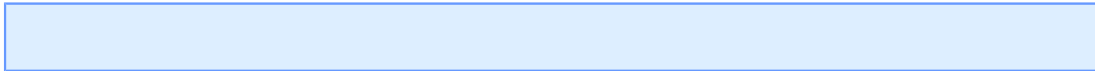


Débutez en VBA Word

par Olivier Lebeau ([Heureux-oli sur DVP](#))

Date de publication : 25 janvier 2009

Dernière mise à jour :



1 - Introduction.....	3
2 - L'objet Application Word.....	3
3 - Les documents.....	3
4 - Range et Selection.....	3
5 - Les paragraphes.....	3
6 - Les tableaux.....	3
7 - Les signets.....	3
7-A - Méthodes.....	7
7-A-1 - Exist.....	7
7-A-2 - Delete.....	7
7-A-3 - Add.....	7
7-A-5 - Exist.....	7
7-B - Propriétés.....	8
7-B-1 - Count.....	8
7-B-2 - Name.....	8
7-C - Start - End.....	8
8 - Les champs.....	8
8-A - Méthodes.....	9
8-A-1 - Add.....	9
8-A-2 - ToggleShowCodes.....	12
8-A-3 - Update.....	12
8-A-4 - Unlink.....	13
8-B - Propriétés.....	13
8-B-1 - Code.....	13
8-B-2 - ShowCodes.....	14
8-B-3 - Count.....	14
9 - Les formulaires.....	14
9-A - Méthodes.....	15
9-A-1 - Add.....	15
9-A-2 - Select.....	15
9-B - Propriétés.....	15
9-B-1 - CalculateOnExit.....	15
9-B-2 - DropDown.....	15
9-B-3 - EntryMacro et ExitMacro.....	16
9-B-4 - Result.....	16
9-B-5 - Protection du document.....	16
10 - Les images et autres objets graphiques.....	17
10-A - Les Shapes.....	18
10-B - Les InlineShapes.....	18
11 - Les objets Range et Selection.....	18
11-A - Méthodes.....	19
11-B - Propriétés.....	19
12 - Les en-têtes et les pieds de page.....	19
13 - Solutions.....	20
13-A - Solutions.....	20
13-A-1 - FileDialog.....	20
13-A-2 - ChangeFileOpenDirectory.....	20
13-A-3 - Quit.....	20
13-A-4 - PrintOut.....	21
13-A-5 - Compter les mots des phrases.....	22
13-A-6 - Mettre troisième mot en gras et double souligné.....	22
13-A-7 - Ajouter un document contenant une table.....	23
13-A-8 - Table de multiplication.....	23
14 - Exercice complet : une application de gestion de courrier.....	25
15 - Liens Utiles.....	25
16 - Remerciements.....	25

1 - Introduction

2 - L'objet Application Word

3 - Les documents

4 - Range et Selection

5 - Les paragraphes

6 - Les tableaux

7 - Les signets

Les signets sont des objets invisibles situés dans votre document et permettant l'accès direct à la position qu'ils occupent. Lorsque vous manipulez le contenu d'un document en VBA, il est bien plus facile d'atteindre un signet que de faire une recherche sur le contenu du document.

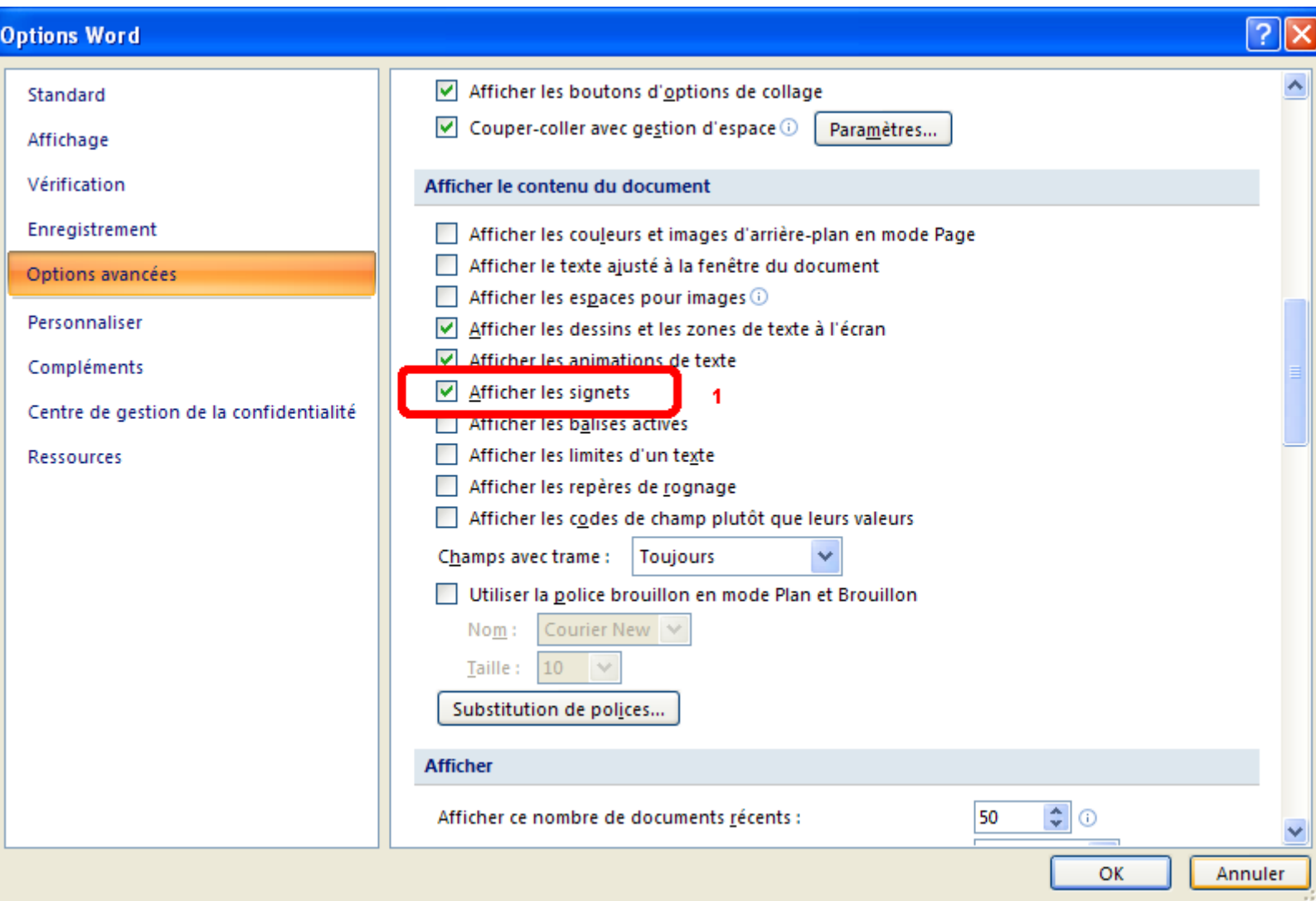
Les signets font partie de la collection **Bookmarks** on peut les atteindre par leur nom ou par leur index, comme pour le contenu des tableaux, on y insère des données ou on en récupère le contenu en passant par la propriété **Range**.

```
ActiveDocument.Bookmarks(1).Range.Text
```

