & reponse & '. Confirmez-vous ?',
vbYesNo + vbQuestion, 'Confirmation')
```

Nous enregistrons dans la **variable reponse**, le texte saisi par l'utilisateur dans la boîte de dialogue (**reponse = InputBox**). Dans cette boîte de dialogue, l'utilisateur voit le message saisi en premier argument : Débutez vous en VBA Access ? . Ce message est écrit entre guillemets car il s'agit d'un texte. Le deuxième argument (Question VBA), définit le titre de la boîte de dialogue. Les arguments d'une fonction ou d'une méthode sont séparés par des virgules.

Nous enregistrons ensuite la valeur retournée en fonction du bouton cliqué dans le MsgBox (**choix = MsgBox**). L'utilisateur visualise le message qui est passé en premier argument de la fonction. Comme il s'agit d'un assemblage de texte statique et de texte mémorisé dans une variable, la phrase est reconstruite par **concaténation**. C'est le **symbole &** (Et Commercial), comme dans les calculs Excel, qui permet de concaténer des bouts de chaines pour faire une phrase.

En deuxième argument de la **fonction MsgBox**, nous définissons les boutons. Intelisense se déclenche d'ailleurs durant la saisie avec des suggestions, comme l'illustre la capture ci-dessous.

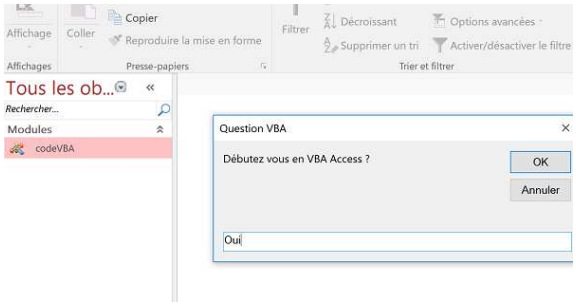
Souvenez-vous, comme en VBA Excel, si vous souhaitez de l'aide sur un élément **VBA** comme **MsgBox**, sélectionnez-le puis enfoncez la **touche F1** du clavier. Vous serez redirigé vers l'aide en ligne pour savoir comment définir les arguments d'une fonction notamment. Ici, nous affichons donc les boutons Oui et Non (**vbYesNo**) accompagnés de l'icône question standard de Windows (**vbQuestion**).




Pour tester ce code, nous allons l'exécuter. Pour ce faire :

- Cliquer sur le bouton **Exécuter** (Flèche verte) de la barre d'outils ou enfoncer la touche **F5**,

La boîte de dialogue **InputBox** apparaît, vous invitant à saisir une réponse à la question posée. Vous cliquez sur Ok après avoir saisir un texte et une autre boîte de dialogue surgit. Il s'agit du **MsgBox** qui ne propose pas de zone de saisie cette fois. En revanche, elle affiche bien la réponse que vous avez saisie, grâce à la **variable reponse** qui a mémorisé cette valeur de retour. Vous cliquez sur Oui ou Non et plus rien ne se produit. Cela signifie que l'exécution du code se termine. Or cette réponse a été stockée dans la **variable choix**. Mais nous ne l'avons pas encore exploitée. Pour connaître cette valeur retournée, afin de la traiter, le plus simple est de l'afficher à l'écran.




- A la suite du code, avant le **End Sub**, ajouter la ligne suivante :

```
MsgBox choix
```

- Exécuter la procédure en enfonçant la touche **F5** du clavier,

Après avoir répondu à l'**InputBox**, lorsque vous cliquez sur le bouton Oui du **MsgBox**, un nouveau **MsgBox** apparaît affichant la valeur 6. Dans le cas contraire, c'est la valeur 7 qui est mémorisée. En d'autres termes, si la valeur 6 a été stockée dans la **variable choix**, nous savons que l'utilisateur a cliqué sur Oui et lorsque c'est la valeur 7, nous savons que l'utilisateur a cliqué sur Non. Il s'agit donc de vérifier une **condition** pour envisager la poursuite du code selon le cas. En **VBA**, il s'agit de l'**instruction If** que nous avons apprise au travers de la formation VBA Excel pour gérer les critères.

