Vos premiers pas dans l'éditeur de macros Excel

par SilkyRoad (silkyroad.developpez.com)

Date de publication : 05/05/2007

Dernière mise à jour :

Cette page propose une description de l'éditeur de macros et montre comment créer vos premières procédures VBA dans Excel.

La présentation correspond à une utilisation dans Excel2002.

I - Introduction

Ce document s'adresse aux utilisateurs d'Excel désirant s'initier au rudiment des macros.

L'article présente les fonctionnalités de l'éditeur de macros, aussi appelé Visual Basic Editor (VBE). Vous y découvrirez une description des outils disponibles pour créer vos premières procédures.

Les macros servent essentiellement à automatiser et personnaliser des actions dans le classeur.

Vous pouvez ainsi écrire des procédures pour les tâches répétitives, mais aussi adapter l'outil Excel pour qu'il réponde exactement à vos besoins particuliers (interagir avec les manipulations de l'utilisateur, piloter d'autres applications...).

Le langage **Visual Basic For Applications** (VBA) est orienté objet et intégré à Excel, ce qui en facilite l'apprentissage et son maniement par les utilisateurs.

Il est constitué d'instructions, de mots clés, de fonctions, de méthodes et de propriétés pour élaborer des codes qui permettent la manipulation des objets de l'application.

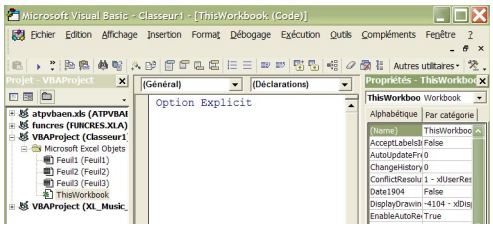
Pour obtenir plus d'informations sur la rédaction des macros, et tout savoir sur les bonnes pratiques dans ce domaine, consultez l'article de **J-M RABILLOUD** pour **programmer efficacement en VBA Excel**.

II - La présentation de l'éditeur

II-A - Comment accéder à l'éditeur

Ouvrez un nouveau classeur Excel puis utilisez le raccourci clavier **Alt+F11**.

Une fenêtre similaire à celle ci apparait à l'écran:



Utilisez le même raccourci clavier pour rebasculer vers la feuille de calcul.

Vous pouvez aussi accéder à l'éditeur en utilisant le menu Outils/Macro/Visual Basic Editor.

Une autre possibilité consiste à cliquer sur le bouton **Visual Basic Editor** dans la barre d'outils Visual Basic.

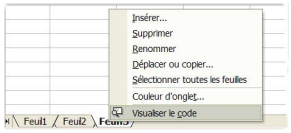
Si cette barre d'outils n'apparait pas dans votre menu Excel:

Utilisez le menu Affichage

Barres d'outils

Sélectionnez l'option "**Visual Basic**".

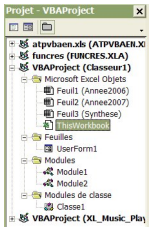
Pour atteindre le module objet d'une feuille de calcul, effectuez un clic droit sur l'onglet et sélectionnez l'option "**Visualiser le code**".



II-B - Description

L'éditeur de macros est constitué de plusieurs fenêtres:

II-B-1 - L'explorateur de projets



L'explorateur de projets permet d'accéder à tous les classeurs et macros complémentaires (.xla) ouverts dans la session Excel.

Si la fenêtre n'apparait pas l'écran, utilisez le menu Affichage/Explorateur de projet (ou le raccourci clavier **Ctrl+R**).

Chaque classeur est constitué:

* D'un **module objet ThisWorkbook**.

* D'un **module objet pour chaque feuille du classeur** (Feuil1, Feuil2, Feuil3 ...).

* De **module objet pour chaque feuille Graphique du classeur**

* De **UserForms** (boîtes de dialogues personnalisées), créées par l'utilisateur.

* De modules standards (Module1, Module2 ...), créés par l'utilisateur.

* De modules de classes (Classe1 ...), créés par l'utilisateur.

Les éléments du dossier "**Microsoft Excel Objects**" existent systématiquement dans tous les classeurs. Vous ne pouvez pas les supprimer depuis l'explorateur de projet.

Les autres éléments sont constitués à partir des ajouts de l'utilisateur.

Double cliquez sur les éléments de l'explorateur pour vous déplacer entres les différents modules et UserForms.