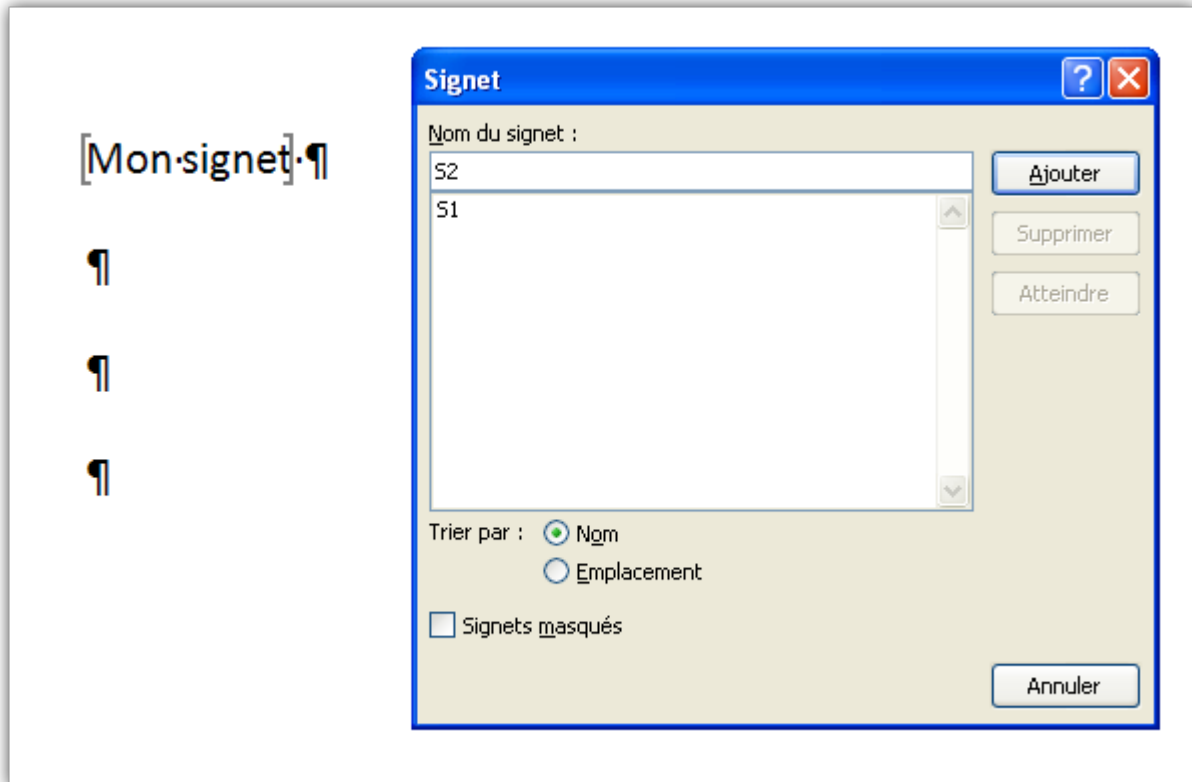
Il existe deux types de signets en VBA, les signets contenant des données et les signets vides qui sont juste un point d'insertion.



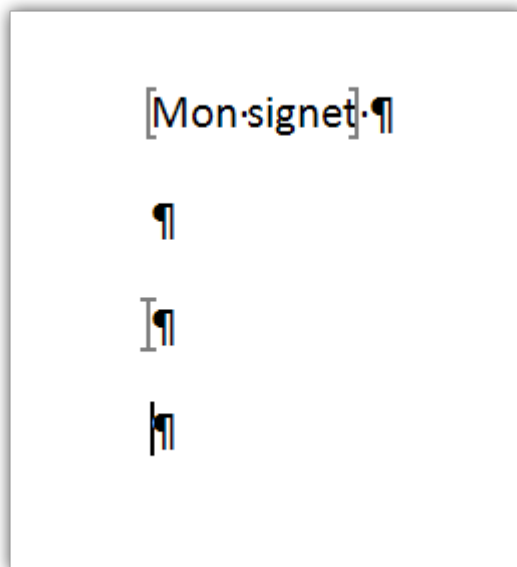
*Pour visualiser les signet présents sur votre document, c'est assez simple, vous devez aller dans les **Options de Word. Options Avancées** et dans la partie **Afficher le contenu du document**, vous devez cocher **Afficher les signets**.*



Vous aurez alors deux crochets gris pour symboliser l'emplacement de vos signets.



Créez un document possédant deux signets. Pour le premier, vous écrivez un mot, vous le sélectionnez et ensuite vous insérez un signet sur ce mot. Pour le second, ajoutez quelques paragraphes et ajoutez un signet là où se trouve votre curseur. Le nom que vous allez donner aux signets n'a pour cet exemple pas d'importance, nous allons les adresser par leur index. Vous devriez obtenir ceci :



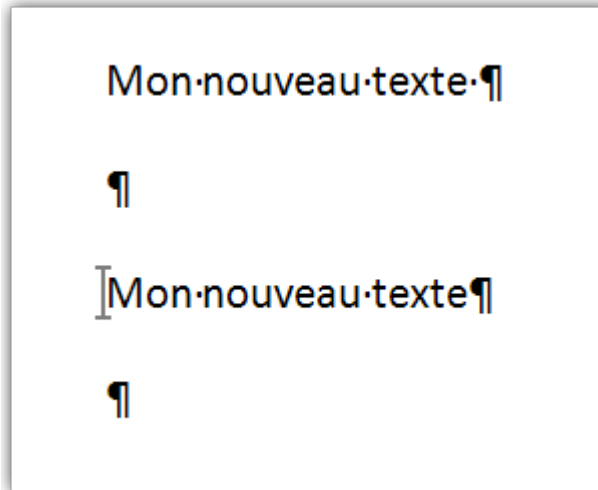
Le premier est un signet normal alors que le second est un point d'insertion. Utilisons un peu de code pour ajouter du texte à nos signets.

```
Sub AjouterTexteSignet ()
Dim stTexte As String
stTexte = "Mon nouveau texte"
ActiveDocument.Bookmarks(2).Range.Text = stTexte
```

```
ActiveDocument.Bookmarks(1).Range.Text = stTexte

End Sub
```

Après exécution, nous obtenons ceci :



Le premier signet est détruit et remplacé par le texte, alors que le second existe toujours. Si nous relançons la même procédure, nous obtenons un message d'erreur.