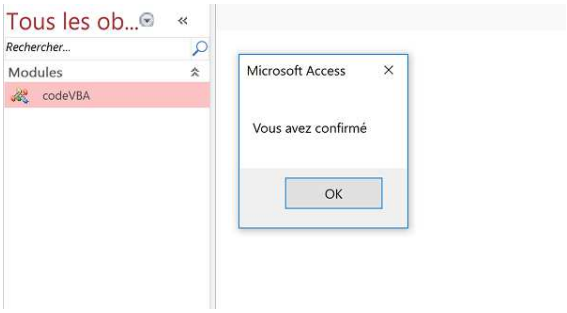
- Supprimer la ligne **MsgBox choix**,
- Puis, ajouter les instructions suivantes :

```
If (choix = 6) Then
MsgBox 'Vous avez confirmé'
Else
MsgBox 'Vous avez infirmé'
End If
```

Après le **mot clé If** suit le critère entre parenthèses. Ces dernières ne sont pas obligatoires. Ce **critère** consiste à comparer la variable choix à une valeur numérique (choix = 6). Le **mot clé Then** est utilisé pour enclencher les actions lorsque ce critère est vérifié, il signifie Alors. Entre le **Then** et le **Else**, nous pouvons lister toutes les actions à réaliser lorsque la condition est vraie. Ici, nous choisissons seulement d'afficher un message à l'écran. Le **mot clé Else** est utilisé pour traduire le Sinon et annoncer l'exécution des instructions dans le cas contraire, lorsque la **condition** n'est pas validée. Là encore, nous choisissons seulement d'afficher un message à l'écran. Enfin, toute **instruction If** doit être fermée par les mots clés **End If**. La **branche Else** de l'**instruction If** n'est pas obligatoire. Nous pourrions très bien envisager des conditions pour lesquelles nous engagerions des actions dans le cas où le critère est vérifié, et de ne rien faire dans le cas contraire.

- Enfoncer la touche **F5** du clavier pour exécuter le code,

Après avoir répondu à l'**InputBox**, en fonction du bouton que vous choisissez sur le premier **MsgBox**, un second apparaît démontrant que votre choix a été intercepté et interprété grâce à l'**instruction If**.



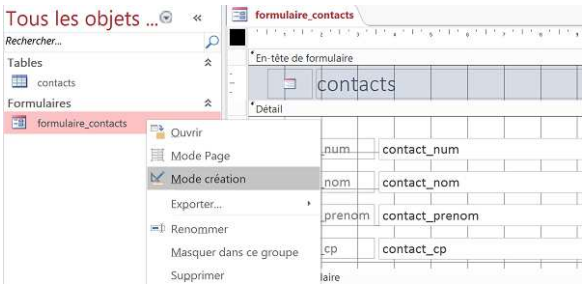
## 1.4. LIER LE CODE AUX OBJETS ACCESS – EVENEMENTS

Jusqu'à présent, le code que nous avons développé est complètement dissocié des **objets Access**. Aucune interaction ne se produit avec un formulaire par exemple, ou encore avec les données de la **base de données**. Un code est d'autant plus précis, lorsqu'il se déclenche sur une action précise, par exemple au clic sur un bouton mais pas seulement. On parle de **gestionnaire d'événements**. Avec **Access**, il est très simple d'associer un code à un ordre précis. Ce dernier se déclenche alors sur événement.

☞ Ouvrir le fichier « **Apprendre à programmer en VBA.accdb** ».

Le terme **base de données** n'est pas vraiment approprié pour décrire ce fichier. Comme l'illustre le volet des objets sur la gauche de la fenêtre, cette base est constituée d'une seule table sur laquelle est bâtie un formulaire.