Les éléments créés par l'utilisateur (modules standards, modules de classe et UserForm) peuvent être supprimés depuis l'explorateur de projet, en effectuant un clic droit sur le module.

Sélectionnez l'option "**Supprimer ...**".

Une boîte de dialogue vous demande si vous souhaitez exporter le module avant de le supprimer (C'est à dire de le sauvegarder sur votre disque dur).

Cliquez sur le bouton "Non" pour confirmer.

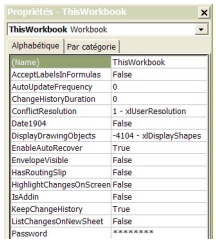
Si vous cliquez sur le bouton **Oui**, une nouvelle fenêtre s'affiche pour préciser l'emplacement de l'enregistrement. Le chemin indiqué par défaut est généralement:

C:\Program Files\Microsoft Office\OfficeXX

Mais vous pouvez bien entendu sauvegardez dans un autre répertoire.

Le fichier sauvegardé porte l'extension **.bas**. Par curiosité, vous pouvez ensuite aller dans l'explorateur Windows pour rechercher ce fichier. Renommez l'extension en **.txt** et ouvrez le comme un simple fichier texte: Il contient les procédures que vous avez saisi dans la fenêtre code (voir le chapitre II-B-3-a).

II-B-2 - La fenêtre Propriétés



Elle permet de consulter et modifier les propriétés des objets (classeur, feuilles, contrôles).

Si la fenêtre n'apparait pas l'écran, utilisez le menu Affichage/Fenêtre Propriétés (ou le raccourci clavier **F4**).

Vous pouvez aussi utiliser le bouton "**Fenêtre propriété**".

II-B-3 - La zone de saisie des macros

II-B-3-a - La fenêtre Code

Chaque type de module et chaque UserForm possède sa propre fenêtre code.

Vous allez rédiger vos procédures dans ces zones de saisie.

Les macros sont constituées de simples textes que vous pouvez entièrement écrire ou copier à partir de codes existants.

Vous pouvez modifier, copier/coller, supprimer une procédure (ou une portion) comme dans un éditeur de texte classique.

II-B-3-b - La couleur du code

Tous les mots clés et les instructions sont automatiquement identifiables par une couleur de police spécifique (généralement bleu par défaut).

Les lignes contenant des erreurs de syntaxe sont identifiables par une couleur de police rouge.

Les lignes de commentaires (précédées par une apostrophe ou par l'instruction **REM**) sont identifiables par une couleur de police verte.

N'hésitez pas à ajouter des annotations dans vos procédures. Cela vous aidera pour la relecture et la compréhension des macros.

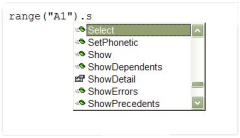
Ces couleurs peuvent être personnalisées depuis le menu Outils/Options/Onglet "**Format de l'éditeur**".

II-B-3-c - La saisie semi automatique

L'éditeur possède un outil de saisie semi automatique.

Cette option est activée en cochant la ligne "**Complément automatique des instructions**" dans le menu Outils/Options/Onglet "**Editeur**".

Dès que vous saisissez un point (.) après un élément, l'environnement VBA propose la liste des méthodes et propriétés associées. La liste sera restreinte en fonction des premières lettres saisies.



La saisie semi automatique limite les risques d'erreurs de syntaxe dans vos procédures.

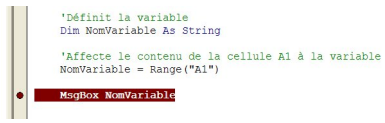
II-B-3-d - Les points d'arrêt

Les Points d'arrêt permettent de stopper le déroulement d'une macro à un emplacement précis.

Lorsque vous lancez la procédure, celle ci s'exécutera jusqu'à la ligne où figure le point d'arrêt.

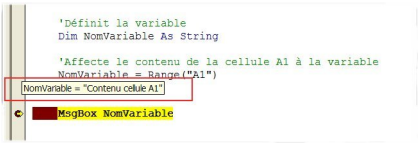
Cliquez dans la marge à gauche de la zone de saisie, pour appliquer un point d'arrêt sur la ligne (ou utilisez la touche raccourci **F9** à l'emplacement du curseur).

Les points d'arrêt ne fonctionnent pas sur les lignes vides.



