



Excel VBA 程序代码库

46

完美 Excel 出品
微信公众号: *excelperfect*



内容提要

Excel VBA 能够快速实现 Excel 操作的自动化，甚至帮助我们打造一套便捷的管理系统，而这些都是由一段段程序代码驱动的。

本电子书《**Excel VBA 程序代码库**》收集整理了 46 个能够实现某些功能的 VBA 程序代码，它们或者能够扩展 Excel 的功能，或者能够提高 Excel 的使用效率。读者可以直接使用这些代码，也可以研读并实践这些代码，从而快速提升 VBA 应用能力。



目录

1. 列出名称及相应的引用	1
2. 计算任意公式的自定义函数	5
3. 打开并处理指定文件夹中的所有工作簿	9
4. 在用户窗体标题栏添加最大化和最小化按钮	11
5. 列出工作簿中的所有公式	15
6. 在工作表中自动添加形状	19
7. 整齐排列工作表中的图表	23
8. 创建并自动排列图表	27
9. 加密解密数据	35
10. 获取文本字符串中的文本或者数字	39
11. 通过右键快捷菜单导航工作簿和工作表	41
12. 删除工作簿中除指定工作表之外的所有工作表	45
13. 记录单元格中存放过的数据	47
14. 处理文件名的自定义函数	51
15. 将 Excel 图表导出为图片	55
16. 将 Excel 图表导出为图片（增强版）	59
17. 从关闭的工作簿中取值	65
18. 合并文件夹中的所有工作簿	67



19.合并工作簿中的所有工作表	69
20.保护含有公式的单元格	73
21.项目管理 根据工作分解结构自动编号	75
22.查找工作表中含有指定日期的所有单元格	81
23.查找并返回满足条件的多个值	83
24.设置页眉/页脚的高效代码	85
25.删除工作表中的所有形状	87
26.将工作簿生成 PDF 文件	89
27.在 Excel 中创建可视化的标签云	93
28.根据编号自动缩进文字	97
29.合并单元格并保留所合并单元格的全部数据	101
30.将数字转换成中文	103
31.将二维数组中的元素连接成字符串	107
32.删除工作簿中的所有 VBA 代码	109
33.列出工作表中设置的条件格式清单	111
34.将 Excel 工作表导出到 Word 文档	115
35.快速备份工作簿中所有的代码模块	119
36.打印多个工作表	123
37.在工作表中添加 ActiveX 控件	125
38.列出工作簿中的所有批注	129
39.列出当前工作表中公式所引用的其他工作表名称	133
40.获取单元格链接的地址	137
41.使用 VBA 代码在默认浏览器中打开链接网站	141
42.修改 MsgBox 信息框中文本的颜色	143



43.获取文件名	147
44.列出指定目录下的所有文件或子文件夹中的所有文件	149
45.获取当前文件所在的文件夹路径及子文件夹数.....	153
46.统计指定文件夹或子文件夹中的工作簿数量.....	157
关于完美 Excel	161
完美 Excel 微信公众号使用指南	162



温馨提示：

在完美 Excel 微信公众号中发送消息：**VBA 小程序**，即可获得本电子书文档下载链接和密码。



如果您对本书有什么建议或者还有什么好的示例 ,欢迎前往完美 Excel 微信公众号沟通交流。

欢迎分享本电子书，让更多的人方便地得到所需要的知识。



本章内容 2018 年 1 月 20 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
VBA 实用小程序 1：列出名称及相应的引用

1. 列出名称及相应的引用

在构建或者研究一个较复杂的工作簿时，往往需要知道在该工作簿中定义的名称及相应的引用。一种简单的方法是使用“公式”选项卡中的“粘贴名称”功能，如图 1.1 所示。在完美 Excel 微信公众号的《[Excel 技巧视频 16：在工作表中显示名称及相应的命名区域](#)》一文中，我们详细介绍了这个技巧。

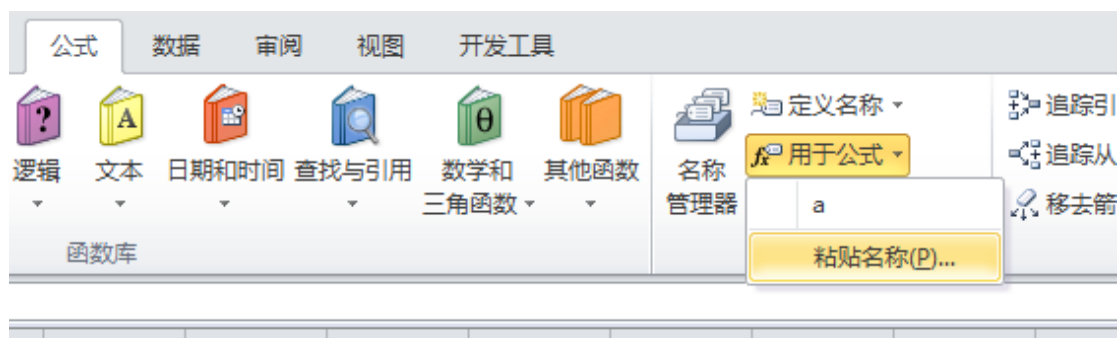


图 1.1

然而，“粘贴名称”功能存在小小的缺陷，当所定义的名称的引用超过 256 个字符时，会截掉后面的部分，如图 1.2 所示。

[illegible]

图 1.2

在图 1.2 中，我们使用“粘贴名称”功能列出了工作簿中所有的名称及对应的引用，但是可以看出，名称 sq_x 引用的数组并没有列全。

可以调用下面的 VBA 程序，列出工作簿中所有的名称引用：

```
Function GetNameRefersTo(TheName As String) As String
    Dim S As String
    Dim HasRef As Boolean
    Dim R As Range
    Dim NM As Name

    Set NM = ThisWorkbook.Names(TheName)

    On Error Resume Next

    Set R = NM.RefersToRange

    If Err.Number = 0 Then
        HasRef = True
    Else
        HasRef = False
    End If

    If HasRef = True Then
        S = R.Text
    Else
        S = NM.RefersTo
        S = "" & S
    End If
    GetNameRefersTo = S
End Function
```

如果想要去掉引用前面的“=”号或者文本常量的双引号，可以在 IF 语句的 Else 子句中使用下面的代码：

```
If StrComp(Mid(S, 2, 1), Chr(34), vbBinaryCompare) = 0 Then
    S = Mid(S, 3, Len(S) - 3)
Else
    S = Mid(S, 2)
End If
```

下面的程序调用 GetNameRefersTo 过程，在当前工作表中输出工作簿中所有的名称及相应的引用：

```
Sub NameLists()

    Dim nmName As Name

    Dim str As String

    Dim i As Long

    i = 2
```



本章内容 2018 年 1 月 24 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 02：计算任意公式的自定义函数

2. 计算任意公式的自定义函数

在 Excel 中，没有一个能够计算指定自变量的任意公式的值的通用函数。例如，我在单元格 B2 中放置公式 $mx+b$ ，其中 m 和 b 是具体的数字，在单元格 A2 中放置自变量 x 的值，没有类似下面的函数来计算这个公式的值：

`=CalculateThisFormula(B2,A2)`

为此，[Jon Peltier](#) 在其博客中编写了一个自定义函数，代码如下：

```
Function CalculateThisFormula(Formula As Variant, Xvalue As Variant) As Variant
    Dim sFormula As String
    Dim sXvalue As String

    Select Case TypeName(Formula)
    Case "Range"
        If Formula.Cells.Count = 1 Then
            sFormula = Formula.Value
        Else
            CalculateThisFormula = CVErr(xlErrValue)
            Exit Function
        End If
    Case "String"
        sFormula = Formula
    Case Else
        CalculateThisFormula = CVErr(xlErrValue)
        Exit Function
    End Select

    Dim sXvalue As String
    sXvalue = Xvalue

    ' 这里可以添加具体的计算逻辑
    ' 例如，对于公式 mx+b，可以解析出 m 和 b，然后代入 x 计算
    ' 这里只是一个占位符，表示计算过程
    CalculateThisFormula = sFormula & " (X=" & sXvalue & ")"
End Function
```



```

End Select

Select Case TypeName(Xvalue)
Case "Range"
    If Xvalue.Cells.Count = 1 Then
        sXvalue = Xvalue.Value
    Else
        CalculateThisFormula = CVErr(xlErrValue)
        Exit Function
    End If
Case "String"
    sXvalue = Xvalue
Case "Double"
    sXvalue = CStr(Xvalue)
Case Else
    CalculateThisFormula = CVErr(xlErrValue)
    Exit Function
End Select

sFormula = Replace(sFormula, "[x]", "(" &sXvalue& ")")
sFormula = Replace(sFormula, "[X]", "(" &sXvalue& ")")

CalculateThisFormula = Evaluate(sFormula)
End Function

```

说明：

- 为了避免将公式中的 x 与函数中的字母 x 混淆，例如 `exp()`，代码在方括号中放置自变量 x，即[x]。
- 这个自定义函数的语法为：

```
CalculateThisFormula(Formula, Xvalue)
```

其中，**Formula**——代表公式的字符串，或者是对包含公式字符串的单元格的引用。例如， $y=f(x)$ ，那么 **Formula** 为 `f(x)`，且自变量 x 必须表示为[x]，并且在公式



表达式中必须显示包含所有运算符。

xvalue——表示自变量 **x** 的值的数字或字符串，或者对包含数字值的单元格的引用。

- 如果任一参数包含多个单元格或者不恰当的输入，那么公式将返回错误。

下图 2.1 所示为自定义函数在工作表中的运用：

B2		fx =calculatethisformula(B\$1,\$A2)				
	A	B	C	D	E	
1		$2+3*[x]$	$1/[x]$	$4*[x]^2-3*[x]+5$	$\exp([x])$	
2	2	8	0.5	15	7.389056099	
3	3	11	0.3333333333	32	20.08553692	
4	4	14	0.25	57	54.59815003	
5	5	17	0.2	90	148.4131591	
6	6	20	0.1666666667	131	403.4287935	
7						
8						

图 2.1

在单元格 B2 中输入公式：

`=CalculateThisFormula(B$1,$A2)`

向下向右拖动至单元格 E6。

这个自定义函数也可以这样使用：

`=CalculateThisFormula("4*[x]^2-3*[x]+5",2)`

`=CalculateThisFormula("1/[x]",4)`





本章内容 2018 年 1 月 30 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
**VBA 实用小程序 3: 打开并处理指定文件夹
中的所有工作簿**

3. 打开并处理指定文件夹中的所有工作簿

有时候，我们需要使用代码打开并处理指定文件夹中的所有工作簿，然后再关闭这些工作簿。例如，打开指定文件夹中的工作簿，获取工作簿中的数值，然后关闭工作簿。

下面的 VBA 代码小程序列出指定文件夹中所有工作簿文件名：

```
Sub ProcessAllWorkbook(sPath As String)
    Dim Wb As Workbook, sFile As String

    If Right(sPath, 1) <> "\" Then sPath = sPath & "\"

    sFile = Dir(sPath & "*.xls*")

    Application.ScreenUpdating = False

    '遍历指定文件路径中的所有工作簿文件
    Do While sFile <> ""
        Set Wb = Workbooks.Open(sPath & sFile)
        '可以在这里编写处理工作簿的代码
        '本例中为打印工作簿名称
        Debug.Print Wb.Name
        '关闭工作簿,不保存任何修改
        Wb.Close False
        sFile = Dir
    End Do
End Sub
```



```
Loop

Application.ScreenUpdating = True
End Sub
```

我们使用下面的代码来测试 ProcessAllWorkbook 过程：

```
Sub test()
    ProcessAllWorkbook ("I:\ExcelTest")
End Sub
```

上面的代码列出文件夹 ExcelTest 中的所有的工作簿名称，如图 3.1 所示。

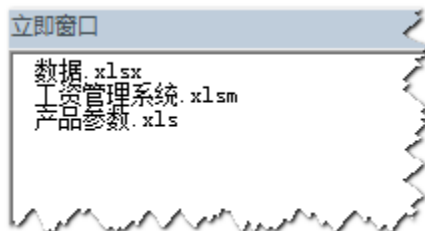


图 3.1

ProcessAllWorkbook 过程给出了一个代码框架，你可以替换 Debug.Print 语句，修改为你的代码，来实现对打开的工作簿的处理。



本章内容 2018 年 2 月 3 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 4: 在用户窗体标题栏添加
最大化和最小化按钮**

4. 在用户窗体标题栏添加最大化和最小化按钮

有时候，我们可能想在用户窗体标题栏添加最大化和最小化按钮，以方便对用户窗体的操控。下面是我找到的一段实现此功能的 VBA 代码，供大家需要时调用。

```
#If Win64 Then

Private Declare PtrSafe Function GetWindowLongPtr _
    Lib "user32.dll" Alias "GetWindowLongPtrA" ( _
        ByVal hwnd As LongPtr, _
        ByVal nIndex As Long) As LongPtr

Private Declare PtrSafe Function SetWindowLongPtr _
    Lib "user32.dll" Alias "SetWindowLongPtrA" ( _
        ByVal hwnd As LongPtr, _
        ByVal nIndex As Long, _
        ByVal dwNewLong As LongPtr) As LongPtr

Private Declare PtrSafe Function FindWindowA _
    Lib "user32.dll" ( _
        ByVal lpClassName As String, _
        ByVal lpWindowName As String) As LongPtr

Private Declare PtrSafe Function DrawMenuBar _
    Lib "user32.dll" ( _
        ByVal hwnd As LongPtr) As Long
```



```

#Else

Private Declare Function GetWindowLongPtr _
    Lib "user32.dll" Alias "GetWindowLongA" ( _
        ByVal hwnd As Long, _
        ByVal nIndex As Long) As Long

Private Declare Function SetWindowLongPtr _
    Lib "user32.dll" Alias "SetWindowLongA" ( _
        ByVal hwnd As Long, _
        ByVal nIndex As Long, _
        ByVal dwNewLong As Long) As Long

Private Declare Function FindWindowA _
    Lib "user32.dll" ( _
        ByVal lpClassName As String, _
        ByVal lpWindowName As String) As Long

Private Declare Function DrawMenuBar _
    Lib "user32.dll" ( _
        ByVal hwnd As Long) As Long

#End If

Private Sub UserForm_Initialize()
    CreateMenu
End Sub

Private Sub CreateMenu()
    Const GWL_STYLE As Long = -16
    Const WS_SYSMENU As Long = &H80000
    Const WS_MINIMIZEBOX As Long = &H20000
    Const WS_MAXIMIZEBOX As Long = &H10000

```



```

#If Win64 Then
    Dim lngFrmWndHdl As LongPtr
    Dim lngStyle As LongPtr
#Else
    Dim lngFrmWndHdl As Long
    Dim lngStyle As Long
#End If

lngFrmWndHdl = FindWindowA(vbNullString, Me.Caption)

lngStyle = GetWindowLongPtr(lngFrmWndHdl, GWL_STYLE)
lngStyle = lngStyle Or WS_SYSMENU      '添加系统菜单
lngStyle = lngStyle Or WS_MINIMIZEBOX '添加最小化框
lngStyle = lngStyle Or WS_MAXIMIZEBOX '添加最大化框

SetWindowLongPtr lngFrmWndHdl, GWL_STYLE, lngStyle

DrawMenuBar lngFrmWndHdl
End Sub

```

将上述代码放置在用户窗体代码模块中，运行后的效果如下图 4.1 所示：

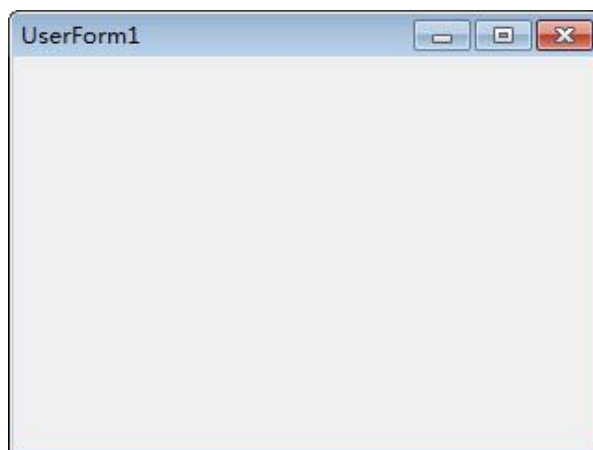


图 4.1





本章内容 2018 年 2 月 10 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 5：列出工作簿中的所有公式

5. 列出工作簿中的所有公式

一个较复杂的工作簿中往往包含有许多的公式，无论是创建还是研究这样的工作簿，都需要清楚地了解这些公式的作用，最直观的方法就是列出这些公式，看看它们到底是什么。

下面的代码在包含公式的工作簿中创建一个新工作表，然后在这个工作表中列出工作簿中的所有公式、公式所在的工作表名和单元格地址。

```
Sub AllFormulasLists()  
    Dim wks As Worksheet  
    Dim wksName  
    Dim rngUsedRange As Range  
    Dim rngFormulaRange As Range  
    Dim rng As Range  
    Dim lngLastRow As Long  
  
    ' 要求输入新工作表名称  
    Do  
        wksName = Application.InputBox("请输入放置公式列表的工作表的名称.", "新工作表名")  
        If wksName = False Then Exit Sub  
    Loop While WorksheetIfExists(wksName)  
  
    ' 添加新的工作表, 并将其移至最后  
    ' 设置新工作表的标题及名称
```



```

Worksheets.Add.Move After:=Worksheets(Worksheets.Count)
With ActiveSheet
    .Range("A1").Value = "工作表名称"
    .Range("B1").Value = "单元格地址"
    .Range("C1").Value = "公式文本"
    .Name = wksName
End With

'遍历工作簿中的工作表并获取工作表中的公式及相关信息
For Each wks In ActiveWorkbook.Worksheets
    If wks.Name <> wksName Then
        Set rngUsedRange = wks.UsedRange
        On Error Resume Next
        Set rngFormulaRange = rngUsedRange.SpecialCells(xlCellTypeFormulas)
        For Each rng In rngFormulaRange
            lngLastRow = lastRow(wksName)
            With Sheets(wksName).Range("A" & lngLastRow +
1)
                .Value = wks.Name
                .Offset(0, 1).Value = Application.WorksheetFunction.Substitute(rng.Address, "$",
"")
                .Offset(0, 2).Value = Mid(rng.Formula, 2,
(Len(rng.Formula)))
            End With
        Next rng
    End If
    Set rngFormulaRange = Nothing
    Set rngUsedRange = Nothing
Next wks

Sheets(wksName).Activate

```




```

        ActiveSheet.Columns("A:C").AutoFit
End Sub

'判断该名称的工作表是否已存在
Function WorksheetIfExists(wksNM) As Boolean
    Dim wksWorksheet As Worksheet

    On Error Resume Next

    Set wksWorksheet = Sheets(wksNM)

    If Not wksWorksheet Is Nothing Then
        WorksheetIfExists = True
    Else
        WorksheetIfExists = False
    End If
End Function

```

```

'获取工作表中的最后一行
Function lastRow(wks) As Long
    lastRow = Sheets(wks).Range("A" & Rows.Count).End(xlUp).Row
End Function

```

在示例工作簿中运行代码后的效果如下图 5.1 所示：

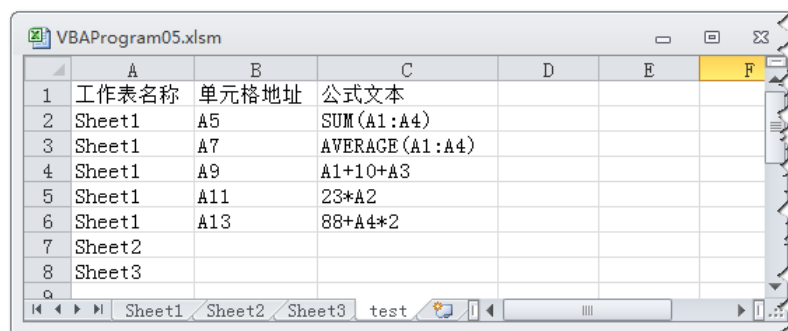


图 5.1





本章内容 2018 年 2 月 25 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 6: 在工作表中自动
添加形状**

6. 在工作表中自动添加形状

下面的代码在当前工作表中自动添加一个矩形。

在代码运行过程中，会要求用户输入形状的大小、颜色，设置形状上显示的文本格式并要求输入显示文本，要求指定单击该形状后运行的宏。

```
Sub 自动添加矩形()  
    Dim wks As Worksheet  
    Dim shp As Object  
    Dim strText As String  
    Dim strDimensions As String  
    Dim rngDimensions As Range  
    Dim iColor As Integer  
    Dim str As String  
  
    On Error Resume Next  
  
    Set wks = ActiveSheet  
  
    strDimensions = Trim(Application.InputBox("请输入形状的维  
数(行数×列数)", _  
        "形状维数", "3×3", , , , 2))  
  
    iColor = Trim(Application.InputBox("请输入形状的颜色:1=蓝  
色,2=绿色,3=红色", _  
        "形状填充的颜色", "2", , , , 1))
```



```

'确保颜色在 0-3 之间

iColor = WorksheetFunction.Min(iColor, 3)
iColor = WorksheetFunction.Max(iColor, 0)

Set rngDimensions =
Selection.Cells(1).Resize(CDbl(Split(strDimensions,
"×")(0)), _
    CDbl(Split(strDimensions, "×")(1)))

With rngDimensions
    Set shp =
wks.Shapes.AddShape(msoShapeRoundedRectangle, .Left, .Top,
-
    .Width, .Height)
End With

With shp
    .Name = "运行的宏名"

'水平居中

With .TextFrame2.TextRange.Characters(1,
Len(strText)).ParagraphFormat
    .FirstLineIndent = 0
    .Alignment = msoAlignCenter
End With

'垂直居中

With .TextFrame2
    .VerticalAnchor = msoAnchorMiddle
End With

With .Fill
    .ForeColor.RGB = Choose(iColor, RGB(0, 176, 210),
RGB(146, 208, 80), RGB(255, 0, 0))

```



```

        .Transparency = 0
        .Solid
    End With

    With .Line
        .ForeColor.RGB = shp.Fill.ForeColor.RGB
        .Transparency = shp.Fill.Transparency
    End With

    .Placement = xlMove
'xlMoveAndSize=1,xlMove=2,xlFreeFloating=3

    .Select
    Application.Dialogs(xlDialogAssignToObject).Show

    str = Split(.OnAction, "!")(1)
    If Len(str) = 0 Then str = .OnAction

    strText = Trim(Application.InputBox("请输入显示在形状
上的文本", "形状文本", str, , , , 2))

    If strText = "False" Then strText = "添加标题"
    If Len(strText) = 0 Then strText = "添加标题"

    With .TextFrame.Characters
        .Text = strText
        .Font.Color = vbWhite
        .Font.Bold = True
    End With

    rngDimensions.Cells(1).Select
End With

```



```
On Error GoTo 0

Set wks = Nothing

End Sub
```

如果用户执行上述代码，设置形状大小为默认的 3×3（即占 3 行 3 列），颜色为 2-绿色，选取单击形状时要执行的宏，并输入显示在形状上的文本，结果如下图 6.1 所示。

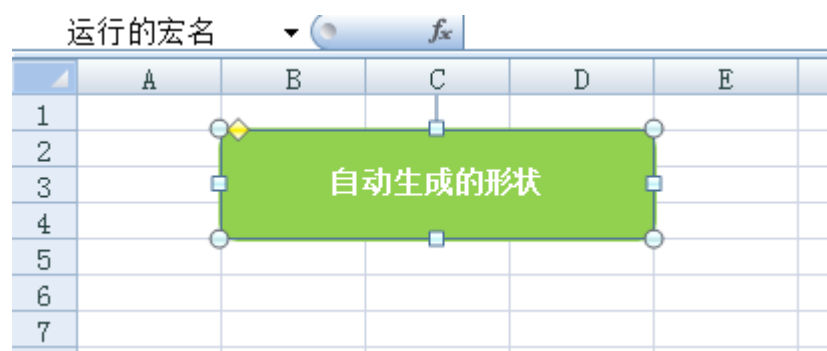


图 6.1



本章内容 2018 年 2 月 28 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
VBA 实用小程序 7：整齐排列工作表中的图表

