Calling Worksheet Functions In VBA

Nearly all worksheet functions can be called from VBA using the Application or Application.Worksheet objects. Excel functions that have native VBA equivalents, such as Month, are not available. The syntax of a worksheet function call is the same as worksheet function itself. For example, a worksheet function in a cell might be:  
  
=VLOOKUP(123,A1:C100,3,FALSE)  
  
To use code in VBA that does the same thing, you would use:

Dim Res As Variant

Res = Application.WorksheetFunction.VLookup(123,Range("A1:C100"),3,FALSE)

The number of parameters and their meanings are the same when calling the function from VBA as they are when calling the function from a worksheet cell. As the code above is written, you will get a runtime error if the value 123 is not found in the range. Therefore, you need to put in some error trapping code:

Dim Res As Variant

On Error Resume Next

Err.Clear

Res = Application.WorksheetFunction.VLookup(123,Range("A1:C100"),3,FALSE)

If Err.Number = 0 Then

''''''''''''''''''''''''''''''''

' Value was found. Continue normal code execution

''''''''''''''''''''''''''''''''

Else

''''''''''''''''''''''''''''''''

' Value was not found. Error code goes here.

''''''''''''''''''''''''''''''''

End If

SectionBreak

Error Handling With Worksheet Functions

This brings us to the topic of error handling when calling worksheet functions in VBA. As noted earlier, there are two basic syntaxes you can use. You can either use an On Error statement and then test the Err.Number value to see if an error occurred, or you can declare the result variable as a Variant type and use IsError to see if that variable is an Error type variant. Which method you use depends on whether you use the WorksheetFunction property in your code. If you do include the WorksheetFunction property, errors will manifest themselves as runtime error that need an On Error statement and a test of the Err.Number value. If you do not include the WorksheetFunction property, you must declare the result variable as a Variant type and test that variable with the IsError function. Examples of both are shown below.

''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Runtime error trapping

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Dim Res As Variant

On Error Resume Next

Err.Clear

Res = Application.WorksheetFunction.VLookup(123,Range("A1:C100"),2,False)

If Err.Number = 0 Then

''''''''''''''''''''''''''''''''''''''''''''''''''''

' Value found by VLookup. Continue normal execution.

'''''''''''''''''''''''''''''''''''''''''''''''''''''

Else

''''''''''''''''''''''''''''''''''''''''''''''''''''

' Value NOT found by VLookup. Error handling code here.

'''''''''''''''''''''''''''''''''''''''''''''''''''''

End If

''''''''''''''''''''''''''''''''''''''''''''''''''''''''

' Using IsError to detect errors

'''''''''''''''''''''''''''''''''''''''''''''''''''''''''

Dim Res As Variant

Res = Application.VLookup(123,Range("A1:C100"),2,False)

If IsError(Res) = False Then

''''''''''''''''''''''''''''''''''''''''''''''''''''

' Value found by VLookup. Continue normal execution.

'''''''''''''''''''''''''''''''''''''''''''''''''''''

Else

''''''''''''''''''''''''''''''''''''''''''''''''''''

' Value NOT found by VLookup. Error handling code here.

'''''''''''''''''''''''''''''''''''''''''''''''''''''

End If

There is no significant difference between the two syntaxes of calling worksheet function. It is largely a matter of personal preference whether to use the WorksheetFunction property. In general, I omit it and use IsError to detect an error.

SectionBreak

Calling Analysis Tool Pack Function In VBA

Prior to Excel 2007, the functions provided by the Analysis Tool Pack (ATP) are provided by a separate add-in. You cannot call them using the methods shown above. First, you must load the *Analysis Tool Pack - VBA* add-in. Note that this is different from the *Analysis Tool Pack* item. Once this add-in is loaded, go to VBA, open your project, and choose *References* from the *Tools* menu. In that dialog, choose *atpvbaen.xls* in the list of references. Once you have that reference in place in your VBA code, you can call the functions in the ATP as if they were native VBA functions. For example,

Dim Res As Variant

Res = MRound(123.456, 0.5)

Debug.Print Res

If there is the possibility that you will have a procedure with the same name as an ATP function, you can prefix the function name with the library name to ensure you are calling the correct function. For example,

Dim Res As Variant

Res = [atpvbaen.xls].MRound(123.456, 0.5)

Debug.Print Res

Note that the square brackets ([ ]) are required in the library name because the name contains a period. The name of the add-in is an abbreviation of Analysis Tool Pack VBA ENglish.. If you are using a language other than English, the last two characters of the add-in name will be the language name abbreviation.