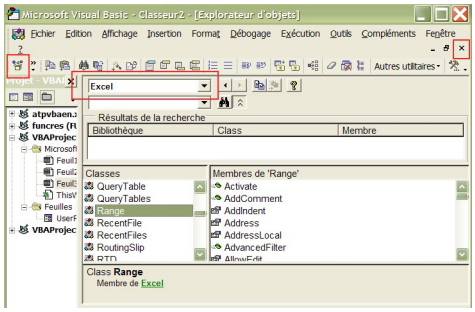
Cette option est pratique pour par exemple vérifier les valeurs attribuées **aux variables**.

En effet, si vos variables sont correctement paramétrées, un passage de la souris sur leur nom fait apparaitre une info-bulle qui indique la valeur prise.



Vous pouvez interrompre totalement l'exécution en appuyant sur le bouton "**Réinitialiser**", lorsque vous êtes en mode "**Arrêt**".

II-B-4 - L'Explorateur d'objets



Cliquez sur l'icône "**L'explorateur d'objets**" (ou le raccourci clavier **F2**) pour afficher cette fenêtre.

Cet outil liste tous les éléments disponibles dans l'éditeur, pour toutes les bibliothèques actives.

Sélectionnez un élément et appuyez sur la touche **F1** pour atteindre l'aide associée.

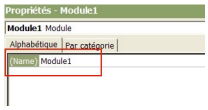
Utilisez la croix sur la droite de l'écran pour refermer cette fenêtre.

II-B-5 - Les modules standards

Les modules standards sont les principaux supports pour écrire vos procédures.

Ajoutez un module standard dans votre projet en utilisant le menu Insertion/Module, depuis l'éditeur de macros.

Par défaut, le premier module créé sera nommé "**Module1**". Vous pouvez personnaliser ce nom en modifiant le champ **(Name)** dans la fenêtre **Propriétés**. Ce nom ne doit pas contenir d'espaces ni de caractères spéciaux.



Vous pouvez créer autant de modules que vous souhaitez dans un classeur, dans la limite de l'espace mémoire disponible.

Toutefois, pour faciliter la lecture des procédures et pour un gain de place, il est préférable de regrouper dans un même module des macros destinées à fonctionner ensemble.

II-B-6 - Les modules de classe

Les classes définissent le fonctionnement des objets, sur le principe de la Programmation Orientée Objet. Un objet peut être défini par une propriété, une méthode ou un évènement.

L'emploi des classes s'adresse à des utilisateurs confirmés ayant de bonnes connaissances en programmation.

Vous pouvez ajouter un module de classe en utilisant le menu Insertion/Module de classe.

II-C - Créez votre première macro

II-C-1 - L'écriture de la procédure

Tout d'abord, ajoutez un module standard dans votre projet.

Une fois le module créé, vous pouvez écrire ou copier la macro ci dessous dans la zone de saisie (fenêtre Code).

Cet exemple affiche un message d'information à l'écran.

Vba

```
Sub Test()
'Environ("username") renvoie le nom de l'utilisateur pour la session Windows ouverte
MsgBox "Bonjour " & Environ("username")
End Sub
```

II-C-2 - Description d'une procédure

les procédures commencent par l'instruction **Sub** et se terminent par l'instruction **End Sub**.

Le code doit être écrit entre **Sub** et **End Sub**.

End sub s'ajoute automatiquement si vous appuyez sur la touche **Entrée** après avoir saisi le nom de la macro ("**Test**" dans l'exemple du chapitre précédent).

Le nom de la macro doit commencer par un caractère alphabétique et pas excéder 255 caractères. Le nom ne doit pas contenir de caractères spéciaux. Le caractère underscore _ est accepté. Essayez d'attribuer des noms les plus explicites possibles afin de faciliter la relecture de votre programme.

Un module ne peut pas contenir deux macros ayant le même nom. Plus généralement, il est déconseillé d'avoir deux macros utilisant le même nom dans un projet.

N'utilisez pas un nom de macro qui est aussi une référence à une cellule.

les parenthèses placées après le nom de la macro Test(), servent à spécifier des arguments (des paramètres) lors de appel à cette procédure. Les parenthèses sont obligatoires même si la procédure s'exécute sans arguments.

