



Excel VBA 解读 • Workbook 对象篇

完美 Excel 出品
微信公众号: [excelperfect](#)



内容提要

Workbook 对象代表工作簿，全部的 **Workbook 对象**组成 **Workbooks 集合**，可以使用 **Workbooks 集合**来引用要操作的工作簿。本电子书《Excel VBA 解读 Workbook 对象篇》深入讲解了 **Workbook 对象**的常用属性、方法和事件，并列举了大量的实用示例，帮助读者快速提升 VBA 应用能力。



目录

主要内容.....	I
1. 发掘 Workbook 对象中的“好东西”.....	1
2. 创建新工作簿 —— Add 方法.....	3
示例代码 1：创建新工作簿并指定工作簿中的工作表数	3
示例代码 2：基于现有工作簿创建新工作簿	4
示例代码 3：将创建的新工作簿赋值	4
3. 用 VBA 代码打开工作簿 — Open 方法.....	5
示例代码 1：基于现有工作簿创建新工作簿	6
示例代码 2：将打开的工作簿赋值给变量	6
示例代码 3：测试是否已经打开了工作簿	6
4. 保存工作簿.....	9
示例代码 1：保存所有打开的工作簿	10
示例代码 2：添加新工作簿并以指定的名称保存该工作簿	10
示例代码 3：替换现有的工作簿	10
5. 关闭工作簿 —— Close 方法.....	13
示例代码 1：关闭所有打开的工作簿	14
示例代码 2：关闭指定的工作簿	14
示例代码 3：模仿在关闭的工作簿中取值	14



示例代码 4：拆分工作簿	16
6. 工作簿的基本属性	17
Name 属性	17
FullName 属性	18
Path 属性	18
CodeName 属性	19
FileFormat 属性	19
示例：将含有代码的工作簿保存为启用宏的工作簿	20
ReadOnly 属性	20
Saved 属性	20
示例 1：获取工作簿基本信息	21
示例 2：获取当前工作簿名称	22
示例 3：获取当前工作簿所在文件路径和名称	23
7. 工作簿中的链接	25
示例 1：获取当前工作簿的链接更新设置信息	26
示例 2：将当前工作簿中第 1 个链接公式转换成值	26
示例 3：更新链接	27
示例 4：查看链接状态	28
示例 5：获取工作簿中的链接信息	30
示例 6：打开当前工作簿链接到的所有工作簿	30
示例 7：获取指定工作簿中所有链接的状态	31
8. 文档属性	33
BuiltinDocumentProperties 属性	33
CustomDocumentProperties 属性	34



示例：在自定义文档属性中存储值.....	37
9. 看看工作簿中有哪些事件.....	39
Workbook 对象相关事件	39
Workbook 对象相关事件的位置	41
10. Workbook 对象的 Open 事件和 BeforeClose 事件.....	43
Workbook_Open 事件	43
Workbook_BeforeClose 事件.....	44
示例 1：设定特定用户才能操作工作表	44
示例 2：要求用户输入指定值	45
示例 3：添加/删除自定义快捷菜单	45
11. Workbook 对象的 SheetActivate 事件、SheetDeactivate 和 SheetSelectionChange 事件.....	49
Workbook_SheetActivate 事件.....	49
Workbook_SheetDeactivate 事件.....	50
Workbook_SheetSelectionChange 事件.....	50
示例 1：只允许访问指定的工作表	51
示例 2：限制用户必须包含指定内容	51
示例 3：限制用户必须在指定区域中操作	51
示例 4：阻止用户修改工作表名称	53
12. Workbook 对象的 BeforePrint 事件.....	55
示例 1：打印前重新计算工作表	55
示例 2：打印前添加页眉和页脚	56
示例 3：跟踪工作簿打印情况	56
13. Workbook 对象的 WindowResize 事件.....	57
示例 1：禁止调整工作簿窗口大小	57



示例 2：在状态栏中显示正在调整窗口大小的工作簿名称	58
14.Workbook 对象的 BeforeSave 事件	59
示例 1：让用户决定是否保存工作簿	59
示例 2：限定用户必须在指定的单元格中输入数据	60
15.工作簿事件示例：在单元格快捷菜单中添加自定义列表	61
16.工作簿事件示例：强制用户必须在指定单元格中输入数据	67
17.工作簿事件示例：强制用户必须启用宏	71
关于完美 Excel	75
完美 Excel 微信公众号使用指南	76