Il est possible de recréer le signet après avoir inséré du texte. Pour y parvenir, nous devons repérer la position de départ du signet et ensuite, connaître la longueur de la chaîne ajoutée au signet. Nous allons utiliser deux procédures pour cette manipulation, une fonction et une sub.

La fonction

```
Function RemplacerTexteSignet(MyBM As Bookmark, stTexte As String) As Boolean
'Déclaration des variables
'intI pour le début de notre Bookmark
Dim intI As Long
'stBM pour le nom de notre Bookmark
Dim stBM As String
'un objet range pour le range de notre Bookmark
Dim MyRng As Range

'Récupération du nom du signet
stBM = MyBM.Name
'Récupération de la position de départ de notre signet
intI = MyBM.Start
'Affectation du texte à notre Bookmark
MyBM.Range.Text = stTexte
'Affectation de l'objet Range, où la position de départ est
' la même que pour le Bookmark et la fin sera le début augmenté
' de la longueur du texte
Set MyRng = ActiveDocument.Range(Start:=intI, End:=intI + Len(stTexte))
' Crée le Bookmark sur l'objet Range
ActiveDocument.Bookmarks.Add stBM, MyRng
Set MyRng = Nothing
'Affectation de la valeur True à la fonction
RemplacerTexteSignet = True
End Function
```

Nous allons passer en paramètres à cette fonction le signet (MyBM As Bookmark) et le texte à remplacer (stTexte As String).

Il nous reste l'appel de cette fonction :

```
Sub AppelRemplacement ()  
If RemplacerTexteSignet (ActiveDocument.Bookmarks ("S2"), "Le fameux texte à remplacer") Then  
    MsgBox "Le remplacement s'est " & vbCrLf _  
        & "correctement déroulé !"  
End If  
End Sub
```

Si le remplacement s'est correctement déroulé, nous affichons un message.

7-A - Méthodes

7-A-1 - Exist

Cette méthode permet de savoir si le signet existe dans la collection.

```
If ActiveDocument.Bookmarks.Exists ("S1") Then  
    ActiveDocument.Bookmarks ("S1").Delete  
End If
```

Cette méthode renvoie la valeur **True** si le signet existe.

7-A-2 - Delete

Nous l'avons abordée dans l'exemple précédent. Un signet peut être supprimé par le code.

```
ActiveDocument.Bookmarks ("S1").Delete
```

7-A-3 - Add

Nous avons déjà fait appel à cette propriété dans notre exemple :

```
ActiveDocument.Bookmarks.Add stBM, MyRng
```

Cette méthode prend deux arguments, le premier (Name:=) est le nom du signet et le second (Range:=), la plage de données qui sera utilisée pour le signet, cette plage peut se résumer à un point d'insertion.

7-A-5 - Exist

Cette méthode permet de vérifier l'existence d'un signet. Si le signet existe, l'expression renvoie Vrai, Faux dans le cas contraire.

```
ActiveDocument.Bookmarks.Exists ("S1")  
' Avec S1 comme nom de signet
```

Cette méthode ne s'utilise qu'avec le nom du signet.

7-B - Propriétés

7-B-1 - Count

C'est probablement la propriété la plus célèbre, elle permet de connaître le nombre de signets contenu dans la collection des signets (**Bookmarks**).

```
Debug.Print ActiveDocument.Bookmarks.Count
```

7-B-2 - Name

Cette propriété renvoie le nom du signet, cette propriété est en lecture écriture, nous l'avons déjà utilisée pour notre exemple.

```
Debug.Print ActiveDocument.Bookmarks(1).Name
```

7-C - Start - End

Ces deux propriétés s'utilisent ensembles. Elles renvoient ou définissent le début et la fin d'un signet


```
ActiveDocument.Paragraphs(6).Range.Select
Selection.Bookmarks.Add Name:="S2"
With ActiveDocument.Bookmarks("S2")
    .Start = ActiveDocument.Paragraphs(6).Range.Characters(3).Start
    .End = ActiveDocument.Paragraphs(6).Range.Characters(6).End
End With
```

Dans cet exemple, nous ajoutons un signet sur le 6^{ième} paragraphe du document, pour ensuite placer ce signet avec pour début le 3^{ième} caractère et la fin au 6^{ième} caractère.

8 - Les champs

Il est plus courant d'utiliser les champs au travers de l'interface graphique, mais heureusement, leur utilisation est également possible dans votre code. L'utilisation des champs au travers de l'interface graphique est décrite détaillée dans ce tutoriel : [Découvrez les champs et leurs applications sous Word](#).

Les champs les plus utilisés sont les champs de fusion et de publipostage, les champs contenant des images ou du texte, les champs relatifs aux propriétés du document et les champs relatifs aux tables et index.

 *Si vous utilisez un chemin dans vos champs, Word ne les prend en comptes que si les "\" sont doublés.*

Ex : C:\Temp\la.docm devient C:\\Temp\la.docm

Insertion d'une image

```
Sub insereruneimage()
ActiveDocument.Fields.Add Range:=Selection.Range, _
    Type:=wdFieldIncludePicture, Text:="C:\\Temp\\a.jpg"
End Sub
```


L'insertion d'un champ se fait par l'utilisation de la méthode **Add**, on passe à cette méthode quatre arguments,
 Range : l'endroit où sera inséré le champ
 Type : constante Word déterminant le type de champ inséré
 Text : Le texte qui sera utilisé pour le champ.
 PreserveFormatting : Conserve ou non le format

Insertion le nom de l'utilisateur

```
Sub InserterNomUtilisateur()
ActiveDocument.Fields.Add Range:=Selection.Range, Type:=wdFieldEmpty, Text:= _
    "USERNAME ", PreserveFormatting:=True
End Sub
```

Dans le cas du nom de l'utilisateur, si vous ne mettez rien, c'est le nom d'utilisateur de la suite Office qui est utilisé. Vous pouvez imposer un autre nom.

```
Sub InsererNomUtilisateur()
ActiveDocument.Fields.Add Range:=Selection.Range, Type:=wdFieldEmpty, Text:= _
    "USERNAME Olivier \* Caps ", PreserveFormatting:=True
End Sub
```

En changeant le type de champ, vous pouvez insérer la date du jour.

Insertion de la date du jour

```
Sub AjouterDate()
ActiveDocument.Fields.Add Range:=Selection.Range, Type:=wdFieldDate
End Sub
```

Nous venons de voir comment ajouter un champ et le "code" qu'il contient, nous pouvons aussi faire la manipulation inverse, pour chaque champ récupérer son code.

```
Sub RecupererCode()
' Déclaration d'un variable de type champ
Dim oFld As Field
' Boucle sur tous les champs du document
For Each oFld In ActiveDocument.Fields
    Debug.Print oFld.Code
Next oFld
End Sub
```

8-A - Méthodes

8-A-1 - Add

Nous avons déjà utilisé cette méthode, elle permet l'ajout d'un champ à la collection des champs du document. Nous avons également vu que cette méthode peut recevoir jusqu'à 4 arguments.

