Chapitre 2 : Premiers pas en VBA

Emmanuel Hyon
by way of L. Mesnager

Université Paris Ouest Nanterre

2014-2015

Sommaire

Sommaire

Le langage VBA

VBA pour Excel est une extension du langage `Visual Basic`.

Un concept central en VBA est le concept d'objet.

Un exemple

Une cellule contient une valeur, qui peut être le résultat d'une formule.

En plus de la valeur contenue, une cellule possède de nombreuses autres caractéristiques : couleur de fond, couleur du texte, taille de la police, alignement du texte dans la cellule (centré, aligné à gauche, ...), etc ...

Et surtout, il y a différentes actions possibles sur une cellule : effacer son contenu, la supprimer, la copier dans le presse-papiers, etc ...

Un exemple

En résumé, pour agir sur une cellule, il faut à la fois disposer

d'un ensemble de variables pour enregistrer	d'une liste d'opérations permettant
▶ sa valeur, ▶ une formule, ▶ la couleur de fond, ▶ l'alignement du texte, ▶ etc ...	▶ d'effacer son contenu, ▶ de la supprimer, ▶ etc ...

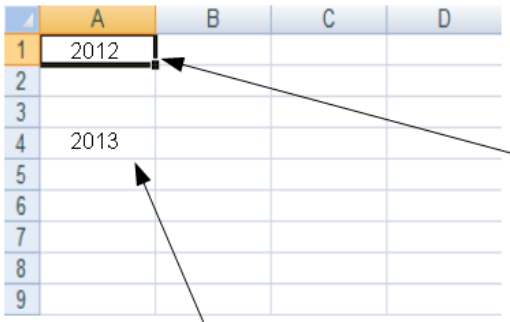
Un exemple

Un objet permet de regrouper ensemble et dans un même emplacement en mémoire des variables et des opérations sur ces variables.

Les variables contenues dans un objet sont appelées les propriétés de l'objet.

Les opérations contenues dans un objet sont appelées les méthodes de l'objet.

Un exemple



Objet Cellule associé à la cellule $A$1

Formule= «2012»

Valeur = 2012

Effacer le contenu

Objet Cellule associé à la cellule $A$4

Formule= « =$A$1+1»

Valeur = 2013

Effacer le contenu

A chaque cellule de la feuille de calcul, on associe une structure contenant les propriétés de la cellule et la liste des différentes actions applicables à cette cellule

Un exemple

1. Il ne faut pas confondre la cellule et l'objet associé !

   En effet, à la même cellule peuvent être associés plusieurs objets.

2. Un objet est une zone mémoire identifiée par son adresse en mémoire.

   Pour accéder à cet objet, on **utilise** l'adresse de cet objet dans un programme VBA.
   Généralement, on enregistre cette adresse dans une variable ce qu'on appelle une référence.

   Attention, référence $\neq$ objet !

   Il peut y avoir plusieurs références vers un même objet !

3. Comme pour les variables, il existe différents types d'objets.

   Chacun de ces types correspond à un des composants d'`Excel` (cellules et plages de cellules, feuilles de calcul, classeurs)

Sommaire

Associer une cellule à un objet

Le type des objets associés aux cellules est le type `Range`.

Pour associer une cellule de la feuille de calcul active, on peut procéder en deux étapes :

1. Déclarer une référence vers un objet de type `Range`

```
Dim c As Range
```

2. Créer un objet en l'associant à une des cellules de la feuille active et affecter à la référence c l'adresse de cet objet.

```
Set c = Range("A1")
```

L'instruction `Range` crée un objet associée à la cellule `A1` et retourne son adresse.

Associer une cellule à un objet

Pour faciliter la programmation, `Excel` fournit un autre moyen de créer un objet associé à une cellule :

```
Set c = [A1]
```

On peut utiliser indifféremment l'une ou l'autre des deux méthodes précédentes pour associer une cellule si on dispose de la référence de la cellule dans la feuille de calcul.

Associer une cellule à un objet

Par contre, si la référence de la cellule est calculée, on ne peut utiliser que l'instruction `Range` :

```
Dim i As Integer, c As Range

i = ...

Set c = Range("A" & i)
```

Remarque. La référence d'une cellule étant indiquée à l'aide d'une chaîne de caractères, on peut utiliser l'opérateur de concaténation & pour construire la chaîne de caractères contenant la référence de la cellule.

Associer une cellule à un objet

On peut créer plusieurs objets associés à une cellule :

```
Dim c1 As Range, c2 As Range

Set c1 = Range("A1")

Set c2 = Range("A1")
```

c1 et c2 sont deux références vers des objets distincts bien qu'associés à la même cellule.