7. 整齐排列工作表中的图表

我们可以手工拖放工作表中的图表，使它们整齐排列，甚至在拖放时使用 ALT 来调整图表尺寸并让图表与单元格边框对齐。

然而，如果工作表中有很多图表，手工拖放图表将是一项非常费时费力的操作。幸好，VBA 能够帮助我们自动调整图表尺寸并整齐排列图表。

下面是一段实现此功能的 VBA 代码，供大家学习研究。

```
Sub PlaceChartsInOrder()  
    '图表大小  
    Const nRowsTall As Long = 6  
    Const nColsWide As Long = 3  
  
    '图表布局  
    Const nChartsPerRow As Long = 3  
    Const nSkipRows As Long = 2  
    Const nSkipCols As Long = 1  
    Const nFirstRow As Long = 3  
    Const nFirstCol As Long = 2  
  
    Dim iChart As Long  
    Dim chtOb As ChartObject  
    Dim dWidth As Double  
    Dim dHeight As Double
```



```

Dim rData As Range

Dim dFirstChartTop As Double

Dim dFirstChartLeft As Double

Dim dRowsBetweenChart As Double

Dim dColsBetweenChart As Double

If ActiveSheet.ChartObjects.Count > 0 Then
    With ActiveSheet.Cells(nFirstRow, nFirstCol)
        If nRowsTall * nColsWide > 0 Then
            dWidth = nColsWide * .Width
            dHeight = nRowsTall * .Height
        Else
            If Not ActiveChart Is Nothing Then
                Set chtOb = ActiveChart.Parent
            Else
                Set chtOb = ActiveSheet.ChartObjects(1)
            End If
            dWidth = chtOb.Width
            dHeight = chtOb.Height
        End If

        dFirstChartLeft = .Left
        dFirstChartTop = .Top
        dRowsBetweenChart = nSkipRows * .Height
        dColsBetweenChart = nSkipCols * .Width
    End With

    For iChart = 1 To ActiveSheet.ChartObjects.Count
        Set chtOb = ActiveSheet.ChartObjects(iChart)
        With chtOb
            .Left = ((iChart - 1) Mod nChartsPerRow) * _

```




```

        (dWidth + dColsBetweenChart) + dFirstChartLeft
        .Top = Int((iChart - 1) / nChartsPerRow) * _
            (dHeight + dRowsBetweenChart) + dFirstChartTop
        .Width = dWidth
        .Height = dHeight
    End With
Next
End If
End Sub

```

在示例工作簿中运行后的效果如下图 7.1 所示：

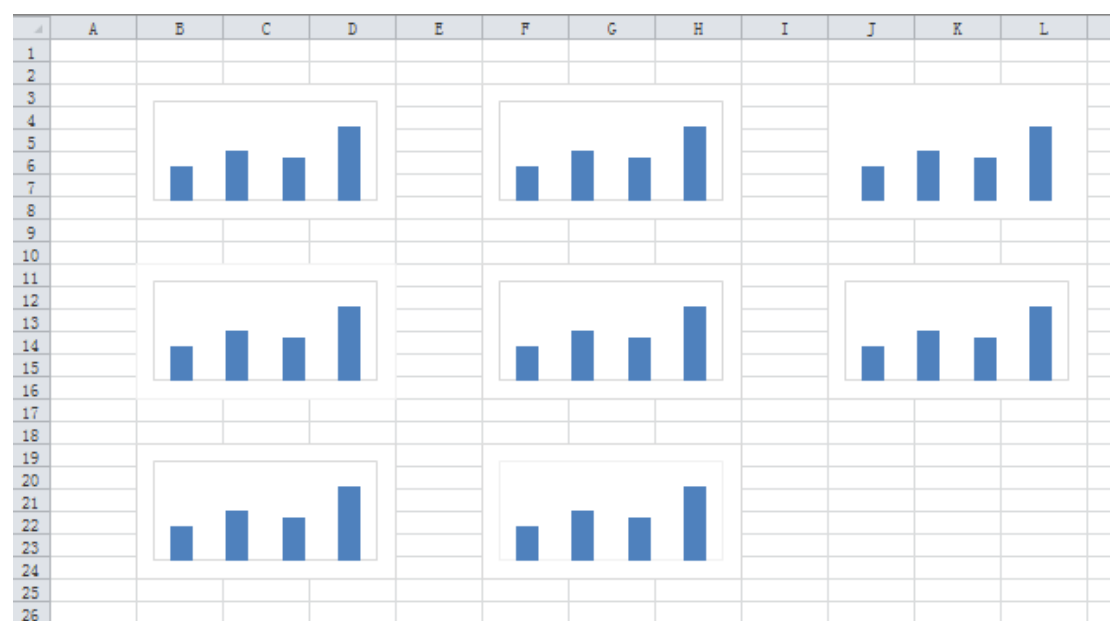


图 7.1

这个 VBA 程序并不复杂，首先指定每个图表占用的高度和宽度，每排多少个图表，以及每一个图表左上角和图表之间的行列间隔，以此来排列工作表中的图表。





本章内容 2018 年 3 月 1 日首发于
 [完美 Excel] 微信公众号 [excelperfect](#)
 原标题为
VBA 实用小程序 8: 创建并自动排列
图表

8. 创建并自动排列图表

在上一节中，我们给出了一段 VBA 程序，能够帮助我们自动调整工作表中的图表尺寸并整齐排列图表。

有时，想要根据工作表中的数据绘制图表并自动排列这些图表，本文给出的 VBA 程序用来实现这个功能。

下图 8.1 是放置示例数据的工作表。

	A	B	C	D	E	F	G	H	I	J	K
1											
2		RowsHigh	ColsWide	FirstRow	FirstCol	SkipRows	SkipCols	NAcross	FontSize	RowHeight	ColWidth
3		10	2	3	2	0	0	4	8		
4											
5		Jan	Feb	Mar	Apr	May	Jun				
6		Goal	25	30	35	40	45	50			
7											
8		Jan	Feb	Mar	Apr	May	Jun				
9		ABC	25.65	21.23	37.34	37.64	38.59	50.32			
10		DEF	22.77	38.35	37.09	40.43	44.54	47.75			
11		GHI	28.38	36.14	28.09	35.59	43.87	47.2			
12		JKL	31.28	26.77	36.43	40.26	42.59	51.25			
13		MNO	24.14	28.86	33.25	38.59	38.55	56.03			
14		PQR	24.36	25.69	30.16	36.62	45.25	54.5			
15		STU	30.27	34.07	28.67	35.02	51.79	46.94			
16		VWX	29.81	31.41	37.04	34.84	44.72	50.7			
17		YZA	20.22	25.55	38.48	38.35	45.66	52.99			
18		BCD	28.76	27.86	32.6	43.18	44.93	47.69			
19		EFG	28.95	28.61	36.35	38.47	41.17	51.79			
20		HIJ	31.03	29.62	36.77	37.66	47.84	47.64			
21											

图 8.1

在图 8.1 中，黄色单元格区域 B2:K3 命名为 Settings，用来存放图表大小、位置等数据，单元格区域 B2:K2 中的说明如下：

RowsHigh: 每个图表的高度，占据的工作表行

ColsWide: 每个图表的宽度，占据的工作表列



FirstRow: 图表占据的最顶部的工作表行
FirstCol: 图表占据的最左侧的工作表列
SkipRows: 图表之间间隔的工作表行
SkipCols: 图表之间间隔的工作表列
NAcross: 每排的图表数
FontSize: 图表使用的文本大小
RowHeight: 工作表行高，以磅为单位
ColWidth: 工作表列宽，以默认字体的字符大小为单位

在图 8.1 中，背景为绿色的单元格 B5 命名为 **DataAll**，单元格区域 C5:H5 是类别标签，单元格 B6 是系列名，单元格区域 C6:H6 中的值是每个图表中都要绘制的系列的数据。

在图 8.1 中，背景为紫色的单元格 B8 命名为 **DataEach**，该区域顶部行区域 C8:H8 是类别标签，左侧的单元格列中的值是系列名称，其余的行数据是每个图表中的系列数据值，本示例中有 12 行，因此这些数据绘制在 12 个图表中。

下面是 VBA 程序代码，供大家学习研究。在运行代码前，图 8.1 所示的工作表应该为活动工作表。

```
Sub CreateAndArrayCharts()  
    Dim vSettings As Variant  
    Dim rDataAll As Range  
    Dim rDataEach As Range  
  
    '图表大小  
    Dim nRowsTall As Long  
    Dim nColsWide As Long  
    Dim nFontSize As Double  
  
    '图表布局  
    Dim nChartsPerRow As Long  
    Dim nSkipRows As Long
```



```
Dim nSkipCols As Long
Dim nFirstRow As Long
Dim nFirstCol As Long

Dim wsData As Worksheet
Dim wsChart As Worksheet
Dim iChart As Long
Dim chtOb As ChartObject
Dim srs As Series
Dim dWidth As Double
Dim dHeight As Double
Dim rData As Range
Dim dFirstChartTop As Double
Dim dFirstChartLeft As Double
Dim dRowsBetweenChart As Double
Dim dColsBetweenChart As Double
Dim dRowHeight As Double
Dim dColWidth As Double
```

```
Application.ScreenUpdating = False
```

'定义工作表

```
Set wsData = ActiveSheet
Set wsChart = Worksheets.Add(After:=wsData)
```

'定义数据区域

```
vSettings = wsData.Range("Settings").Value
Set rDataAll = wsData.Range("DataAll").CurrentRegion
Set rDataEach = wsData.Range("DataEach").CurrentRegion
```

'图表布局



```

nRowsTall = vSettings(2, 1)
nColsWide = vSettings(2, 2)
nFirstRow = vSettings(2, 3)
nFirstCol = vSettings(2, 4)
nSkipRows = vSettings(2, 5)
nSkipCols = vSettings(2, 6)
nChartsPerRow = vSettings(2, 7)
nFontSize = vSettings(2, 8)
dRowHeight = vSettings(2, 9)
dColWidth = vSettings(2, 10)

If dRowHeight > 0 Then
    wsChart.Rows.RowHeight = dRowHeight
End If

If dColWidth > 0 Then
    wsChart.Columns.ColumnWidth = dColWidth
End If

With wsChart.Cells(1, 1)
    dWidth = nColsWide * .Width
    dHeight = nRowsTall * .Height
    dFirstChartLeft = (nFirstCol - 1) * .Width
    dFirstChartTop = (nFirstRow - 1) * .Height
    dRowsBetweenChart = nSkipRows * .Height
    dColsBetweenChart = nSkipCols * .Width
End With

For iChart = 1 To rDataEach.Rows.Count - 1
    '创建图表
    Set chtOb = wsChart.ChartObjects.Add( _

```



```

        ((iChart - 1) Mod nChartsPerRow) * (dWidth + dColsBetweenChart) +
dFirstChartLeft, _
        Int((iChart - 1) / nChartsPerRow) * (dHeight + dRowsBetweenChart) +
dFirstChartTop, _
        dWidth, dHeight)

```

'添加标准系列(对所有图表都相同)

```
Set srs = chtOb.Chart.SeriesCollection.NewSeries
```

```
With srs
```

```

        .Name = rDataAll.Cells(2, 1)
        .Values = rDataAll.Cells(2, 2).Resize(, rDataAll.Columns.Count - 1)
        .XValues = rDataAll.Cells(1, 2).Resize(, rDataAll.Columns.Count - 1)
        .ChartType = xlLine
        .Border.Weight = xlMedium

```

```
End With
```

'添加不同的系列(每个图表自己的系列)

```
Set srs = chtOb.Chart.SeriesCollection.NewSeries
```

```
With srs
```

```

        .Name = rDataEach.Cells(iChart + 1, 1)
        .Values = rDataEach.Cells(iChart + 1, 2).Resize(,
rDataEach.Columns.Count - 1)
        .XValues = rDataEach.Cells(1, 2).Resize(, rDataEach.Columns.Count - 1)
        .ChartType = xlLine
        .Border.Weight = xlMedium

```

```
End With
```

'格式化图表

```
With chtOb.Chart
```

```

        .HasTitle = True
        .ChartTitle.Characters.Text = srs.Name
        With .ChartArea

```



```

        .AutoScaleFont = False
        .Font.Size = nFontSize
        .Border.ColorIndex = 2
    End With
    .HasLegend = False
    With .Axes(xlValue)
        .HasMajorGridlines = False
        .Border.ColorIndex = 48
    End With
    With .Axes(xlCategory)
        .Border.ColorIndex = 48
        With .TickLabels
            .Orientation = xlHorizontal
            .Offset = 0
        End With
    End With
    End With
    With .PlotArea
        .Border.ColorIndex = 48
        .Interior.ColorIndex = 2
        .Left = 0
        .Top = 0
        .Height = chtOb.Chart.ChartArea.Height
        .Width = chtOb.Chart.ChartArea.Width - 7
    End With
    With .ChartTitle
        .Font.Bold = True
        .Top = chtOb.Chart.PlotArea.InsideTop
        .Left = chtOb.Chart.PlotArea.InsideLeft + 3
    End With
    End With
    Next

```




```
Application.ScreenUpdating = True  
End Sub
```

在示例工作簿中运行后的效果如下图 8.2 所示：

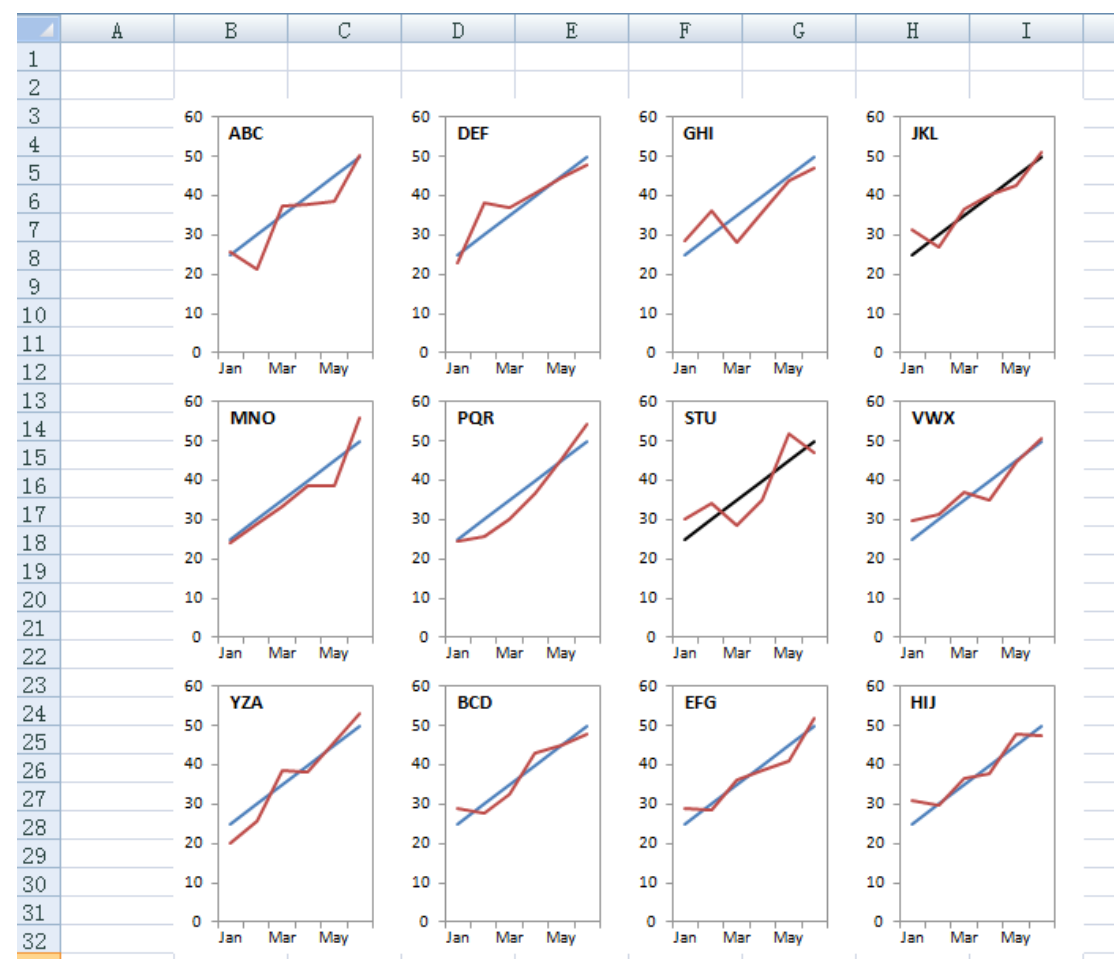


图 2

通过调整图 8.1 所示工作表中 **Settings** 区域的值，可以得到不同排列形式的图表。





本章内容 2018 年 3 月 18 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[VBA 实用小程序 9：加密/解密数据](#)

9. 加密/解密数据

对大多数程序员来说，加密和解密算法看起来似乎很神秘。下面的示例程序演示了一种在 VBA 应用程序中加密和解密数据的非常简单的方法。虽然对于专业密码的专家来说是小儿科，但它很容易阻止 99.99% 的其他用户访问敏感数据。尽管此示例使用短字符串进行演示，但所提供的加密算法可以非常快速地处理包含各种数据的非常大的文件。

这段程序是由《[Excel 2007 VBA 参考大全](#)》的作者之一 Rob Bovey 提供的，特在此分享，供有用时参考。

```
''' 注意：要加密的数据必须存储在支持 Unicode 文本的位置。
''' 这样的位置包括注册表,甚至纯文本文件。
''' 为了演示的目的,这里简单地使用了一个公共字符串变量。
Public gszData As String

''' 这个程序要求用户输入一个字符串,然后加密这个字符串。
Public Sub DemoEncryptData()

    gszData = CStr(Application.InputBox("输入一个要加密的字符串.", "加密演示"))

    ''' 如果用户没有取消输入框或者没有输入而单击确定,则继续。
    If (gszData <> CStr(False)) And (Len(gszData) > 0) Then
        ''' 通过一次传递到加密过程来加密指定的数据字符串。
        EncryptDecrypt gszData
        MsgBox "加密后的字符串是:" & vbCrLf & vbCrLf & gszData, vbInformation, "加密演示"
    Else
        gszData = vbNullString
    End If
End Sub
```



```

        End If
    End Sub

    ''' 这个程序解密由上面的程序加密的字符串.

    Public Sub DemoDecryptData()
        ''' 如果有加密的数据存储在公共字符串变量中.....

        If Len(gszData) > 0 Then
            ''' 再次通过加密过程解密指定的数据字符串.

            EncryptDecrypt gszData

            MsgBox "解密后的字符串是:" & vbCrLf & vbCrLf & gszData, vbInformation, "加密演示"

        Else
            MsgBox "没有要解密的存储数据.", vbExclamation, "加密演示"

        End If
    End Sub

    .....

    ''' 说明:    执行 Xor 加密/解密字符串数据.
    '''          先传递字符串到程序来加密它.
    '''          再次传递字符串来解密它.
    '''

    ''' 参数:    szData           [in|out] 要加密或解密的数据字符串
    '''

    Private Sub EncryptDecrypt(ByRef szData As String)

        Const IKEY_VALUE As Long = 215

        Dim bytData() As Byte
        Dim ICount As Long

        bytData = szData
    
```



```
For ICount = LBound(bytData) To UBound(bytData)
    bytData(ICount) = bytData(ICount) Xor IKEY_VALUE
Next ICount

szData = bytData

End Sub
```

运行效果如下图 9.1 所示：

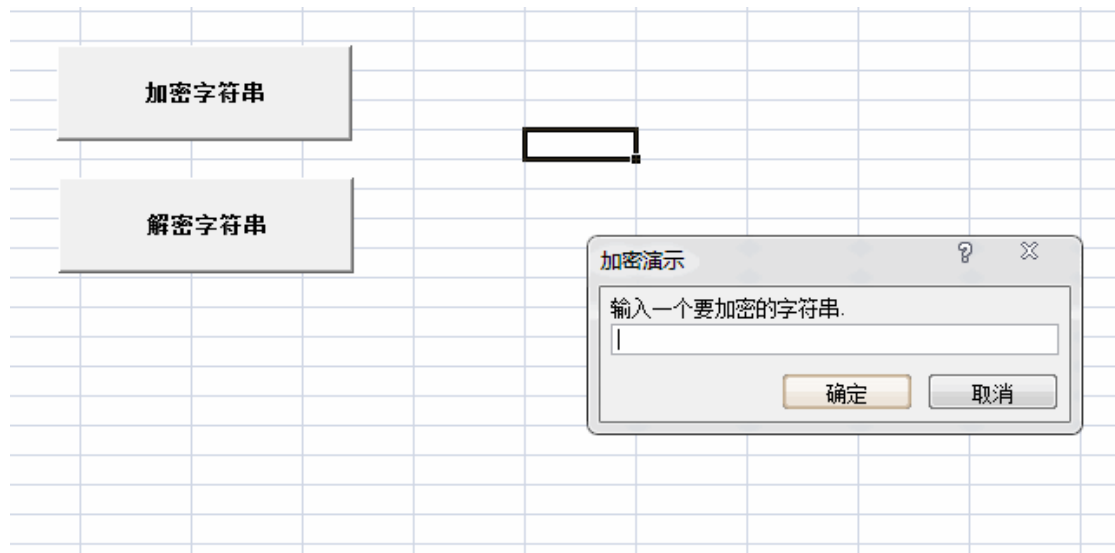


图 9.1





本章内容 2018 年 3 月 25 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 10: 获取文本字符串中的文本或者数字

10. 获取文本字符串中的文本或者数字

在工作表单元格中，包含有文本和数字混合的字符串，如图 10.1 所示，如何提取字符串中的数字或者文本？

	A	B	C
1	数据	文本部分	数字部分
2	《完美Excel大全》订购价¥39	《完美Excel大全》订购价¥	39
3	期末分数99	期末分数	99
4	员工编号10001	员工编号	10001
5			
6			

图 1

可以在工作表中使用 VBA 自定义函数来实现。

下面的 VBA 代码创建的自定义函数从字符串中提取数字。

```
Function GetNum(rng As String)

    Dim lngLen As Long
    Dim i As Long, result
    lngLen = Len(rng)
    For i = 1 To lngLen
        If IsNumeric(Mid(rng, i, 1)) Then
            result = result & Mid(rng, i, 1)
        End If
    Next i
    GetNum = result
End Function
```



下面的 VBA 代码创建的自定义函数从字符串中提取文本。

```
Function GetText(rng As String)
    Dim lngLen As Long
    Dim i As Long, result
    lngLen = Len(rng)
    For i = 1 To lngLen
        If Not (IsNumeric(Mid(rng, i, 1))) Then
            result = result & Mid(rng, i, 1)
        End If
    Next i
    GetText = result
End Function
```

在 VBE 编辑器的标准模块中输入上述代码后，在工作表中像使用 Excel 函数一样输入公式：

```
=GetNum(A2)
```

或者

```
=GetText(A2)
```

提取单元格中字符串里的数字或者文本。

两个函数都只接受一个参数，该参数指定要从中提取数字或文本的单元格。



本章内容 2018 年 3 月 29 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 11: 通过右键快捷
菜单导航工作簿和工作表**