Voici la liste des valeurs que peut prendre l'argument **Type**.

Nom	Description
wdFieldAddin	Champ de complément. Non disponible dans la boîte de dialogue Champ. Utilisé pour

	stocker les données cachées dans l'interface utilisateur.
wdFieldAddressBlock	Champ AddressBlock.
wdFieldAdvance	Champ Advance.
wdFieldAsk	Champ Ask.
wdFieldAuthor	Champ Author.
wdFieldAutoNum	Champ AutoNum.
wdFieldAutoNumLegal	Champ AutoNumLgl.
wdFieldAutoNumOutline	Champ AutoNumOut.
wdFieldAutoText	Champ AutoText.
wdFieldAutoTextList	Champ AutoTextList.
wdFieldBarCode	Champ BarCode.
wdFieldBidiOutline	Champ BidiOutline.
wdFieldComments	Champ Comments.
wdFieldCompare	Champ Compare.
wdFieldCreateDate	Champ CreateDate.
wdFieldData	Champ Data.
wdFieldDatabase	Champ Database.
wdFieldDate	Champ Date.
wdFieldDDE	Champ DDE. Non disponible dans la boîte de dialogue Champ, mais pris en charge pour les documents créés dans les versions précédentes de Word.
wdFieldDDEAuto	Champ DDEAuto. Non disponible dans la boîte de dialogue Champ, mais pris en charge pour les documents créés dans les versions précédentes de Word.
wdFieldDocProperty	Champ DocProperty.
wdFieldDocVariable	Champ DocVariable.
wdFieldEditTime	Champ EditTime.
wdFieldEmbed	Champ Embedded.
wdFieldEmpty	Champ Empty. Agit comme un espace réservé pour les contenus de champ qui ne sont pas encore ajoutés. Un champ ajouté en utilisant le raccourci Ctrl+F9 dans l'interface utilisateur est un champ Empty.
wdFieldExpression	Champ = (Formule).
wdFieldFileName	Champ FileName.
wdFieldFileSize	Champ FileSize field.
wdFieldFillIn	Champ Fill-In.
wdFieldFootnoteRef	Champ FootnoteRef. Non disponible dans la boîte de dialogue Champ. Inséré par programmation ou en mode interactif.
wdFieldFormCheckBox	Champ FormCheckBox.
wdFieldFormDropDown	Champ FormDropDown.
wdFieldFormTextInput	Champ FormText.
wdFieldFormula	Champ EQ (Equation).
wdFieldGoToButton	Champ GoToButton.
wdFieldGreetingLine	Champ GreetingLine.
wdFieldHyperlink	Champ Hyperlink.
wdFieldIf	Champ If.
wdFieldImport	Champ Import. Ne peut pas être ajouté dans la boîte de dialogue Champ, mais

	peut être ajouté en mode interactif ou par programmation (rédaction de code).
wdFieldInclude	Champ Include. Ne peut pas être ajouté dans la boîte de dialogue Champ, mais peut être ajouté en mode interactif ou par programmation (rédaction de code).
wdFieldIncludePicture	Champ IncludePicture.
wdFieldIncludeText	Champ IncludeText.
wdFieldIndex	Champ Index.
wdFieldIndexEntry	Champ XE (Index Entry).
wdFieldInfo	Champ Info.
wdFieldKeyWord	Champ Keywords.
wdFieldLastSavedBy	Champ LastSavedBy.
wdFieldLink	Champ Link.
wdFieldListNum	Champ ListNum.
wdFieldMacroButton	Champ MacroButton.
wdFieldMergeField	Champ MergeField.
wdFieldMergeRec	Champ MergeRec.
wdFieldMergeSeq	Champ MergeSeq.
wdFieldNext	Champ Next.
wdFieldNextIf	Champ NextIf.
wdFieldNoteRef	Champ NoteRef.
wdFieldNumChars	NumChars.
wdFieldNumPages	Champ NumPages.
wdFieldNumWords	Champ NumWords.
wdFieldOCX	Champ OCX. Ne peut pas être ajouté dans la boîte de dialogue Champ, mais peut être ajouté par programmation (rédaction de code), par le biais de la méthode

	AddOLEControl de la collection Shapes ou de la collection InlineShapes.
wdFieldPage	Champ Page.
wdFieldPageRef	Champ PageRef.
wdFieldPrint	Champ Print.
wdFieldPrintDate	Champ PrintDate.
wdFieldPrivate	Champ Private.
wdFieldQuote	Champ Quote.
wdFieldRef	Champ Ref.
wdFieldRefDoc	Champ RD (Reference Document).
wdFieldRevisionNum	Champ RevNum.
wdFieldSaveDate	Champ SaveDate.
wdFieldSection	Champ Section.
wdFieldSectionPages	Champ SectionPages
wdFieldSequence	Champ Seq (Sequence).
wdFieldSet	Champ Set.
wdFieldShape	Champ Shape. Créé automatiquement pour les images dessinées.
wdFieldSkipIf	Champ SkipIf.
wdFieldStyleRef	Champ StyleRef.
wdFieldSubject	Champ Subject.
wdFieldSymbol	Champ Symbol.
wdFieldTemplate	Champ Template.
wdFieldTime	Champ Time.
wdFieldTitle	Champ Title.
wdFieldTOA	Champ TOA (Table of Authorities).
wdFieldTOAEntry	Champ TOA (Entrée dans la table des références).
wdFieldTOC	Champ TOC (Table of Contents).
wdFieldTOCEntry	Champ TOC (Entrée dans la table des matières).
wdFieldUserAddress	Champ UserAddress.
wdFieldUserInitials	Champ UserInitials.
wdFieldUserName	Champ UserName.
wdFieldBibliography	Champ Bibliography.
wdFieldCitation	Champ Citation.

Cet exemple montre comment ajouter un champ date à votre document :

```
Sub AjouterChampDate()
ActiveDocument.Fields.Add Range:=Selection.Range, Type:=wdFieldDate
End Sub
```

8-A-2 - ToggleShowCodes

Cette méthode correspond à la combinaison de touches Shift + F9, elle permet l'affichage des codes de champ.

```
Selection.Fields.ToggleShowCodes
```

8-A-3 - Update

Lorsque vous utilisez cette méthode, vous mettez à jour le ou les champs du document.