Associer une cellule à un objet

On peut déclarer plusieurs références vers un même objet :

```
Dim c1 As Range, c2 As Range

Set c1 = Range("A1")

Set c2 = c1
```

c1 et c2 sont deux références vers le même objet.

L'instruction Is

Le langage VBA fournit l'instruction Is qui permet de déterminer si deux références sont des références vers un même objet ou des objets distincts :

```
Dim c1 As Range, c2 As Range, c3 As Range
Dim b1 As Boolean, b2 As Boolean

Set c1 = Range("A1")
Set c2 = Range("A2")
Set c3 = c1

b1 = c2 Is c1 ' b vaut False
b2 = c3 Is c1 ' b vaut True
```

Sommaire

Contenu d'une cellule

Une cellule d'une feuille de calcul peut :

- être vide,
- contenir une valeur saisie par l'utilisateur
- contenir une valeur qui est le résultat de l'évaluation d'une formule

Pour savoir si une cellule est vide ou non, le langage VBA fournit l'instruction `IsEmpty` :

```
Dim c As Range, b As Boolean

Set c = [A1]

b = IsEmpty(c)
```

La variable `b` vaudra :

- `True` si la cellule `A1` est vide
- `False` si la cellule `A1` n'est pas vide (contient une valeur)

Valeurs et formules

- La propriété `Value` permet d'accéder à la valeur que contient une cellule,
- la propriété `Formula` permet d'accéder à la formule enregistrée dans une cellule.

Attention. Si la cellule contient une formule, la valeur de la cellule est le résultat de l'évaluation de la formule. Si on a enregistré une valeur dans la cellule, la propriété `Formula` est alors la valeur enregistrée.

La propriété Value

Une cellule pouvant contenir différents types de valeurs : valeurs numériques, chaînes de caractères, dates, etc ... La propriété `Value` est d'un type particulier : le type `Variant`.

Une variable de type `Variant` peut prendre n 'importe quel type de valeurs (de type `Double`, de type `String`, etc ...).

```
Dim v As Variant

v = 1
v = 2.17
v = "master"
v = True
```

Un exemple de lecture de la valeur contenue dans une cellule

```
Dim c As Range

Set c = [A1]

MsgBox c.Value
'affiche la valeur contenue dans la cellule A1
```

Un exemple d'écriture d'une valeur dans une cellule

```
Dim c As Range

Set c = [A1]

c.Value = 3.14
c.Value = "Master ISIFAR"
c.Value = True
```

Les fonctions TypeName et VarType

La fonction `TypeName(v)` retourne le nom du type de données que contient `v` :

```
Dim c As Range

Set c = Range("A1")

c.Value = "Master"

TypeName(c.Value) 'retourne String
```

Les fonctions TypeName et VarType

La fonction `varType(v)` retourne un entier représentant le type de valeurs que contient `v` :

```
Dim c As Range

Set c = Range("A1")

c.Value = "Master"

VarType(c.Value) 'retourne 8
```

Remarque. Il existe des constantes prédéfinies pour chaque type de données : `vbDouble`, `vbString`, etc ... Donc, pour savoir si une variable `v` contient un `Double` ou non, il suffit de faire le test

```
VarType(v) = vbDouble
```

La propriété `Formula`

La propriété `Formula` est une variable de type `String`.

Si la cellule contient une formule, la propriété `Formula` contient la formule sous la forme d'une chaîne de caractères commençant par le symbole =.

Remarque. Dans ce cas, la propriété `Value` est le résultat de l'évaluation de la formule.

Remarque. Si la cellule contient une valeur et non une formule, cette propriété prend comme valeur le résultat de la conversion de cette valeur en chaîne de caractères.

Un conseil. Si la cellule est vide ou ne contient pas de formule, n'utilisez pas la propriété `Formula`.

La propriété HasFormula

Pour savoir si une cellule contient ou non une formule, on utilise la propriété `HasFormula` de la cellule qui vaut `True` si la cellule contient une formule et `False` sinon.

```
If Range("A1").HasFormula Then
    MsgBox "la cellule A1 contient une formule"
Else
    MsgBox "la cellule A1 ne contient pas de formule"
End If
```

Attention. Une cellule vide ne contient pas de formule !

