What is an Object in VBA?

What is an Object?

What is an Object?

In VBA, an Object is essentially a "thing" that you can manipulate through your code. These "things" are components of Excel, like workbooks, worksheets, charts, or ranges. Application, Workbooks, Worksheets, and Range are the common Objects we use while coding.

Applications of Objects in VBA

Interacting with Excel Components: Objects allow VBA to interact with Excel, letting you automate tasks like formatting a range, copying data, or generating charts.

Building Flexible Code: Objects make your code modular and adaptable. For example, you can write a macro that works with any workbook by referencing objects dynamically instead of hardcoding values.

Objects have three primary aspects:

Properties: Properties describe the characteristics or attributes of an object. Each Object has its specific characteristics.
Example: For a Range object representing a cell, properties include:
• Value (the value of the cell) > `Range("A1").Value = "Hello"`
• Font (the font style and size) > `Range("A1").Font.Bold = True`

Methods: Methods are the actions or operations that an object can perform. Methods allow you to perform actions on objects.
Example: For a Workbook object, methods include:
• Save (saves the workbook) > `Workbooks("Report.xlsx").Save`
• Close (closes the workbook) > `Workbooks("Report.xlsx").Close`

Event: Events are triggers that occur when something happens to an object or specific triggers that objects respond to. Events enable dynamic behavior in your VBA code by responding to user actions or system processes.
Example: A Worksheet object has events such as:
• Worksheet_Change (triggered when a cell's value is changed) >
```
Private Sub Worksheet_Change(ByVal Target As Range)
        MsgBox "You changed the value in " & Target.Address
End Sub
```

Hierarchy of Objects

Objects in Excel are organized in a hierarchical structure, called the **Object Model**. The hierarchy starts from the top-level Excel application and narrows down to specific components like cells/ranges. This structure ensures you can access any part of Excel logically.

Common Object Hierarchy:
1. Application: Represents the Excel application itself.
Example: `Application.ActiveWorkbook` > refers to the currently active workbook.
2. Workbook: Represents an Excel workbook.
Example: `Workbooks("Budget.xlsx")` > refers to a specific workbook named "Budget.xlsx"
3. Worksheet: Represents a sheet within a workbook.
Example: `Workbooks("Budget.xlsx").Worksheets("Sheet1")` > accesses a specific worksheet (Sheet1) of Budget.xlsx
4. Range: Represents a cell or group of cells within a worksheet.
Example: `Workbooks("Budget.xlsx").Worksheets("Sheet1").Range("A1")` > refers to cell A1 in Sheet1 of Budget.xlsx

How to Work with Objects in VBA

Use Existing Objects

Referencing Objects: Use dot notation to navigate the object hierarchy and access specific objects, properties, or methods.

Create New Objects

Creating and Managing Objects: New Objects can be created as variables

Existing types: You can create a new object as a Workbook, etc.
Example: Create a new workbook:
```
Dim MyWB As Workbook
Set MyWB = Workbooks.Add
```
To release an object reference:
```
Set MyWB = Nothing
```