- Cliquer avec le bouton droit de la souris sur l'objet **formulaire_contacts**,
- Dans le menu contextuel, choisir **Mode Création**,




Le **formulaire** s'affiche ainsi en mode conception, au centre de la fenêtre. Ce mode permet de le personnaliser pour modifier notamment l'apparence et la disposition des champs. Pour rappel, ces champs sont ceux définis lors de la **création de la table**, sur laquelle le **formulaire** est construit.

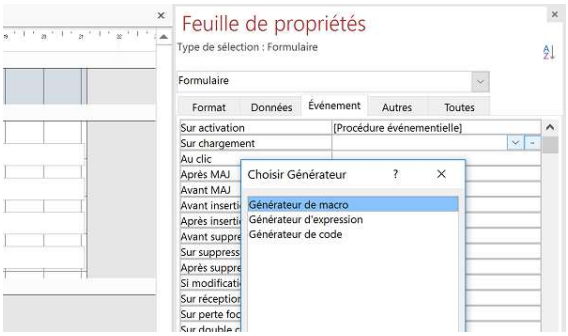
En même temps que le formulaire apparaît, trois rubans dédiés s'affichent en haut de la fenêtre. Il s'agit des rubans **Création**, **Organiser** et **Format** qui offrent les fonctionnalités pour paramétrer ce formulaire.

- Dans la section Outils du ruban Création, cliquer sur le bouton **Feuille de propriétés**,

Un volet apparaît sur la droite de la fenêtre. Il propose toutes les propriétés à personnaliser pour l'objet sélectionné sur le formulaire. Comme nous n'avons sélectionné aucun objet, il s'agit des propriétés du formulaire lui-même. Et c'est ainsi que vous constatez qu'**Access** est naturellement très proche des langages de programmation orientés objets.

- Cliquer sur l'**onglet Evénement** de la feuille de Propriétés,

De nombreux **événements** peuvent être interceptés et gérés pour chaque objet **Access**. Il suffit de leur associer une ou des actions pour réaliser des tâches précises au moment même où l'événement se génère. Nous pourrions par exemple demander le mot de passe administrateur à l'ouverture du formulaire, pour n'autoriser l'accès qu'aux ayant droits. Mais il s'agit pour nous d'une découverte de **VBA Access**, nous nous contenterons dans un premier temps de faire des remarques sur les possibilités d'interactions offertes.




Comme l'illustre la capture ci-dessus, les actions qui peuvent être associées aux **événements** se construisent soit avec une **macro**, soit en générant des expressions avec des opérateurs et fonctions soit avec du **code VBA**.

- Pour l'**événement Sur activation**, choisir **[Procédure événementielle]** à l'aide de la liste déroulante,
- Puis, cliquer sur le petit bouton à l'extrémité droite,

Vous basculez ainsi dans l'**éditeur de code**, entre les bornes d'une **procédure événementielle**.

```
Private Sub Form_Current()

End Sub
```

Tout code saisi entre les bornes de cette procédure se déclenchera à l'activation du formulaire, c'est-à-dire avant même son affichage. Cette procédure n'est cette fois pas écrite dans un module. Elle est directement rattachée à l'objet formulaire, comme en atteste l'explorateur de projet sur la gauche. C'est pourquoi les propriétés listées dans la fenêtre Propriétés, en bas à gauche de la fenêtre, sont beaucoup plus nombreuses que précédemment.

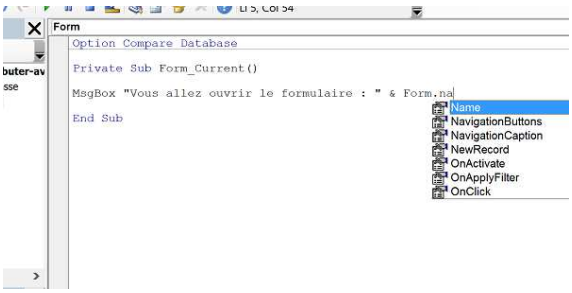
Pour réaliser des actions intéressantes, notamment sur les données et opérer de vraies interactions entre les **objets de base de données**, nous avons d'abord besoin d'avancer dans l'apprentissage du **VBA Access**. C'est pourquoi, nous nous contenterons ici de déclencher un code basique, selon les prérequis que nous avons abordés précédemment.