11. 通过右键快捷菜单导航工作簿和工作表

下面的 VBA 程序在单元格快捷菜单中添加一个名为“工作簿导航”的命令，通过该命令快速地在当前所有打开的工作簿及其工作表之间导航，如图 11.1 所示。

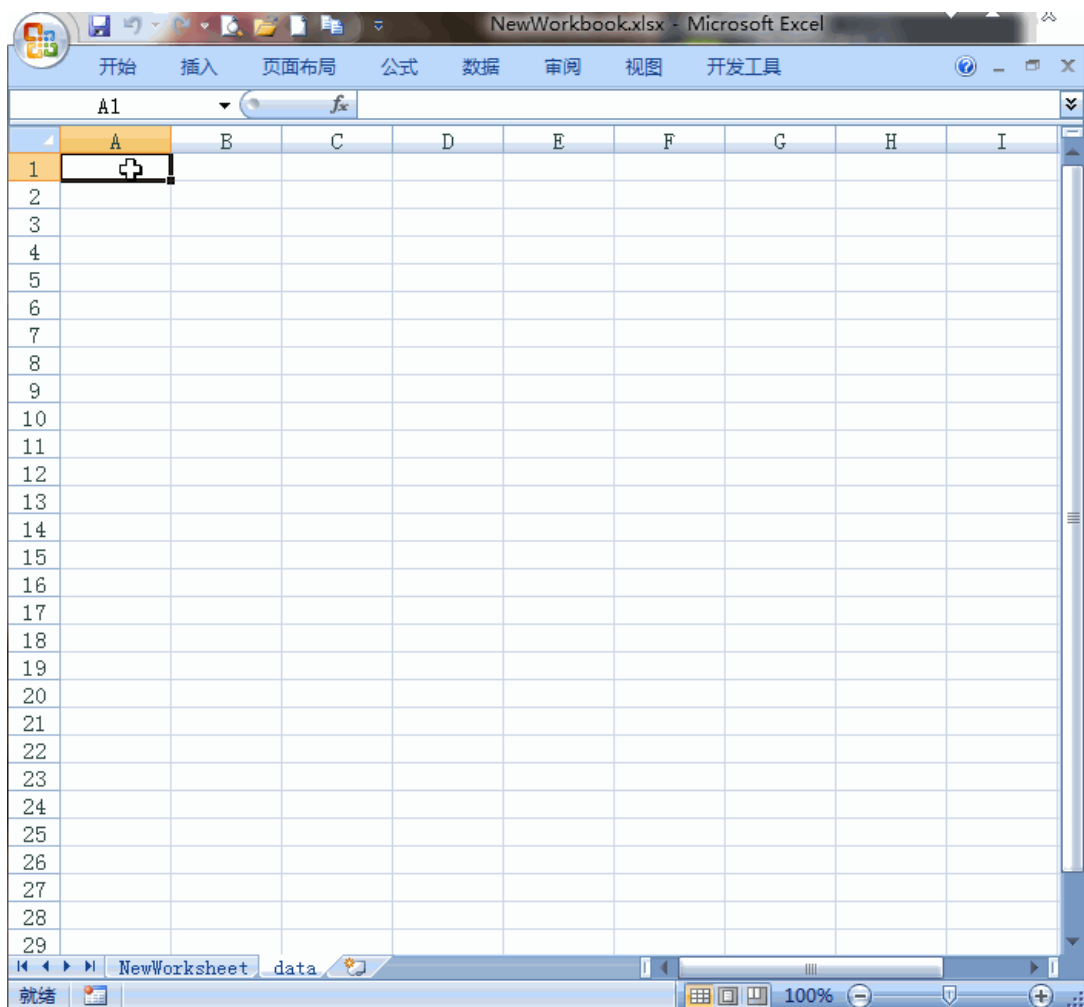


图 11.1

在 VBE 中添加一个标准模块，输入下面的代码：



```

Sub CustomMenu()
    On Error GoTo ExitSub:
    Dim cmb As CommandBarControl
    On Error Resume Next

    Application.CommandBars("Cell").Controls(" 工 作 簿 导 航").Delete

    Set cmb = Application.CommandBars("Cell").Controls.Add _
        (Type:=msoControlPopup, Temporary:=True)

    cmb.Caption = "工作簿导航"
    cmb.OnAction = "AddControlsInCustomMenu"
ExitSub:
    Exit Sub
End Sub

Sub AddControlsInCustomMenu()
    Dim wb As Workbook
    Dim ws As Worksheet
    Dim cmb As CommandBarControl

    Dim cmbCtl1 As CommandBarControl, cmbCtl2 As CommandBarControl

    For Each cmb In Application.CommandBars("Cell"). _
        Controls(" 工 作 簿 导 航").Controls
        On Error Resume Next
        cmb.Delete
    Next

    For Each wb In Application.Workbooks
        Set cmbCtl1 = Application.CommandBars("Cell"). _
            Controls(" 工 作 簿 导 航").Controls.Add(Type:=msoControlPopup)
        With cmbCtl1

```



```

        .Caption = wb.Name
        .OnAction = "ActivateWB"
    End With
    For Each ws In wb.Sheets
        If ws.Visible = xlSheetVisible Then
            Set cmbCtl2 = cmbCtl1.Controls.Add( _
                Type:=msoControlButton)
            With cmbCtl2
                .Caption = ws.Name
                .OnAction = "ActivateWS"
            End With
        End If
    Next ws
Next wb
End Sub

Sub ActivateWB()
    On Error Resume Next

    Windows(Application.CommandBars.ActionControl.Caption)
        .Activate
End Sub

Sub ActivateWS()
    On Error Resume Next

    Sheets(Application.CommandBars.ActionControl.Caption)
        .Activate
End Sub

```

在 **ThisWorkbook** 模块中，输入下面的代码：

```

Private Sub Workbook_Open()
    On Error Resume Next

    Application.CommandBars("Cell").Controls(" 工 作 簿 导 航

```



```
) .Delete  
    Call CustomMenu  
End Sub  
  
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    On Error Resume Next  
    Application.CommandBars("Cell").Controls(" 工 作 簿 导 航"  
).Delete  
End Sub
```



本章内容 2018 年 4 月 8 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 12：删除工作簿中
除指定工作表之外的所有工作表

12. 删除工作簿中除指定工作表之外的所有工作表

有时候，我们想快速删除工作簿中的一些工作表，但保留指定名称的工作表。如下图 12.1 所示，删除工作簿中除名称为“Sheet1”、“Sheet2”和“完美 Excel”外的所有工作表。

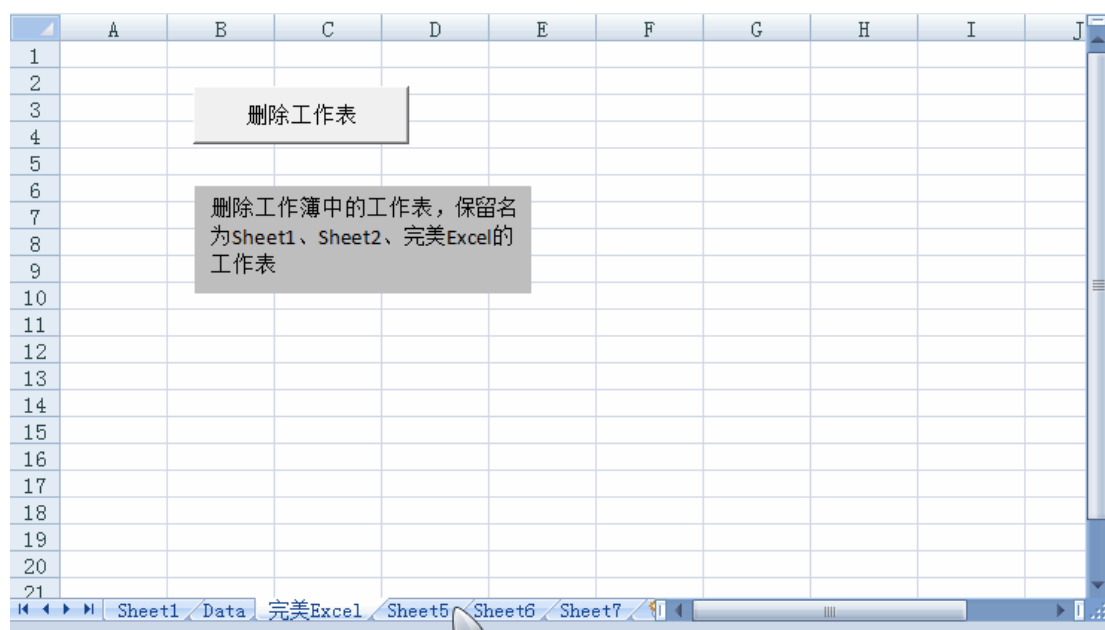


图 12.1

VBA 代码如下：

```
Sub 删除指定工作表之外的工作表()  
    Dim strWorksheet As String  
    Dim i As Long
```



```
Dim wks As Worksheet

' 要保留的工作表名
' 注意最后的逗号分隔符
strWorksheet = "Sheet1,Sheet2,完美 Excel,"

Application.DisplayAlerts = False

' 遍历工作表
For Each wks In ThisWorkbook.Worksheets
    ' 检查工作表名是否在字符串中
    ' 注意后面的逗号分隔符
    If InStr(1, strWorksheet, wks.Name & ",") = 0 Then
        wks.Delete
    End If
Next wks

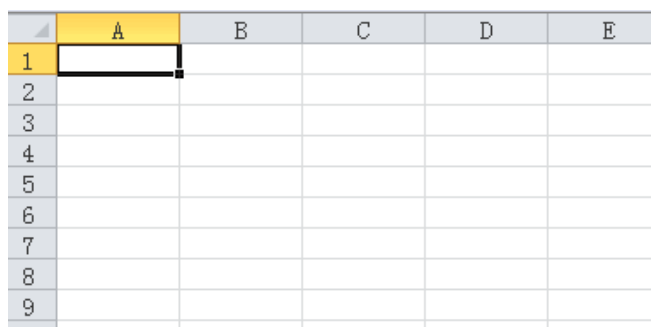
Application.DisplayAlerts = True
End Sub
```



本章内容 2018 年 4 月 20 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 13: 记录单元格中
存放过的数据

13. 记录单元格中存放过的数据

如果我们能够记录单元格中曾经放置过的数据,就可以清楚地看到该单元格的编辑痕迹。如图 13.1 所示,在工作表 Sheet1 的单元格批注中,显示出该单元格中所有存放过的数据,包括当前正存放的数据。



	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					

图 13.1 (本图为视频动画截图,详细的动画演示参见[完美 Excel](#) 微信公众号)

在工作表 Sheet1 的代码模块中,输入的代码如下:

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Dim str As String
    '存储输入的数据
    str = Sheet2.Range(Target.Address) & Target & " , "
    Sheet2.Range(Target.Address) = str
    '清除多于一个单元格导致的错误
    On Error Resume Next
    '不能覆盖已存在的批注
    Target.ClearComments
End Sub
```



```

'添加单元格批注

With Target

    '当值改变量获取前一个值

    .AddComment

    .Comment.Visible = False

    .Comment.Text Text:="本单元格中依次输入的值是：" & _
        & Left(Sheet2.Range(Target.Address), _
            Len(Sheet2.Range(Target.Address)) - 3)

End With

End Sub

```

使用下面的代码，可以只显示该单元格中上次存放的数据。

```

Private Sub Worksheet_SelectionChange(ByVal Target As Range)

    '将前一个值复制到另一个工作表相对应的单元格中

    Sheet2.Range(Target.Address) = Target

End Sub

Private Sub Worksheet_Change(ByVal Target As Range)

    '清除多于一个单元格导致的错误

    On Error Resume Next

    '不能覆盖已存在的批注

    Target.ClearComments

    With Target

        '当值改变量获取前一个值

        .AddComment

        .Comment.Visible = False

        .Comment.Text Text:="单元格中上次输入的值是：" & _
            Sheet2.Range(Target.Address)

    End With

End Sub

```



代码运行后的结果如图 13.2 所示。

	A	B	C	D	E
1					
2					
3					
4					
5					
6					
7					
8					
9					

图 13.2（本图为视频动画截图，详细的动画演示参见[完美 Excel 微信公众号](#)）

注意，代码要放置在工作表代码模块中，本例是放置在工作表 **Sheet1** 代码模块中。**Worksheet_SelectionChange** 事件发生在由当前单元格转移到的下一个单元格中，例如当前单元格为 **A1**，下一个单元格为 **A2**，那么该事件发生在单元格 **A2** 中。





本章内容 2018 年 4 月 24 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 14: 处理文件名的
自定义函数

14.处理文件名的自定义函数

下面列出了 [Jon Peltier](#) 编写的一些处理文件名的自定义函数,与大家分享。

'获取文件名,包括扩展名

```
Function FullNameToFileName(strFullName As String) As String

    Dim i As Integer
    Dim strTest As String

    '是否包含方括号
    If InStr(1, strFullName, "[") > 0 Then
        i = InStr(1, strFullName, "[")
        strTest = Mid$(strFullName, i + 1, InStr(1, strFullName, "]") - i - 1)
    Else
        '文件路径中反斜杠的位置
        For i = Len(strFullName) To 1 Step -1
            If Mid$(strFullName, i, 1) = "\" Then Exit For
        Next i
        strTest = Mid$(strFullName, i + 1, Len(strFullName) - i)
    End If

    FullNameToFileName = strTest
End Function
```

'获取文件名,不包括扩展名

```
Function FullNameToRootName(strFullName As String) As String

    Dim i As Integer
```



```

Dim strTest As String
strTest = FullNameToFileName(strFullName)
'获取文件名中后缀点所在的位置
For i = Len(strTest) To 1 Step -1
    If Mid$(strTest, i, 1) = "." Then Exit For
Next
strTest = Left$(strTest, i - 1)
FullNameToRootName = strTest
End Function

'获取文件的扩展名
Function FullNameToFileExt(strFullName As String) As String
    Dim i As Integer
    Dim strTest As String
    strTest = FullNameToFileName(strFullName)
    '获取文件名中后缀点所在的位置
    For i = Len(strTest) To 1 Step -1
        If Mid$(strTest, i, 1) = "." Then Exit For
    Next
    If i > 0 Then
        strTest = Right$(strTest, Len(strTest) - i)
    Else
        strTest = ""
    End If
    FullNameToFileExt = strTest
End Function

'获取文件所在的路径
Function FullNameToPath(strFullName As String) As String
    '不包括尾斜杠
    Dim i As Integer

```



'获取文件路径中反斜杠位置

```
For i = Len(strFullName) To 1 Step -1
    If Mid$(strFullName, i, 1) = "\" Then Exit For
Next
If i < 1 Then
    FullNameToPath = ""
Else
    FullNameToPath = Mid$(strFullName, 1, i - 1)
End If
End Function
```

'判断指定文件是否存在

```
Function FileExists(ByVal FileSpec As String) As Boolean
    Dim Attr As Long
    On Error Resume Next
    '获取文件或文件夹属性
    Attr = GetAttr(FileSpec)
    If Err.Number = 0 Then
        '不是文件夹
        FileExists = Not ((Attr And vbDirectory) = vbDirectory)
    End If
End Function
```

'判断指定的文件夹是否存在

```
Function DirExists(ByVal FileSpec As String) As Boolean
    Dim Attr As Long
    On Error Resume Next
    '获取文件或文件夹属性
    Attr = GetAttr(FileSpec)
    If Err.Number = 0 Then
        '不是文件夹
```



```

DirExists = (Attr And vbDirectory) = vbDirectory

End If

End Function

```

与 Excel 内置函数一样，在工作表中输入函数名并指定参数，即可获得相应的结果，如图 14.1 所示。

	A	B	C	D	E	F	G
1							
2		文件路径和文件名	G:\09. Excel\06.1 VBA实用小程序\14\test.xlsx				
3		文件夹	G:\09. Excel\06.1 VBA实用小程序\14\				
4							
5							
6		文件名（带扩展名）	test.xlsx				
7		公式	=FullNameToFilename(C2)				
8		文件名（不带扩展名）	test				
9		公式	=FullNameToRootName(C2)				
10		文件扩展名	xlsx				
11		公式	=FullNameToFileExt(C2)				
12		文件所在路径	G:\09. Excel\06.1 VBA实用小程序\14				
13		公式	=FullNameToPath(C2)				
14		文件是否存在	TRUE				
15		公式	=FileExists(C2)				
16		文件夹是否存在	TRUE				
17		公式	=DirExists(C3)				
18							
19							

图 14.1



本章内容 2018 年 4 月 30 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
VBA 实用小程序 15: 将 Excel 图表
导出为图片

15.将 Excel 图表导出为图片

下面是 **Jon Peltier** 编写的一段简单的 VBA 小程序，用来将当前图表导出为图片并保存在当前工作簿相同的文件夹中。在程序中，我添加了一些注释，以方便理解。

```
Sub ExportChart()  
    Dim strChtName As String  
    Dim strPrompt As String  
    Dim strDefault As String  
  
    If ActiveSheet Is Nothing Then GoTo ExitSub  
    If ActiveChart Is Nothing Then GoTo ExitSub  
  
    strPrompt = "图表将被导出到当前工作簿所在文件夹"  
    strPrompt = strPrompt & vbCrLf & vbCrLf  
    strPrompt = strPrompt & "输入文件名,带扩展名(例如,.png,.gif)."  
    strDefault = ""  
  
    '确定图片文件名及图片格式  
    Do  
        strChtName = Application.InputBox(strPrompt, "导出图表为图片",  
strDefault, , , , 2)  
        If Len(strChtName) = 0 Then GoTo ExitSub
```



```
If strChtName = "False" Then GoTo ExitSub
```

```
Select Case True
```

```
Case UCase$(Right(strChtName, 4)) = ".PNG"
```

```
Case UCase$(Right(strChtName, 4)) = ".GIF"
```

```
Case UCase$(Right(strChtName, 4)) = ".JPG"
```

```
Case UCase$(Right(strChtName, 4)) = ".JPE"
```

```
Case UCase$(Right(strChtName, 5)) = ".JPEG"
```

```
Case Else
```

```
strChtName = strChtName & ".png" '图片格式默认为 png
```

```
End Select
```

'调用自定义函数判断具有该名字的文件是否存在

```
If Not FileExists(ActiveWorkbook.Path & "\" & strChtName) Then Exit Do
```

```
strPrompt = "名为" & strChtName & "的文件已存在!"
```

```
strPrompt = "图表将被导出到当前工作簿所在文件夹"
```

```
strPrompt = strPrompt & vbCrLf & vbCrLf
```

```
strPrompt = strPrompt & "请选择一个唯一的文件名,带扩展名(例如,.png,.gif)."
```

```
strDefault = strChtName
```

```
Loop
```

'将图表导出为指定格式的图片并存储在当前工作簿所在文件夹

```
ActiveChart.Export ActiveWorkbook.Path & "\" & strChtName
```

```
ExitSub:
```

```
End Sub
```



'判断指定文件是否存在

```
Function FileExists(ByVal FileSpec As String) As Boolean
```

```
    Dim Attr As Long
```

```
    On Error Resume Next
```

'获取文件或文件夹属性

```
    Attr = GetAttr(FileSpec)
```

```
    If Err.Number = 0 Then
```

'不是文件夹

```
        FileExists = Not ((Attr And vbDirectory) = vbDirectory)
```

```
    End If
```

```
End Function
```

在运行代码之前，先选择要导出为图片的图表。

运行程序，会弹出如图 15.1 所示的对话框，要求你输入图片的文件名以及扩展名。如果没有输入扩展名，则使用默认的扩展名 PNG。

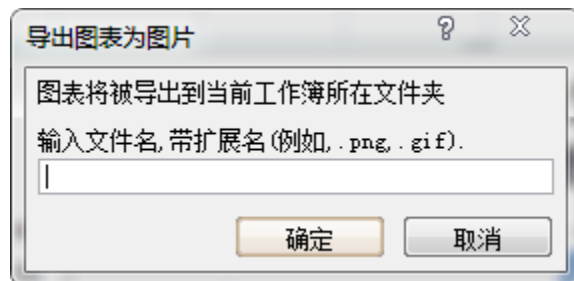


图 15.1

输入文件名后，单击“确定”，在当前工作簿所在文件夹中生成一个当前图表的图片文件。





本章内容 2018 年 5 月 3 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 16：将 Excel 图表
导出为图片（增强版）

16.将 Excel 图表导出为图片（增强版）

本节介绍的程序也是 [Jon Peltier](#) 编写的，我对代码稍作修改并添加了一些注释，以方便理解。

```
Sub ExportChartAsPicture()  
    Dim strChtName As String  
    Dim strFileName As String  
    Dim strPathName As String  
    Dim strPrompt As String  
    Dim strCurDir As String  
    Dim lngOverwrite As Long  
  
    '获取当前路径  
    strCurDir = CurDir  
  
    If ActiveSheet Is Nothing Then GoTo ExitSub  
    If ActiveChart Is Nothing Then GoTo ExitSub  
  
    '获取工作簿所在路径并修改当前路径  
    strPathName = ActiveWorkbook.Path  
    If Len(strPathName) > 0 Then  
        '改变当前路径  
        ChDrive strPathName  
        '设置为缺省路径  
        ChDir strPathName
```



```

End If

strFileName = "MyChartPic.png"

Do
    strChtName = Application.GetSaveAsFilename(strFileName, "所有文件 (*.*)", *.*", , _
        "浏览文件夹并输入文件名")
    If Len(strChtName) = 0 Then GoTo ExitSub
    If strChtName = "False" Then GoTo ExitSub

    Select Case True
        Case UCase$(Right(strChtName, 4)) = ".PNG"
        Case UCase$(Right(strChtName, 4)) = ".GIF"
        Case UCase$(Right(strChtName, 4)) = ".JPG"
        Case UCase$(Right(strChtName, 4)) = ".JPE"
        Case UCase$(Right(strChtName, 5)) = ".JPEG"
        Case Else
            If Right$(strChtName, 1) <> "." Then strChtName = strChtName &
                "."
            strChtName = strChtName & ".png" '图片格式默认为 png
    End Select

    '调用自定义函数判断具有该名字的文件是否存在
    If Not FileExists(strChtName) Then Exit Do

    strFileName = FullNameToFileName(strChtName)
    strPathName = FullNameToPath(strChtName)

    strPrompt = "名为" & strChtName & "的文件已存在于" & strPathName &
        ""中."

    strPrompt = strPrompt & vbCrLf & vbCrLf & "你想覆盖现有文件吗?"

```



```
IngOverwrite = MsgBox(strPrompt, vbYesNoCancel + vbQuestion, "图片文件已存在")
```

```
Select Case IngOverwrite
```

```
Case vbYes
```

```
Exit Do
```

```
Case vbNo
```

```
'空
```

```
Case vbCancel
```

```
GoTo ExitSub
```

```
End Select
```

```
Loop
```

```
'将图表导出为指定格式的图片并存储设定的文件夹中
```

```
ActiveChart.Export strChtName
```

```
ExitSub:
```

```
'恢复为原有路径
```

```
ChDrive strCurDir
```

```
ChDir strCurDir
```

```
End Sub
```

```
'判断指定文件是否存在
```

```
Function FileExists(ByVal FileSpec As String) As Boolean
```

```
Dim Attr As Long
```

```
On Error Resume Next
```

```
'获取文件或文件夹属性
```



```

Attr = GetAttr(FileSpec)

If Err.Number = 0 Then
    '不是文件夹
    FileExists = Not ((Attr And vbDirectory) = vbDirectory)
End If
End Function

'获取文件名,包括扩展名
Function FullNameToFileName(strFullName As String) As String
    Dim i As Integer
    Dim strTest As String
    '是否包含方括号
    If InStr(1, strFullName, "[") > 0 Then
        i = InStr(1, strFullName, "[")
        strTest = Mid$(strFullName, i + 1, InStr(1, strFullName, "]") - i - 1)
    Else
        '文件路径中反斜杠的位置
        For i = Len(strFullName) To 1 Step -1
            If Mid$(strFullName, i, 1) = "\" Then Exit For
        Next i
        strTest = Mid$(strFullName, i + 1, Len(strFullName) - i)
    End If
    FullNameToFileName = strTest
End Function

'获取文件所在的路径
Function FullNameToPath(strFullName As String) As String
    '不包括尾斜杠
    Dim i As Integer
    '获取文件路径中反斜杠位置
    For i = Len(strFullName) To 1 Step -1

```