主要内容

每次使用 Excel 时，我们都会很自然地打开工作簿，然后在其中进行操作。在 VBA 中，Workbook 对象代表一个单独的工作簿，具有许多属性和方法，然而，我们只是经常使用其中的一些。

本书详细讲解了 Workbook 对象常用的属性、方法和事件，并分享了很多实用示例，相信会帮助你提升 VBA 编程应用能力。下面列出了本书各章的主要内容。

1.发掘 Workbook 对象中的“好东西”

简要介绍了 Workbooks 集合和 Workbook 对象，以及 Workbook 对象常用的属性、方法和事件，为接下来要详细讲解的内容作铺垫。

2.创建新工作簿——Add 方法

使用 Add 方法来创建新工作簿，详细讲解 Add 方法的语法。文中的示例：①创建新工作簿并指定工作簿中的工作表数；②基于现有工作簿创建新工作簿；③将创建的新工作簿赋值。

3.用 VBA 代码打开工作簿——Open 方法

详细讲解了 Open 方法的语法。文中的示例：①基于现有工作簿创建新工作簿；②将打开的工作簿赋值给变量；③测试是否已经打开了工作簿。

4.保存工作簿

使用 Workbook 对象的 Save 方法保存工作簿，详细讲解了 Save 方法的语法。文中的示例：①保存所有打开的工作簿；②添加新工作簿并以指定的名称保存该



工作簿；③替换现有的工作簿。

5.关闭工作簿——Close 方法

详细讲解 `Close` 方法的语法。文中的示例：①关闭所有打开的工作簿；②关闭指定的工作簿；③模仿在关闭的工作簿中取值；④拆分工作簿。

6.工作簿的基本属性

讲解了用来获取工作簿基本信息的一些属性，包括：`Name` 属性、`FullName` 属性、`Path` 属性、`CodeName` 属性、`FileFormat` 属性、`ReadOnly` 属性、`Saved` 属性，等等。文中的示例：①获取工作簿基本信息；②获取当前工作簿名称；③获取当前工作簿所在文件路径和名称。

7.工作表簿中的链接

讲解 `Workbook` 对象返回链接信息的一些方法和属性。文中的示例：①获取当前工作簿的链接更新设置信息；②将当前工作簿中第 1 个链接公式转换成值；③更新链接；④查看链接状态；⑤获取工作簿中的链接信息；⑥打开当前工作簿链接到的所有工作簿；⑦获取指定工作簿中所有链接的状态。

8.文档属性

讲解 `Workbook` 对象的内置文档属性 `BuiltinDocumentProperties` 属性和自定义文档属性 `BuiltinDocumentProperties` 属性。示例：在自定义文档属性中存储值。

9.看看工作簿中有哪些事件

详细列出了 `Workbook` 对象相关事件及发生的情形。

10.Workbook 对象的 Open 事件和 BeforeClose 事件

详细讲解 `Workbook` 对象常用的两个事件：`Open` 事件和 `BeforeClose` 事件的语法及说明。文中的示例：①设定特定用户才能操作工作表；②要求用户输入指定值；③添加/删除自定义快捷菜单。



11.Workbook 对象的 SheetActivate 事件、SheetDeactivate 和 SheetSelectionChange 事件

详细讲解 Workbook 对象的 3 个事件：SheetActivate 事件、SheetDeactivate 事件和 SheetSelectionChange 事件的语法及说明。文中的示例：①只允许访问指定的工作表；②限制用户必须包含指定内容；③限制用户必须在指定区域中操作；④阻止用户修改工作表名称。

12.Workbook 对象的 BeforePrint 事件

详细讲解了 Workbook 对象的 Workbook_BeforePrint 事件的语法及说明。文中的示例：①打印前重新计算工作表；②打印前添加页眉和页脚；③跟踪工作簿打印情况。

13.Workbook 对象的 WindowResize 事件

详细讲解了 Workbook 对象的 Workbook_WindowResize 事件的语法及说明。文中的示例：①禁止调整工作簿窗口大小；②在状态栏中显示正在调整窗口大小的工作簿名称。

