

# Excel® VBA

## Notes for Professionals

### Chapter 10: Workbooks

#### Section 10.1: When To Use ActiveWorkbook and ThisWorkbook

It's a VBA Best Practice to **always** specify which workbook your VBA code refers. If this specification is omitted, then VBA assumes the code is directed at the currently active workbook (ActiveWorkbook).

```
--- the currently active workbook (and worksheet) is implied  
Range("A1").value = 3.1415  
Cells(1, 1).value = 3.1415
```

However, when several workbooks are open at the same time – particularly and especially when VBA code is running from an Excel Add-In – references to the ActiveWorkbook may be confused or misdirected. For example, an add-in with a UDF that checks the time of day and compares it to a value stored on one of the add-in's worksheets (that are typically not readily visible to the user) will have to explicitly identify which workbook is being referenced. In our example, our open (and active) workbook has a formula in cell A1 ("EarlyOrLate") and does NOT have any VBA written for that active workbook. In our add-in, we have the following User Defined Function (UDF):

```
Public Function EarlyOrLate() As String  
    If Hour(Time) > ThisWorkbook.Sheets("WatchTime").Range("A1") Then  
        EarlyOrLate = "It's Late!"  
    Else  
        EarlyOrLate = "It's Early!"  
    End If  
End Function
```

The code for the UDF is written and stored in the installed Excel add-in. It uses data stored on a worksheet in the add-in called "WatchTime". If the UDF had used ActiveWorkbook instead of ThisWorkbook, then it would never be able to guarantee which workbook was intended.

#### Section 10.2: Changing The Default Number of Worksheets in A New Workbook

The "factory default" number of worksheets created in a new Excel workbook is generally set to three. Your VBA code can explicitly set the number of worksheets in a new workbook.

```
--- save the current Excel global setting  
With Application  
    Dim oldSheetsCount As Integer  
    oldSheetsCount = SheetsInNewWorkbook  
    Dim myNewWB As Workbook  
    SheetsInNewWorkbook = 1  
    Set myNewWB = Workbooks.Add  
    --- restore the previous setting  
    SheetsInNewWorkbook = oldSheetsCount  
End With
```

#### Section 10.3: Application Workbooks

In many Excel applications, the VBA code takes actions directed at the workbook in which it's contained. Yet that workbook with a ".xlsm" extension and the VBA macros only focus on the worksheets and data WITHIN. However, there are often times when you need to combine or merge data from other workbooks, or write your data to a separate workbook. Opening, closing, saving, creating, and deleting other workbooks is a common need for many VBA applications.

Excel® VBA Notes for Professionals

### Chapter 12: Loop through all Sheets in Active Workbook

#### Section 12.1: Retrieve all Worksheets Names in Active Workbook

Option Explicit

```
Sub LoopAllSheets()  
    ---  
End Sub
```

```
Dim wb As Excel.Workbook  
--- declare an array of type String without committing to maximum number of members  
Dim strNames() As String  
Dim i As Integer
```

```
--- get the number of worksheets in Active Workbook, and put it as the maximum number of members in the array  
ReDim strNames(1 To ActiveWorkbook.Worksheets.Count)
```

```
i = 1
```

```
--- loop through all worksheets in Active Workbook  
For Each wb In ActiveWorkbook.Worksheets
```

```
    strNames(i) = wb.Name  
    i = i + 1  
Next wb
```

```
--- get the name of each worksheet and save it in the array  
End Sub
```

#### Section 12.2: Loop Through all Sheets in all Files in a Folder

```
Sub TheLoopOfFile()  
    ---  
End Sub
```

```
Dim wb As Workbook  
Dim filename As String  
Dim path As String  
Dim rng As Range  
Dim ws As Worksheet  
Dim sheet As Worksheet
```

```
path = "pathof(files)" & "\*"  
filename = Dir(path & "*.*")  
Set wb = ThisWorkbook.Sheets("sheet") 'included in case you need to differentiate  
--- between workbooks i.e. currently opened workbook vs workbook containing code
```

```
Do While Len(filename) > 0
```

```
    Dim ws As Worksheet
```

```
    Set ws = Worksheets.Open(path & filename, True, True)
```

```
    For Each sheet In ActiveWorkbook.Worksheets 'this needs to be adjusted for identifying  
        sheet.Range("A1:A1000").Clear 'clear sheet data
```

```
    For Each rng In sheet.Range("A1:A1000").UsedRange  
        If rng.Value = " " And rng.Column = 1 Then  
            rng.Value = " " 'add that does stuff
```

```
        End If  
    Next rng  
Next sheet
```

```
End While
```

```
End Sub
```