```

        If Mid$(strFullName, i, 1) = "\" Then Exit For
    Next
    If i < 1 Then
        FullNameToPath = ""
    Else
        FullNameToPath = Mid$(strFullName, 1, i - 1)
    End If
End Function

```

在运行代码之前，先选择要导出为图片的图表。

运行程序，会弹出如图 16.1 所示的对话框，可以选择在哪里保存图片文件，默认为当前工作簿所在的文件夹。如果没有输入扩展名，则使用默认的扩展名 PNG。

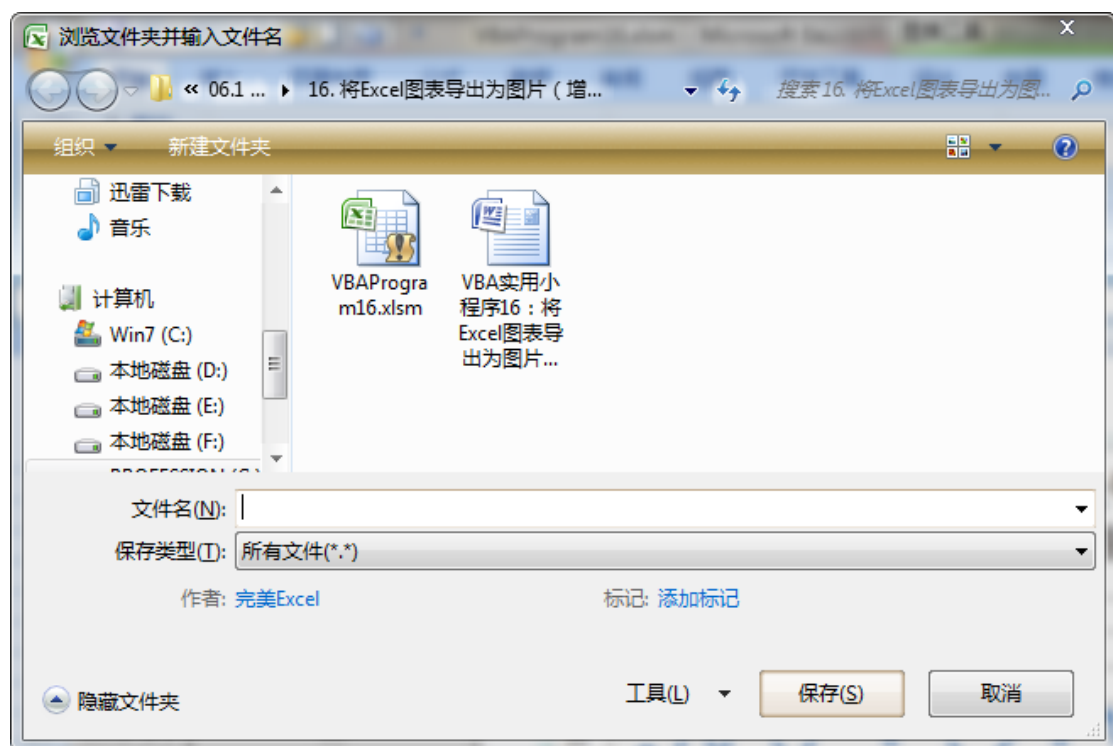


图 16.1

输入文件名后，单击“确定”，在所选文件夹中生成一个当前图表的图片文件。如果文件名与文件夹中已有文件相同，那么可以选择是否覆盖该文件。





本章内容 2018 年 5 月 10 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 17：从关闭的工作簿中取值

17.从关闭的工作簿中取值

有时，我们想要从其他工作簿中获取数值但又不想打开该工作簿，可以使用下面的 VBA 程序来实现。

从关闭的工作簿中取值的自定义函数：

```
Private Function GetData(Path, File, Sheet, Address)

    Dim Data As String

    Data = "" & Path & "[" & File & "]" & Sheet & "!" & _
        Range(Address).Range("A1").Address(, , xlR1C1)

    GetData = ExecuteExcel4Macro(Data)

End Function
```

自定义函数中使用了 XLM 宏从关闭的文件中提取数据。

下面的代码演示了使用 GetData 函数从指定的关闭工作簿中取值：

```
Sub GetDataDemo()

    Dim FilePath As String, Address As String

    Dim Row As Long, Column As Long

    '可以修改下面的常量以满足实际需要

    Const FileName As String = "Book1.xlsx"

    Const SheetName As String = "Sheet1"

    Const NumRows As Long = 10
```



```

Const NumColumns As Long = 10
FilePath = ActiveWorkbook.Path & "\"

DoEvents
Application.ScreenUpdating = False

If Dir(FilePath & FileName) = Empty Then
    MsgBox "没有找到工作簿" & FileName, , "文件不存在"
    Exit Sub
End If

For Row = 1 To NumRows
    For Column = 1 To NumColumns
        Address = Cells(Row, Column).Address
        Cells(Row, Column) = GetData(FilePath, FileName, SheetName, Address)
        Columns.AutoFit
    Next Column
Next Row

Application.ScreenUpdating = True
End Sub

```

可以根据实际需要，修改程序代码中的常量，以指定要从中获取值的工作簿以及哪个工作表和工作表中的行列。



本章内容 2018 年 5 月 19 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
**VBA 实用小程序 18：合并文件夹中的
所有工作簿**

18.合并文件夹中的所有工作簿

在当前工作簿所在的文件夹中，存在着一些其他的工作簿文件，我们想将这些工作簿中的工作表合并到当前工作簿中，可以使用下面的代码。

在当前工作簿的代码模块中输入下面的代码：

```
Sub CombineWorkbooks()  
    Dim strPath As String  
    Dim strFilename As String  
    Dim wkb As Workbook  
    Dim wks As Worksheet  
  
    Application.EnableEvents = False  
    Application.ScreenUpdating = False  
  
    '获取当前工作簿所在路径  
    strPath = ActiveWorkbook.Path  
    '获取当前工作簿所在文件夹中的 Excel 文件  
    strFilename = Dir(strPath & "\*.xls*", vbNormal)  
  
    '遍历文件夹直至找不到文件为止  
    Do Until strFilename = ""  
        '判断是否为当前工作簿
```



```

If strFilename <> ActiveWorkbook.Name Then
    '打开找到的工作簿并复制其工作表到当前工作簿
    Set wkb = Workbooks.Open(Filename:=strPath & "\" & strFilename)
    For Each wks In wkb.Worksheets
        wks.Copy
        After:=ThisWorkbook.Sheets(ThisWorkbook.Sheets.Count)
    Next wks
    '关闭打开的工作簿
    wkb.Close False
End If
'获取下一个文件
strFilename = Dir()
Loop

Application.EnableEvents = True
Application.ScreenUpdating = True
End Sub

```

代码中，我们指定变量 `strPath` 为当前工作簿所在的文件夹。可以将变量 `strPath` 指定的路径修改为想要合并的工作簿所在的文件夹路径，从而合并指定文件夹中的所有工作簿。



本章内容 2018 年 5 月 24 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 19：合并工作簿中的
所有工作表**

19.合并工作簿中的所有工作表

本程序假设当前工作簿中所有工作表都有相同的表结构，相同的列标题和列顺序。
新建一个名为 Main 的工作表，将所有工作表中的数据复制到新工作表 Main 中。

在标准模块中输入下面的代码：

```
Sub CombineWorksheets()  
    Dim wbk As Workbook  
    Dim wks As Worksheet  
    Dim wksMain As Worksheet  
    Dim rng As Range  
    Dim lngColCount As Long  
  
    '设置变量 wbk 为当前工作簿  
    Set wbk = ActiveWorkbook  
  
    '如果工作表名已存在,则给出提示信息  
    For Each wks In wbk.Worksheets  
        If wks.Name = "Main" Then  
            MsgBox "已经存在一个名为 Main 的工作表." & vbCrLf & _  
                "请删除或者重命名这个工作表." & vbCrLf & _  
                "我们将使用工作表 Main 合并其他工作表.", _  
                vbOKOnly + vbExclamation, "错误"  
        End If  
    End For  
End Sub
```



```

Exit Sub

End If

Next wks

Application.ScreenUpdating = False

'添加新工作表并放置在最后
With wbk
    Set wksMain = .Worksheets.Add(After:=.Worksheets(.Worksheets.Count))
End With

'将新工作表命名为 Main
wksMain.Name = "Main"

'从第 1 个工作表中获取列标题和第 1 行的列数
Set wks = wbk.Worksheets(1)
lngColCount = wks.Cells(1, Columns.Count).End(xlToLeft).Column

'将列标题输入到新工作表中
With wksMain.Cells(1, 1).Resize(1, lngColCount)
    .Value = wks.Cells(1, 1).Resize(1, lngColCount).Value
    .Font.Bold = True
End With

'遍历工作簿中的工作表
For Each wks In wbk.Worksheets
    '若工作表为新添加的工作表,则退出程序
    If wks.Index = wbk.Worksheets.Count Then
        Exit For
    End If
    '获取工作表中的数据区域,从第 2 行开始
    With wks

```



```

        Set rng = .Range(.Cells(2, 1), .Cells(Rows.Count, 1).End(xlUp).Resize(,
lngColCount))
    End With
    '将获取的数据输入到新添加的工作表
    wksMain.Cells(Rows.Count, 1).End(xlUp).Offset(1).Resize(rng.Rows.Count,
rng.Columns.Count).Value = rng.Value
    Next wks

    '自动调整列宽
    wksMain.Columns.AutoFit

    Application.ScreenUpdating = True
End Sub

```





本章内容 2018 年 5 月 29 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
**VBA 实用小程序 20：保护含有公式
的单元格**

20.保护含有公式的单元格

很多时候，你都不想别人修改或者删除你的工作表中含有公式的单元格，因为这样会打乱你的工作表结构。

下面的代码锁定当前工作表中含有公式的所有单元格，你不能删除或者修改这些单元格，除非你解除工作表保护。当然，你可以修改或者删除其他单元格（不能删除公式单元格所在的行）。

```
Sub LockCellsWithFormulas()  
    With ActiveSheet  
        .Unprotect  
        .Cells.Locked = False  
        .Cells.SpecialCells(xlCellTypeFormulas).Locked = True  
        .Protect AllowDeletingRows:=True  
    End With  
End Sub
```

运行上面的代码后，当你要删除工作表中含有公式的单元格或该单元格所在行时，Excel 会弹出下图 20.1 所示的提示信息。

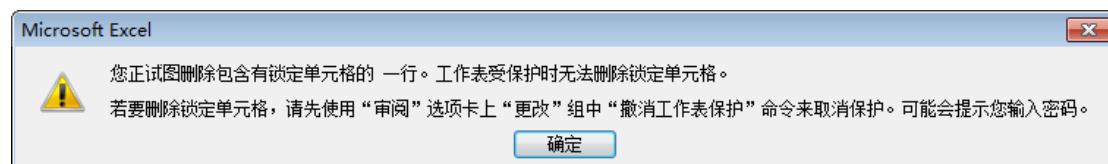


图 20.1



当你要修改工作表中含有公式的单元格时，Excel 会弹出下图 20.2 所示的提示信息。

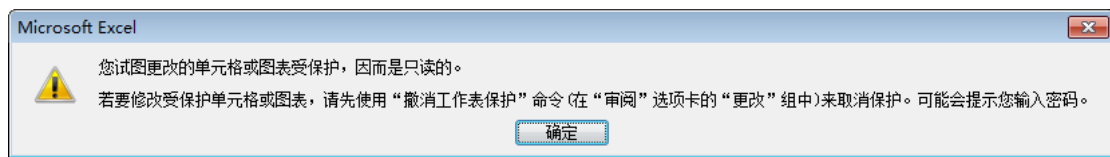


图 20.2



本章内容 2018 年 6 月 3 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 21: 项目管理|根据
工作分解结构自动编号

21.项目管理 | 根据工作分解结构自动编号

工作分解结构是一种项目管理技术，就是把一个项目，按一定的原则分解，项目分解成任务，任务又分解成子任务，直到分解不下去为止。工作分解结构简称 WBS，是制定项目进度计划、资源需求、成本预算、风险管理计划和采购计划等工作的重要基础。

例如，下图 21.1 所示，在列 B 中，我们将工作进行了分解，缩进量多的工作任务是前面缩进量少或没有缩进的工作任务的子任务，即利用缩进的形式来表达 WBS 的层次结构图。

	A	B	
1	序号	工作任务	
2		基础工程	
3		放线	
4		开挖	
5		钢筋制安	
6		下料	
7		绑扎	
8		建筑工程	
9		混凝土	
10		钢筋绑扎	
11		立模	
12		浇筑	
13		养护	
14		拆模	
15		外墙装饰	
16		室内装饰	
17		室外工程	
18		项目结束	
19			

图 21.1



我们可以使用 VBA 代码，给上面的 WBS 层次结构进行自动编号，如图 22.2 所示。

	A	B	C
1	序号	工作任务	
2	1	基础工程	
3	1.1	放线	
4	1.2	开挖	
5	1.3	钢筋制安	
6	1.3.1	下料	
7	1.3.2	绑扎	
8	2	建筑工程	
9	2.1	混凝土	
10	2.1.1	钢筋绑扎	
11	2.1.2	立模	
12	2.1.3	浇筑	
13	2.1.4	养护	
14	2.1.5	拆模	
15	3	外墙装饰	
16	4	室内装饰	
17	5	室外工程	
18		项目结束	
19			

图 22.2

代码如下：

```
Sub WBSNumber()
    '假设工作表中的布局为:
    '第 1 行为列标题
    '列 A 为工作分解结构的编号
    '列 B 包含工作任务描述,并进行适当的缩进

    On Error Resume Next

    '隐藏分页线并禁用屏幕更新
    Application.ScreenUpdating = False
    ActiveSheet.DisplayPageBreaks = False

    '将编号列格式化为文本
```



```
ActiveSheet.Range("A:A").NumberFormat = "@"
```

```
Dim r As Long '行计数
```

```
Dim depth As Long '每项任务的缩进数
```

```
Dim wbsarray() As Long '存放每层级任务的编号的数组
```

```
Dim basenum As Long '编号中的整数
```

```
Dim wbs As String '存放每层级任务的编号
```

```
Dim aloop As Long '循环变量
```

```
r = 2 '开始行
```

```
basenum = 0 '初始整数
```

```
ReDim wbsarray(0 To 0) As Long '初始化存放编号的数组
```

```
'遍历含有项目工作任务的单元格,直至遇到内容为"项目结束"的单元格
```

```
Do While Cells(r, 2) <> "项目结束"
```

```
    '忽略空工作任务
```

```
    If Cells(r, 2) <> "" Then
```

```
        '跳过隐藏的行
```

```
        If Rows(r).EntireRow.Hidden = False Then
```

```
            '获取列 B 中工作任务的缩进层级
```

```
            depth = Cells(r, 2).IndentLevel
```

```
            '如果没有缩进
```

```
            If depth = 0 Then
```

```
                '编号增加 1
```

```
                basenum = basenum + 1
```

```
                wbs = CStr(basenum)
```

```
                ReDim wbsarray(0 To 0)
```

```
            '如果有缩进,则是子任务或子-子任务
```

```
            Else
```

```
                '根据当前的缩进数重新调整数组大小
```



```

ReDim Preserve wbsarray(0 To depth) As Long
depth = depth - 1
If wbsarray(depth) <> 0 Then
    wbsarray(depth) = wbsarray(depth) + 1
Else
    wbsarray(depth) = 1
End If

If wbsarray(depth + 1) <> 0 Then
    For aloop = depth + 1 To UBound(wbsarray)
        wbsarray(aloop) = 0
    Next aloop
End If

wbs = CStr(basenum)

For aloop = 0 To depth
    wbs = wbs & "." & CStr(wbsarray(aloop))
Next aloop
End If

'使用编号填充单元格
Cells(r, 1).Value = wbs

'消除"数字作为文本存储"错误
Cells(r, 1).Errors(xlNumberAsText).Ignore = True

'如果有子任务,则应用格式
If Cells(r + 1, 2).IndentLevel > Cells(r, 2).IndentLevel Then
    Cells(r, 1).Font.Bold = True
    Cells(r, 2).Font.Bold = True

```



```
Else
    Cells(r, 1).Font.Bold = False
    Cells(r, 2).Font.Bold = False
End If

If Cells(r, 2).IndentLevel = 0 Then
    Cells(r, 1).Font.Bold = True
    Cells(r, 2).Font.Bold = True
End If
End If
End If

'增加行数
r = r + 1

Loop

'恢复屏幕更新
Application.ScreenUpdating = True
End Sub
```

注意，列 B 中的数据应进行正确的缩进，才能获得正确的编号。





本章内容 2018 年 6 月 8 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
VBA 实用小程序 22：查找工作表中
含有指定日期的所有单元格

22. 查找工作表中含有指定日期的所有单元格

有时候，我们想要知道工作表中含有某日期的单元格有哪些，那么，下面的代码可以帮助你实现。

在标准模块中输入下面的代码：

```
Function FindDates(ByRef FindDate As Date, Optional LookInRange As Range) As Range

    '声明变量

    Dim rngTarget As Range, strFirstAddr As String

    Dim rngFind As Range, rngLastCell As Range

    '如果没有指定区域,则设置为工作表中当前已使用区域

    If LookInRange Is Nothing Then

        Set LookInRange = ActiveSheet.UsedRange

    End If

    With LookInRange

        '获取查找区域的最后一个单元格

        Set rngLastCell = .Cells(.Cells.Count)

        '从最后一个单元格开始向前查找

        Set rngTarget = .Find(What:=FindDate, _

                               After:=rngLastCell, _

                               LookIn:=xlFormulas, _
```



```

SearchOrder:=xlByRows)

'如果找到单元格
If Not rngTarget Is Nothing Then
    Set rngFind = rngTarget
    strFirstAddr = rngTarget.Address
    Do
        '将找到的单元格合并
        Set rngFind = Union(rngFind, rngTarget)
        '查找下一个单元格
        Set rngTarget = .FindNext(After:=rngTarget)
        '如果找不到单元格了或者找到的单元格是最先找到的单元格,则退出循环
    Loop Until rngTarget Is Nothing Or rngTarget.Address = strFirstAddr
    End If
End With

'返回值
If rngFind Is Nothing Then
    Set FindDates = Nothing
Else
    Set FindDates = rngFind
End If

Set LookInRange = Nothing
Set rngTarget = Nothing
Set rngFind = Nothing
Set rngLastCell = Nothing
End Function

```

给上述自定义函数传递要查找的日期以及单元格区域,将返回单元格区域中所有含有该日期的单元格。如果没有指定单元格区域,那么使用当前工作表中已使用的区域。



本章内容 2018 年 6 月 26 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 23：查找并返回满足条件的多个值

23.查找并返回满足条件的多个值

在 Excel 中，查找满足条件的数据时，一般只能返回单个值。要一次返回满足条件的多个值，需要使用复杂的数组公式。

下面的 VBA 自定义函数，可以一次返回满足条件的多个值。在 VBE 标准模块中，输入下面的代码：

```
Function GetDatas(LookupValue As String, _  
                  LookupRange As Range, _  
                  ColNum As Integer)  
  
    Dim i As Long  
    Dim Result As String  
  
    For i = 1 To LookupRange.Columns(1).Cells.Count  
        If LookupRange.Cells(i, 1) = LookupValue Then  
            Result = Result & " " & LookupRange.Cells(i, ColNum) & ","  
        End If  
    Next i  
  
    GetDatas = Left(Result, Len(Result) - 1)  
End Function
```

然后，在工作表中，像输入普通的函数一样，输入=GetDatas 并输入必需的参数。该函数有 3 个参数：



- 参数 LookupValue: 需要查找的字符串
- 参数 LookupRange: 被查找的单元格区域
- 参数 ColNum: 被查找的单元格区域中返回找到的数据的列

如下图 23.1 所示。

	A	B	C	D	E	F
1						
2		超市名称	特色物品		超市名称	特色物品
3		东区	香蕉		东区	香蕉, 西红柿, 冬瓜
4		西区	苹果		南区	核桃, 西兰花, 杏仁, 甜瓜
5		北区	榴莲		西区	苹果, 葛根
6		南区	核桃		北区	榴莲, 黄瓜
7		中区	香梨		中区	香梨, 脐橙
8		东区	西红柿			
9		北区	黄瓜			
10		南区	西兰花			
11		西区	葛根			
12		中区	脐橙			
13		东区	冬瓜			
14		南区	杏仁			
15		南区	甜瓜			

图 23.1

数据列表为单元格区域 B3:C15, 这是被查找的区域。在列 E 的单元格中列出了查找的字符串, 列 F 中为查找到的数据。在单元格 F3 中的公式为:

```
=getdatas(E3,$B$3:$C$15,2)
```

下拉至单元格可 F7。



本章内容 2018 年 6 月 30 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 24: 设置页眉/页脚
的高效代码**

24.设置页眉/页脚的高效代码

通常，在 VBA 中我们可以利用 PageSetup 对象的属性来设置页眉或者页脚。例如，下面的代码来自定义页脚：

```
Sub SetFooters()  
    With Worksheets("Sheet1").PageSetup  
        .LeftFooter = "完美 Excel"  
        .RightFooter = "excelperfect"  
    End With  
End Sub
```

然而，在改变 PageSetup 对象的任何属性时，Excel 都会重新编制页码，检查自动缩放或自动分页功能是否也需要随之调整，还要与打印机驱动程序进行通信，从而导致速度变慢。

为避免这种情形，可以使用 XLM 函数 PAGE.SETUP。下面的代码与上文代码效果相同：

```
Sub SetFooters()  
    Worksheets("Sheet1").Activate  
    ExecuteExcel4Macro "PAGE.SETUP(, ""&L 完美 Excel &R excelperfect""")  
End Sub
```

代码中使用的 PAGE.SETUP 函数可以在 Microsoft 的 macrofun.hlp 文件找到详细信息，其语法如下：



```
PAGE.SETUP(head, foot, left, right, top, bot, size, h_cntr, v_cntr, orient, paper_size,  
scale, pg_num, bw_chart, quality, head_margin, foot_margin, draft)
```



本章内容 2018 年 7 月 10 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
**VBA 实用小程序 25：删除工作表中的
所有形状**

25.删除工作表中的所有形状

下面的代码删除当前工作表中的所有形状：

```
Sub DeleteAllShapesInWorksheet()  
    Dim shp As Shape  
  
    For Each shp In ActiveSheet.Shapes  
        shp.Delete  
    Next shp  
End Sub
```

然而，有时我们并不想将所有形状都删除，想要保留指定类型的形状，例如图表或者单元格批注（Excel 将它们都当作为形状）。下面的代码删除指定类型形状（图表和单元格批注）之外的所有形状：

```
Sub DeleteAllShapesExcludeCertainShapes()  
    Dim shp As Shape  
  
    For Each shp In ActiveSheet.Shapes  
        If shp.Type <> msoChart And shp.Type <> msoComment Then  
            shp.Delete  
        End If  
    Next shp  
End Sub
```



在上面的代码中，我们使用 IF 语句来判断形状的类型，避免删除指定类型的形状。下表列出了所有的形状类型。

msoShapeType	枚举值	类型
msoAutoShape	1	自选图形
msoCallout	2	标注
msoCanvas	20	画布
msoChart	3	图表
msoComment	4	批注
msoContentApp	27	Content Office Add-in
msoDiagram	21	图示
msoEmbeddedOLEObject	7	嵌入的OLE对象
msoFormControl	8	窗体控件
msoFreeform	5	任意多边形
msoGraphic	28	图形
msoGroup	6	组合
msoIgxGraphic	24	SmartArt图形
msoInk	22	墨迹
msoInkComment	23	墨迹批注
msoLine	9	线条
msoLinkedGraphic	29	链接的图形
msoLinkedOLEObject	10	链接的OLE对象
msoLinkedPicture	11	链接的图片
msoMedia	16	媒体
msoOLEControlObject	12	OLE控件对象
msoPicture	13	图片
msoPlaceholder	14	占位符
msoScriptAnchor	18	脚本定位标记
msoShapeTypeMixed	-2	混合形状类型
msoTable	19	表
msoTextBox	17	文本框
msoTextEffect	15	文本效果
msoWebVideo	26	网络视频



本章内容 2018 年 7 月 24 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 26：将工作簿生成
PDF 文件

26.将工作簿生成 PDF 文件

下面的代码将当前工作簿生成成为指定名称的 PDF 文件：

