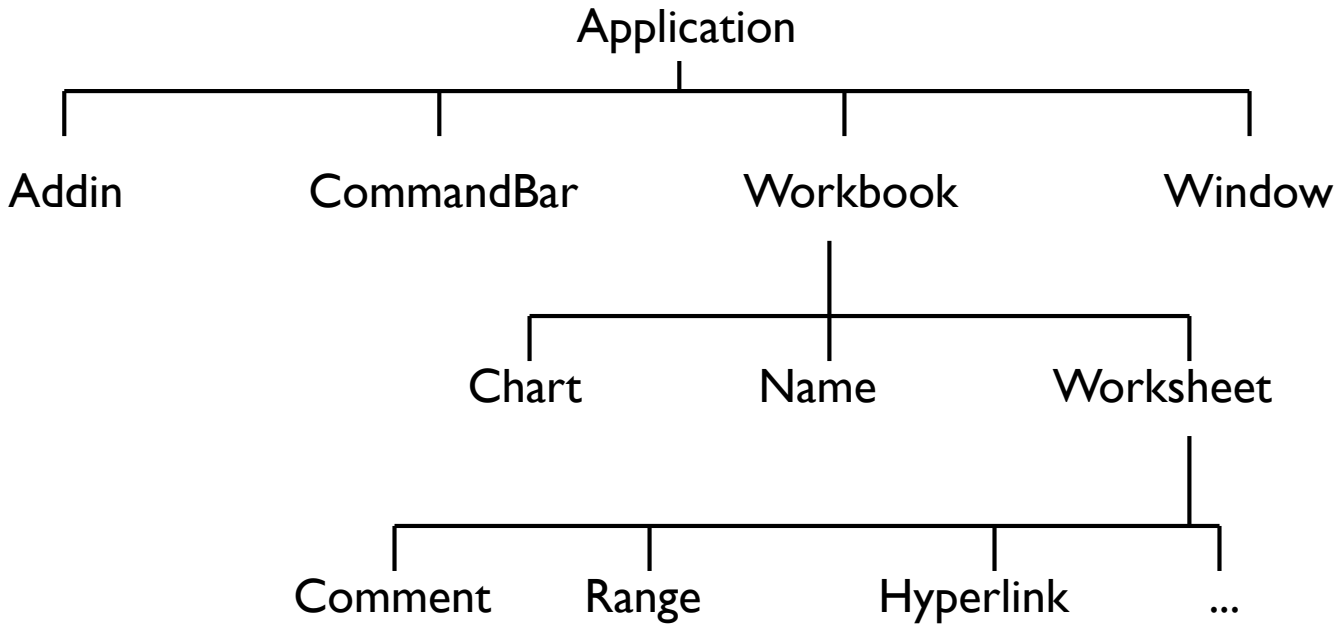
Excel
VBA programming
Visual Basic for Applications

Herve Hocquard

http://www.labri.fr/perso/hocquard

THE OBJECT MODEL IN VBA

- An object consists of attributes (or properties) and methods associated with it
- The existing objects are constituted in hierarchy (composition relation)



- Key concept
- We add an "s"!
  - Workbooks: collection of Workbook objects
  - Worksheets: Collection of Worksheet objects
  - … Etc.

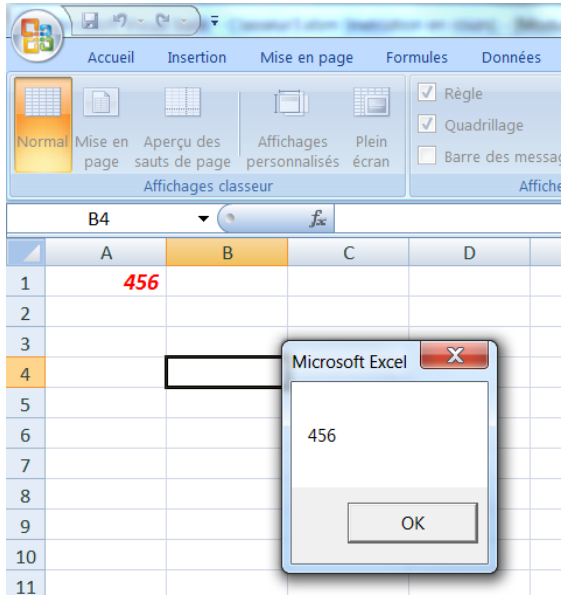
- Calling on an element of a collection: 2 methods:
  - Call by element name
    - Ex: Worksheets ("Sheet1")
  - Call by index
    - Ex: Worksheets (1)