Excel® VBA Notes for Professionals

**100+ pages**  
of professional hints and tricks

# Contents

<b>About</b>	1
<b>Chapter 1: Getting started with Excel VBA</b>	2
Section 1.1: Opening the Visual Basic Editor (VBE)	3
Section 1.2: Declaring Variables	5
Section 1.3: Adding a new Object Library Reference	6
Section 1.4: Hello World	10
Section 1.5: Getting Started with the Excel Object Model	12
<b>Chapter 2: Arrays</b>	16
Section 2.1: Dynamic Arrays (Array Resizing and Dynamic Handling)	16
Section 2.2: Populating arrays (adding values)	16
Section 2.3: Jagged Arrays (Arrays of Arrays)	17
Section 2.4: Check if Array is Initialized (If it contains elements or not)	17
Section 2.5: Dynamic Arrays [Array Declaration, Resizing]	17
<b>Chapter 3: Conditional statements</b>	19
Section 3.1: The If statement	19
<b>Chapter 4: Ranges and Cells</b>	21
Section 4.1: Ways to refer to a single cell	21
Section 4.2: Creating a Range	21
Section 4.3: Offset Property	23
Section 4.4: Saving a reference to a cell in a variable	23
Section 4.5: How to Transpose Ranges (Horizontal to Vertical & vice versa)	23
<b>Chapter 5: Named Ranges</b>	25
Section 5.1: Define A Named Range	25
Section 5.2: Using Named Ranges in VBA	25
Section 5.3: Manage Named Range(s) using Name Manager	26
Section 5.4: Named Range Arrays	28
<b>Chapter 6: Merged Cells / Ranges</b>	29
Section 6.1: Think twice before using Merged Cells/Ranges	29
<b>Chapter 7: Locating duplicate values in a range</b>	30
Section 7.1: Find duplicates in a range	30
<b>Chapter 8: User Defined Functions (UDFs)</b>	32
Section 8.1: Allow full column references without penalty	32
Section 8.2: Count Unique values in Range	33
Section 8.3: UDF - Hello World	33
<b>Chapter 9: Conditional formatting using VBA</b>	36
Section 9.1: FormatConditions.Add	36
Section 9.2: Remove conditional format	37
Section 9.3: FormatConditions.AddUniqueValues	37
Section 9.4: FormatConditions.AddTop10	38
Section 9.5: FormatConditions.AddAboveAverage	38
Section 9.6: FormatConditions.AddIconSetCondition	38
<b>Chapter 10: Workbooks</b>	41
Section 10.1: When To Use ActiveWorkbook and ThisWorkbook	41
Section 10.2: Changing The Default Number of Worksheets In A New Workbook	41
Section 10.3: Application Workbooks	41
Section 10.4: Opening A (New) Workbook, Even If It's Already Open	42

Section 10.5: Saving A Workbook Without Asking The User .....	43
<b>Chapter 11: Working with Excel Tables in VBA .....</b>	<b>44</b>
Section 11.1: Instantiating a ListObject .....	44
Section 11.2: Working with ListRows / ListColumns .....	44
Section 11.3: Converting an Excel Table to a normal range .....	44
<b>Chapter 12: Loop through all Sheets in Active Workbook .....</b>	<b>45</b>
Section 12.1: Retrieve all Worksheets Names in Active Workbook .....	45
Section 12.2: Loop Through all Sheets in all Files in a Folder .....	45
<b>Chapter 13: Use Worksheet object and not Sheet object .....</b>	<b>47</b>
Section 13.1: Print the name of the first object .....	47
<b>Chapter 14: Methods for Finding the Last Used Row or Column in a Worksheet .....</b>	<b>48</b>
Section 14.1: Find the Last Non-Empty Cell in a Column .....	48
Section 14.2: Find the Last Non-Empty Row in Worksheet .....	48
Section 14.3: Find the Last Non-Empty Column in Worksheet .....	49
Section 14.4: Find the Last Non-Empty Cell in a Row .....	50
Section 14.5: Get the row of the last cell in a range .....	50
Section 14.6: Find Last Row Using Named Range .....	50
Section 14.7: Last cell in Range.CurrentRegion .....	51
Section 14.8: Find the Last Non-Empty Cell in Worksheet - Performance (Array) .....	51
<b>Chapter 15: Creating a drop-down menu in the Active Worksheet with a Combo Box .....</b>	<b>54</b>
Section 15.1: Example 2: Options Not Included .....	54
Section 15.2: Jimi Hendrix Menu .....	55
<b>Chapter 16: File System Object .....</b>	<b>57</b>
Section 16.1: File, folder, drive exists .....	57
Section 16.2: Basic file operations .....	57
Section 16.3: Basic folder operations .....	58
Section 16.4: Other operations .....	58
<b>Chapter 17: Pivot Tables .....</b>	<b>60</b>
Section 17.1: Adding Fields to a Pivot Table .....	60
Section 17.2: Creating a Pivot Table .....	60
Section 17.3: Pivot Table Ranges .....	63
Section 17.4: Formatting the Pivot Table Data .....	63
<b>Chapter 18: Binding .....</b>	<b>64</b>
Section 18.1: Early Binding vs Late Binding .....	64
<b>Chapter 19: autofilter ; Uses and best practices .....</b>	<b>66</b>
Section 19.1: Smartfilter! .....	66
<b>Chapter 20: Application object .....</b>	<b>70</b>
Section 20.1: Simple Application Object example: Display Excel and VBE Version .....	70
Section 20.2: Simple Application Object example: Minimize the Excel window .....	70
<b>Chapter 21: Charts and Charting .....</b>	<b>71</b>
Section 21.1: Creating a Chart with Ranges and a Fixed Name .....	71
Section 21.2: Creating an empty Chart .....	72
Section 21.3: Create a Chart by Modifying the SERIES formula .....	73
Section 21.4: Arranging Charts into a Grid .....	75
<b>Chapter 22: CustomDocumentProperties in practice .....</b>	<b>79</b>
Section 22.1: Organizing new invoice numbers .....	79
<b>Chapter 23: PowerPoint Integration Through VBA .....</b>	<b>82</b>
Section 23.1: The Basics: Launching PowerPoint from VBA .....	82