```
Sub CovertWorkbookToPDF()  
    Dim FileName As String  
  
    '使用相应的参数调用生成 PDF 的函数  
    FileName = CreatePDF(Source:=ActiveWorkbook, _  
                        FixedFilePathName:="", _  
                        OverwriteIfFileExist:=True, _  
                        OpenPDFAfterPublish:=True)  
  
    '可以在参数 FixedFilePathName 中指定带路径的文件名,例如  
    'FixedFilePathName:="D:\Test\ThisPDFFile.pdf"  
  
    If FileName <> "" Then  
        '如果 FileName 存在,则可以在此编写处理 PDF 的代码  
        '例如,作为邮件来发送 PDF 文件  
    Else  
        MsgBox "不能创建 PDF 文件, 可能的原因是:" & vbNewLine & _  
            "1.没有安装相应的 Microsoft Add-in" & vbNewLine & _  
            "2.在另存为对话框中选择了取消" & vbNewLine & _  
            "3.保存文件的路径不正确" & vbNewLine & _
```



"4.不想覆盖已有的 PDF 文件"

End If

End Sub

Function CreatePDF(Source As Object, _

FixedFilePathName As String, _

OverwriteIfFileExist As Boolean, _

OpenPDFAfterPublish As Boolean) As String

Dim FileFormatstr As String

Dim Fname As Variant

If FixedFilePathName = "" Then

'打开另存为对话框,输入 PDF 文件名

FileFormatstr = "PDF Files (*.pdf), *.pdf"

Fname = Application.GetSaveAsFilename("", FileFilter:=FileFormatstr, _
Title:="创建 PDF 文件")

'如果取消该对话框则退出函数

If Fname = False Then Exit Function

Else

Fname = FixedFilePathName

End If

'如果 OverwriteIfFileExist = False

'测试是否该 PDF 文件已存在

'如果为 True 则退出函数

If OverwriteIfFileExist = False Then

If Dir(Fname) <> "" Then Exit Function

End If

'PDF 文件名正确,发布 PDF 文件



```

On Error Resume Next

Source.ExportAsFixedFormat _
    Type:=xlTypePDF, _
    FileName:=Fname, _
    Quality:=xlQualityStandard, _
    IncludeDocProperties:=True, _
    IgnorePrintAreas:=False, _
    OpenAfterPublish:=OpenPDFAfterPublish

On Error GoTo 0

'如果发布成功,函数将返回该文件名

If Dir(Fname) <> "" Then
    CreatePDF = Fname
End If

End Function

```

可以指定生成的 PDF 文件保存的路径和文件名。如果没有指定，那么会弹出“创建 PDF 文件”对话框（与“另存为”对话框相同），可以选择保存的文件夹并指定名称。

将上述代码复制到要生成 PDF 的工作簿的 VBE 标准模块中，运行 CovertWorkbookToPDF 过程，选择相应的文件夹并输入文件名称，生成该工作簿的 PDF 文件。





本章内容 2018 年 8 月 1 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 27: 在 Excel 中创建
可视化的标签云

27.在 Excel 中创建可视化的标签云

在网页中，我们经常会看到标签云，关键词或标签的使用数量越多，该关键词或标签在网页中显示的字体就越大。在 Excel 中，也可以创建类似的效果，如图 27.1 所示。

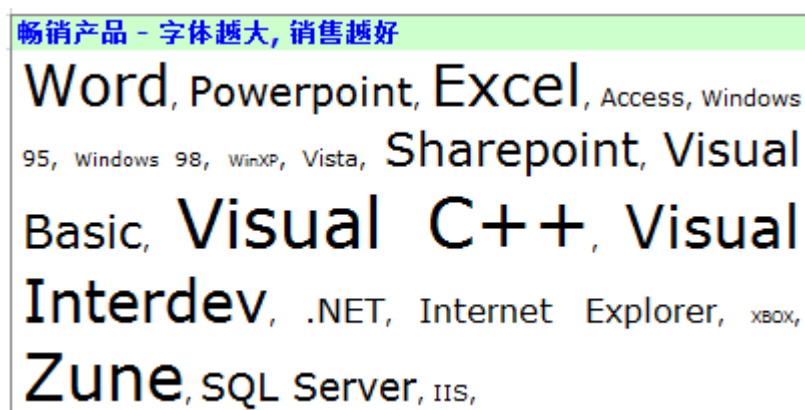


图 27.1

这是我在 [chandoo.org](#) 上学到的一段代码，在这里与大家分享。图 2 为数据表，一列为创建标签云的文本，另一列是决定其大小的数字（数字越大，在标签云中的文本显示就越大）。



产品名称	盈利能力
Word	104%
Powerpoint	68%
Excel	105%
Access	22%
Windows 95	18%
Windows 98	8%
WinXP	3%
Vista	15%
Sharepoint	75%
Visual Basic	77%
Visual C++	141%
Visual Interdev	114%
.NET	52%
Internet Explorer	43%
XBOX	0%
Zune	126%
SQL Server	60%
IIS	13%

图 27.2

下面是创建标签云的代码：

' 本过程基于列表标签名、标签重要性创建标签云
' 标签重要性能够设置任意值, 这里以 8 至 20 之间的值为标准

```
Sub createCloud()
    On Error GoTo tackle_this
    Dim cntr, i, cell, strt, taglist
    Dim size As Integer

    size = Selection.Count / 2

    Dim tags() As String
    Dim importance()

    ReDim tags(1 To size) As String
    ReDim importance(1 To size)

    Dim minImp As Integer
    Dim maxImp As Integer
```



```

cntr = 1
i = 1

For Each cell In Excel.Selection
    If cntr Mod 2 = 1 Then
        taglist = taglist & cell.Value & ", "
        tags(i) = cell.Value
    Else
        importance(i) = Val(cell.Value)
        If importance(i) > maxImp Then
            maxImp = importance(i)
        End If
        If importance(i) < minImp Then
            minImp = importance(i)
        End If
        i = i + 1
    End If
    cntr = cntr + 1
Next cell

' 在单元格 E26 中粘贴值
Range("e26").Select
ActiveCell.Value = taglist
ActiveCell.Font.size = 8

strt = 1

For i = 1 To size
    With ActiveCell.Characters(Start:=strt,
Length:=Len(tags(i))).Font
        .size = 6 + Math.Round((importance(i) - minImp)
/ (maxImp - minImp) * 14, 0)
    End With
    strt = strt + Len(tags(i))
Next i

```



```

        .Strikethrough = False
        .Superscript = False
        .Subscript = False
        .OutlineFont = False
        .Shadow = False
        .Underline = xlUnderlineStyleNone
        .ColorIndex = xlAutomatic
    End With
    strt = strt + Len(tags(i)) + 2
Next i

Exit Sub
tackle_this:
    ' 这里放置错误处理
    MsgBox "需要选择一个表或区域以便创建标签云", vbCritical +
vbOKOnly, _
        "Wow, 似乎发生了错误!"
End Sub

```

运行上述代码前，应该先选择要创建标签云的单元格区域或列表。正如图 2 所示，示例使用了含 2 列数据的列表。

代码中，使用了两个数组，一个数组用于存储每个标签的文本，另一个数组用于存储相对应的数值。

代码中，将所有标签文本放置在指定的单元格中，示例中为单元格 E26，然后基于 6 至 20 之间的标准值设置字体大小。

你可以根据实际情况，对代码略作修改后适应具体的情形。



本章内容 2018 年 8 月 10 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 28：根据编号自动
缩进文字**

28.根据编号自动缩进文字

在 21：项目管理 | 根据工作分解结构自动编号中，我们根据工作任务文本的缩进量来自动进行编号，如图 28.1 所示。

	A	B	C
1	序号	工作任务	
2	1	基础工程	
3	1.1	放线	
4	1.2	开挖	
5	1.3	钢筋制安	
6	1.3.1	下料	
7	1.3.2	绑扎	
8	2	建筑工程	
9	2.1	混凝土	
10	2.1.1	钢筋绑扎	
11	2.1.2	立模	
12	2.1.3	浇筑	
13	2.1.4	养护	
14	2.1.5	拆模	
15	3	外墙装饰	
16	4	室内装饰	
17	5	室外工程	
18		项目结束	
19			

图 28.1

在文章发表后，一位网友问：能不能反向实现，即有正确的编号，让文字列自动根据编号层次缩进？

下面，我来介绍实现代码。

仍然以前面的数据为例，如图 28.2 所示。



	A	B	C
1	序号	工作任务	
2	1	基础工程	
3	1.1	放线	
4	1.2	开挖	
5	1.3	钢筋制安	
6	1.3.1	下料	
7	1.3.2	绑扎	
8	2	建筑工程	
9	2.1	混凝土	
10	2.1.1	钢筋绑扎	
11	2.1.2	立模	
12	2.1.3	浇筑	
13	2.1.4	养护	
14	2.1.5	拆模	
15	3	外墙装饰	
16	4	室内装饰	
17	5	室外工程	
18		项目结束	
19			
20			

图 28.2

根据列 A 中的序号，缩进列 B 中的相应文本。其中，子任务即有一个小数点的，缩进一级，再下一级子任务即有两个小数点的，缩进二级，依此类推。

下面是代码：

```
Sub WBSIndent ()
    '假设工作表中的布局为：
    '第 1 行为列标题
    '列 A 为工作分解结构的编号
    '列 B 包含工作任务描述

    On Error Resume Next

    '隐藏分页线并禁用屏幕更新
    Application.ScreenUpdating = False
    ActiveSheet.DisplayPageBreaks = False
```



```

'将编号列格式化为文本
ActiveSheet.Range("A:A").NumberFormat = "@"

Dim r As Long
Dim iIndent As Integer

r = 2 '开始行

Do While Cells(r, 2) <> "项目结束"
    '忽略空工作任务
    If Cells(r, 1) <> "" Then
        '跳过隐藏的行
        If Rows(r).EntireRow.Hidden = False Then
            iIndent = DotNum(Cells(r, 1))
            Cells(r, 2).IndentLevel = iIndent
        End If
    End If
    r = r + 1
Loop

'恢复屏幕更新
Application.ScreenUpdating = True
End Sub

'统计文本中的小数点个数
Function DotNum(str As String)
    Dim i As Integer
    Dim iNum As Integer

    For i = 1 To Len(str)
        If Mid(str, i, 1) = "." Then iNum = iNum + 1
    Next i

```



```
DotNum = iNum  
End Function
```

其中，使用了一个自定义函数 **DotNum** 来统计列 **A** 单元格中的小数点个数，从而确定缩进的层级。代码运行后的效果如上图 **1** 所示。



本章内容 2018 年 8 月 16 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 29：合并单元格并
保留所合并单元格的全部数据

29. 合并单元格并保留所合并单元格的全部数据

在 Excel 中，使用合并单元格功能时，最后的结果将只是所合并区域左上角单元格中的数据，如图 29.1 所示。

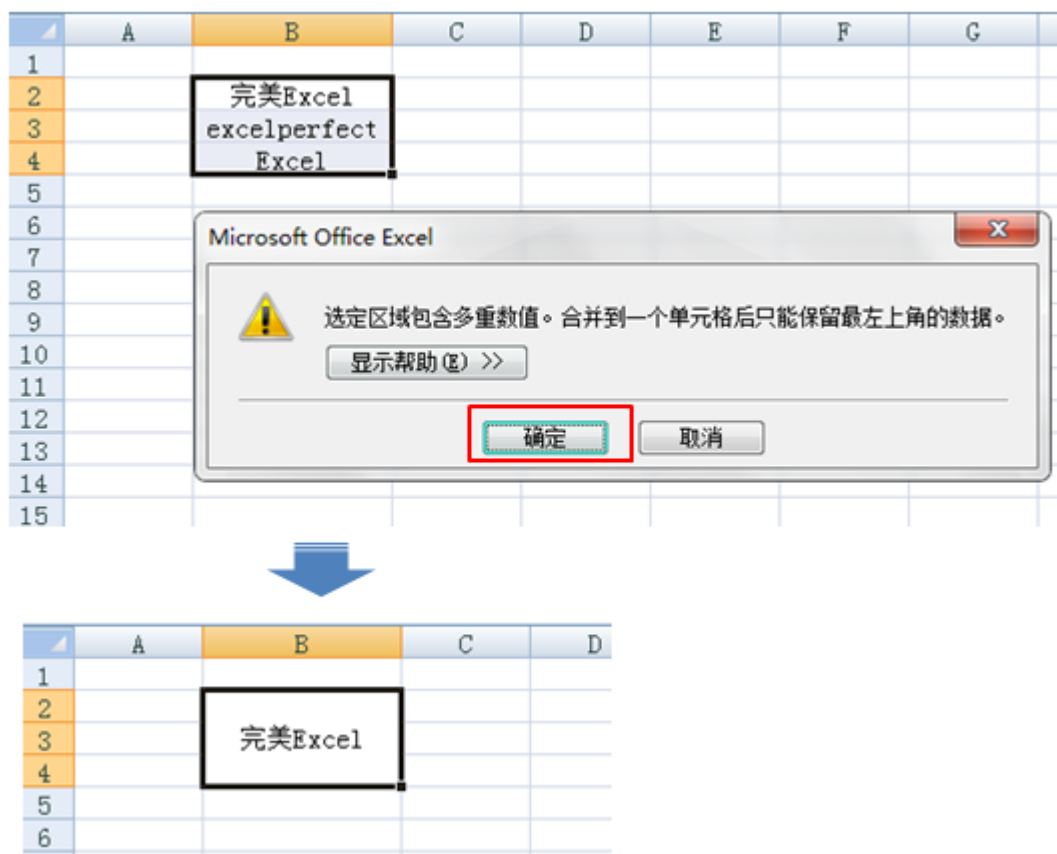


图 29.1

下面的程序在合并单元格后将保留合并单元格区域中所有单元格中的数据。

```
Sub MergeCellsAndValues()  
    Dim rng As Range
```



```

Dim strResult As String

On Error Resume Next

For Each rng In Selection
    '可以将" "更换为你自己的分隔符号
    strResult = strResult & rng.Value & " "
Next rng

strResult = Left(strResult, Len(strResult) - 2)

With Selection
    .Clear
    .Cells(1).Value = strResult
    .Merge
    .HorizontalAlignment = xlLeft
    .VerticalAlignment = xlCenter
    .WrapText = True
End With
End Sub

```

代码首先遍历所要合并的单元格并将这些单元格中的值连接成一个字符串存储在指定的变量中，示例中使用空格（" "）来分隔各单元格的数据，你可以修改为其他分隔符，例如逗号。代码运行后的效果如图 29.2 所示。

	A	B	C
1			
2		完美Excel	
3		excelperfect	
4		Exce	
5			
6			

图 29.2



本章内容 2018 年 8 月 22 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 30：将数字转换成中文

30.将数字转换成中文

有时候，我们可能想要将阿拉伯数字转换成中文数字表示。在 [dailydoseofexcel.com](#) 中，[DICK KUSLEIKA](#) 发表了 5 篇文章，为我们提供了转换为不同大小的英文版数字的实现过程。在这里，我将代码进行了修改，使之能够将数字转换成中文。

下面是程序代码：

```
Function ProcessTriplet(ByVal dNumber As Double, Optional  
ByVal strSuffix As String) As String  
    Dim varSingles As Variant  
    Dim varTens As Variant  
    Dim strReturn As String  
  
    varSingles = Split("零,一,二,三,四,五,六,七,八,九", ",")  
    varTens = Split("零,一十,二十,三十,四十,五十,六十,七十,八十,  
九十", ",")  
  
    If dNumber >= 100 Then  
        strReturn = strReturn & varSingles(dNumber \ 100) &  
"百"  
        dNumber = dNumber - (dNumber \ 100) * 100  
    End If  
  
    If dNumber > 9 Then  
        strReturn = strReturn & varTens(dNumber \ 10)  
        dNumber = dNumber - (dNumber \ 10) * 10  
    End If  
    strReturn = strReturn & varSingles(dNumber)  
    If strSuffix <> "" Then  
        strReturn = strReturn & strSuffix  
    End If  
    ProcessTriplet = strReturn  
End Function
```



```

End If

strReturn = strReturn & varSingles(dNumber)
strReturn = strReturn & strSuffix

ProcessTriplet = Trim(strReturn)
End Function

Function NumbersToWords(ByVal dNumbers As Double) As String
    Dim strReturn As String
    Dim dRemainder As Double
    Dim varTriplets As Variant
    Dim i As Long
    Dim lFixed As Long

    varTriplets = Split(",十,百,千,万,十万,百万,千万,亿,十亿,百  

    亿,千亿,万亿", ",")

    If dNumbers = 0 Then
        strReturn = "零"
    Else
        dRemainder = dNumbers
        For i = 12 To 0 Step -1
            If dRemainder >= 10 ^ i Then
                lFixed = Fix(Int(dRemainder + 0.5) / 10 ^ i)
                strReturn = strReturn & ProcessTriplet(lFixed,  

varTriplets(i))
                dRemainder = dRemainder - (lFixed * 10 ^ i)
                If dRemainder < 10 ^ (i - 1) Then strReturn =  

strReturn & "零"
            End If
        Next i
    End If
End Function

```




```

    If Right(strReturn, 1) = "零" Then strReturn =
Left(strReturn, Len(strReturn) - 1)

    NumbersToWords = Trim$(strReturn)
End Function

Sub test()
    Debug.Print NumbersToWords(100101)
End Sub

```

在程序中，使用 test 过程来测试代码，将数字 100101 转换成中文数字，代码运行后的效果如图 30.1 所示。

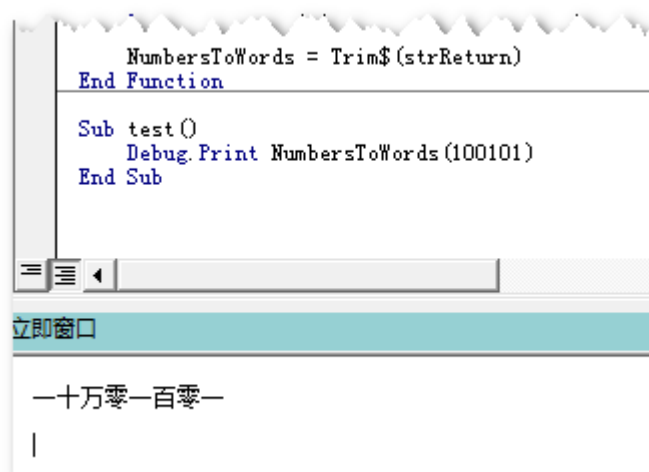


图 30.1

也可以在工作表中使用自定义函数，如图 30.2 所示。

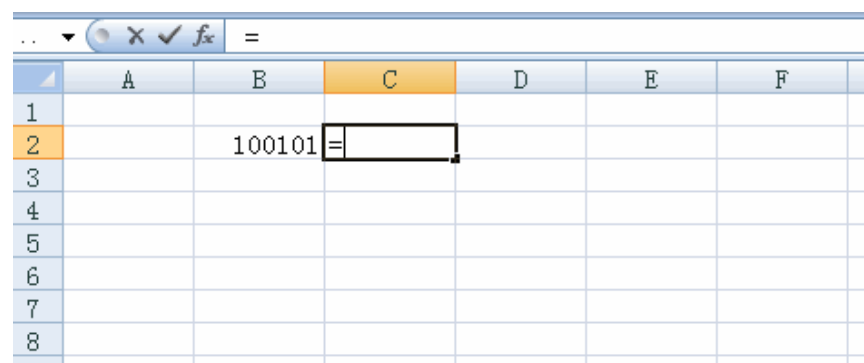


图 30.2（视频截图，完整视频参见[完美 Excel 微信公众号](#)）



当然，上面只是一种实现数字转换的代码，还可以编写其他的实现同样效果的代码，例如使用二维数组来实现转换。



本章内容 2018 年 8 月 28 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 31：将二维数组中的
元素连接成字符串**

31.将二维数组中的元素连接成字符串

使用 Join 函数能够将传递给它的数组按指定的分隔符连接成一个字符串，但是 Join 函数仅能处理一维数组。有时候，我们可能想要处理二维数组，将其中的元素连接成字符串。在 [dailydoseofexcel.com](#) 中，DICK KUSLEIKA 给出了一个实现此功能的函数，我将其辑录如此，与大家分享。

自定义函数 Join2DArray 的代码如下：

```
Function Join2DArray(ByVal varArray As Variant, _  
                    Optional ByVal strWordDelim As String =  
" ", _  
                    Optional ByVal strLineDelim As String =  
vbNewLine) As String  
    Dim i As Long, j As Long  
    Dim aReturn() As String  
    Dim aLine() As String  
  
    ReDim aReturn(LBound(varArray, 1) To UBound(varArray, 1))  
    ReDim aLine(LBound(varArray, 2) To UBound(varArray, 2))  
  
    For i = LBound(varArray, 1) To UBound(varArray, 1)  
        For j = LBound(varArray, 2) To UBound(varArray, 2)  
            aLine(j) = varArray(i, j)  
        Next j  
        aReturn(i) = Join(aLine, strWordDelim)  
    Next i
```



```

Join2DArray = Join(aReturn, strLineDelim)
End Function

```

在程序中，遍历二维数组的第一维，并使用参数 strLineDelim 指定的分隔符来连接每行；在内部的遍历中，使用参数 strWordDelim 指定的分隔符来连接第二维中的每个元素。

使用下面的 test 过程来测试代码：

```

Sub test()
    Dim a(1 To 2, 1 To 2) As String
    a(1, 1) = "完美"
    a(1, 2) = "Excel"
    a(2, 1) = "Excel"
    a(2, 2) = "Perfect"

    Debug.Print Join2DArray(a)
    Debug.Print
    Debug.Print Join2DArray(a, ",")
    Debug.Print
    Debug.Print Join2DArray(a, , "|")
    Debug.Print
    Debug.Print Join2DArray(a, ";", "||")
End Sub

```

代码运行后的结果如图 31.1 所示。

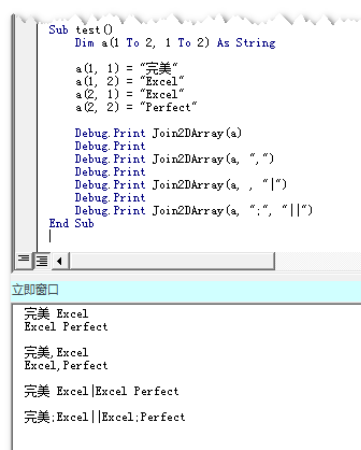


图 31.1



本章内容 2018 年 9 月 8 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 32：删除工作簿中的
所有 VBA 代码**

32.删除工作簿中的所有 VBA 代码

在开发工作簿时，我们往往会创建很多代码模块，不仅有标准模块，还有用户窗体和其他的一些代码。如果你想删除这些代码模块，那么可以使用手工一个一个移除。当然，你也可以使用下面的代码一次性移除所有代码模块，包括下面的代码自身所在的模块。

```
Sub DeleteAllCode()  
    Dim i As Integer  
    Dim result As VbMsgBoxResult  
    Dim strPrompt As String  
    Dim strTitle As String  
  
    strPrompt = "你确定要删除" & _  
                ActiveWorkbook.Name & _  
                "中的所有 VBA 代码吗?"  
    strTitle = "确认是否删除代码"  
  
    result = MsgBox(strPrompt, vbYesNo + vbQuestion,  
strTitle)  
    If result = vbNo Then  
        MsgBox "取消删除操作", vbInformation, "操作中止"  
        Exit Sub  
    End If
```