14.Workbook 对象的 BeforeSave 事件

详细讲解了 Workbook 对象的 Workbook_BeforeSave 事件的语法及说明。文中的示例：①让用户决定是否保存工作簿；②限制用户必须在指定的单元格中输入数据。

15.工作簿事件示例——在单元格快捷菜单中添加自定义列表

使用 Workbook 对象的 SheetBeforeRightClick 事件，当右击单元格时在其快捷菜单中添加自定义列表的技巧。

16. 工作簿事件示例——强制用户必须在指定单元格中输入数据

使用 Workbook_BeforeClose 事件强制用户必须在指定的单元格中输入数据，



否则就不能关闭该工作簿。如果用户想要关闭工作簿但没有在指定的所有单元格中都输入数据，那么 Excel 会弹出提示信息，列出还没有输入数据的单元格，并将这些单元格的背景设置为黄色。

17. 工作簿事件示例——强制用户必须启用宏

使用 `Workbook_Open` 事件和 `Workbook_BeforeClose` 事件来实现在打开工作簿时，Excel 提示用户必须启用宏，否则工作簿中数据工作表均不可见。

温馨提示：

在完美 Excel 微信公众号中发送消息：**工作簿对象**，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 2017 年 12 月 17 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读 \(83\): 发掘 Workbook 对象中的“好东西”](#)

1. 发掘 Workbook 对象中的“好东西”

在《Excel VBA 解读：基础入门篇》中，我们接触到了 **Workbook 对象**，知道了 **Workbook 对象** 代表工作簿，全部的 **Workbook 对象** 组成 **Workbooks 集合**，可以使用 **Workbooks 集合** 来引用要操作的工作簿，如图 1.1 所示。