**Form** est un objet **VBA Access** désignant un formulaire. On peut lui passer en argument, l'index du formulaire à piloter. Mais par défaut, cet objet désigne le formulaire actif. Comme tout objet, il propose des propriétés et méthodes.

- Entre les bornes de la procédure événementielle, ajouter la ligne de **code** suivante :

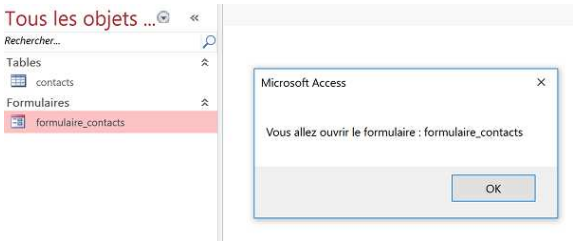
*MsgBox 'Vous allez ouvrir le formulaire : ' & Form.Name*

Nous utilisons de nouveau la fonction **MsgBox** pour afficher un message à l'écran. Comme ce code doit se déclencher sur activation du formulaire, son affichage doit intervenir avant celui du formulaire. Nous associons à un texte entre guillemets (**&**), le nom du formulaire que nous activons en appelant la propriété **Name** de l'objet **Form**. Une propriété ou une méthode s'appelle en tapant un point juste après le nom de l'objet. D'ailleurs vous notez l'apparition instantanée d'une liste de propositions des propriétés et méthodes de l'objet dans l'éditeur de code. C'est ce qu'illustre la capture ci-dessous. Une **propriété** est accompagnée d'une icône grisée tandis qu'une **méthode** s'affiche préfixée d'une icône verte.




- Réaliser le raccourci clavier **ALT + F11** pour revenir sur l'application **Access**,
- Fermer le formulaire en cliquant sur la croix de son onglet en enregistrant les modifications,
- Double cliquer sur **formulaire_contacts** dans le volet des **objets Access** sur la gauche,

Vous commandez ainsi son ouverture en mode Formulaire ou en exécution si vous préférez. Avant même que le formulaire ne s'affiche, vous remarquez l'irruption de la boîte de dialogue **MsgBox** que nous avons programmée. Le **gestionnaire d'événements** a intercepté l'activation du formulaire et a ordonné le déclenchement du **code** saisi entre les bornes de la **procédure événementielle**. Cette boîte de message affiche bien le nom du formulaire grâce à la propriété **Name** de l'objet **Form**. Ces découvertes nous seront fort utiles pour des développements ultérieurs, au fur et à mesure de la progression pédagogique.




- Cliquer sur le bouton Ok de la boîte de dialogue pour poursuivre le déroulement du code,

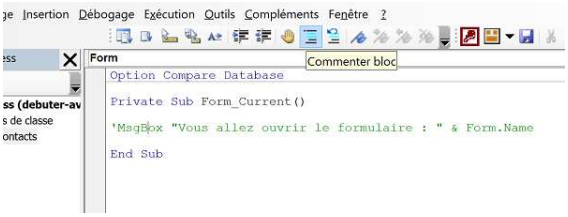
Comme nous n'avons plus de ligne de code, le programme atteint la fin de la procédure (**End Sub**) et autorise ensuite l'affichage du formulaire. Les enregistrements des contacts de la table s'affichent dans une vue graphique. Vous pouvez utiliser les petites flèches de navigation en bas de la fenêtre pour les faire défiler.

- Cliquer sur la flèche du bouton **Affichage** à gauche du ruban Accueil,
- Dans la liste, choisir **Mode Création**,

Nous revenons dans la vue précédente, la conception du formulaire.