```

On Error Resume Next

With ActiveWorkbook.VBProject
    For i = .VBComponents.Count To 1 Step -1
        .VBComponents.Remove .VBComponents(i)
    Next i

    For i = .VBComponents.Count To 1 Step -1
        .VBComponents(i).CodeModule.DeleteLines 1, _
            .VBComponents(i).CodeModule.CountOfLines
    Next i
End With

On Error GoTo 0
End Sub

```

要想正常运行上述代码，必须在宏设置中选中“信任对 VBA 工程对象模型的访问”复选框，如下图 32.1 所示。

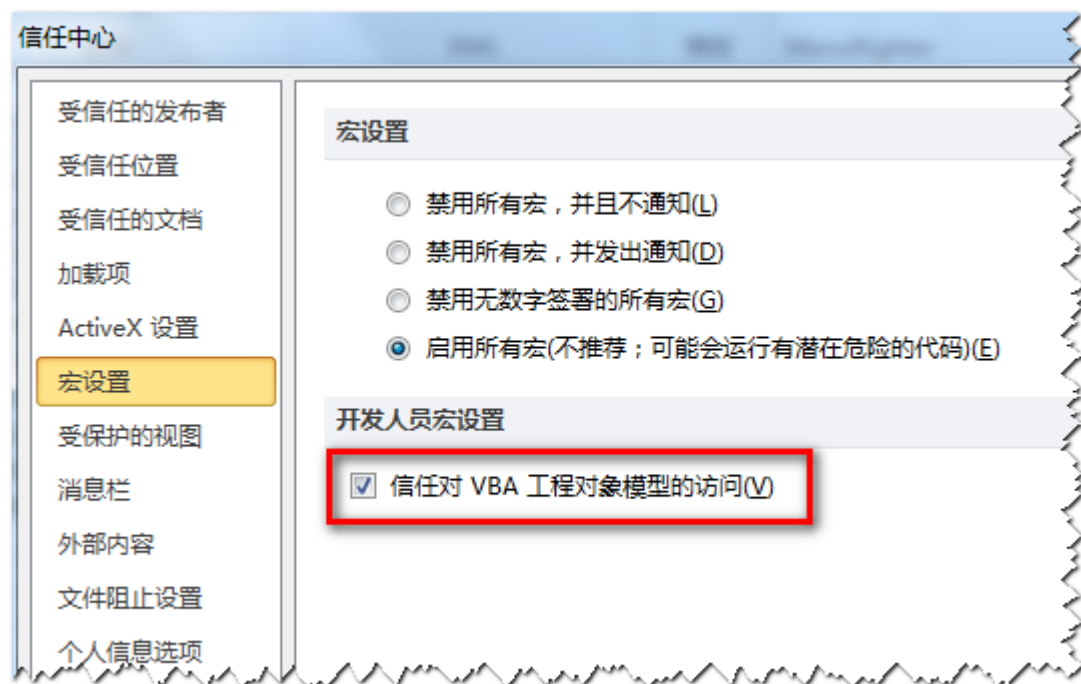


图 32.1



本章内容 2018 年 9 月 14 日首发于
[完美 Excel] 微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 33：列出工作表中
设置的条件格式清单

33.列出工作表中设置的条件格式清单

条件格式是 Excel 中的一个非常强大的功能，让数据更清晰明确。有时候，工作表中会设置一些条件格式以突出显示相关的数据。通常，我们可以使用“条件格式规则管理器”查看所使用的条件格式，如图 33.1 所示。

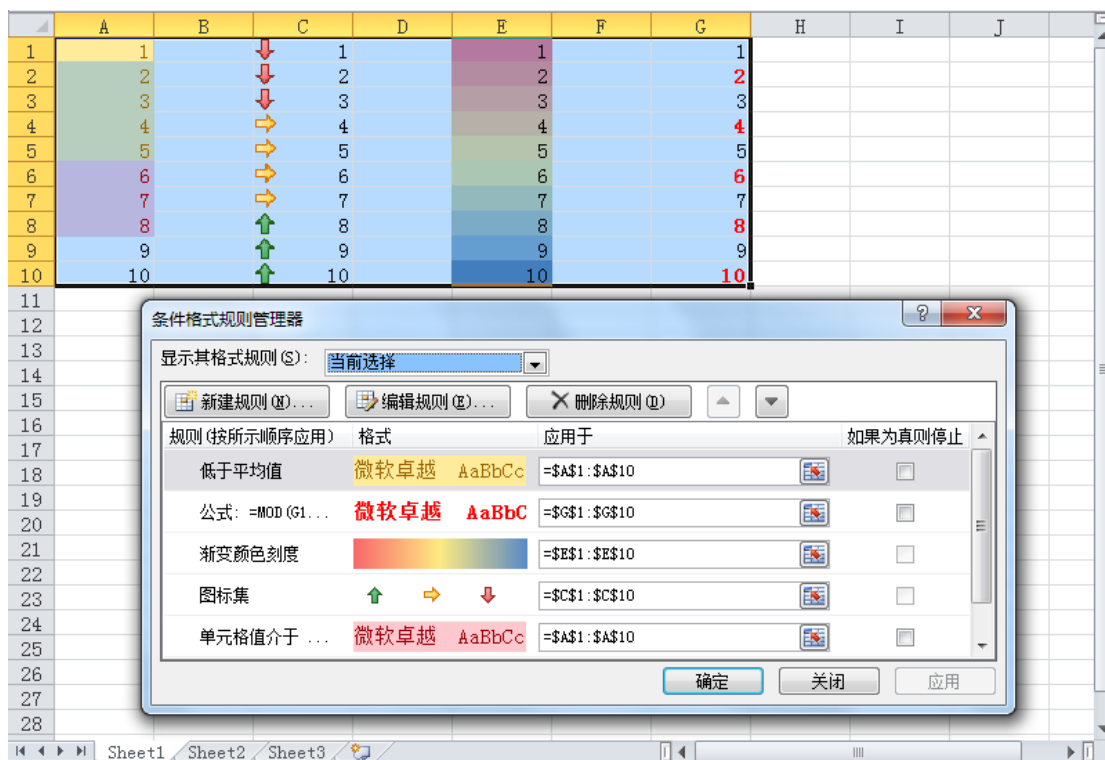


图 33.1

如果能够将工作表中使用的条件格式全部列出来，对于理解工作表将会有很大的帮助。下面的程序代码学习整理自 [dailydoseofexcel.com](#)，运行代码后，将新建一个工作簿，并在其中列出原工作簿工作表 Sheet1 中设置的条件格式。

```
Sub ListConditionalFormatting()
```



```

Dim cf As Variant
Dim rng As Range
Dim colConFormat As Collection
Dim i As Long
Dim wks As Worksheet
Dim var As Variant

Set colConFormat = New Collection

For Each rng In
Sheet1.Cells.SpecialCells(xlCellTypeAllFormatConditions)
    For i = 1 To rng.FormatConditions.Count
        With rng.FormatConditions
            On Error Resume Next
            colConFormat.Add .Item(i),
FCSignature(.Item(i))
            On Error GoTo 0
        End With
    Next i
Next rng

ReDim var(1 To colConFormat.Count + 1, 1 To 5)

Set wks = Workbooks.Add.Worksheets(1)
var(1, 1) = "类型"
var(1, 2) = "单元格"
var(1, 3) = "如果为真则停止"
var(1, 4) = "公式 1"
var(1, 5) = "公式 2"

For i = 1 To colConFormat.Count
    Set cf = colConFormat.Item(i)

```




```

        var(i + 1, 1) = FCTypeFromIndex(cf.Type)
        var(i + 1, 2) = cf.AppliesTo.Address
        var(i + 1, 3) = cf.StopIfTrue
        On Error Resume Next
        var(i + 1, 4) = "'" & cf.Formula1
        var(i + 1, 5) = "'" & cf.Formula2
        On Error GoTo 0
    Next i

    wks.Range("A1").Resize(UBound(var, 1), UBound(var, 2)).Value = var
    wks.UsedRange.EntireColumn.AutoFit
End Sub

Function FCSignature(ByRef cf As Variant) As String
    Dim strResult(1 To 3) As String

    strResult(1) = cf.AppliesTo.Address
    strResult(2) = FCTypeFromIndex(cf.Type)
    On Error Resume Next
    strResult(3) = cf.Formula1
    FCSignature = Join(strResult, vbNullString)
End Function

Function FCTypeFromIndex(lngIndex As Long) As String
    Select Case lngIndex
        Case 1: FCTypeFromIndex = "单元格值"
        Case 2: FCTypeFromIndex = "表达式"
        Case 3: FCTypeFromIndex = "色阶"
        Case 4: FCTypeFromIndex = "数据条"
        Case 5: FCTypeFromIndex = "前 10?"
        Case 6: FCTypeFromIndex = "图标集"
    End Select
End Function

```



```

Case 8: FCTypeFromIndex = "唯一值"

Case 9: FCTypeFromIndex = "文本"

Case 10: FCTypeFromIndex = "空值"

Case 11: FCTypeFromIndex = "时间段"

Case 12: FCTypeFromIndex = "高于平均值"

Case 14: FCTypeFromIndex = "非空值"

Case 16: FCTypeFromIndex = "错误"

Case 17: FCTypeFromIndex = "无错误"

Case Else: FCTypeFromIndex = "未知"

End Select

End Function

```

对于上图 33.1 所示设置了条件格式的工作表，运行 ListConditionalFormatting 过程后，列出其设置的条件格式清单如下图 33.2 所示。

	A	B	C	D	E
1	类型	单元格	如果为真则停止	公式1	公式2
2	高于平均值	\$A\$1:\$A\$10	FALSE		
3	单元格值	\$A\$1:\$A\$10	FALSE	=3	=8
4	图标集	\$C\$1:\$C\$10	FALSE		
5	色阶	\$E\$1:\$E\$10	FALSE		
6	表达式	\$G\$1:\$G\$10	FALSE	=MOD(G1, 2)=0	
7					

图 33.2



本章内容 2018 年 9 月 27 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[VBA 实用小程序 34: 将 Excel 工作表
导出到 Word 文档](#)

34.将 Excel 工作表导出到 Word 文档

经常有人问怎么样使用 VBA 实现将 Excel 工作表导出到 Word 文档中，下面的程序给出了一个范例，其功能为将工作簿中的所有工作表中的内容导出到 Word 文档：

```
Sub ExportExcelToWord()  
    Dim wd As Word.Application  
    Dim wdDoc As Word.Document  
    Dim wks As Worksheet  
  
    Application.ScreenUpdating = False  
  
    Set wd = New Word.Application  
    Set wdDoc = wd.Documents.Add  
  
    For Each wks In ActiveWorkbook.Worksheets  
        wks.UsedRange.Copy  
  
        wdDoc.Paragraphs(wdDoc.Paragraphs.Count).Range.InsertParagraphAfter  
  
        wdDoc.Paragraphs(wdDoc.Paragraphs.Count).Range.Paste  
        Application.CutCopyMode = False  
  
        wdDoc.Paragraphs(wdDoc.Paragraphs.Count).Range.InsertParagraphAfter  
    End For  
End Sub
```



```

        If Not wks.Name = Worksheets(Worksheets.Count).Name
Then
            With
wdDoc.Paragraphs(wdDoc.Paragraphs.Count).Range
                .InsertParagraphBefore
                .Collapse Direction:=wdCollapseEnd
                .InsertBreak Type:=wdPageBreak
            End With
        End If
    Next wks

    Set wks = Nothing

    With wd.ActiveWindow
        If .View.SplitSpecial = wdPaneNone Then
            .ActivePane.View.Type = wdPrintView
        Else
            .View.Type = wdPrintView
        End If
    End With

    Set wdDoc = Nothing
    wd.Visible = True
    Set wd = Nothing

    Application.ScreenUpdating = True
End Sub

```

在VBE标准模块中输入上述代码之前,先要在VBE中单击菜单“工具——引用”,在弹出的“引用”对话框中勾选“Microsoft Word 14.0 Object Library”,如下图34.1所示。



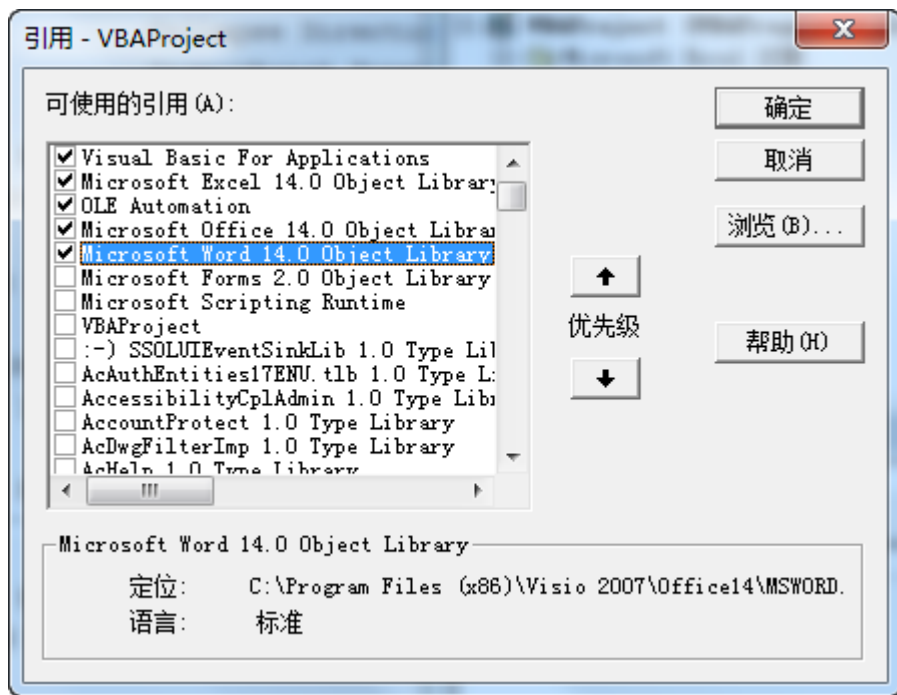


图 34.1

如下图 34.2 所示的工作簿，其工作表 Sheet1 和 Sheet2 中均有内容，运行 ExportExcelToWord 过程后，将新建一个 Word 文档，分别插入这两个工作表中的内容。

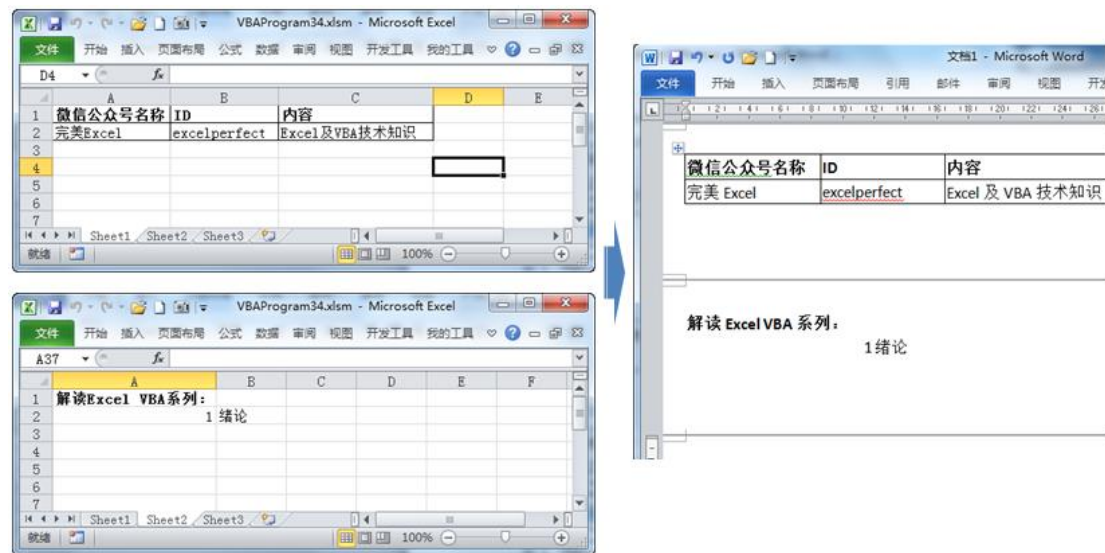


图 34.2





本章内容 2018 年 10 月 9 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 35：快速备份工作簿中所有的代码模块

35.快速备份工作簿中所有的代码模块

下面的程序代码可以实现一次快速导出工作簿中所有的代码模块，你所要做的就是选择一个放置代码模块的文件夹，所有的代码模块将放置在该文件夹中。

' Excel 和 VBA 没有提供方便的文件夹目录选择器或者文件选择对话框。
' 下面的函数提供对带有必要功能的系统 DLL 的引用。

```
Private Type BROWSEINFO ' 用于 GetFolderName 函数
    hOwner As Long
    pidlRoot As Long
    pszDisplayName As String
    lpszTitle As String
    ulFlags As Long
    lpfn As Long
    lParam As Long
    iImage As Long
End Type

Private Declare Function SHGetPathFromIDList Lib "shell32.dll" _
    Alias "SHGetPathFromIDListA" (ByVal pidl As Long, ByVal pszPath As String) As Long

Private Declare Function SHBrowseForFolder Lib "shell32.dll" _
    Alias "SHBrowseForFolderA" (lpBrowseInfo As BROWSEINFO) As Long
```



' 返回用户所选择的文件夹的名称

```
Function GetFolderName(Msg As String) As String
    Dim bInfo As BROWSEINFO, Path As String, R As Long
    Dim x As Long, pos As Integer
    bInfo.pidlRoot = 0& ' 根文件夹 = 桌面
    If IsMissing(Msg) Then
        ' 对话框标题
        bInfo.lpszTitle = "选择文件夹."
    Else
        bInfo.lpszTitle = Msg
    End If
    bInfo.ulFlags = &H1 ' 返回的目录类型
    x = SHBrowseForFolder(bInfo) ' 显示对话框
    ' 解析结果
    Path = Space$(512)
    R = SHGetPathFromIDList(ByVal x, ByVal Path)
    If R Then
        pos = InStr(Path, Chr$(0))
        GetFolderName = Left(Path, pos - 1)
    Else
        GetFolderName = ""
    End If
End Function
```

```
Sub ExportMods()
    Dim Path As String
    Dim objMyProj As VBProject
    Dim objVBComp As VBComponent

    Set objMyProj = Application.VBE.ActiveVBProject

    Path = GetFolderName("选择放置 BAS 文件的文件夹:")
```




```

If Path = "" Then
    MsgBox ("你没有选择导出目录. 导出失败.")
    Exit Sub
End If

For Each objVBComp In objMyProj.VBComponents
    If objVBComp.Type = vbext_ct_ClassModule Or
vbext_ct_MSForm Or vbext_ct_StdModule Then
        objVBComp.Export Path & "\" & objVBComp.Name &
".bas"
    End If
Next
End Sub

```

要正常运行上面的代码，需要先设置对“Microsoft Visual Basic for Applications Extensibility 5.3”库的引用。在 VBE 中，单击菜单“工具——引用”，在“引用”对话框中，找到并选取“Microsoft Visual Basic for Applications Extensibility 5.3”库，如下图 35.1 所示。

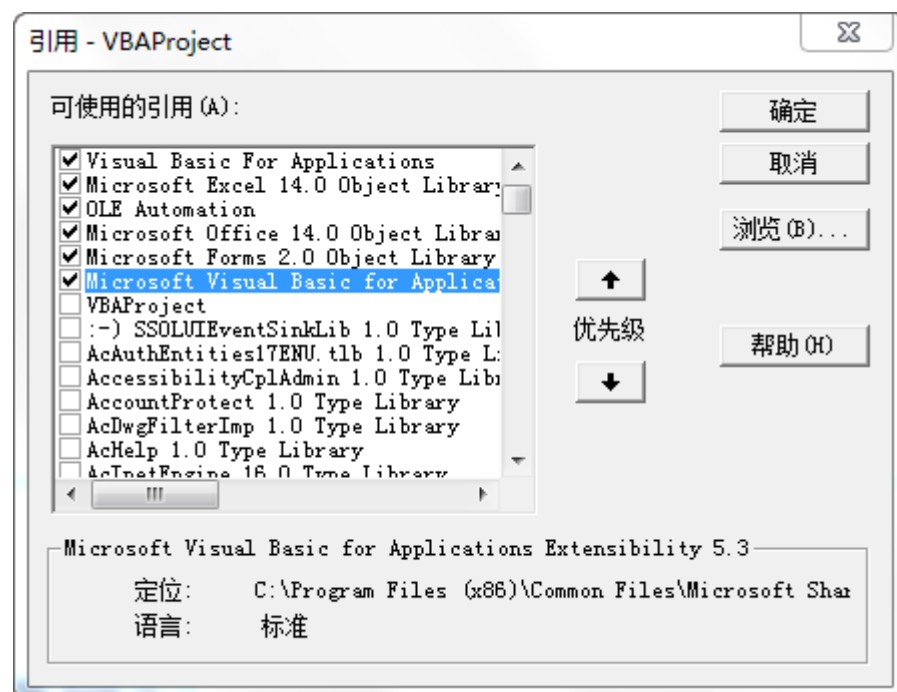


图 35.1

运行 ExportMods 过程，将弹出选择文件夹的对话框，选取要放置代码模块的



文件夹，如下图 35.2 所示。

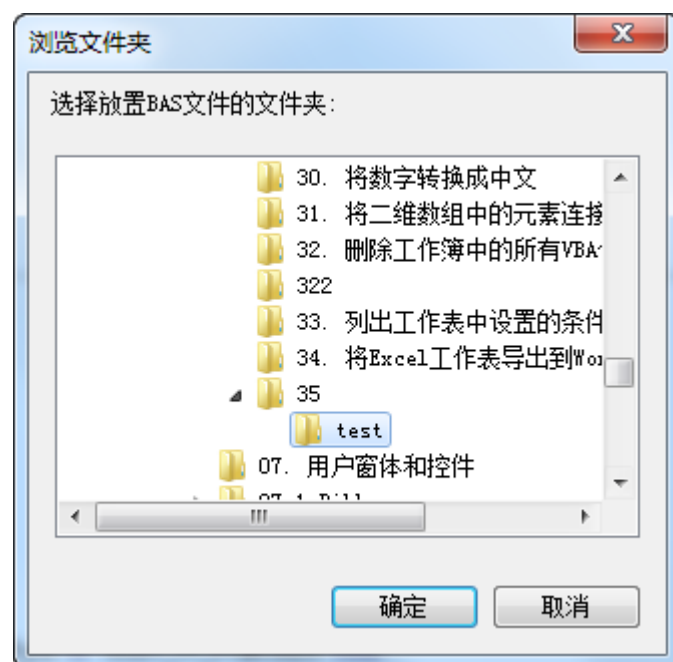


图 35.2

单击“确定”，工作簿中的代码模块将导入到所选文件夹中，如下图 35.3 所示。

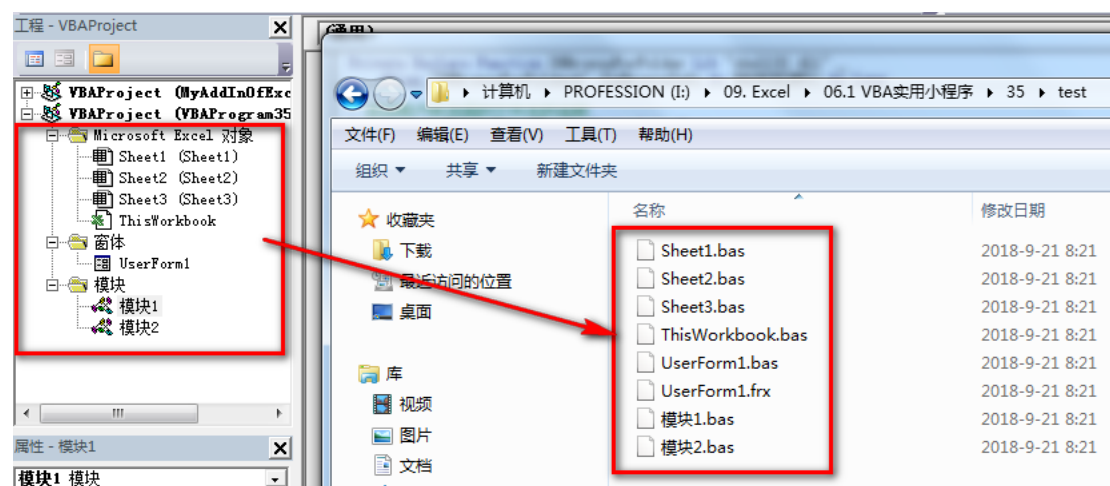


图 35.3



本章内容 2018 年 10 月 12 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 36：打印多个工作表

36.打印多个工作表

在 Excel 中，默认打印当前活动工作表。如果你想打印多个工作表，先按下 Ctrl 键或者 Shift 键选取想要打印的工作表，然后单击打印。也可以在打印时出现的“打印内容”对话框中，选取“整个工作簿”来打印工作簿中所有的工作表。

下面的程序可用于打印多个工作表。

程序 1：