Utiliser une formule Excel

On peut utiliser une formule Excel pour donner une valeur à une cellule. Il suffit d'écrire la formule entre crochets.

```
Dim c As Range

Set c = [A6]

c.Value = [=SUM(A1:A5)]

Set c = [A7]

c.Value = [=2*A6]
```

La valeur de la cellule `A6` est égale à la somme des valeurs contenues dans la plage `A1:A5`.

La valeur de la cellule `A7` est le double de la valeur de la cellule `A6`.

Attention. On utilise les noms anglophones des fonctions Excel.

Ecrire une formule Excel

Pour écrire une formule dans une cellule, on écrit une chaîne de caractères contenant une formule Excel dans la propriété `Formula`.

```
Dim c As Range

Set c = [A9]

c.Formula = "=SUM(A1:A5)"
```

Attention. On utilise les noms anglophones des fonctions `Excel`.

Ecrire une formule Excel

Pour utiliser les noms francophones des fonctions `Excel`, les objets de type `Range` ont une autre propriété : la propriété `FormulaLocal` qui permet d'indiquer une formule en utilisant les noms en français des fonctions `Excel` :

```
Dim c As Range

Set c = [A9]

c.FormulaLocal = "=SOMME(A1:A5)"
```

Sommaire

Effacer le contenu d'une cellule

Pour effacer le contenu d'une cellule (valeur ou formule), on peut utiliser la méthode `ClearContents` :

```
Dim c As Range

Set c = [A1]

c.ClearContents
'la cellule A1 est désormais vide !
```

Sommaire

Connaître la référence d'une cellule

La méthode `Address` est une variable de type `string` contenant la référence, dite absolue (avec les dollars), d'une cellule.

```
Dim c As Range

Set c = [A1]

'c.Address = "$A$1"
```

Accéder aux propriétés associées au format d'une cellule

Attention. Ne pas confondre le contenu d'une cellule (valeur, formule) et le format d'une cellule (son apparence).

- la couleur de fond : `Interior.Color`. Il existe des couleurs prédéfinies : `vbBlack`, `vbRed`, `vbBlue`, `vbMagenta`, `vbYellow`,`vbCyan`,`vbGreen`.
- l'alignement horizontal du texte : `HorizontalAlignment` qui peut prendre une des valeurs `xlCenter`, `xlRight`, `xlLeft`.
- l'alignement vertical du texte : `VerticalAlignment` qui peut prendre une des valeurs `xlUp`, `xlCenter`, `xlBottom`, `xlTop`.
- la couleur du contenu : `Font.Color`.
- mettre en gras le contenu : `Font.Bold` qui prend la valeur `True` ou `False`
- mettre en italique le contenu : `Font.Italic` qui prend la valeur `True` ou `False`
- et bien d'autres ...

Effacer le formattage d'une cellule

A l'instar de la commande `clearContents`, il existe une méthode qui permet d'effacer le formattage d'une cellule (c'est-à-dire réinitialiser aux valeurs par défaut toutes les propriétés d'une cellule définissant sont format) : c'est la méthode `ClearFormats`.

```
Dim c As Range

Set c = [A1]

c.ClearFormats
```

Le bloc d'instructions `With ... End With`

Lorsqu'on modifie plusieurs propriétés d'une cellule on peut utiliser le bloc d'instructions `With ... End With` qui permet d'éviter de répéter la référence de l'objet lié à la cellule :

```
Dim c As Range

Set c = [A1]

With c

.Value = 6
    .Interior.Color = vbYellow
    .Font.Bold = True
    .HorizontalAlignment = xlCenter

End With
```

Attention. Un point « . » doit être mis devant chaque propriété !

Le bloc d'instructions With ... End With

Les instructions précédentes sont équivalentes à la séquence d'instructions suivantes :

```
Dim c As Range

Set c = [A1]

c.Value = 6

c.Interior.Color = vbYellow

c.Font.Bold = True

c.HorizontalAlignment = xlCenter
```