- Réaliser le raccourci clavier **ALT + F11** pour basculer dans l'éditeur de code,
- Faire un clic droit n'importe où sur la barre d'outils de l'éditeur,
- Dans la liste, choisir **Edition** pour afficher la barre d'outils Edition,
- Dans le code, cliquer n'importe où dans la ligne du **MsgBox** pour y placer le point d'insertion,
- Puis, cliquer sur le bouton **Commenter bloc** de la barre d'outils Edition,

La ligne de code concernée apparaît en vert, préfixée d'une apostrophe. Elle est passée en **commentaire**. Elle est désormais ignorée et donc neutralisée. Nous aurions tout simplement pu saisir une apostrophe devant la ligne, pour la commenter manuellement. Si vous commandez de nouveau l'affichage du formulaire en mode exécution, vous constaterez que le **MsgBox** n'intervient plus.




- Dans le formulaire en mode création, sélectionner la zone de saisie **contact_nom**,
- Dans l'onglet Evénement de sa feuille de propriétés, cliquer sur le petit bouton à droite de l'**événement Au Clic**,
- Dans la boîte de dialogue, choisir **Générateur de code**,

Vous basculez dans l'**éditeur de code**, entre les bornes de la **procédure contact_nom_Click()**. **contact_nom** est le nom de l'objet zone de saisie sur le formulaire. Ce nom a été emprunté au nom du champ qu'il désigne et tel qu'il a été créé dans la table. Le code saisi entre ces bornes se déclenchera lorsque l'événement du clic, précisément sur cette zone de saisie sera intercepté.

- Entre les bornes de cette procédure, saisir les lignes de code suivantes :

```
With contact_nom
.FontBold = True
.FontSize = 14
.ForeColor = RGB(255, 0, 0)
End With
```

Tout d'abord, nous exploitons le **bloc With** afin de regrouper et lister les **propriétés** de l'**objet contact_nom**, de façon structurée. La présentation est plus propre et nous évitons de répéter inutilement le nom de l'objet en question. Un **bloc With** se boucle nécessairement par l'instruction **End With**. Les propriétés **FontBold** et **FontSize** de cet objet zone de texte permettent respectivement, de définir la police en gras (**True**) et la taille à 14, par affectation (**=**). Enfin la propriété **ForeColor** désigne la couleur de premier plan de l'objet, soit la couleur du texte. Nous la réglons sur du rouge, grâce à la fonction **RGB()** que nous avons aussi déjà utilisée en **VBA Excel**. Cette fonction demande les trois composantes primaires des couleurs à mélanger, rouge, vert et bleu. Comme nous réglons le rouge sur la valeur maximale, nous définissons la couleur de police sur du rouge.

- Réaliser le raccourci clavier **ALT + F11** pour revenir sur le formulaire en création,
- Cliquer sur le bouton de l'**événement Sur perte focus** de la feuille de propriétés de l'**objet contact_nom**,
- Dans la boîte de dialogue, choisir de nouveau **Générateur de code**,
- Entre les bornes de la procédure événementielle ainsi créée, ajouter le code suivant :

```
With contact_nom
.FontBold = False
.FontSize = 12
.ForeColor = RGB(0, 0, 0)
End With
```

Il s'agit d'un code identique au précédent, sauf que nous attribuons des valeurs de propriétés conformes à celles d'origine. Ce code est susceptible de se déclencher sur l'**événement Sur perte focus**, c'est-à-dire lorsqu'un autre objet est sélectionné par exemple. Plus généralement, cet événement se déclenche lorsque l'objet est désactivé.

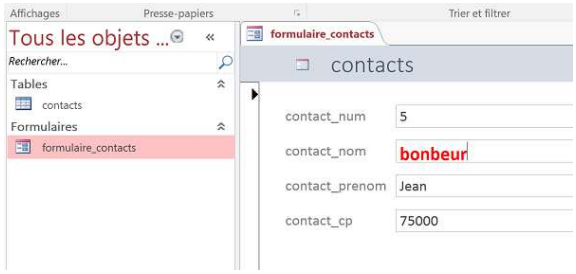
- Réaliser le raccourci **ALT + F11** pour revenir sur le formulaire,
- Enfoncer la touche **F5** du clavier pour exécuter le formulaire,
- Cliquer dans la zone de saisie **Contact_nom**,

Vous remarquez que le texte qu'elle contient subit les modifications imposées par le code, déclenché sur l'événement du clic. Le nom du contact apparaît plus gros, en gras et en rouge.