Mise à jour d'un champ

```
ActiveDocument.Fields(1).Update
```

Mise à jour de tous les champs

```
ActiveDocument.Fields.Update
```

L'utilisation de cette méthode renvoie un **Long**, si la valeur renvoyée est 0, c'est que la mise à jour s'est correctement déroulée, si la mise à jour a généré une erreur, le nombre renvoyé correspond à l'index du premier champ qui pose problème.

```
If ActiveDocument.Fields.Update = 0 Then  
    MsgBox "La mise à jour s'est correctement déroulée"  
Else  
    MsgBox "Le champ " & ActiveDocument.Fields.Update & _  
        " a généré une erreur"  
End If
```

8-A-4 - Unlink

Cette méthode permet dans certains cas de briser le lien entre le champ la source et de remplacer le champ par sa valeur la plus récente.

```
ActiveDocument.Fields(1).Unlink
```

Cette méthode peut-être appliquée à la collection.

```
ActiveDocument.Fields.Unlink
```

Dans ce cas, tous les champs seront remplacés.

8-B - Propriétés

8-B-1 - Code

Cette propriété renvoie le code du champ. Elle est en lecture-écriture. c'est donc grâce à cette propriété que vous pourrez récupérer les codes d'un champ ou encore en modifier le code.

Nous avons vu comment ajouter un champ date.

```
Sub AjouterChampDate()  
ActiveDocument.Fields.Add Range:=Selection.Range, Type:=wdFieldDate  
End Sub
```

Nous allons modifier ce champ pour lui ajouter un masque.

```
Sub AjouterMasque()  
' Déclaration des variables  
Dim stChamp As String  
Dim oFld As Field  
Dim intI As Integer  
Dim stContent() As String
```

```

'Affectation des objets
Set oFld = ActiveDocument.Fields(1)

stChamp = oFld.Code
'Boucle pour déterminer si le champ est un champ date
If oFld.Type = wdFieldDate Then
    'Remplissage du tableau
    stContent = Split(stChamp, "\")
    Debug.Print LBound(stContent)
    stChamp = ""
End If
'Boucle sur le contenu du tableau
For intI = 0 To UBound(stContent)
    Debug.Print stContent(intI)
    'Ajout du masque avant le second élément du tableau
    If intI = 1 Then stChamp = stChamp & "@ " & Chr(34) & "dddd dd/MM/yyyy" & Chr(34) & " \"
    If intI = UBound(stContent) Then
        stChamp = stChamp & stContent(intI)
    Else
        stChamp = stChamp & stContent(intI) & " \"
    End If
Next intI
Debug.Print stChamp
oFld.Code.Text = stChamp
oFld.Update

End Sub

```

8-B-2 - ShowCodes

Alors que ToggleShowCode bascule l'affichage du code tous les champs, ShowCodes permet de ne le faire que pour un seul champ.

```
ActiveDocument.Fields(1).ShowCodes = True
```

8-B-3 - Count

Les fields formant une collection, **Count** permet de connaître leur nombre.

```
ActiveDocument.Fields.Count
```

9 - Les formulaires

Cette partie va traiter des formulaires Word, vous ne devez pas confondre pas confondre UserForm et formulaire, les UserForms sont des objets VBA alors que les formulaires sont des objets du document Word. Nous n'allons traiter que les contrôles formulaires "hérités".



Les champs de formulaire ne sont fonctionnels que si le formulaire est verrouillé en mode remplissage de formulaire.

9-A - Méthodes

9-A-1 - Add

Cette méthode nous permet d'ajouter un champ de formulaire à la collection. Elle reçoit deux arguments, le premier est l'endroit où sera inséré le champ et le second, le type de champ.

Ajout d'un champ de formulaire

```
ActiveDocument.FormFields.Add Range:=Selection.Range, Type:=wdFieldFormTextInput
```

wdFieldFormCheckBox	Champ FormCheckBox.
wdFieldFormDropDown	Champ FormDropDown.
wdFieldFormTextInput	Champ FormText.

9-A-2 - Select

Cette méthode sélectionne un élément de la collection. Si vous souhaitez lors de l'ouverture de votre document vous positionner sur un champ précis, c'est avec cette méthode que vous le ferez.

9-B - Propriétés

9-B-1 - CalculateOnExit


9-B-2 - DropDown

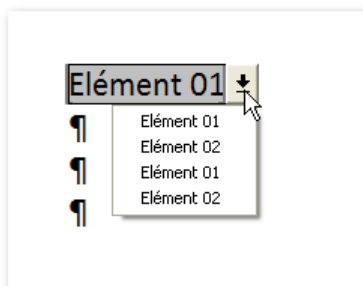
La propriété **DropDown** donne accès à la propriété **ListEntries** permettant l'ajout d'éléments dans une liste déroulante. Elle n'est utilisable qu'avec une liste déroulante.

Ajout d'élément à une liste

```
Sub AjouterElements()
Dim oFrmFld As FormField

Set oFrmFld = ActiveDocument.FormFields("dropDown01")
With oFrmFld.DropDown.ListEntries
.Add Name:="Elément 01"
.Add Name:="Elément 02"
End With
Set oFrmFld = Nothing
End Sub
```

 *Si vous exécutez votre code plusieurs fois, lors de chaque exécution, vous ajoutez les éléments à la liste.
Vous obtiendrez plusieurs fois les mêmes éléments.*

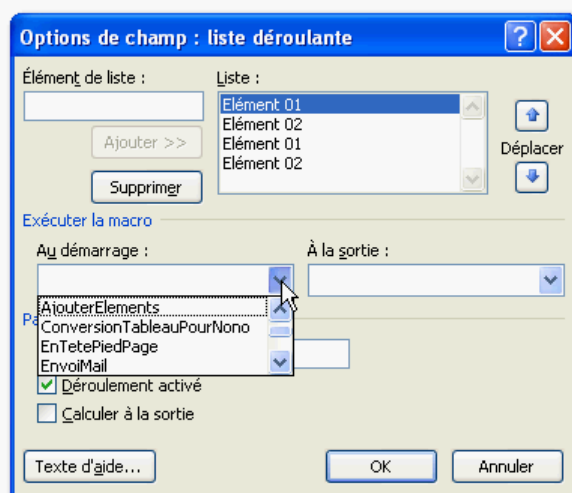


9-B-3 - EntryMacro et ExitMacro

Ces deux propriétés permettent d'attribuer à un champ de formulaire le nom de la macro qui sera exécutée sur l'évènement Sortie ou Entrée de ce dernier.

```
ActiveDocument.FormFields(1).EntryMacro = "MiseEnForme"
```

Ça correspond à ce choix dans la boîte de dialogue propriété du champ.