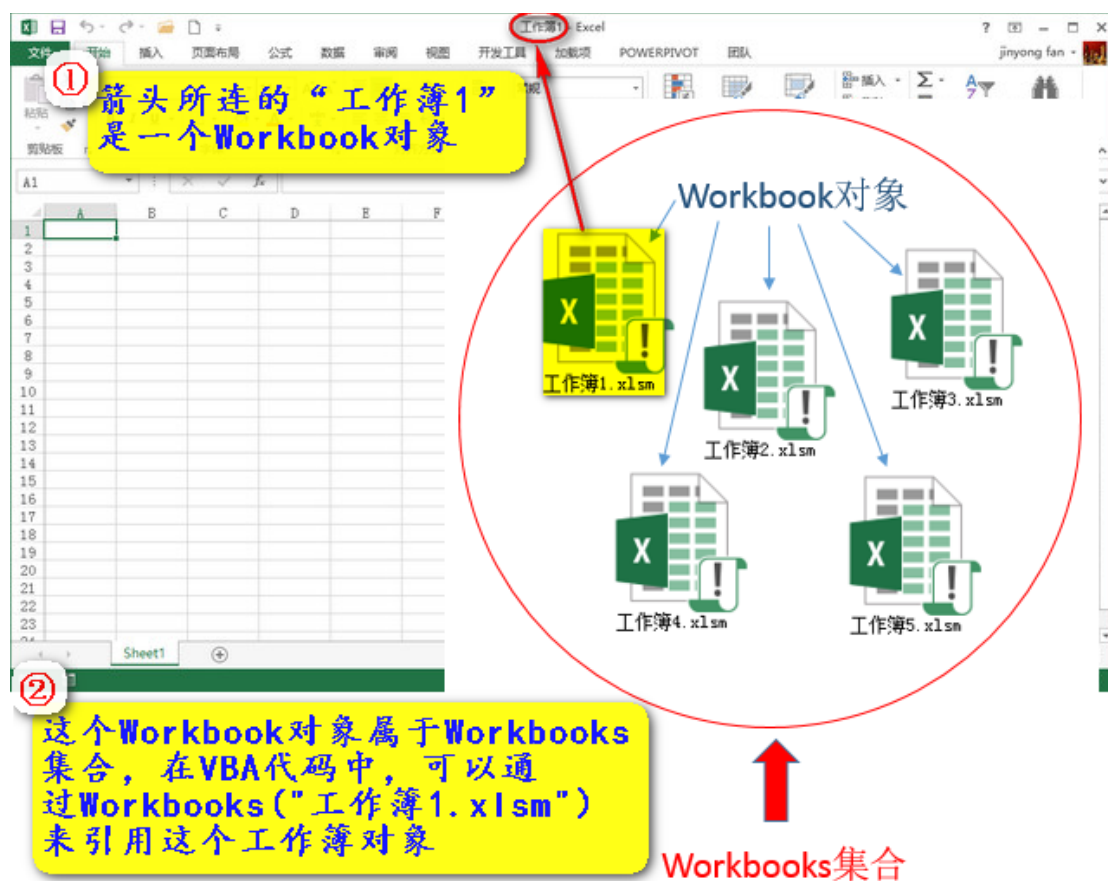


图 1.1

同时，也学习了使用 **Application 对象** 的 **ThisWorkbook 属性** 返回代码所在的工作簿对象，使用 **ActiveWorkbook 属性** 获得当前工作簿对象。



可以使用 **Workbooks 对象** 的 **Add 方法** 来创建一个新工作簿，并且可以指定该工作簿中包含的工作表模板。

打开和关闭工作簿是最常用不过的操作了。在 VBA 中，使用 **Workbooks 对象** 的 **Open 方法** 来打开一个工作簿，使用 **Workbooks 对象** 的 **Close 方法** 关闭所有打开的工作簿，使用 **Workbook 对象** 的 **Close 方法** 关闭指定的工作簿。

Workbook 对象 提供了能够获取工作簿基本信息的属性。**Name 属性** 返回工作簿名称，**FullName 属性** 返回带有完整路径的工作簿名称，**CodeName 属性** 返回工作簿对象名称；**Path 属性** 返回工作簿所在目录；**FileFormat 属性** 返回代表工作簿类型的常量；**ReadOnly 属性** 返回工作簿是否只读，**Saved 属性** 返回工作簿是否已保存。

Workbook 对象 还提供了很多事件，允许创建由程序或者用户操作工作簿时自动响应的动作。例如，在打开工作簿时，会执行 **Open 事件** 中的代码；当工作簿中任意工作表上的单元格被改变时，发生 **SheetChange 事件**。

本书接下来的内容，将深入讲解工作簿对象常用的一些属性、方法和事件，并附有很多实用示例，助你提升 VBA 编程应用能力。



本章内容 2017 年 12 月 20 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读（84）：创建新工作簿——Add 方法](#)

2. 创建新工作簿 —— Add 方法

使用 Workbooks 对象的 Add 方法创建新工作簿，其语法为：

Workbooks 对象.Add(Template)

说明：

- 参数 Template 可选，确定如何创建新工作簿。如果该参数指定的字符串是已有的 Excel 文件名，那么使用指定的文件作为模板创建新工作簿。如果该参数指定为 xlWBATemplate 常量之一，则新工作簿包含单个指定类型的工作表。如果忽略该参数，那么新工作簿包含标准的空工作表，可以使用 SheetsInNewWorkbook 属性设置工作表数。
- xlWBATemplate 常量为：xlWBATChart、xlWBATExcel4IntlMacroSheet、xlWBATExcel4MacroSheet、xlWBATWorksheet，分别代表图表工作表、宏工作表以及标准工作表。
- 创建的新工作簿成为当前活动工作簿。

示例代码 1：创建新工作簿并指定工作簿中的工作表数

下面的代码创建含有 5 个标准工作表的新工作簿。

```
Sub testWBAdd()  
    ' 设置新工作簿中的工作表数  
    Application.SheetsInNewWorkbook = 5  
    ' 创建新工作簿  
    Workbooks.Add  
End Sub
```

如果不指定新工作簿中工作表数，那么创建的新工作簿基于 Workbook 对象默认属性创建新的空工作簿。



示例代码 2：基于现有工作簿创建新工作簿

下面的代码以工作簿 excelvba81.xlsm 为模板创建新工作簿。

```
Sub testWBAdd1()  
    Workbooks.Add Template:="G:\09. Excel\01. 解读 Excel  
VBA\Excel VBA 解读 (81): 工作表事件示例\excelvba81.xlsm"  
End Sub
```