- Cliquer dans la zone de saisie **Contact_prenom**,

Comme **Contact_nom** n'est plus sélectionné, l'**événement Sur perte focus** est intercepté. Le code qu'il contient est donc déclenché et les réglages de police sont rétablis, comme nous l'avons ordonné avec les propriétés de l'objet.

Avec le peu de connaissance dont nous disposons pour l'instant, nous entrevoyons déjà la puissance du **VBA** mêlée à **Access** et les interactions qu'il est possible de réaliser avec les **objets de base de données**.




- Cliquer sur la flèche du **bouton Affichage**, à gauche dans le ruban Accueil,
- Dans la liste, choisir **Mode Création**,
- Glisser la zone du pied de formulaire vers le bas de façon à créer de l'espace en dessous de la dernière zone de saisie **contact_cp**,
- Dans les visuels de la section Contrôles du ruban Création, choisir le **bouton**,
- Le tracer sur le formulaire, dans l'espace vide sous l'objet contact_cp,
- Cliquer sur le bouton Annuler de la boîte de dialogue qui se déclenche,
- Activer l'**onglet Autres** de sa feuille de propriétés,
- Remplacer la valeur de sa **propriété Nom** par le texte **Suivant**,
- Activer l'**onglet Format** de sa feuille de propriétés,
- Régler sa **propriété Légende** sur **Suivant**,

La **propriété Nom** permet de définir le nom de l'objet, soit du bouton, pour le piloter par le code à l'aide de ses **propriétés** et **méthodes**. La propriété Légende permet de modifier le texte qu'il affiche pour le rendre plus explicite à l'utilisation.

- Activer désormais l'**onglet Evénement** de sa feuille de propriétés,
- Cliquer sur le petit bouton à droite de l'événement **Au clic**,
- Dans la boîte de dialogue, choisir **Générateur de code** et valider par Ok,

Nous basculons de nouveau dans l'éditeur de code entre les bornes de la **procédure Suivant_Click()**. Le code saisi entre ces bornes se déclenchera donc au clic sur le bouton nommé **Suivant**. Il s'agit fort logiquement de l'événement naturel associé à un bouton.

- Saisir la ligne de code suivante :

```
Form.Recordset.MoveNext
```

**Form** est toujours l'**objet VBA Access** qui représente le formulaire actif par défaut, au moment de l'exécution du code. Si nous souhaitions désigner un autre formulaire, il faudrait le spécifier, en paramètre de l'**objet Form**. **Recordset** est un objet dérivé qui désigne les **enregistrements** que manipule le formulaire. Il s'agit en fait des enregistrements de la table sur laquelle le formulaire a été bâti. Un enregistrement ici, concerne un contact complet, soit le numéro, le nom, le prénom et le code postal. La méthode **MoveNext** de l'objet **Recorset** permet de placer le focus sur l'enregistrement suivant, soit le contact suivant.

- Réaliser le raccourci **ALT + F11** pour revenir sur le formulaire,
- Enfoncer la touche **F5** du clavier pour exécuter le formulaire,
- Cliquer à plusieurs reprises sur le **bouton Suivant**,

Vous constatez que les enregistrements défilent puisque nous passons en revue les contacts, les uns après les autres. Si vous allez trop loin, une erreur d'exécution se déclenche. Nous ne gérons en effet pas l'exception qui consiste à ne plus avancer, lorsque la fin des enregistrements est atteinte. Mais ce n'est pas l'objectif ici, puisqu'il s'agissait simplement de découvrir les potentialités d'**Access** pour le **développement**, notamment au travers de la **gestion des événements**.

Dans les prochaines formations, nous apprendrons les **objets de programmation** importants, notamment ceux permettant de manipuler les données ou encore ceux permettant de relier les **objets Access**. Au fur et à mesure, en imbriquant ces connaissances, nous serons capables de bâtir de sérieuses applications professionnelles.