Un exemple de procédure utilisant un argument, à placer dans un module standard:

Vba

```vba
Sub TestMacro()

    'Apelle la procédure MultiplicationCellule en indiquant la référence à
    'la cellule A1 en argument.
    MultiplicationCellule Range("A1")

End Sub


Sub MultiplicationCellule(Cible As Range)
    'Vérifie si la cellule contient bien une valeur numérique
    '(Sinon la suite de la macro va provoquer une erreur)
    If Not IsNumeric(Cible) Then Exit Sub

    'multiplie le contenue de la cellule par 2
    Cible = Cible * 2
End Sub
```

Quand vous lancez la macro depuis la feuille de calcul (raccourci **Alt+F8**), vous constatez que la procédure contenant des arguments (**MultiplicationCellule**) n'est pas visible dans la boîte de dialogue. Il n'y a que les procédures sans arguments qui apparaissent dans la liste.

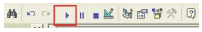
Si vous utilisez l'instruction **Private**:

Private Sub TestMacro()

Dans ce cas la macro **TestMacro** n'est accessible que pour les procédures du module dans lequel elle est déclarée. Sa portée est limitée et elle n'est plus visible dans la boîte de dialogue (**Alt+F8**).

II-C-3 - Lancer la procédure

Pour lancer votre macro depuis l'éditeur de macros, placez le curseur à l'intérieur de la procédure et cliquez sur le bouton "**Exécuter Sub/UserForm**".



Vous pouvez aussi utiliser le menu Exécution/Exécuter Sub/UserForm (ou le raccourci clavier **F5**).

Pour lancer une macro depuis la feuille de calcul:

Utilisez le menu Outils/Macro/Macros (ou le raccourci clavier **Alt+F8**)

Sélectionnez la procédure dans la liste et cliquez sur le bouton **Exécuter**.

Le bouton "**Exécuter une macro**", dans la barre d'outils "**Visual Basic**", permet aussi d'afficher la boîte de dialogue pour sélectionner la procédure à lancer.

Nous n'allons pas entrer dans le détail dans cet article, mais il existe de nombreuses manières pour automatiser le déclenchement d'une macro:

* A partir d'un bouton dans la barre de menus.

* A partir d'un contrôle/objet.

* A partir d'une forme automatique.

* A partir d'un évènement.

* A partir d'une autre macro.

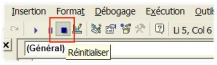
* A partir d'un raccourci clavier (Attention: Il annule tout raccourci équivalent défini par défaut dans Excel, pendant que le classeur qui contient la macro est ouvert.)

II-C-4 - Le mode pas à pas

Vous pouvez exécuter vos macros en mode pas à pas, afin de suivre chaque étape de l'exécution. Placez le curseur dans votre procédure (entre Sub et End Sub) et appuyez sur la touche **F8**. Chaque utilisation de la touche **F8** exécute une ligne du code et attend un nouvel appui pour continuer.

Comme pour le point d'arrêt, cet outil est très pratique pour voir la valeur prise par les variables, à chaque étape de la macro.

Vous pouvez terminer l'exécution à n'importe quel moment en appuyant sur le bouton "Réinitialiser".



II-C-5 - Les fonctions

Contrairement aux procédures **Sub** qui ne renvoient pas de données, la fonction (**Function**) est un type de procédure qui renvoie une valeur.

L'environnement VBA dispose de nombreuses fonctions prédéfinies mais vous pouvez aussi créer vos fonctions personnelles. Ces dernières sont utilisées de la même manière que les fonctions prédéfinies.

les fonctions commencent par l'instruction **Function** et se terminent par l'instruction **End Function**.

Le code doit être écrit entre **Function** et **End Function**.

Les fonctions acceptent des arguments obligatoires et optionnels. Les arguments optionnels sont placés en fin de la déclaration.

Vous pouvez définir le type de résultat retourné de la même manière que pour une variable.

Voici un exemple de fonction à placer dans un module standard:

Vba

```
Function MaFonctionPerso(NbValeurs As Long, Optional Coeff As Variant) As Long

    'Spécifie que la fonction doit être recalculée à chaque fois qu'un calcul
    'est effectué dans une cellule quelconque de la feuille.
    Application.Volatile

    'Vérifie si l'argument optionnel à été indiqué
        'Nota: Coeff doit être impérativement de type Variant.
    If IsMissing(Coeff) Then
        'S'il n'y a pas d'argument optionnel:
        MaFonctionPerso = (NbValeurs + 3)
    Else
        'S'il y a un argument optionnel:
```

```vba
            MaFonctionPerso = (NbValeurs + 3) * Coeff
        End If

    End Function
```