运行后创建的工作簿如下图 2.1 所示。

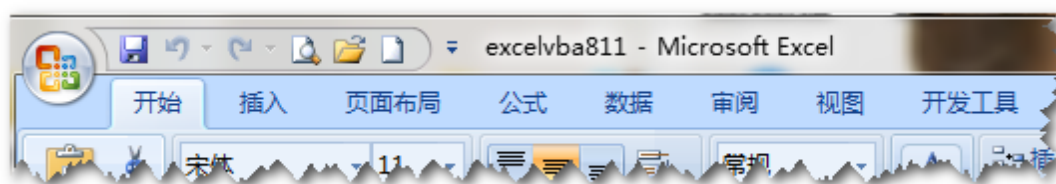


图 2.1

可以看出，新工作簿名在模板工作簿名的后面加了一个数字 1。如果在此基础上再创建一个新工作簿，则会在模板工作簿名后添加数字 2，依此类推。

示例代码 3：将创建的新工作簿赋值

下面的代码将新创建的工作簿赋给对象变量。

```
Sub testWBAdd2()  
    Dim wb1 As Workbook  
    Dim wb2 As Workbook  
  
    Set wb1 = Workbooks.Add  
    Set wb2 = Workbooks.Add(Template:="G:\09. Excel\01. 解  
读 Excel VBA\Excel VBA 解读 (81): 工作表事件示例  
\excelvba81.xlsm")  
End Sub
```

代码将基于默认属性创建的新工作簿赋值给变量 wb1，将基于工作簿 excelvba81.xlsm 创建的新工作簿赋值给变量 wb2。

这样，在后面的代码中，可以直接使用对象变量来操作相应的工作簿。



3. 用 VBA 代码打开工作簿 — Open 方法

要操作工作簿，要使用工作表分析和处理数据，都需要先打开工作簿。

在实际操作中，打开工作簿是再平常不过的操作了。双击桌面上的 Excel 快捷方式图标，或者点击桌面左下角开始菜单，找到 Excel 程序，单击即可打开工作簿。

在 VBA 中，我们可以使用 Workbooks 对象的 Open 方法打开工作簿，其语法为：

```
Workbooks 对象.Open(FileName, [UpdateLinks], [ReadOnly], [Format], [Password], [WriteResPassword], [IgnoreReadOnlyRecommended], [Origin], [Delimiter], [Editable], [Notify], [Converter], [AddToMru], [Local], [CorruptLoad])
```

说明：

- 参数 FileName 必需，指定一个字符串，代表要打开的工作簿文件名。
- 参数 UpdateLinks 可选，指定文件中外部引用（链接）的方式。如果忽略该参数，那么将提示用户指定如何更新链接。指定值为 0 将不更新外部引用（链接），指定值为 1 将更新外部引用（链接）但不更新远程引用（链接），指定值为 2 将更新远程引用（链接）但不更新外部引用（链接），指定值为 3 将更新外部引用（链接）。
- 参数 ReadOnly 可选，如果设置其值为 True，将在只读模式下打开工作簿。
- 参数 Format 可选，如果打开的是文本文件，那么该参数指定分隔符；如果忽略该参数，那么使用当前分隔符。可以指定下列值来确定文件的分隔符：1 代表 Tab，2 代表逗号，3 代表空格，4 代表分号，5 代表 Nothing，6 代表自定义分隔符。
- 参数 Password 可选，指定打开受保护工作簿时需要的密码。如果忽略该参



数但是工作簿受密码保护，则会提示用户输入密码。

- 参数 `Delimiter` 可选，如果打开的文件是文本文件且参数 `Format` 是 6，那么该参数指定用于作为分隔符的字符。例如，对于 `tab` 键使用 `Chr(9)`，对于逗号使用 “，”，对于分号使用 “；”，对于自定义字符串，仅使用其第一个字符。
- 其他参数使用不多，可参见 Excel VBA 帮助。

示例代码 1：基于现有工作簿创建新工作簿

下面的代码打开工作簿 `excelvba81.xlsm`。

```
Sub testWBOpen()  
  
Workbooks.Open Filename:="I:\09. Excel\01. 解读 Excel  
VBA\Excel VBA 解读 (81): 工作表事件示例\excelvba81.xlsm"  
  
End Sub
```