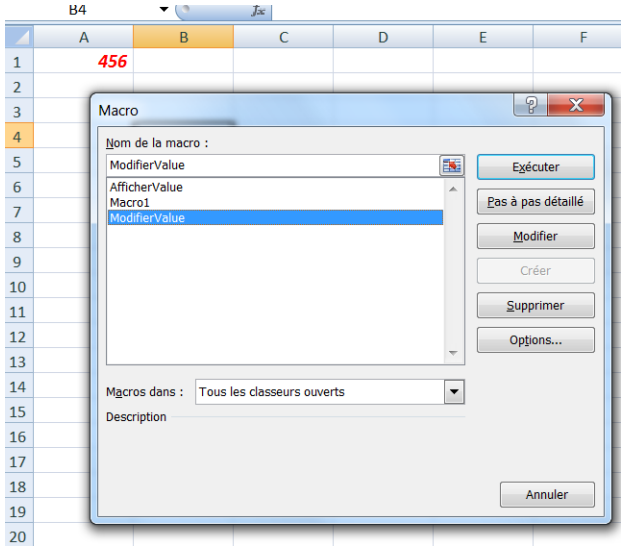
- **Dot operator (.)**
  - Example: Application.Workbooks("Workbook1.xlsx").Worksheets(1).Range ("A1").Value = 9
- **Simplification: for example if Workbook1.xlsx is the active workbook:**
  - Worksheets(1).Range ("A1").**Value**= 9

Properties of an object

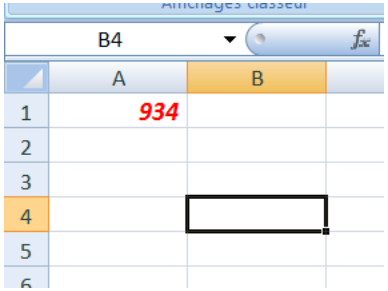
```
Sub AfficherValue()
'
Contents = Worksheets("Feuil1").Range("A1").Value
MsgBox Contents


    ActiveWorkbook.Save
End Sub
```






```
Sub ModifierValue()
    Worksheets("Feuil1").Range("A1").Value = 934
End Sub
```



- **Action relating to an object**

- **Examples:**
  - Worksheets("Sheet1").Activate
  - Range ("A1").Copy Range ("B1")

- **A method takes 0, 1 or more arguments.**
  - The first argument is separated from the method by a space, the arguments are separated between them by commas
  - OR use of parentheses

- **Objects:** VBA manipulates objects contained in its host application. (In this case, Excel is the host application.) Excel provides you with more than 100 classes of objects to manipulate.

  Examples of objects include a workbook, a worksheet, a range on a worksheet, a chart, and a shape. Many more objects are at your disposal, and you can use VBA code to manipulate them. Object classes are arranged in a hierarchy.

- **Objects** also can act as containers for other objects. For example, Excel is an object called **Application**, and it contains other objects, such as **Workbook objects**. The Workbook object contains other objects, such as **Worksheet objects** and **Chart objects**. A Worksheet object contains objects such as **Range objects**, **PivotTable objects**, and so on. The arrangement of these objects is referred to as Excel's object model.

- **Collections:** Like objects form a collection. For example, the Worksheets collection consists of all the worksheets in a particular workbook. Collections are objects in themselves.

- **Object hierarchy:** When you refer to an object, you specify its position in the object hierarchy by using a period (also known as a dot) as a separator between the container and the member. For example, you can refer to a workbook named Book1.xlsx as Application.Workbooks(''Book1.xlsx'')

  – This code refers to the Book1.xlsx workbook in the Workbooks collection. The Workbooks collection is contained in the Excel Application object. Extending this type of referencing to another level, you can refer to Sheet1 in Book1 as

    Application.Workbooks(''Book1.xlsx'').Worksheets(''Sheet1'')

  – You can take it to still another level and refer to a specific cell as follows:

    Application.Workbooks(''Book1.xlsx'').Worksheets(''Sheet1'').Range(''A1'')

- **Active objects:** If you omit a specific reference to an object, Excel uses the active objects. If Book1 is the active workbook, the preceding reference can be simplified as

  ```
  Worksheets(''Sheet1'').Range(''A1'')
  ```

  If you know that Sheet1 is the active sheet, you can simplify the reference even more:

  ```
  Range(''A1'')
  ```

- **Objects properties:** Objects have properties. A property can be thought of as a setting for an object. For example, a range object has properties such as Value and Address. A chart object has properties such as HasTitle and Type. You can use VBA to determine object properties and also to change them. Some properties are read-only properties and can't be changed by using VBA.