Vous pouvez ensuite appeler cette fonction depuis une procédure **Sub**.

```vba
    Sub AppelFonction()
        MsgBox MaFonctionPerso(10)
        MsgBox MaFonctionPerso(10, 5)
    End Sub
```

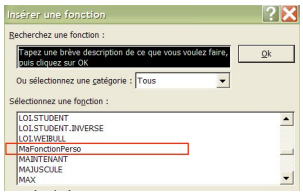
Mais aussi l'utiliser depuis la feuille de calcul:

```formules
    =MaFonctionPerso(10)
    =MaFonctionPerso(10;5)

    'Si le premier argument fait référence à la cellule A1:
    =MaFonctionPerso(A1;5)
```

Les fonctions n'apparaissent pas dans la boîte de dialogue des macros (**Alt+F8**). Par contre elles sont intégrées dans la liste des fonctions Excel tant que le classeur qui les contient est ouvert:

Pour la visualiser dans la liste, utilisez le menu Insertion/Fonction:



Si vous utilisez l'instruction **Private**:

Private MaFonctionPerso(NbValeurs As Long, Optional Coeff As Variant) As Long

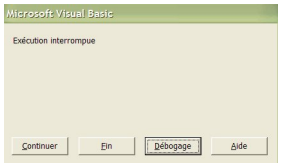
Dans ce cas la fonction **MaFonctionPerso** n'est accessible que pour les procédures du module dans lequel elle est déclarée.

Consultez le tutoriel pour créer et utiliser les fonctions.

II-C-6 - Arrêter une procédure en cours d'exécution

Vous pouvez arrêter l'exécution d'une macro en utilisant la touche **Echap** ou par le raccourci clavier **Ctrl+Pause**.

L'application interrompt l'exécution et affiche une boîte de dialogue.



Vous avez ensuite le choix entre plusieurs options:

* Continuer l'exécution.

* Arrêter définitivement l'exécution.

* Effectuer une pause (Comme si vous étiez en mode pas à pas).

* Afficher l'aide.

III - L'enregistreur de macros

Si vous ne savez pas comment écrire une procédure, sollicitez l'enregistreur de macros.

Lancez l'enregistreur depuis la feuille de calcul.

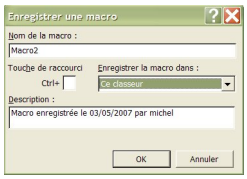
Une boîte de dialogue vous permet de spécifier préalablement:

* Le nom de la macro

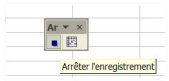
* Dans quel classeur enregistrer la macro

* Un raccourci clavier pour déclencher la macro (facultatif)

* Un commentaire en début de macro (facultatif).



Ensuite effectuez une série d'actions dans la feuille (Sélection et coloriage de cellules, écriture dans les cellules, ajout de feuille ...etc...), puis arrêtez l'enregistreur.



Retournez dans l'éditeur VBE.

Vous constatez qu'un nouveau module standard vient d'être créé. Il contient une traduction VBA des opérations précédemment effectuées.

Si vous recherchez une syntaxe adéquate pour vos procédures, utilisez l'enregistreur, analysez le résultat puis adaptez le code à votre besoin.

Si vous n'êtes pas un spécialiste du VBA et que vous recherchez une simple macro répétitive, conservez la macro enregistrée telle quelle et ré-exécutez la.

L'enregistreur est toutefois limité: Il fonctionne uniquement pour des actions successives. Les notions de boucles et de conditions ne sont pas prises en compte. Il crée uniquement des procédures de type **Sub**. Les macros enregistrées contiennent parfois des portions de codes qui ne servent pas, ou ne sont pas directement en rapport avec les actions effectuées. Celles ci sont donc améliorables.

Par exemple, la macro ci dessous a été créée depuis l'enregistreur pour colorier en jaune la cellule A1:

Vba

```
Range("A1").Select
With Selection.Interior
    .ColorIndex = 6
    .Pattern = xlSolid
End With
```

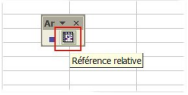
Vous pouvez obtenir le même résultat en écrivant directement:

Vba

```
Range("A1").Interior.ColorIndex = 6
```

Les références relatives:

Si vous voulez enregistrer une macro par rapport à la position de la cellule active, utilisez l'option des références de cellules relatives.