打开后的工作簿 `excelvba81.xlsm` 成为当前工作簿。

示例代码 2：将打开的工作簿赋值给变量

下面的代码将打开的工作簿赋给对象变量。

```
Sub testWBOpen1()  
  
    Dim wb As Workbook  
  
    Set wb = Workbooks.Open(Filename:="I:\09. Excel\01. 解  
读 Excel VBA\excelvbaSample.xlsm")  
  
End Sub
```

打开工作簿 `excelvbaSample.xlsm` 并将该工作簿赋值给变量 `wb`。在代码中，可以直接使用对象变量 `wb` 来操作该工作簿。

示例代码 3：测试是否已经打开了工作簿

下面的自定义函数 `blnWBOpen` 检查指定名称的工作簿是否已经被打开。




```

Function blnWBOpen(strWBName As String) As Boolean
    Dim wb As Workbook

    On Error Resume Next
    Set wb = Workbooks(strWBName)
    If Not wb Is Nothing Then
        blnWBOpen = True
    End If
End Function

```

代码首先将对工作簿的引用赋值给对象变量，然后检查是否赋值成功。如果找到指定的工作簿，blnWBOpen 函数返回 True，否则返回 False。

代码使用 On Error Resume Next 语句确保在没有找到指定的工作簿时不会发生运行时错误。

使用下面的代码测试 blnWBOpen 函数：

```

Sub testWBOpen2()
    Dim strWBName As String

    strWBName = "excelvbaSample.xlsm"

    If blnWBOpen(strWBName) Then
        MsgBox "指定的工作簿已打开!"
    Else
        MsgBox "没有打开指定的工作簿!"
    End If
End Sub

```

下面的自定义函数 blnWBOpen1 也可以用来检查指定名称的工作簿是否已经被打开：

```

Function blnWBOpen1(strWBName As String) As Boolean
    Dim wb As Workbook

    On Error Resume Next

```



```
Set wb = Workbooks(strWBName)
If Err.Number = 0 Then
    blnWBOpen1 = True
Else
    blnWBOpen1 = False
End If
End Function
```

还可以使用下面的自定义函数 blnWBOpen2 检查指定名称的工作簿是否已经被打开：

```
Function blnWBOpen2(strWBName As String) As Boolean
    Dim wb As Workbook
    Err.Clear
    On Error Resume Next
    Set wb = Workbooks(strWBName)
    blnWBOpen2 = Not wb Is Nothing
    Err.Clear
    On Error GoTo 0
End Function
```



本章内容 2018 年 3 月 15 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读（86）：保存工作簿](#)

4. 保存工作簿

可以使用 Workbook 对象的 Save 方法保存工作簿，其语法为：

Workbook 对象.Save

下面的示例代码用来保存当前工作簿：

```
Sub SaveWB()  
    ActiveWorkbook.Save  
End Sub
```

如果是首次保存工作簿，那么要使用 SaveAs 方法来指定工作簿的名称。其语法为：

Workbook 对象.SaveAs (FileName, FileFormat, Password, WriteResPassword, ReadOnlyRecommended, CreateBackup, AccessMode, ConflictResolution, AddToMru, TextCodepage, TextVisualLayout, Local)

将某个工作簿另存为指定名称的工作簿。可以看到，SaveAs 方法的参数很多，其中：

- 所有参数均为可选参数。
- 参数 FileName 指定要保存的工作簿的名称。可以提供保存的路径，否则将保存到当前文件夹中。
- 参数 FileFormat 指定工作簿保存的格式。
- 参数 Password 指定保存的工作簿的密码，区分大小写。
- 参数 WriteResPassword 指定工作簿的写保护密码。如果使用密码保存工作簿并且在打开时未提供密码，则以只读方式打开该工作簿。
- 参数 CreateBackup 设置为 True 时，创建工作簿的备份。
- 参数 AddToMru 设置为 True 时，将工作簿添加到最近使用文件列表中。



示例代码 1：保存所有打开的工作簿

下面的代码保存所有打开的工作簿。

```
Sub SaveAllWb()  
    Dim wb As Workbook  
  
    For Each wb In Workbooks  
        wb.Save  
    Next wb  
End Sub
```