<b>Chapter 24: How to record a Macro</b>	83
Section 24.1: How to record a Macro	83
<b>Chapter 25: SQL in Excel VBA - Best Practices</b>	85
Section 25.1: How to use ADODB.Connection in VBA?	85
<b>Chapter 26: Excel-VBA Optimization</b>	87
Section 26.1: Optimizing Error Search by Extended Debugging	87
Section 26.2: Disabling Worksheet Updating	88
Section 26.3: Row Deletion - Performance	88
Section 26.4: Disabling All Excel Functionality Before executing large macros	89
Section 26.5: Checking time of execution	90
Section 26.6: Using With blocks	91
<b>Chapter 27: VBA Security</b>	93
Section 27.1: Password Protect your VBA	93
<b>Chapter 28: Debugging and Troubleshooting</b>	94
Section 28.1: Immediate Window	94
Section 28.2: Use Timer to Find Bottlenecks in Performance	95
Section 28.3: Debugger Locals Window	95
Section 28.4: Debug.Print	96
Section 28.5: Stop	97
Section 28.6: Adding a Breakpoint to your code	97
<b>Chapter 29: VBA Best Practices</b>	98
Section 29.1: ALWAYS Use "Option Explicit"	98
Section 29.2: Work with Arrays, Not With Ranges	100
Section 29.3: Switch off properties during macro execution	101
Section 29.4: Use VB constants when available	102
Section 29.5: Avoid using SELECT or ACTIVATE	103
Section 29.6: Always define and set references to all Workbooks and Sheets	105
Section 29.7: Use descriptive variable naming	105
Section 29.8: Document Your Work	106
Section 29.9: Error Handling	107
Section 29.10: Never Assume The Worksheet	109
Section 29.11: Avoid re-purposing the names of Properties or Methods as your variables	109
Section 29.12: Avoid using ActiveCell or ActiveSheet in Excel	110
Section 29.13: WorksheetFunction object executes faster than a UDF equivalent	111
<b>Chapter 30: Excel VBA Tips and Tricks</b>	113
Section 30.1: Using xlVeryHidden Sheets	113
Section 30.2: Using Strings with Delimiters in Place of Dynamic Arrays	114
Section 30.3: Worksheet .Name, .Index or .CodeName	114
Section 30.4: Double Click Event for Excel Shapes	116
Section 30.5: Open File Dialog - Multiple Files	117
<b>Chapter 31: Common Mistakes</b>	118
Section 31.1: Qualifying References	118
Section 31.2: Deleting rows or columns in a loop	119
Section 31.3: ActiveWorkbook vs. ThisWorkbook	119
Section 31.4: Single Document Interface Versus Multiple Document Interfaces	120
<b>Credits</b>	122
<b>You may also like</b>	124

# About

Please feel free to share this PDF with anyone for free,  
latest version of this book can be downloaded from:  
<https://goalkicker.com/ExcelVBABook>

This *Excel® VBA Notes for Professionals* book is compiled from [Stack Overflow Documentation](#), the content is written by the beautiful people at Stack Overflow. Text content is released under Creative Commons BY-SA, see credits at the end of this book whom contributed to the various chapters. Images may be copyright of their respective owners unless otherwise specified

This is an unofficial free book created for educational purposes and is not affiliated with official Excel® VBA group(s) or company(s) nor Stack Overflow. All trademarks and registered trademarks are the property of their respective company owners

The information presented in this book is not guaranteed to be correct nor accurate, use at your own risk

Please send feedback and corrections to [web@petercv.com](mailto:web@petercv.com)

[Click here to download full PDF material](#)