9-B-4 - Result

Cette propriété permet de récupérer le résultat du champ.

```
Debug.Print ActiveDocument.FormFields(1).Result
```

9-B-5 - Protection du document

Pour pouvoir utiliser un formulaire, vous devez le protéger. La protection doit autoriser le remplissage des champs de formulaire. Le code ci-dessous vous montre comment verrouiller votre document pour autoriser le remplissage des formulaires.

```
Sub ProtegerDocument()  
  
With ActiveDocument  
    .Protect wdAllowOnlyFormFields  
End With
```



```
End Sub
```

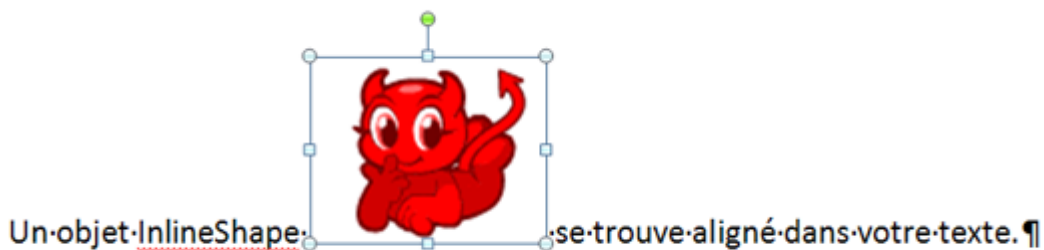
Il y a cependant un problème, tout le document est protégé, si nous avons du texte libre à ajouter au document, nous devons ajouter une section et déprotéger cette section.

```
Sub ProtegerDocument ()
With ActiveDocument
.Protect wdAllowOnlyFormFields
.Sections(2).ProtectedForForms = False
End With
End Sub
```

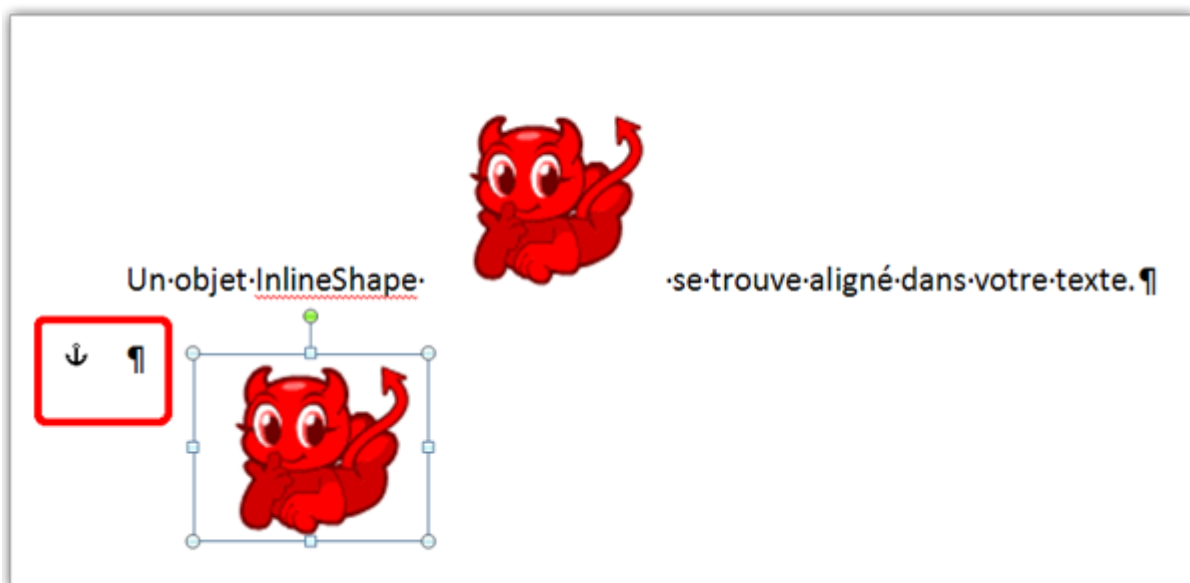
Le code ci-dessus montre comment verouiller tout le document et supprimer la protection de la seconde section.

10 - Les images et autres objets graphiques

Dans un document Word, nous retrouvons deux types d'objets graphiques, les **Shapes** et les **InlineShapes**. La différence se situe au niveau de la position qu'occupe l'objet sur le document. Les objets InlineShapes se trouvent dans l'alignement du texte, alors que les objets Shapes peuvent se trouver n'importe où sur votre document.



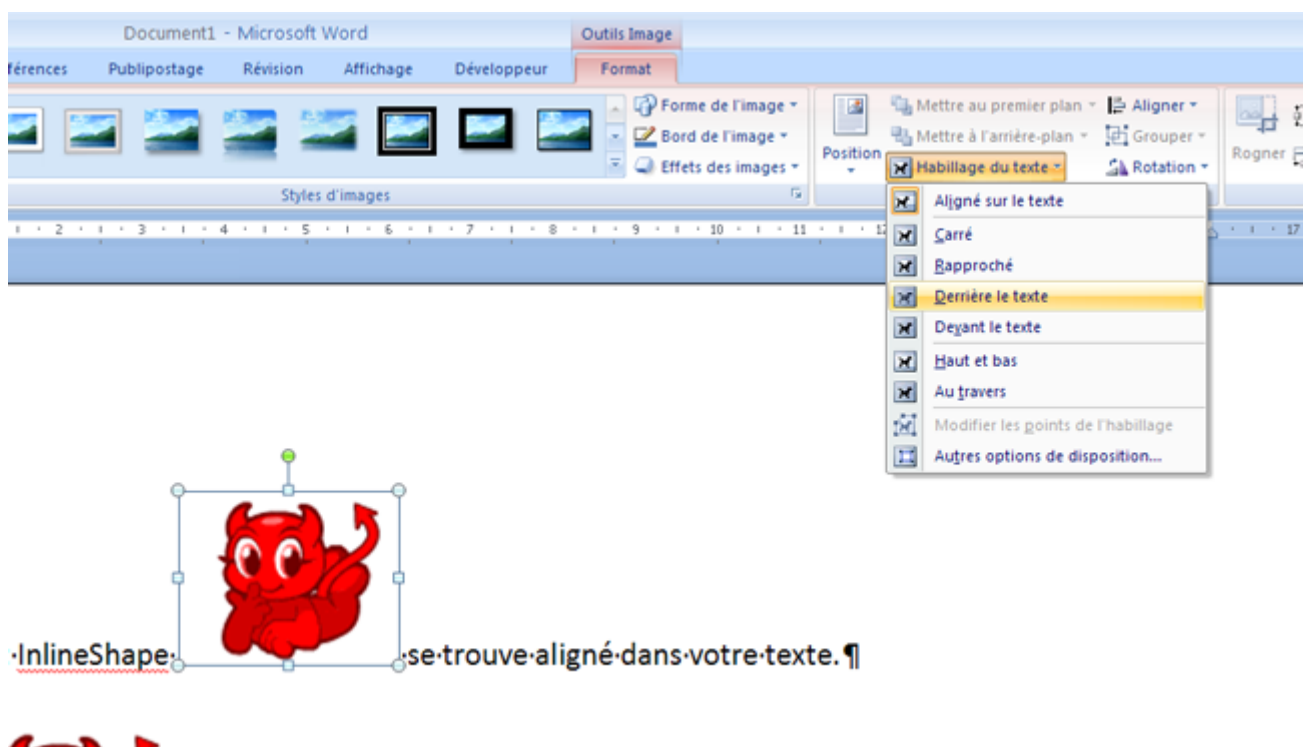
Un objet InlineShape



Un objet Shape

Lorsque vous sélectionnez l'objet Shape, une petite ancre apparaît devant un paragraphe. Ce symbole vous indique où l'objet est accroché dans votre texte.