示例代码 2：添加新工作簿并以指定的名称保存该工作簿

下面的代码先添加一个新工作簿，然后以变量 `strName` 指定的字符串作为名称保存该工作簿。

```
Sub AddWBAndSave()  
    Dim wbNewWorkbook As Workbook  
    Dim strName As String  
  
    strName = "test3"  
  
    Set wbNewWorkbook = Workbooks.Add  
  
    wbNewWorkbook.SaveAs Filename:=strName  
End Sub
```

示例代码 3：替换现有的工作簿

在使用 `SaveAs` 方法指定文件名保存工作簿时，如果刚好存在相同名称的工作簿文件，那么 Excel 就会给用户发出一条警告消息，提示用户确定是否覆盖已存在的文件。



如果希望覆盖已存在的文件，但不想每次都弹出这样的警告消息，那么可以使用下面的代码：

```
Sub ReplaceExistWb()  
    Dim wbNewWorkbook As Workbook  
    Dim strName As String  
  
    strName = "test3"  
  
    Set wbNewWorkbook = Workbooks.Add  
  
    Application.DisplayAlerts = False  
    wbNewWorkbook.SaveAs Filename:=strName  
    Application.DisplayAlerts = True  
End Sub
```

其中，使用了 Application 对象的 DisplayAlerts 属性来屏蔽警告消息。

如果不希望覆盖已存在的工作簿文件，那么可以给工作簿提供一个新的名称：

```
Sub SaveAsWbAvoidNameSame()  
    Dim wbNewWorkbook As Workbook  
    Dim strName As String  
    Dim strNewName As String  
    Dim i As Integer  
  
    strName = "test3"  
    i = 0  
    Set wbNewWorkbook = Workbooks.Add  
  
    strNewName = strName  
  
    Do While blnFileExists(strNewName & ".xls*")  
        i = i + 1
```



```
        strNewName = strName & i
    Loop

    wbNewWorkbook.SaveAs Filename:=strNewName
End Sub

Function blnFileExists(strFile As String) As Boolean
    If Dir(strFile) <> "" Then
        blnFileExists = True
    End If
End Function
```

代码使用了自定义函数 `blnFileExists` 来检查是否存在同名文件，如果存在则返回 `True`。并且，如果存在同名文件，则以该文件名后加上一个顺序数字来命名新工作簿。



本章内容 2018 年 3 月 16 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读\(87\): 关闭工作簿——Close 方法](#)

5. 关闭工作簿 —— Close 方法

可以使用 Workbooks 对象或者 Workbook 对象的 Close 方法关闭工作簿。Workbooks 对象的 Close 方法关闭所有打开着的工作簿，而 Workbook 对象的 Close 方法则关闭一个工作簿。

其语法分别为：

Workbooks 对象.close

关闭所有工作簿。如果有些工作簿发现修改，则会弹出是否保存对该工作簿所作的修改的提示框。

Workbook 对象.close (SaveChanges, Filename, RouteWorkbook)

关闭某个工作簿。其中：

- 所有参数均为可选参数。
- 参数 SaveChanges 指定是否保存对工作簿所作的修改。如果没有修改过工作簿，则忽略此参数。如果修改了工作簿，则使用此参数指定是否保存修改；如果参数值设置为 True，则关闭时将保存修改，此时如果工作簿还没有被命名，则使用参数 Filename 指定的名称，若没有指定 Filename 参数，则要求用户输入文件名；如果此参数值设置为 False，则关闭时将不会保存任何对工作簿的修改。
- 参数 Filename 指定保存修改的工作簿名称。

注意，上述操作仅对同一窗口中打开的工作簿有效。



示例代码 1：关闭所有打开的工作簿

下面的代码关闭所有打开的工作簿。

```
Sub CloseAllWB()  
    Workbooks.Close  
End Sub
```

对于 Excel 2007 及以后的版本来说，所有打开的工作簿必须处于同一个实例窗口中。

示例代码 2：关闭指定的工作簿

下面的代码关闭工作簿 test1.xlsx，并保存对该工作簿所作的修改。

```
Sub CloseWorkbook()  
    Workbooks("test1.xlsx").Close SaveChanges:=True  
End Sub
```