  You refer to properties by combining the object with the property, separated by a period. For example, you can refer to the value in cell A1 on Sheet1 as

  ```
  Worksheets(''Sheet1'').Range(''A1'').Value
  ```

- **VBA variables:** You can assign values to VBA variables. Think of a variable as a name that you can use to store a particular value. To assign the value in cell A1 on Sheet1 to a variable called Interest, use the following VBA statement:

```
Interest = Worksheets(''Sheet1'').Range(''A1'').Value
```

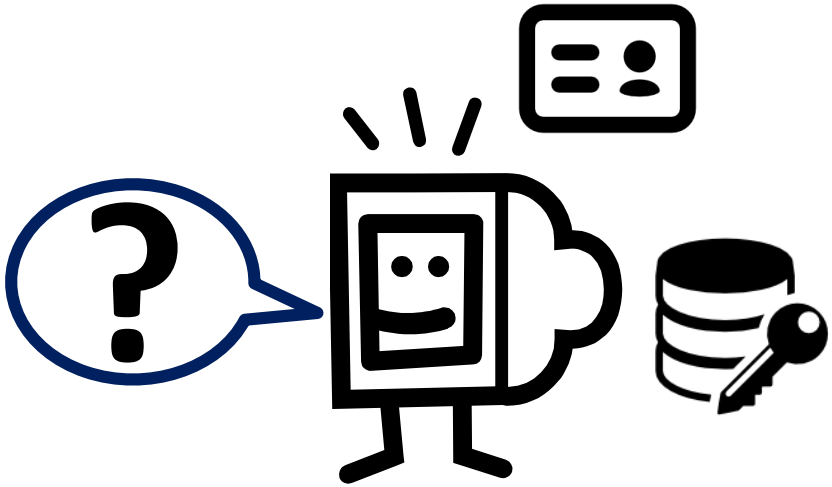
- **Object methods:** Objects have methods. A method is an action that is performed with the object. For example, one of the methods for a Range object is ClearContents. This method clears the contents of the range. You specify methods by combining the object with the method, separated by a period. For example, to clear the contents of cell A1 on the active worksheet, use

```
Range(''A1'').ClearContents
```

- **Standard programming constructs:** VBA also includes many constructs found in modern programming languages, including arrays, conditional statements, and loops.

- **Events:** Some objects recognize specific events, and you can write VBA code that is executed when the event occurs. For example, opening a workbook triggers a Workbook_Open event. Changing a cell in a worksheet triggers a Worksheet_Change event.

- Believe it or not, the preceding section pretty much summarizes what VBA is all about and how it works with Excel. Now you just need to learn the details…