On obtient un objet **Shape** par la transformation d'un objet **InlineShape**.



Transformation d'un objet InlineShape en objet Shape

10-A - Les Shapes

10-B - Les InlineShapes

11 - Les objets Range et Selection

Ces deux "objets" seront traités dans le même chapitre parce qu'ils ont pratiquement les mêmes méthodes et propriétés. Ils permettent d'insérer ou de récupérer les données qui se trouvent dans le document.

L'objet sélection possède une petite particularité, il représente le texte sélectionné du document.

Petit essai, vous insérez au clavier un morceau de texte, vous en sélectionnez une partie à la souris ou au clavier, pour ensuite utiliser ce morceau de code :

```
Sub RecupererSelection()
    MsgBox Selection.Text
End Sub
```

Si vous n'avez pas commis d'erreur, vous avez obtenu un message contenant le texte sélectionné.

Nous allons faire une manipulation similaire pour l'objet Range.

```
Sub AfficherRange ()
```

```
ActiveDocument.Paragraphs(1).Range.Text
End Sub
```

Vous avez obtenu un message avec le contenu du premier paragraphe de votre document.

11-A - Méthodes

11-B - Propriétés

12 - Les en-têtes et les pieds de page

Les en-têtes et les pieds de pages dépendent de deux facteurs, le premier est la section, le second la mise en page. Lorsque vous utilisez des en-têtes et pieds de page, vous pouvez les définir pour chaque section de votre document, mais lors de la mise en page, vous pouvez aussi choisir une première page différente et également une page paire et impaire différente. Tous ces choix ont une influence sur la façon dont nous allons les adresser en VBA.

Pour les adresser en fonction de la section, c'est très simple, il suffit de faire référence à la section en utilisant son index.

```
ActiveDocument.Section(1)
```

Ensuite, selon que l'on souhaite adresser l'en-tête ou le pied de page :

Pour l'en-tête

```
Activedocument.Sections(1).Headers(wdHeaderFooterPrimary)
```

Pour le pied de page

```
Activedocument.Sections(1).Footers(wdHeaderFooterPrimary)
```

Section()	Section adressée par le code
Headers()	Selon le type
Footers()	Selon le type

Il existe trois types d'en-têtes ou de pieds de page.

wdHeaderFooterEvenPages	Renvoie tous les en-têtes ou pieds de page figurant sur les pages paires.
wdHeaderFooterFirstPage	Renvoie le premier en-tête ou pied de page spécifié dans un document ou une section.
wdHeaderFooterPrimary	Renvoie l'en-tête ou le pied de page de toutes les pages d'un document ou d'une section, à l'exception de celui de la première page.

13 - Solutions

13-A - Solutions

13-A-1 - FileDialog

Dans l'énoncé, nous avons deux arguments à passer à la fonction, le répertoire initial et le titre de la boîte de dialogue. La fonction devra retourner le nom du fichier choisi.

Fonction FileDialog (Solution)

```
Function strFileName( strInitialFolder As String, strTitre As String ) As String
Dim dlg As FileDialog

Set dlg = Application.FileDialog(msoFileDialogFilePicker)
With dlg
.InitialFileName = strInitialFolder
.Title = strTitre
.Show
End With
strFileName = dlg.SelectedItems(1)
Set dlg = Nothing
End Function
```

L'appel de la fonction

Appel Fonction FileDialog

```
Sub ObtenirFichier()
Debug.Print strFileName("c:\windows\", "www.developpez.com")

End Sub
```

13-A-2 - ChangeFileOpenDirectory

Pour y arriver, vous devez déclarer la variable dans le module et pas dans la procédure.

```
Public oldPath As String

'*****
Sub CahngerDefPath()
oldPath = Application.ChangeFileOpenDirectory
Application.ChangeFileOpenDirectory = "C:\Temp\"
End Sub
'*****

Sub RestaurerDefPath()
Application.ChangeFileOpenDirectory = oldPath
End Sub
```

13-A-3 - Quit

Nous avons abordé comment affecter un objet à une variable. Il existe cinq solutions à cet exercice.

Solution N° 1

```
Dim objWapp As Word.Application
```

Solution N° 1

```
set objWapp = New Word.Application
...
...
objWapp.Quit
Set objWapp = Nothing
```

Solution N° 2

```
Dim objWapp As New Word.Application
...
...
objWapp.Quit
Set objWapp = Nothing
```

Solution N° 3

```
Dim objWapp As Word.Application

set objWapp = Word.Application
...
...
objWapp.Quit
Set objWapp = Nothing
```

Solution N° 4

```
Dim objWapp As Object

set objWapp = CreateObject("Word.Application")
...
...
objWapp.Quit
Set objWapp = Nothing
```

Solution N° 5

```
Dim objWapp As Word.Application

set objWapp = GetObject("Word.Application")
...
...
objWapp.Quit
Set objWapp = Nothing
```

Dans tous les cas de figure, il faut utiliser **objet.Quit** suivi de **Set objet = Nothing**.

13-A-4 - PrintOut

Pour ouvrir le fichier via une boîte de dialogue, nous avons vu comment manipuler l'objet FileDialog, nous allons l'utiliser. L'étape suivante est l'ouverture du fichier suivie de son impression et finalement sa fermeture.

Exercice PrintOut

```
Sub ImpressionFichier()
Dim objDlg As FileDialog
Dim strNomFichier As String
'Affectation des objets
Set objDlg = Application.FileDialog(msoFileDialogFilePicker)

objDlg.Show
' Test de vérification de fichier choisi
```

Exercice PrintOut

```

' si pas de fichier choisi, on sort de la procédure
If objDlg.SelectedItems.Count = 0 Then
    Set objDlg = Nothing
    Exit Sub
End If
'Récupération du nom de fichier
strNomFichier = objDlg.SelectedItems(1)
'Ouverture du fichier
Documents.Open strNomFichier
'Impression
ActiveDocument.PrintOut
'Fermeture
ActiveDocument.Close
'libération des objets
Set objDlg = Nothing

End Sub

```

Dans notre code, nous avons introduit une condition supplémentaire, si nous n'avons pas de fichier choisi, nous générons une erreur, en faisant ce test, nous sortons de la procédure s'il n'y a pas de fichier choisi.