L'enregistrement s'effectue par rapport à la cellule active tant que vous ne quittez pas Excel ou ne cliquez pas à nouveau sur le bouton "**Référence relative**", pour la désélectionner.

IV - La sécurité des macros

Tout classeur Excel peut potentiellement contenir des macro-virus. C'est l'inconvénient de cet outil puissant qu'est le VBA.

De nombreux fichiers sont échangés chaque jour par le biais des messageries électroniques ou téléchargés depuis des forums et autres sites spécialisés: Ce qui augmente d'autant le risque de propagation des virus.

Même sans mauvaise intention, certaines macros mal maitrisées peuvent causer des dégâts.

Par exemple: La modification de la base de registre est toujours risquée pour votre PC. La manipulation des barres d'outils est aussi une source de problème dans l'application Excel.

Il convient donc d'être prudent.

Règle importante:

Quand vous recevez un classeur dont vous ne connaissez l'origine, ayez le réflexe de désactiver les macros lors de la première ouverture.

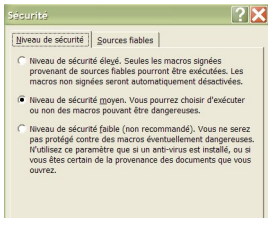
Ensuite vérifiez que les procédures contenues dans le fichier ne risquent de provoquer des actions indésirables sur votre PC.

S'il n'y a pas de problème, refermez le classeur puis ré-ouvrez le en activant les macros.

Un anti virus régulièrement mis à jour est aussi nécessaire pour vous protéger des macros malveillantes.

Pour paramétrer le niveau de sécurité, utilisez le menu Outils/Macro/Sécurité.

Sélectionnez le niveau de sécurité **Moyen** pour être alerté avant l'ouverture d'un classeur qui contient des macros.



Cliquez sur le bouton **OK** pour valider.

V - Conseils et astuces

Utilisez l'indentation pour aérer vos procédures et en améliorer la lisibilité.

L'indentation est le décalage des portions de macro par l'insertion de tabulations.

Sélectionnez votre portion de code:

```
Sub ModifieCellule_A1()
With Range("A1")
'Ecrit dans la cellule
.Value = "DVP"
'Modifie la couleur
.Interior.ColorIndex = 6
End With
End Sub
```

Appuyez sur la touche **Tabulation** (ou le menu Edition/Retrait).

```
Sub ModifieCellule_A1()
With Range("A1")
    'Ecrit dans la cellule
    .Value = "DVP"
    'Modifie la couleur
    .Interior.ColorIndex = 6
End With
End Sub
```

Le texte sélectionné est décalé sur la droite. Appuyez de nouveau sur la touche **Tabulation** pour ajouter un nouveau retrait.

La combinaison **MAJ+Tabulation** effectue un retrait négatif.

Il existe des compléments gratuits qui proposent des outils de gestion pour l'éditeur de macros (Aide à l'écriture des codes, vérification du contenu, supports de mise en forme ...).

Par exemple: **MZTools**.

Ajoutez des commentaires pour la faciliter relecture et la compréhension des macros.

Il est possible de copier un module ou un UserForm vers un autre classeur ouvert, en effectuant un Glisser/Déposer depuis l'explorateur de projets.

Rédigez les mots clé Excel, les instructions, les fonctions, les méthodes et les propriétés en minuscule. Si les mots sont correctement écrits, les majuscules s'afficheront automatiquement. C'est un moyen fiable pour vérifier la syntaxe.

Les variables sont des éléments très importants dans une macro.

Elles servent à stocker et manipuler des informations pendant le déroulement de vos procédures. Des noms de variables clairs et quelques bonnes pratiques d'utilisation amélioreront vos codes.

Pour plus de détail, consultez **le tutoriel qui leur est consacré**.

Utilisez toujours **Option Explicit** par défaut:

L'instruction **Option Explicit** est placée en tout début de module. Cela oblige à déclarer toutes les variables: Chaque variable non déclarée ou mal orthographiée sera ainsi signalée lors de l'exécution de la procédure.

Pour qu'**Option Explicit** soit ajouté automatiquement dans tous vos nouveaux classeurs:

Menu Outils/Options/Onglet "Editeur"/Cochez l'option "Déclaration des variables obligatoire".

Il existe des raccourcis claviers intéressants à connaitre dans l'éditeur VBE.