```
Sub PrintMultiSheets()  
    Dim vwksList As Variant  
    vwksList = Array("Sheet1", "Sheet3")  
    ThisWorkbook.Sheets(vwksList).PrintOut  
End Sub
```

代码打印指定的工作表，其中，指定要打印的工作表名作为 Array 函数的参数。

程序 2：

```
Sub PrintAllSheets()  
    Dim strWksList() As String  
    Dim lngNum As Long  
  
    ReDim strWksList(1 To ThisWorkbook.Sheets.Count)  
  
    For lngNum = LBound(strWksList) To UBound(strWksList)  
        strWksList(lngNum) =  
ThisWorkbook.Sheets(lngNum).Name  
    Next lngNum  
End Sub
```



```
Next lngNum

ThisWorkbook.Sheets(strWksList).PrintOut
End Sub
```

代码打印工作簿中所有工作表，其中，遍历所有工作表并将其名字添加到数组中。

程序 3:

```
Sub PrintAllSheets2()
    Dim wks As Object
    Dim lngNum As Long

    Set wks = ThisWorkbook.ActiveSheet

    For lngNum = 1 To ThisWorkbook.Sheets.Count
        ThisWorkbook.Sheets(lngNum).Select False
    Next lngNum

    ThisWorkbook.PrintOut

    wks.Select True
End Sub
```

代码打印工作簿中所有工作表，其中遍历所有工作表并将工作表成组选择。

上述代码都可以通过 IF 语句来指定或排除不想要打印的工作表，而要打印所有工作表，最简单的代码是使用语句：

```
Sheets.PrintOut
```

或者

```
Worksheets.PrintOut
```



本章内容 2018 年 10 月 26 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 37：在工作表中添加 ActiveX 控件

37.在工作表中添加 ActiveX 控件

在工作表中，利用控件可以直观地表达数据的状态。例如，下图 37.1 所示，我需要逐项核对物品清单列表，如果已核对，则在前面的复选框中打勾。



	A	B	C
1	物品编号	物品名称	
2	C0001	水泥	
3	C0005	钢筋	
4	C0016	钢板	
5	C0188	螺栓	
6	C9001	阳光板	
7	C9801	型钢	
8			

	A	B	C
1	是否核对	物品编号	物品名称
2	<input type="checkbox"/>	C0001	水泥
3	<input checked="" type="checkbox"/>	C0005	钢筋
4	<input type="checkbox"/>	C0016	钢板
5	<input checked="" type="checkbox"/>	C0188	螺栓
6	<input type="checkbox"/>	C9001	阳光板
7	<input type="checkbox"/>	C9801	型钢
8			

图 37.1

我们可以使用 VBA 代码来添加 ActiveX 控件，即利用 OLEObjects 集合的 Add 方法。上例中，在列表第 1 列添加复选框控件的代码如下：

```
Sub AddCheckBoxControl()  
    Dim objCheck As OLEObject  
    Dim rng As Range
```



```

Dim lngLast As Long

'清除已有的复选框
For Each objCheck In Sheet1.OLEObjects
    objCheck.Delete
Next objCheck

'统计数据行
lngLast = Sheet1.Range("A" & Rows.Count).End(xlUp).Row

'插入一列
Sheet1.Columns(1).Insert
Sheet1.Range("A1").Value = "是否核对"

'添加复选框
With Sheet1
    For Each rng In .Range("A2:A" & lngLast)
        With .OLEObjects.Add(classtype:="Forms.Checkbox.1", _
            Top:=rng.Top, _
            Left:=rng.Left, _
            Height:=rng.Height, _
            Width:=rng.Width)
            .Object.Caption = ""
            .LinkedCell = rng.Address
            .Object.Value = False
        End With
    Next rng
End With
End Sub

```

如果我们要隐藏未核对的行，即未选取复选框的行，可以使用下面的代码：



```

Sub HideRows()
    Dim rng As Range
    Dim lngLast As Long

    '统计数据行
    lngLast = Sheet1.Range("B" & Rows.Count).End(xlUp).Row

    With Sheet1
        For Each rng In Range("A2:A" & lngLast)
            If Not rng.Value Then
                rng.EntireRow.Hidden = True
            End If
        Next rng
    End With
End Sub

```

代码利用了 AddCheckBoxControl 过程中将添加的复选框的值链接到其所在的单元格，在复选框未被选取时，其值为 False。

因为我们要添加复选框控件，所以代码中使用了代表复选框的名称：

`Forms.Checkbox.1`

一些常用控件的名称如下：

`Forms.ComboBox.1`

`Forms.Optionbutton.1`

`Forms.Textbox.1`

`Forms.ListBox.1`

`Forms.Commandbutton.1`

你可以在代码中使用想要添加的控件的名称来添加该类控件。





本章内容 2018 年 11 月 6 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 38：列出工作簿中的
所有批注**

38.列出工作簿中的所有批注

本文所介绍的程序改编自 [Bill Jelen](#) 的 VBA and Macros for Microsoft Excel，实现在单独的工作表中列出工作簿中所有的批注信息，包括批注所在的工作簿、工作表 and 单元格、批注作者、批注内容。

VBA 代码清单如下：

```
Sub AllCommentsList()  
    ' 声明变量  
    Dim wb As Workbook  
    Dim ws As Worksheet  
    Dim cmt As Comment  
    Dim lngCmtCount As Long  
  
    ' 赋初始值, 代表第 2 行  
    lngCmtCount = 2  
  
    ' 关闭屏幕刷新  
    Application.ScreenUpdating = False  
  
    ' 新建带有 1 个工作表的工作簿  
    Set wb = Workbooks.Add(xlWorksheet)  
  
    ' 在新建的工作簿中输入标题行  
    With wb.Sheets(1)
```



```

        .Range("A1") = "作者"
        .Range("B1") = "工作簿"
        .Range("C1") = "工作表"
        .Range("D1") = "单元格"
        Range("E1") = "批注"
    End With

    '遍历代码所在的工作簿中的工作表
    For Each ws In ThisWorkbook.Worksheets
        '遍历工作表中的批注
        For Each cmt In ws.Comments
            '将批注信息输入新工作簿中
            With wb.Sheets(1)
                '批注作者
                .Cells(lngCmtCount, 1) = cmt.author
                '批注所在工作簿名
                .Cells(lngCmtCount, 2) = cmt.Parent.Parent.Parent.Name
                '批注所在工作表名
                .Cells(lngCmtCount, 3) = cmt.Parent.Parent.Name
                '批注所在单元格地址
                .Cells(lngCmtCount, 4) = cmt.Parent.Address
                '批注内容,调用子过程来清理批注内容
                .Cells(lngCmtCount, 5) = CleanComment(cmt.author, cmt.Text)
            End With

            '增加行计数
            lngCmtCount = lngCmtCount + 1
        Next cmt
    Next ws

```



```

'设置单元格区域不换行
wb.Sheets(1).UsedRange.WrapText = False

'关闭屏幕刷新
Application.ScreenUpdating = True

'释放对象变量
Set ws = Nothing
Set wb = Nothing
End Sub

'清理批注内容
Private Function CleanComment(author As String, cmt As String) As String
    Dim tmp As String

    '去掉批注作者
    tmp = Application.WorksheetFunction.Substitute(cmt, author & ":", "")

    '去掉批注中多余的空格
    tmp = Application.WorksheetFunction.Substitute(tmp, Chr(10), "")

    '返回清理后的批注内容
    CleanComment = tmp
End Function

```

在示例工作簿中运行代码后的结果如下图 38.1 所示。

	A	B	C	D	E	F	G	H	I	J
1	作者	工作簿	工作表	单元格	批注					
2	完美Excel	VBAProgram38.xlsm	Sheet1	\$B\$2	该批注在工作表Sheet1中，内容为空。					
3	完美Excel	VBAProgram38.xlsm	Sheet2	\$C\$2	本批注在工作表Sheet2中，可在此单元格中输入密码。					
4										
5										
6										

图 38.1





本章内容 2018 年 11 月 23 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为

**VBA 实用小程序 39：列出当前工作
表中公式所引用的其他工作表名称**

39. 列出当前工作表中公式所引用的其他工 作表名称

有时候，我们需要知道当前工作表中的公式是否引用了其他工作表。如果引用了其他工作表，那么所引用的工作表是哪些。这样，方便我们了解工作表结构，避免复制、删除等操作破坏现有工作表，从而导致错误。

下面的 VBA 代码列出当前工作表中公式所引用的其他工作表名称，如果没有引用其他工作表，则显示相应的消息。代码清单如下：

' 查找公式中引用的其他工作表

' 列出这些引用的工作表

```
Sub ListWorksheetLinks()  
    Dim rng As Range  
    Dim cell As Range  
    Dim dicWorksheet1 As Object  
    Dim dicWorksheet2 As Object  
    Dim strX, strY, strZ  
    Dim i As Long  
    Dim j As Long  
    Dim k As Long  
    Dim wks As Worksheet  
    Dim strSheets As String  
  
    Set dicWorksheet1 = CreateObject("Scripting.Dictionary")
```



```

Set dicWorksheet2 = CreateObject("Scripting.Dictionary")
Set rng = Cells.SpecialCells(xlCellTypeFormulas)

i = 0

For Each cell In rng
    '判断是否引用了其他工作表
    If InStr(1, cell.Formula, "!") > 0 Then
        strX = Split(cell.Formula, "!")
        If Not dicWorksheet1.exists(strX(0)) Then
            i = i + 1
            dicWorksheet1.Add strX(0), i
        End If
    End If
Next cell

'没有链接到其他工作表的公式
If i = 0 Then
    MsgBox "当前工作表没有链接其他工作表.", vbInformation
    GoTo exitH
End If

strY = dicWorksheet1.keys

'列出工作表名
j = 0
For k = LBound(strY) To UBound(strY)
    For Each wks In ActiveWorkbook.Worksheets
        If InStr(1, strY(k), wks.Name) > 1 Then
            If Not dicWorksheet2.exists(wks.Name) Then
                j = j + 1
                dicWorksheet2.Add wks.Name, j
            End If
        End If
    Next wks
Next k

```



```

        End If
    Exit For
End If
Next wks

Next k

strSheets = Join(dicWorksheet2.keys, vbCrLf)

MsgBox "当前工作表公式引用了工作表:" & vbCrLf & strSheets

exith:

Set dicWorksheet2 = Nothing
Set dicWorksheet1 = Nothing
Set rng = Nothing
End Sub

```

代码中，使用了两个 Dictionary 对象，使用键和内容存储相应的信息。

```

Set dicWorksheet1 = CreateObject("Scripting.Dictionary")
Set dicWorksheet2 = CreateObject("Scripting.Dictionary")

```

代码：

```

Set rng = Cells.SpecialCells(xlCellTypeFormulas)

```

将工作表中公式单元格区域赋值给对象变量 rng。

下面的代码：

```

For Each cell In rng
    '判断是否引用了其他工作表
    If InStr(1, cell.Formula, "!") > 0 Then
        strX = Split(cell.Formula, "!")
        If Not dicWorksheet1.exists(strX(0)) Then
            i = i + 1
            dicWorksheet1.Add strX(0), i
        End If
    End If
End If

```



```
Next cell
```

代码使用 Split 函数拆分出公式中“!”之前的部分，并检查其是否存在于字典对象中，如果不存在，则将其添加到字典对象中。

如果变量 i 的值为 0，表示没有发现工作表中存在引用其他工作表的公式，显示消息告诉用户并退出程序。

```
If i = 0 Then  
    MsgBox "当前工作表没有链接其他工作表.", vbInformation  
    GoTo exitH  
End If
```

接下来，提取第 1 个字典对象中的键值（其中包含着工作表名）到新的数组中：

```
strY = dicWorksheet1.keys
```

然后，遍历这个数组，使用 InStr 函数判断数组中是否包含工作表名。如果包含工作表名，则看其是否存在于字典对象中，如果不存在，则在字典对象中添加该工作表名。此时，这个字典对象的键值即为公式引用的其他工作表名。

使用 Join 函数连接字典对象中的键值，并单独成行，然后使用 MsgBox 显示最后的结果。

```
strSheets = Join(dicWorksheet2.keys, vbCrLf)  
MsgBox "当前工作表公式引用了工作表:" & vbCrLf & strSheets
```

注：这段代码不仅可实现列出当前工作表中公式所引用的其他工作表名称，而且演示了 Dictionary 对象与 Split 函数、Join 函数的应用。



本章内容 2018 年 12 月 12 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 实用小程序 40：获取单元格链接
的地址

40.获取单元格链接的地址

在 Excel 中，我们可以在单元格文本中插入超链接，方便单击后跳转到链接的地址，例如下图 40.1 所示，单元格中带有下划线的蓝色文本就是插入了超链接的，在其上单击，即可跳转到链接的网址或者位置。

	A	B	C
1	完美Excel		
2	VBA		
3	Excel Hero		
4			
5			
6			

图 40.1

选取要设置超链接的单元格，单击“插入”选项卡中的“超链接”按钮，在弹出的“编辑超链接”对话框中输入网址或者选取要跳转的位置，如下图 40.2 所示，即可完成单元格链接设置。

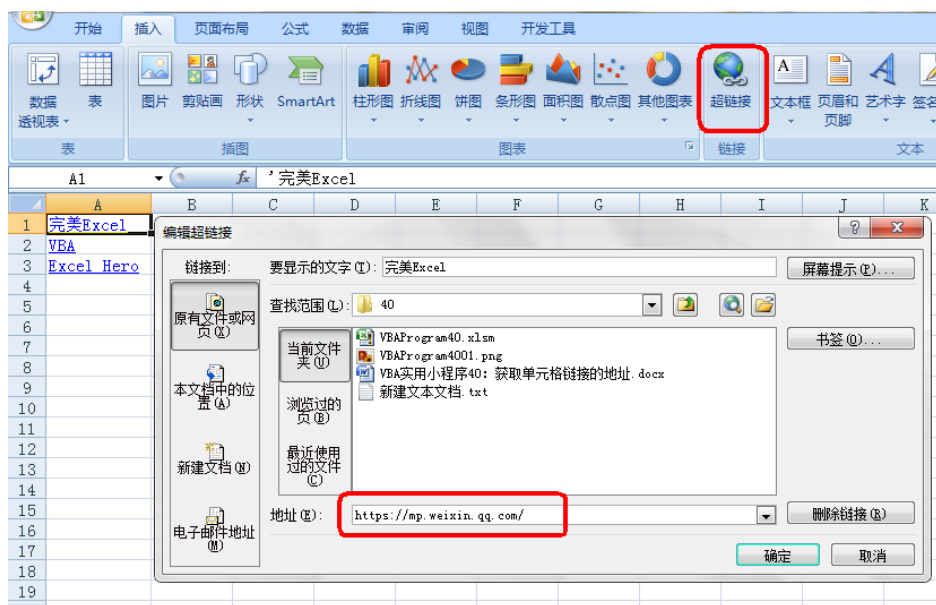


图 40.2



也可以使用 HYPERLINK 函数设置单元格链接, 如下图 40.3 所示, 在单元格 C1 中设置了与单元格 A1 相同的超链接。

	C1								
	A	B	C	D	E	F	G	H	
1	完美Excel		完美Excel						
2	VBA								
3	Excel Hero								
4									
5									

图 40.3

下面是 wellsr.com 中提供的一个 VBA 程序, 能够非常方便地提取单元格超链接地址。

' 获取单元格文本的超链接地址

```
Function GetLink(rng As Range)
```

' 声明变量

```
Dim strFormula As String
```

```
Dim strAddress As String
```

```
Dim i As Long
```

```
Dim lnkHyperlink As Hyperlink
```

```
Dim lnkHyperlinks As Hyperlinks
```

' 单元格公式文本

```
strFormula = rng.Formula
```

' 获取文本中超链接地址的开始位置

' 同时测试是否是超链接公式

```
i = InStr(1, strFormula, "HYPERLINK(", vbBinaryCompare)
```

```
If i > 0 Then
```

' 对于超链接公式, 返回超链接地址

```
strAddress = Mid(strFormula, i + 11)
```

```
strAddress = Left(strAddress, InStr(strAddress, """) - 1)
```



```

Else
    '如果是插入的链接,则返回链接位置
    Set lnkHyperlinks = rng.Worksheet.Hyperlinks
    For Each lnkHyperlink In lnkHyperlinks
        If lnkHyperlink.Range = rng Then
            strAddress = lnkHyperlink.Address
        End If
    Next lnkHyperlink
End If

'将结果赋值给函数
GetLink = strAddress
End Function

```

我们可以像 Excel 内置函数一样，在工作表中输入后提取链接地址，以查看各单元格都链接到什么地方了，如下图 40.4 所示。

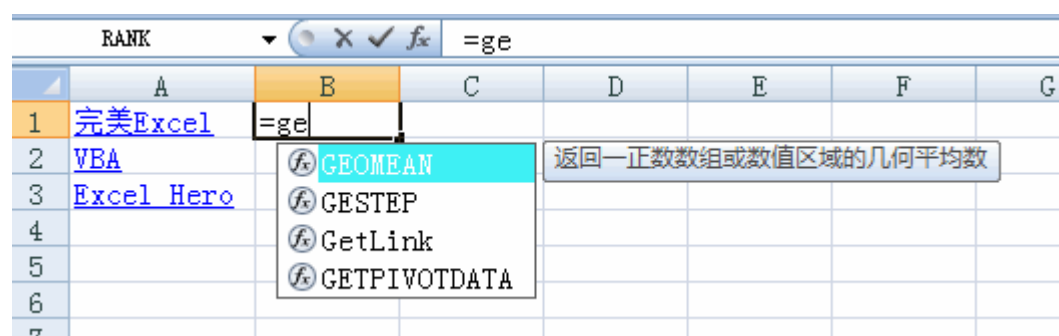


图 40.4

也可以在 VBA 代码中使用，从而获得相应链接地址后进一步使用。

```

Sub ExtractURL()
    Dim strURL As String
    Dim rng As Range

    strURL = "这些单元格的链接地址:" & vbCrLf

    For Each rng In Range("A1:A3")

```



```
strURL = strURL & GetLink(rng) & vbCrLf  
Next rng  
  
MsgBox strURL  
End Sub
```

在图 40.4 所示的工作表中运行上述代码后的结果如下图 40.5 所示。



图 40.5



本章内容 2019 年 1 月 8 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 实用小程序 41：使用 VBA 代码
在默认的浏览器中打开链接网站**

41.使用 VBA 代码在默认的浏览器中打开链接网站

有很多在浏览器中打开某网站的方法。下面是 [wellsr.com](#) 提供的一段程序，在 VBA 中使用 ShellExecute 函数在默认的浏览器中打开所传递的网站。

```
'API 声明, 运行一个外部程序

Private Declare Function ShellExecute _
    Lib "shell32.dll" _
    Alias "ShellExecuteA" ( _
        ByVal hwnd As Long, _
        ByVal lpOperation As String, _
        ByVal lpFile As String, _
        ByVal lpParameters As String, _
        ByVal lpDirectory As String, _
        ByVal nShowCmd As Long) _
    As Long

Private Sub LaunchWebsite(strUrl As String)

    ' 发生错误则跳至标签处

    On Error GoTo wellsrLaunchError

    ' 声明变量

    Dim r As Long
```



```

' 执行后的返回值

r = ShellExecute(0, "open", strUrl, 0, 0, 1)

' 如果执行错误

' 则尝试替代方法

If r = 5 Then
    r = ShellExecute(0, "open", "rundll32.exe", _
        "url.dll,FileProtocolHandler " & strUrl, 0, 1)
End If

Exit Sub

wellsrLaunchError:

MsgBox "尝试打开 URL 时发生错误." & vbNewLine & vbNewLine &
_
    "错误: " & Err.Number & ", " & Err.Description,
vbCritical, "发生错误"

End Sub

```

ShellExecute 函数在尝试启动 URL 时常常会产生“拒绝访问”错误，此时，会尝试执行替代方法。

与访问被拒绝错误 SE_ERR_ACCESSDENIED 相关的错误代码为 5。当遇到此错误代码时，程序会尝试显式调用 rundll32.exe FileProtocolHandler，以便可以无缝地启动你的网站。

下面是调用 LaunchWebsite 过程来打开指定网站的示例代码：

```

Sub URLdemo ()
    LaunchWebsite ("http://mp.weixin.qq.com")
End Sub

```



本章内容 2019 年 1 月 10 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[VBA 实用小程序 42: 修改 MsgBox 信息框中文本的颜色](#)

42.修改 MsgBox 信息框中文本的颜色

使用 Windows API 函数,可以非常容易地改变 MsgBox 信息框中的文本颜色。
代码如下:

```
'API 声明
#If Win64 Then
    Private Declare PtrSafe Function GetSysColor Lib "user32" _
        (ByVal nIndex As Long) As Long
    Private Declare PtrSafe Function SetSysColors Lib "user32" _
        (ByVal nChanges As Long, lpSysColor As Long, _
        lpColorValues As Long) As Long
#Else
    Private Declare Function GetSysColor Lib "user32" _
        (ByVal nIndex As Long) As Long
    Private Declare Function SetSysColors Lib "user32" _
        (ByVal nChanges As Long, lpSysColor As Long, _
        lpColorValues As Long) As Long
#End If

'定义常量
Private Const COLOR_WINDOWTEXT As Long = 8
Private Const CHANGE_INDEX As Long = 1
```



```

Public Sub testMsgBoxColor()
    Dim defaultColor As Long

    '存储默认的系统颜色
    defaultColor = GetSysColor(COLOR_WINDOWTEXT)

    '将系统颜色设置为红色
    SetSysColors CHANGE_INDEX, COLOR_WINDOWTEXT, vbRed
    MsgBox "错误", vbCritical, "结果是....."

    '将系统颜色设置为绿色
    SetSysColors CHANGE_INDEX, COLOR_WINDOWTEXT, RGB(0, 128,
0)
    MsgBox "正确", , "结果是....."