13-A-5 - Compter les mots des phrases

Nous allons parcourir la collection des phrases pour compter le nombre de mots contenu dans chaque phrase du premier paragraphe.

```

Sub CompterMots()
Dim wdSentence
' ne pas donner de type à la variable équivaut à la déclarer
' comme variant
' Faire une boucle sur la collection Sentence et en compter les mots
For Each wdSentence In ActiveDocument.Paragraphs(1).Range.Sentences
    Debug.Print wdSentence.Words.Count
Next wdSentence
End Sub

```

 *Ne pas déclarer le type de la variable ne doit être utilisé que si ce type n'existe pas.*

13-A-6 - Mettre troisième mot en gras et double souligné

Comme nous devons parcourir tous les paragraphes de la collection, nous allons déclarer un objet paragraphe et une boucle **For Each ... Next**. Pour chaque paragraphe, nous allons adresser le troisième mot de la collection des mots de ce paragraphe.

Mot trois gras et souligné

```

Sub MettreMot3Gras()
'Déclaration des variable
Dim pAra As Paragraph
'Boucle sur les paragraphes du document
For Each pAra In ActiveDocument.Paragraphs
    With pAra.Range.Words(3).Font
        .Bold = True
        .Underline = wdUnderlineDouble
    End With
Next pAra

End Sub

```

13-A-7 - Ajouter un document contenant une table

Pour ajouter une table vous devez spécifier le nombre de lignes et de colonnes que votre table va contenir. Pour y parvenir, nous allons utiliser deux variables qui vont contenir le nombre de lignes (intR) et de colonnes (intC). Pour l'ajout de la nouvelle table, nous utiliserons ces deux variables.

```
Sub AjouterDocEtTable()
Dim intR As Integer
Dim intC As Integer

With ActiveDocument.Tables(1)
    intR = .Rows.Count 'Nombre de lignes
    intC = .Columns.Count 'Nombre de colonnes
End With
'Ajout du nouveau document
Documents.Add
'Ajout d'une table
ActiveDocument.Tables.Add Range:=Selection.Range, numRows:=intR, numcolumns:=intC

End Sub
```

Comme le document ajouté devient le Document Actif, pour ajouter la table, nous utiliserons l'**ActiveDocument**.

13-A-8 - Table de multiplication

Avant de créer notre document, nous devons déterminer combien lignes et de colonnes votre table va comporter. L'énoncé demande 10 nombres, 20 valeurs et un titre par colonne et par ligne. Ce qui va nous donner 11 colonnes et 21 lignes.

Nous allons en profiter pour changer l'apparence de notre table en ajoutant des bordures extérieures et intérieures, les titres des lignes et colonnes seront en gras souligné.

Pour l'ajout d'un nouveau document, nous l'avons déjà vu plus tôt.

```
Dim oDoc As Document

Set oDoc = Application.Documents.Add
```

Pour l'ajout d'une table, vous l'avez déjà fait.

```
Dim oTbl As Table
Dim intR As Integer 'Nombre de lignes
Dim intC As Integer 'Nombre de colonnes

' Ajout de notre table
Set oTbl = oDoc.Tables.Add (Range:=Selection.Range, NumRows:= 21 , NumColumns:=11)
```

Pour les résultats, nous allons utiliser une boucle

```
For intR = 2 To 21
    For intC = 2 To 11
        oTbl.Cell(intR, intC).Range.Text = (intC - 1) * ( intR - 1)
    Next intC
Next intR
```

Il nous manque les données pour les titres des lignes et colonnes et nous allons en profiter pour mettre notre test en forme.

```

For intR = 2 To 21
    'Données de la première ligne
    oTbl.Cell(intR, 1).Range.Text = intR - 1
    'Mise en forme de chaque cellule
    With oTbl.Cell(intR, 1).Range.Font
        .Bold = True
        .Underline = wdUnderlineDouble
        .UnderlineColor = wdColorBlack
    End With
Next intR
For intC = 2 To 11
    'Données de la première colonnes
    oTbl.Cell(1, intC).Range.Text = intC - 1
    'Mise en forme
    With oTbl.Cell(1, intC).Range.Font
        .Bold = True
        .Underline = wdUnderlineDouble
        .UnderlineColor = wdColorBlack
    End With
Next intC
    
```

Finalement, il nous reste la mise en forme de la table

```

With oTbl.Borders
    .Enable = True
    .InsideLineStyle = wdLineStyleDot
    .OutsideLineStyle = wdLineStyleSingle
    .InsideLineWidth = wdLineWidth050pt
    .OutsideLineWidth = wdLineWidth100pt
End With
    
```

Si nous assemblons tous nos petits morceaux de code, nous obtenons ceci :

Le code complet

```

Sub TableDeMultiplication()
    Dim oDoc As Document
    Dim oTbl As Table
    Dim intR As Integer 'Nombre de lignes
    Dim intC As Integer 'Nombre de colonnes

    Set oDoc = Application.Documents.Add
    ' Ajout de notre table
    Set oTbl = oDoc.Tables.Add(Range:=Selection.Range, NumRows:=21, NumColumns:=11)
    'Les résultats
    For intR = 2 To 21
        For intC = 2 To 11
            oTbl.Cell(intR, intC).Range.Text = (intR - 1) * (intC - 1)
        Next intC
    Next intR
    'Les titres
    intR = 0
    intC = 0
    For intR = 2 To 21
        oTbl.Cell(intR, 1).Range.Text = intR - 1
        With oTbl.Cell(intR, 1).Range.Font
            .Bold = True
            .Underline = wdUnderlineDouble
            .UnderlineColor = wdColorBlack
        End With
    Next intR
    For intC = 2 To 11
        oTbl.Cell(1, intC).Range.Text = intC - 1
    Next intC
End Sub
    
```


Le code complet

```
With oTbl.Cell(1, intC).Range.Font
    .Bold = True
    .Underline = wdUnderlineDouble
    .UnderlineColor = wdColorBlack
End With
Next intC

'Mise en forme
'Bordure
With oTbl.Borders
    .Enable = True
    .InsideLineStyle = wdLineStyleDot
    .OutsideLineStyle = wdLineStyleSingle
    .InsideLineWidth = wdLineWidth050pt
    .OutsideLineWidth = wdLineWidth100pt
End With

End Sub
```

14 - Exercice complet : une application de gestion de courrier

15 - Liens Utiles

	Titre de l'article
VBA	Généralités sur le VBA
	Initiation au VBA d'Outlook

16 - Remerciements

Je tiens à remercier pour leurs contributions à ce projet :

- **lorenzole+bo**
- **Philippe JOCHMANS**
- **Caro-Line**
- **Arkham46**
- **Jeannot45**