Quelques exemples en complément de ceux déjà indiqués dans les différents chapitres:

* **Ctrl+Espace**: Affiche une liste pour compléter le mot clé que vous êtes en train d'écrire.

* **Ctrl+I**: Affiche la syntaxe de la variable, fonction, instruction, méthode ou propriété sélectionnée.

* **Ctrl+A**: Sélectionne tout le texte de la fenêtre Code.

* **F7**: Affiche la fenêtre de code de l'objet sélectionné dans un UserForm.

* **Ctrl L**: Liste les procédures (Pile des appels) en cours d'exécution. La procédure doit être en mode Pause.

Vous pouvez insérer jusqu'à 1024 caractères par ligne de code mais il est conseillé de ne pas écrire en dehors de la partie visible de l'écran, pour améliorer l'ergonomie de travail dans la fenêtre de saisie.

Si besoin, insérez un espace suivi d'un caractère underscore _ pour revenir à la ligne.

Par exemple:

Vba
```
'Instruction sur une seule ligne.
MsgBox "Vous êtes dans le classeur " & ThisWorkbook.Name

'Résultat équivalent en utilisant des retours à la ligne (caractères underscore)
MsgBox _
    "Vous êtes dans" & _
    "le classeur " & _
    ThisWorkbook. _
    Name
```

Les phases de développement peuvent parfois amener des erreurs ou des plantages involontaires (blocage du classeur, perte de données ...), même chez les programmeurs chevronnés. Pour vous prémunir de ce genre de désagrément, sauvegardez régulièrement des versions différentes de vos classeurs.

Rédigez préalablement un synopsis détaillé de votre projet. Cela vous aidera à garder en vue les objectifs principaux et vous aiguillera pour la mise en oeuvre ainsi que pour la recherche de solutions.

Le partage de votre travail avec d'autres personnes:

Si votre classeur a pour but d'être distribué, faites le tester par plusieurs utilisateurs avant sa mise en service. Attention aux versions d'Office différentes: Assurez vous de la compatibilité avec les versions descendantes (plus anciennes).

VI - Les sources d'aide

Désormais vous savez comment créer une macro, mais une nouvelle question se pose:

Ou trouver de l'aide et des solutions à mes problèmes dans ce vaste domaine qu'est le langage VBA?

Il y a tout d'abord l'aide fournie dans l'application Microsoft Excel.

Quand vous ne connaissez pas la signification d'un terme, écrivez le dans la fenêtre Code, puis positionnez le curseur sur ce mot. Ensuite, appuyez sur la touche **F1** du clavier.

Une nouvelle fenêtre s'affiche à l'écran:

Si le terme recherché est un mot clé Excel, l'aide associée apparaitra à l'écran.

L'aide n'est pas toujours évidente à appréhender, mais vous allez vous rendre compte rapidement que les informations contenues sont très détaillées, et souvent complétées par des exemples.

Une fois affichée, vous pouvez aussi utiliser la fenêtre d'aide pour effectuer d'autres recherches par:

* Sommaire

* Aide intuitive

* Index

Si l'Aide n'est pas disponible sur votre poste, récupérez la depuis le CD-ROM d'installation de votre produit Microsoft Excel.

Il existe aussi beaucoup d'informations sur le web, mises à disposition par des contributeurs et contributrices bénévoles.

Des requêtes, à partir des moteurs de recherche, vous feront découvrir de nombreux cours, démo, trucs et astuces, FAQ, tutoriels et classeurs prêts à l'emploi.

Des sources d'aide très complètes sont notamment disponibles sur le site **Developpez.com**:

* **La FAQ Excel**

* **La FAQ VBA**

* **Les tutoriels**

* **Le Forum Excel**

* **Le forum VBA**

* **Ma page perso** ... Auto promotion ... ;o)

D'autres liens sur le web:

* **La base de connaissance Microsoft**

VII - Conclusion

Cet article ne présente qu'une partie infime des possibilités offertes par l'environnement VBA.

Pour améliorer vos connaissances dans la programmation, le meilleur des apprentissages consiste à mettre en application des exemples concrets: Développez vos projets personnels pour découvrir de nouvelles fonctionnalités. N'hésitez pas à tester de nouveaux codes. Vous vous apercevrez vite que le VBA est un outil très puissant et qu'il existe pratiquement toujours une solution à votre besoin.

VIII - Téléchargement