Le bloc d'instructions `With ... End With`

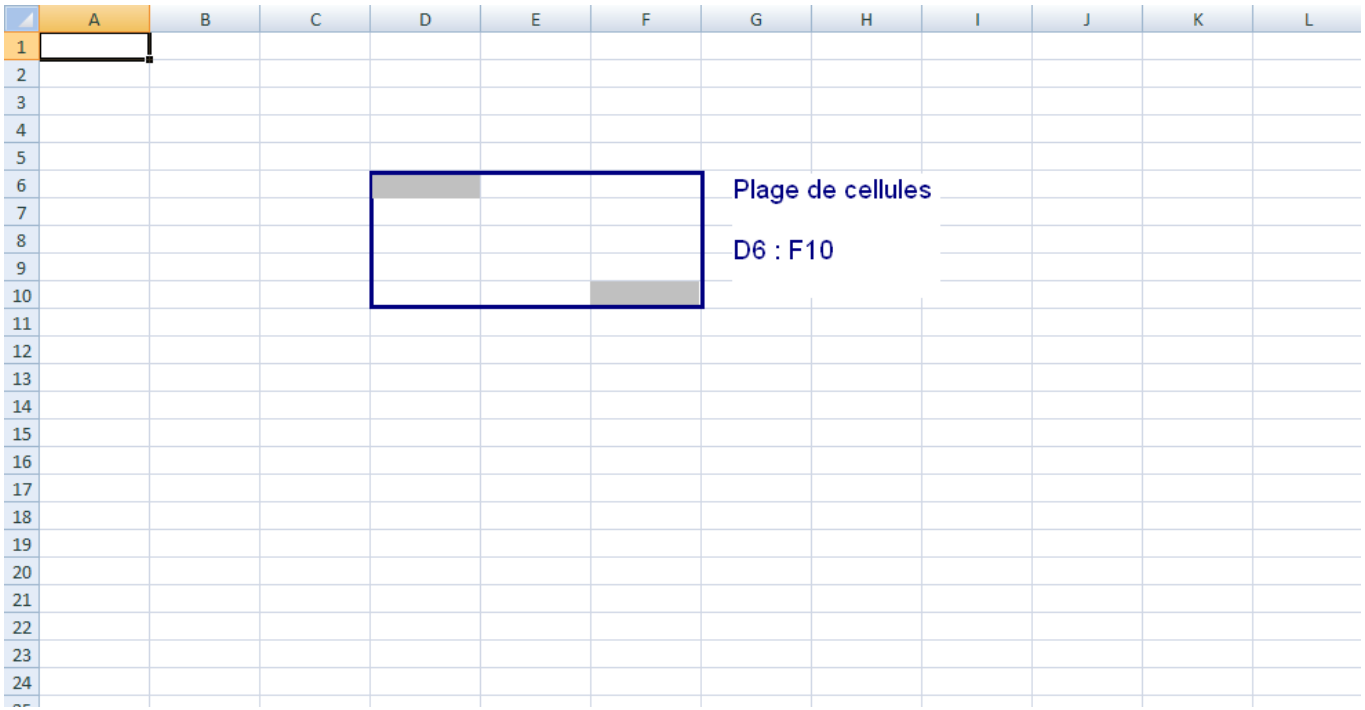
On peut aussi appeler des méthodes de l'objet dans un bloc d'instructions `With ... End With` :

```
Dim c As Range

Set c = [A1]

With c

.ClearContents
.ClearFormats

.Value = 6
    .Interior.Color = vbYellow
    .Font.Bold = True
    .HorizontalAlignment = xlCenter

End With
```

Sommaire

Plage de cellule

Une **plage de cellules contigues** est une zone rectangulaire de cellules contigues. Pour identifier une plage de cellules, on donne la référence du coin supérieur gauche et du coin inférieur gauche séparés par le caractère :



Associer un objet à une plage de cellule

Les objets associés aux plages de cellules sont aussi de type `Range`.

Remarque. Le langage VBA ne distingue pas les cellules des plages de cellules. Une cellule est vue en fait comme une plage de cellules ayant une seule cellule !

A l'instar des cellules, il y a deux manières d'associer un objet à une plage de cellules :

```
Dim p As Range

Set p = [A1:C5]

Set p = Range("A1:C5")
```

Dans les deux cas, la référence `p` contient l'adresse d'un objet associé à la plage de cellules `A1:C5`.

Une remarque importante

Les objets associés aux plages de cellule étant de type `Range`, ils possèdent toutes les propriétés enoncées dans les sections précédentes !

La manipulation de ces propriétés pour les plages de cellules sera abordée dans un autre chapitre.

