**COMWrapperBuilder tool.**

It creates SimpleVariantPlus.bi to make working with COM easier. SimpleVariantPlus.bi is based on OSchmidt's SimpleVariant.bi.  
  
For this tool to work, you need:  
1. Add #include "SimpleVariantPlus.bi" into one of the modules of the project or a single file.  
The tool creates "SimpleVariantPlus.bi" itself if it is not available, otherwise it overwrites.  
2. Select the path to the project or a single file in graphical mode or specify in the command line.  
Has two switches on the command line:  
-p Path to the project  
-s Path to a single file  
3. In the graphical mode you need to click on the button Run. If passed to the command line, the tool automatically works when the tool is loaded  
At the end, reports on the result of the work.  
Can be used in IDEs that support Tools. There, the command lines are set in the settings and you will not manually specify the path. Only you will choose a tool from the menu.  
4. To test, you must compile your code and run your program.

OSchmidt's example can be represented as follows

Code: [Select all](https://freebasic.net/forum/viewtopic.php?f=8&t=28739)

#define UNICODE  
#include once "SimpleVariantPlus.bi"  
  
'Whilst the other examples so far, were creating InProcess-ObjectInstances -  
'the following snippet covers an OutOfProcess-COMServer ("Excel.Application")  
MsgBox "It might take a few seconds before something happens" & Chr(10) & \_  
"(Excel-Startups are not the fastest, especially on a 'cold FileCache')"  
  
ShowCOMErrors = False 'we check the COMErr-Variable (and show an Error-MsgBox when needed) ourselves  
Dim xlApp As vbVariant = CreateObject("Excel.Application")  
If COMErr.Number Then MsgBox COMErr.Description: End ' early exit on Machines without an Excel-Install  
ShowCOMErrors = True 'for the rest of the snippet, we switch the built-in COMErr-Messaging back On  
  
xlApp.WorkBooks.Add  
xlApp.Visible = True  
  
Dim xlSheet As vbVariant = xlApp.ActiveSheet, CellValue As vbVariant  
  
With xlSheet.Range("A1") 'let's operate from the TopLeft-cell  
     
   Dim SArr(0 To 2) As String = {"Hello", "COM-calls", "from FB"}  
   For i As Integer = 0 To UBound(SArr)  
      .Offset(0, i).Value = SArr(i)  
   Next  
     
   CellValue = .Offset(0, 1).Value '<- should return "COM-calls"  
     
End With '<- at this point, the above instantiated A1-Range-Object will be destroyed implicitly  
  
xlSheet.Parent.Saved = True 'let's suppress the "Save Document" Dialogue ...  
xlSheet.Clear '... and destroy the Sheet-Object with the appropriate vbVariant.Method ...  
  
MsgBox CellValue '... now show the CellValue we have retrieved (should contain "COM-calls") ...  
  
xlApp.Quit '...  before we "call it quits" here

You can write code like you work in VB without additional variables in functions, at the end you can run this tool. This tool creates all the necessary methods by analyzing your code.  
  
For example, you can write like this:

Code: [Select all](https://freebasic.net/forum/viewtopic.php?f=8&t=28739)

#include "SimpleVariantPlus.bi"  
  
Dim As vbVariant wordApp  
  
wordApp = CreateObject("Word.Application")  
wordApp.Documents.Add  
wordApp.Visible = True  
  
Sleep

And this code will work after running this tool.