    '恢复默认颜色
    SetSysColors CHANGE_INDEX, COLOR_WINDOWTEXT,
defaultColor
End Sub

```

这段程序通过使用 user32 库中的 SetSysColors API 函数临时改变窗口文本的默认颜色来修改 MsgBox 信息框中的文本颜色。

代码先使用 vbRed 常量将文本颜色修改为红色，接着又修改为绿色。

如果在代码运行的过程中，你打开记事本并在其中输入一些文字，将会看到这些文字使用了改变后的颜色。你也会看到 VBE 中代码的颜色也随着改变了。

下图 42.1 所示为文本颜色修改成了红色的效果。



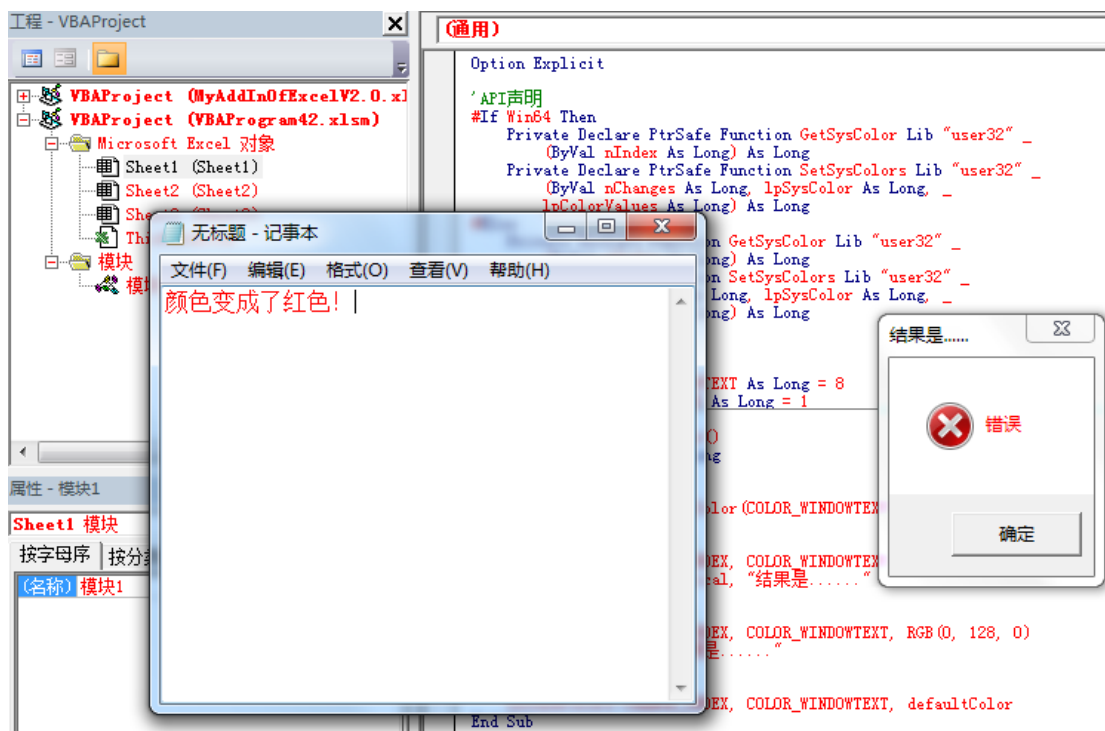


图 42.1

下图 42.2 所示为文本颜色修改成了绿色的效果。

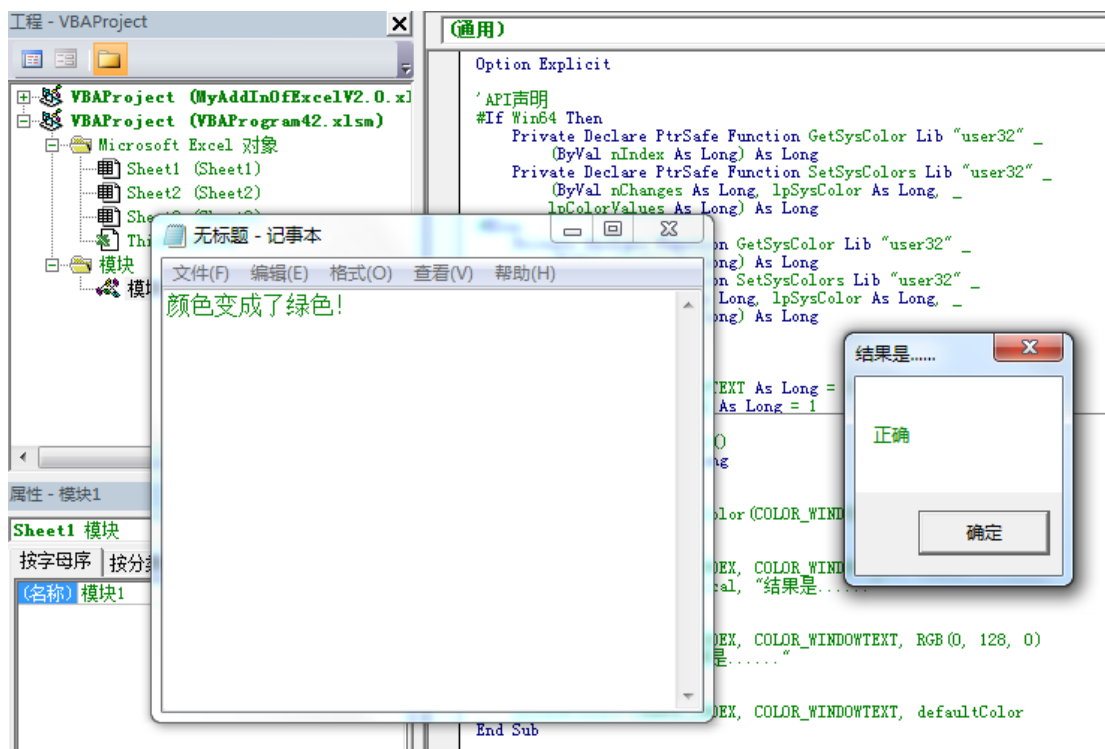


图 42.2

下图 42.3 所示为恢复系统默认色。



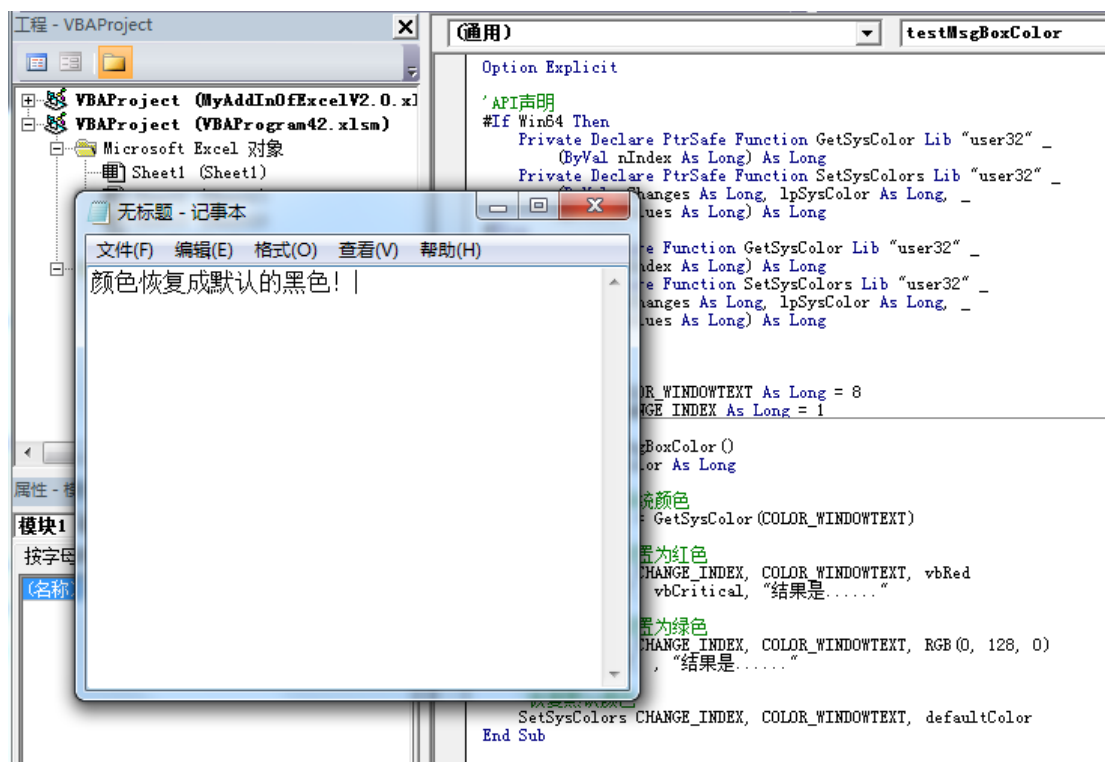


图 42.3

代码中，将最初的颜色存储在变量 defaultColor 中，以便最后能够恢复系统的默认颜色。通常，系统的默认色为黑色，如果你调试时改变了默认颜色，可以单独运行代码：

```
SetSysColors CHANGE_INDEX, COLOR_WINDOWTEXT, vbBlack
```

将系统默认色恢复为黑色。



本章内容 2018 年 10 月 15 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[VBA 代码库 01: 获取文件名](#)

43.获取文件名

下面的自定义函数返回文件名:

```
Private Function GetFileName(strFullPath As String)
    Dim lngPos As Long
    Dim lngStart As Long
    Dim strFilename As String

    lngStart = 1

    Do
        lngPos = InStr(lngStart, strFullPath, "\")
        If lngPos = 0 Then
            strFilename = Right(strFullPath, Len(strFullPath) - lngStart + 1)
        Else
            lngStart = lngPos + 1
        End If
    Loop While lngPos > 0

    GetFileName = strFilename
End Function
```

说明:



- 参数 `strFullPath` 为包含完整路径的文件名或者文件名的字符串。

编写一个简单的测试过程：

```
Sub test()  
    Dim str As String  
    str = "G:\09. Excel\06.2 VBA 代码库\VBACodeLibrary01.xlsm"  
    Debug.Print GetFileName(str)  
End Sub
```

运行测试过程后的结果如下图 43.1 所示。

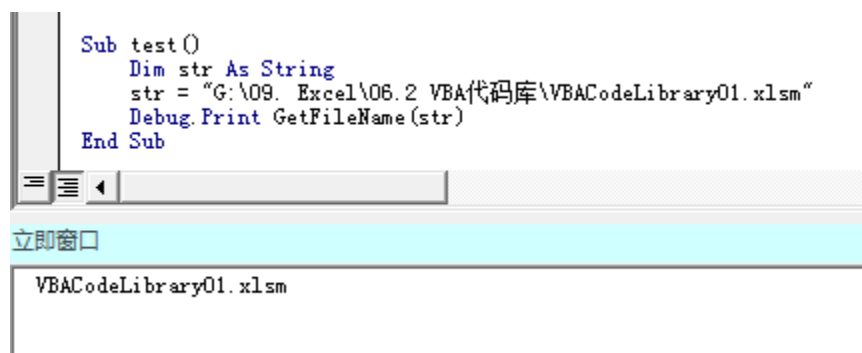


图 43.1



本章内容 2018 年 10 月 27 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为

**VBA 代码库 02: 列出指定目录下的
所有文件或子文件夹中的所有文件**

44. 列出指定目录下的所有文件或子文件夹 中的所有文件

下面的自定义函数用于列出指定目录下的所有文件或者其子文件夹中的所有文件:

```
' 用途: 列出指定目录下的所有文件或子文件夹中的所有文件
' 说明: 使用本函数前, 添加对 Microsoft Scripting Runtime 对象库的引用
' 来源于 MSDN: Working with Files, Folders, and Drives: More VBA Tips and Tricks

Function GetFiles(strPath As String, dctDict As Scripting.Dictionary, Optional blnRecursive As Boolean) As Boolean

    ' 本过程返回目录中的所有文件到 Dictionary 对象中.
    ' 如果递归调用则同时返回子文件夹中的所有文件.

    Dim fsoSysObj As Scripting.FileSystemObject
    Dim fdrFolder As Scripting.Folder
    Dim fdrSubFolder As Scripting.Folder
    Dim filFile As Scripting.file

    ' 返回新的 FileSystemObject.
```



```

Set fsoSysObj = New Scripting.FileSystemObject

On Error Resume Next

' 获取文件夹.
Set fdrFolder = fsoSysObj.GetFolder(strPath)
If Err <> 0 Then
    ' 不正确的路径.
    GetFiles = False
    GoTo GetFiles_End
End If
On Error GoTo 0

' 遍历 Files 集合, 添加到字典.
For Each filFile In fdrFolder.Files
    dctDict.Add filFile.path, filFile.path
Next filFile

' 如果 Recursive 标志为真, 则递归调用.
If blnRecursive Then
    For Each fdrSubFolder In fdrFolder.Subfolders
        GetFiles fdrSubFolder.path, dctDict, True
    Next fdrSubFolder
End If

' 如果没有错误发生则返回 True.
GetFiles = True

GetFiles_End:
Exit Function
End Function

```

说明:



- 参数 `strPath` 为指定的目录路径。
- 参数 `blnRecursive` 可选,若指定为 `True` 则列出子文件夹中的所有文件。

编写一个简单的测试过程:

```
' 测试 GetFiles 函数.
Sub TestGetFiles()
    Dim dctDict As Scripting.Dictionary
    Dim varItem As Variant
    Dim strDirPath As String

    ' 将此路径更改为你想要列出文件的路径.
    strDirPath = "E:\1.专题学习\0.Excel\2.Excel 项目开发"

    ' 创建新的字典.
    Set dctDict = New Scripting.Dictionary

    ' 递归调用, 返回文件到 Dictionary 对象.
    If GetFiles(strDirPath, dctDict, True) Then
        ' 打印字典中的项目.
        For Each varItem In dctDict
            ' 只打印工作簿文件.
            If Right(varItem, 4) Like ".xls" Or Right(varItem,
2) Like ".xls*" Then
                Debug.Print varItem
            End If
        Next
    End If
End Sub
```





本章内容 2018 年 11 月 10 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
**VBA 代码库 03: 获取当前文件所在
的文件夹路径及子文件夹数**

45. 获取当前文件所在的文件夹路径及子文件夹数

下面的两个程序都可以返回当前文件所在的文件夹路径。

· 用途: 获取当前文件所在的文件夹路径

· 说明: 程序 1-使用常规方式

```
Function GetCurFileDir1() As String
    Dim strFileFullName As String
    Dim strFileDir As String
    Dim i As Long
    Dim iDirNum As Integer

    strFileFullName = ThisWorkbook.FullName

    ' 查找最后一个"\"的位置
    For i = Len(strFileFullName) To 1 Step -1
        If Mid(strFileFullName, i, 1) = "\" Then
            iDirNum = i
            Exit For
        End If
    Next i

    ' 获取当前文件所在文件夹目录
    strFileDir = Left(strFileFullName, iDirNum)
    GetCurFileDir1 = strFileDir
```



```
End Function
```

· 用途:获取当前文件所在的文件夹路径

· 说明:程序 2-使用 FileSystemObject 对象

```
Function GetCurFileDir2() As String
    Dim fs As Object
    Dim strFileFullName As String
    Dim strFileDir As String

    Set fs = CreateObject("Scripting.FileSystemObject")

    strFileFullName = ThisWorkbook.FullName
    strFileDir = fs.getparentfoldername(strFileFullName)
    GetCurFileDir2 = strFileDir
End Function
```

下面的测试过程调用 GetCurFileDir1 函数并使用消息框显示当前工作簿所在的文件夹路径:

```
Sub testGetCur()
    MsgBox "本工作簿所在的文件夹位置:" & Chr(10) & Chr(10) &
    GetCurFileDir1
End Sub
```

运行后,在我的计算机中的结果如下图 45.1 所示。

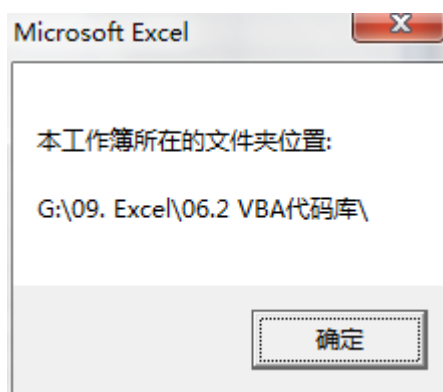


图 45.1



下面的自定义函数使用上面的 GetCurFileDir 函数，获取当前文件所在文件夹中的子文件夹数。

- 用途:返回当前目录下的子文件夹数
- 说明:使用了 GetCurFileDir 系列函数

```
Function lngSubFolderNum() As Long
    Dim fs As Object
    Dim objFolder As Object
    Dim ofsSubFolder As Object
    Dim colFiles

    Set fs = CreateObject("Scripting.FileSystemObject")
    Set objFolder = fs.GetFolder(GetCurFileDir2)
    Set colFiles = objFolder.Subfolders
    lngSubFolderNum = colFiles.Count
End Function
```

下面的过程来测试 lngSubFolderNum 函数：

```
Sub testSubFolderNum()
    MsgBox "本工作簿所在文件夹中含有 " & lngSubFolderNum & " 个子文件夹。"
End Sub
```

运行后，在我的计算机中的结果如下图 45.2 所示。

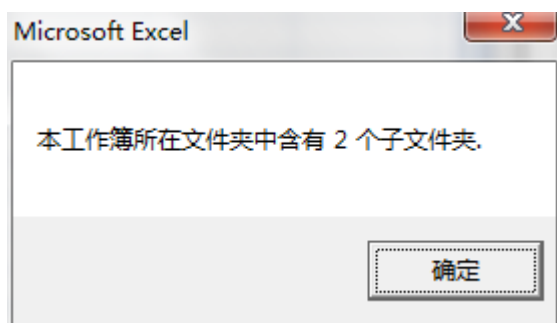


图 45.2





本章内容 2018 年 12 月 4 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
VBA 代码库 04: 统计指定文件夹或
子文件夹中的工作簿数量

46.统计指定文件夹或子文件夹中的工作簿数量

下面的程序可用于统计指定的文件夹中工作簿的数量，如果指定参数 blnRecursive 为 True，则还要统计该文件夹中子文件夹中工作簿的数量。

```
' 用途:统计指定文件夹或子文件夹中的工作簿数量
'      搜索指定文件夹，及其子文件夹，统计所有 Excel 工作簿文件数
' 参数:strPath——要搜索的文件夹
'      dctDict——存放文件的字典对象
'      blnRecursive——是否搜索当前文件夹的子文件夹,可选

Function NumFilesInCurDir2003or2007plus(strPath As String,
dctDict As Scripting.Dictionary, Optional blnRecursive As
Boolean) As Long

' 本过程返回目录中的所有文件到 Dictionary 对象中。
' 如果递归调用则同时返回子文件夹中的所有文件。

Dim fsoSysObj      As Scripting.FileSystemObject
Dim fdrFolder      As Scripting.Folder
Dim fdrSubFolder   As Scripting.Folder
Dim filFile        As Scripting.file
Dim varItem        As Variant
Dim lngWBNum       As Long

' 生成新的 FileSystemObject.
```



```

Set fsoSysObj = New Scripting.FileSystemObject

On Error Resume Next

' 获取文件夹.
Set fdrFolder = fsoSysObj.GetFolder(strPath)
If Err <> 0 Then
    ' 不正确的路径.
    GoTo GetFiles_End
End If
On Error GoTo 0

' 遍历 Files 集合, 添加到字典.
For Each filFile In fdrFolder.Files
    dctDict.Add filFile.path, filFile.path
Next filFile

' 如果 Recursive 标志为真, 则递归调用.
' 搜索文件夹中的子文件夹.
If blnRecursive Then
    For Each fdrSubFolder In fdrFolder.Subfolders
        GetFiles fdrSubFolder.path, dctDict, True
    Next fdrSubFolder
End If

' 如果文件扩展名为工作簿, 则计数
For Each varItem In dctDict
    If Right(varItem, 4) Like ".xls" Or Right(varItem,
5) Like ".xls*" Then
        lngWBNum = lngWBNum + 1
    End If
Next

```



```

' 返回统计的工作簿数
NumFilesInCurDir2003or2007plus = lngWBNum

GetFiles_End:
    Exit Function
End Function

```

下面的简单过程测试自定义的 NumFilesInCurDir2003or2007plus 函数：

```

Sub TestNumFilesInCurDir2003or2007()
    ' 测试 NumFilesInCurDir2003or2007 函数.

    Dim dctDict As Scripting.Dictionary
    Dim varItem As Variant
    Dim strDirPath As String

    ' 可修改为自己的路径.
    strDirPath = "G:\09. Excel\06.2 VBA 代码库"

    ' 创建新的字典.
    Set dctDict = New Scripting.Dictionary

    MsgBox "文件夹" & strDirPath & "中的工作簿数量为:" & _
        NumFilesInCurDir2003or2007plus(strDirPath, dctDict,
        True)
End Sub

```

运行后的结果如下图 46.1 所示。

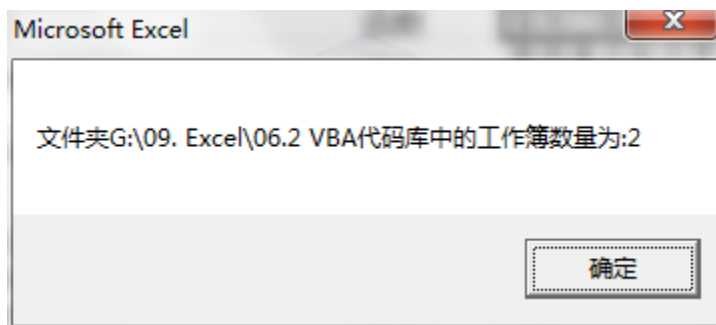


图 46.1





关于完美 Excel

完美 Excel 是一个坚持分享 Excel 与 VBA 技术知识的微信公众号，自 2017 年 5 月 15 日开始，每天推送一篇 Excel 与 VBA 技术和应用方面的文章。目前，共有 670 余篇实用文章可供广大 Excel 爱好者和使用者学习交流。这本电子书就是根据完美 Excel 上发表的 Excel VBA 实用小程序系列和 VBA 代码库系列文章整理而成的。

每天清晨，完美 Excel 微信公众号：**excelperfect** 都会推送一篇关于 Excel 与 VBA 的相关文章。如果你有兴趣学习 Excel 和 VBA 的相关知识和实战技巧，可以关注完美 Excel 微信公众号，绝对不会让你失望！

可以通过下列方法关注[完美 Excel]微信公众号：

方法 1—在通讯录中搜索“完美 Excel”或者“**excelperfect**”后点击关注。

方法 2—扫一扫下面的二维码



完美 Excel 微信公众号使用指南

下图 1 是完美 Excel 微信公众号的界面。公众号名称显示在屏幕正上方，屏幕底部显示有“菜单栏”，目前设置的菜单为“技术精粹”、“VBA 精选”、“联系 me”。在底部左侧的小键盘图标为消息框入口，单击可进入消息框界面给完美 Excel 公众号发送消息。



图 1

下图 2、图 3、图 4 分别为底部 3 个菜单的子菜单。目前，菜单“技术精粹”中设置有“2018 年文章合集”、“480 篇文章合集”、“又一波 20 个函数”、“快速学会 30 个函数”、“玩转数据验证”等 5 个子菜单；菜单“VBA 精选”中设置有“Excel VBA 编程基础”、“最最基础入门篇”、“VBA 学习经验”等 3 个子菜单；菜单“联系 me”中设置有“投稿须知”、“网站集粹”、“个人声明”、“坚持的美好”、“2019 年目标”等 5 个子菜单。





图 2



图 3





图 4

单击这些子菜单会进入详细的文章页面或者文章整理的入口页面，方便读者浏览或查阅本公众号的文章。同时，这些子菜单会随着完美 Excel 微信公众号内容的增加而适时调整。

可以单击底部左侧的小键盘图标，进入发送消息界面，如图 5 所示。在文本框中输入想要发送的文字，单击底部的“发送”按钮，就可以将消息发送给完美 Excel 微信公众号。

大家应留意完美 Excel 微信公众号推送的文章中的一些信息，例如，我会在百度网盘中存放一些文档资料或者示例工作簿文件，并在文章中给出进入百度网盘下载的文本信息，你只需在发送消息框中输入我给出的文本，单击发送后，就会收到一条关于下载链接和密码的信息。单击链接并按提示输入密码后，即可获得



相关的文档资料或示例工作簿文件了。



图 5

例如，在图 5 所示的界面中输入“Excel 动画图 2”后，会自动收到图 6 所示的信息，根据信息即可获取这个 Excel 动画图表文件。



图 6



希望大家在完美 Excel 微信公众号中能够学习到所需要的知识，获取到所需要的 Excel 应用技巧，提高自己的水平。但愿在今后的日子里，完美 Excel 微信公众号能够真正帮助大家发挥 Excel 的威力，为大家解决问题，提高工作效率。