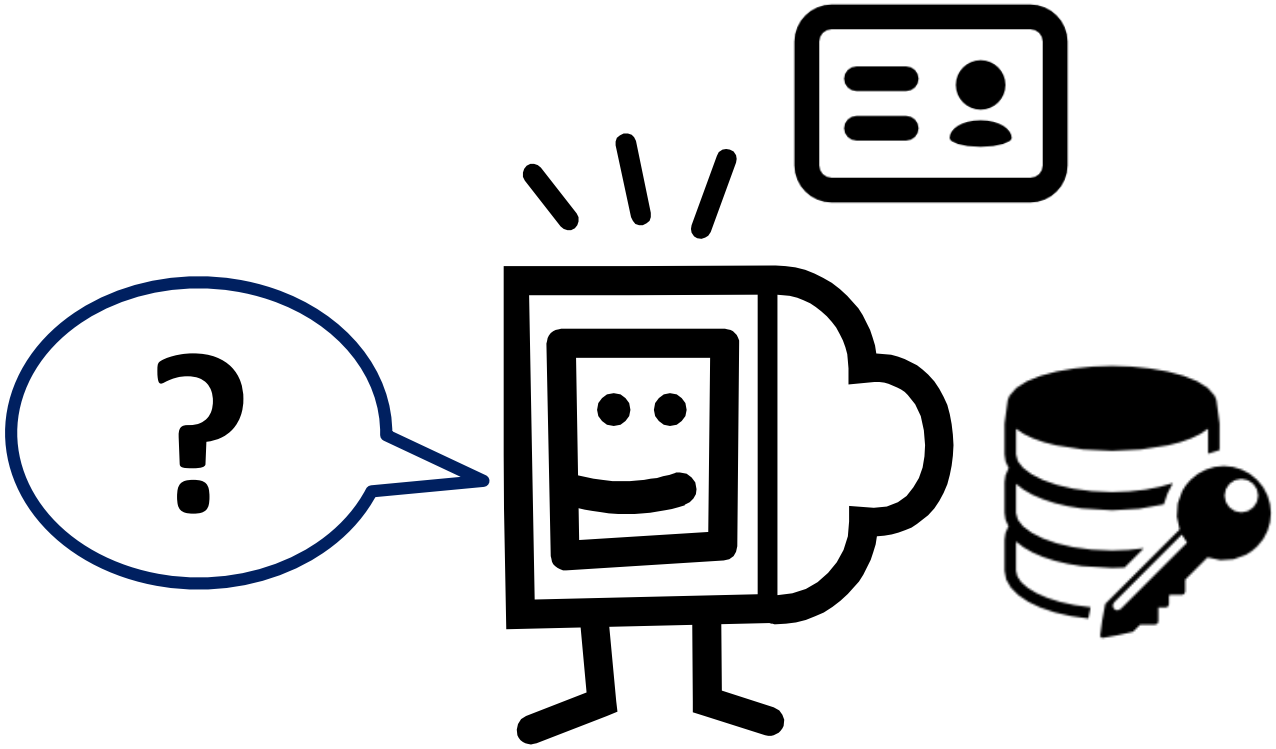
In this analogy, I compare Excel with a fast-food restaurant chain.

The basic unit of Excel is a Workbook object. In a fast-food chain, the basic unit is an individual restaurant. With Excel, you can add workbooks and close workbooks, and the set of all the open workbooks is known as Workbooks (a collection of Workbook objects). Similarly, the management of a fast-food chain can add restaurants and close restaurants — and all the restaurants in the chain can be viewed as the Restaurants collection — a collection of Restaurant objects.

An Excel workbook is an object, but it also contains other objects, such as worksheets, charts, VBA modules, and so on. Furthermore, each object in a workbook can contain its own objects. For example, a Worksheet object can contain Range objects, PivotTable objects, Shape objects, and so on. Continuing with the analogy, a fast-food restaurant (like a workbook) contains objects, such as the Kitchen, DiningArea, and Tables (a collection). Furthermore, management can add or remove objects from the Restaurant object. For example, management can add more tables to the Tables collection. Each of these objects can contain other objects. For example, the Kitchen object has a Stove object, a VentilationFan object, a Chef object, a Sink object, and so on. So far, so good. This analogy seems to work. Let’s see whether I can take it further.

Excel objects have properties. For example, a Range object has properties such as Value and Name, and a Shape object has properties such as Width and Height. Not surprisingly, objects in a fast-food restaurant also have properties. The Stove object, for example, has properties such as Temperature and NumberofBurners. The VentilationFan object has its own set of properties (TurnedOn, RPM, and so on). Besides properties, Excel's objects also have methods, which perform operations on objects. For example, the ClearContents method erases the contents of a Range object.

An object in a fast-food restaurant also has methods. You can easily envision a ChangeThermostat method for a Stove object, or a SwitchOn method for a VentilationFan object. With Excel, methods sometimes change an object's properties. The ClearContents method for a Range object changes the Range Value property. Similarly, the ChangeThermostat method on a Stove object affects its Temperature property. With VBA, you can write procedures to manipulate Excel's objects. In a fast-food restaurant, the management can give orders to manipulate the objects in the restaurants. ("Turn on the stove and switch the ventilation fan to high.")



To be continued with VBA Language…

Thank you

Herve Hocquard(hocquard@labri.fr)

http://www.labri.fr/perso/hocquard/Teaching.html