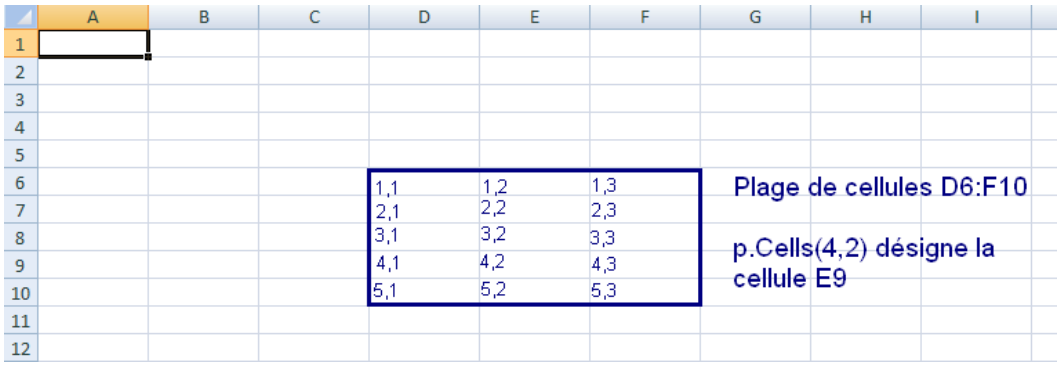
La seule que nous rappelerons est la propriété `Address` :

```
Dim p As Range

Set p = [A1:B9]

'p.Address = "$A$1:$B$9"
```

La propriété Cells

La collection des cellules que contient la plage de cellule est accessible via la propriété `Cells`. Pour accéder à une cellule particulière, on indique à quelle ligne `i` et à quelle colonne `j` elle se trouve :

```
p.Cells(i,j)
```



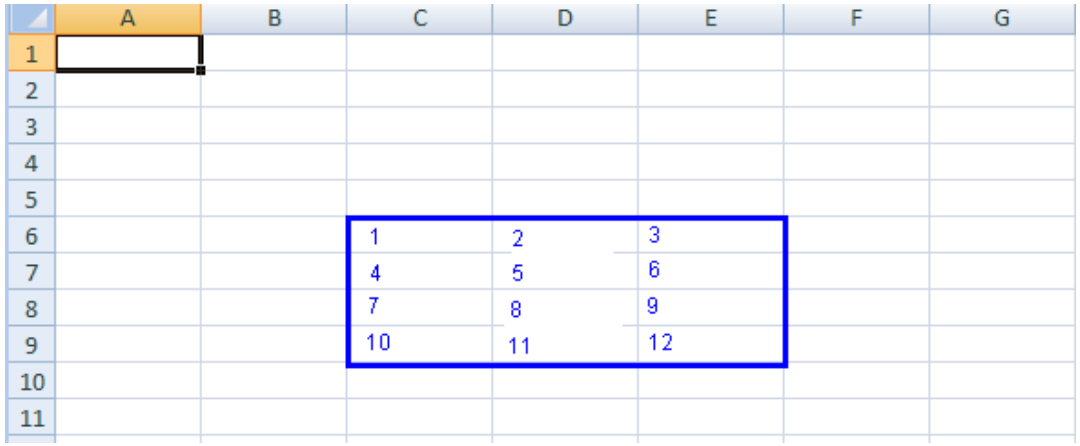
La collection Cells

Il existe une autre façon d'adresser une cellule d'une plage de cellules à partir du numéro de la cellule dans la plage

```
p.Cells(i)
```

désigne la *i*-ème cellule de la plage.

Attention, dans ce cas, l'indice ne correspond ni au numéro de la ligne ni au numéro de la colonne.



Nombre de cellules, de lignes, de colonne d'une plage de cellule

Etant donné une variable `r` associée une plage de cellule

```
Dim r as Range
Set r = Range("A1:C3")
```

on peut connaitre le nombre de cellules de la plage de cellule

```
MsgBox r.Count & " cellules dans la plage"
```

mais aussi le nombre de lignes et de colonnes

```
MsgBox r.Rows.Count & " lignes"
MsgBox r.Columns.Count & " colonnes"
```

Parcourir une plage de cellules

On peut utiliser les deux propriétés précédentes pour parcourir les cellules d'une plage de cellules.

```
Dim lig As Integer, col As Integer

Dim p As Range

Set p = [A1:B9]

For lig = 1 To p.Rows.Count

For col = 1 To p.Columns.Count

MsgBox p.Cells(lig,col)

Next col

Next lig
```

Parcourir une plage de cellules

On peut aussi parcourir la collection des cellules d'une plage à l'aide d'une boucle **For Each** :

```
Dim c As Range

For Each c In p.Cells

    MsgBox c.Value

Next c
```

Utiliser une fonction Excel pour faire un calcul sur une plage

On peut accéder à toutes les fonctions Excel à travers l'objet `Application.WorksheetFunction`.

Exemple.

```
    Dim p As Range, c As Range

    Set c = [A8]

    Set p = [A1:A5]

c.Value = Application.WorksheetFunction.Average(p)
```

La valeur de la cellule `A8` est alors la moyenne des valeurs contenus dans la plage `A1:A5`.