下面的代码关闭工作簿 test1.xlsx，并将对该工作簿的修改保存到 test2.xlsx 工作簿中。

```
Sub CloseWorkbook1()  
    Workbooks("test1.xlsx").Close SaveChanges:=True, _  
        Filename:="test2.xlsx"  
End Sub
```

注意，此时工作簿 test1.xlsx 仍保持原来的内容不变。

示例代码 3：模仿在关闭的工作簿中取值

经常有人提出，如何获取关闭的工作簿中的值？除了真正不打开工作簿来取值外，还可以换一种思路来实现。可以使用程序打开想要在其中取值的工作簿，获取该工作簿中的值，然后将其关闭。由于程序运行速度很快，用户可能根本没有意识到我们打开过该工作簿，就好像我们没有打开该工作簿而获得了其中的值一样。



示例代码如下：

```
Sub GetValue()  
    Workbooks.Open Filename:=ThisWorkbook.Path &  
    "\test2.xlsx"  
    ActiveWorkbook.Worksheets("Sheet1").Range("A1").Copy _  
    ThisWorkbook.Worksheets("Sheet1").Range("A1")  
    Workbooks("test1.xlsx").Close SaveChanges:=False  
End Sub
```

其中，语句 `ThisWorkbook.Path` 获取代码所在工作簿的路径。

运行效果如下图 5.1 所示。

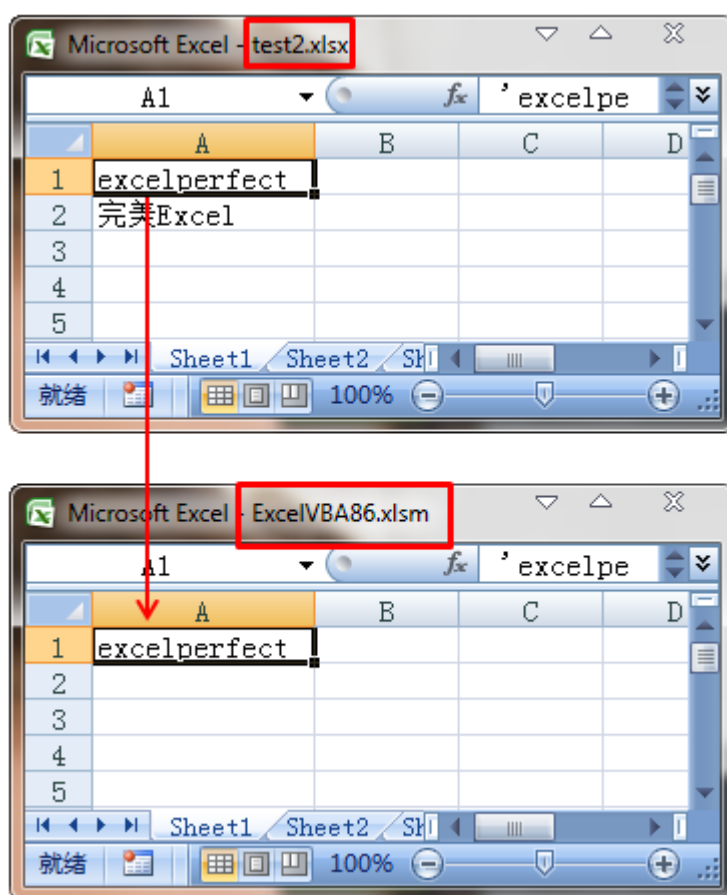


图 5.1



示例代码 4：拆分工作簿

下面的代码将当前工作簿中的每个工作表都保存为一个工作簿，新工作簿保存在与原工作簿相同的文件夹中，其名称为工作表的名称。

```
Sub SplitWorkbook()  
    Dim wks As Worksheet  
    Dim wkb As Workbook  
    Dim strNewWBName As String  
  
    For Each wks In ThisWorkbook.Sheets  
        strNewWBName = ThisWorkbook.Path & "\" & wks.Name &  
        ".xlsx"  
        wks.Copy  
        ActiveWorkbook.Sheets(1).Name = "Sheet1"  
        ActiveWorkbook.SaveAs Filename:=strNewWBName  
        ActiveWorkbook.Close SaveChanges:=False  
    Next wks  
End Sub
```



本章内容 2018 年 3 月 22 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读（88）：工作簿的基本属性](#)

6. 工作簿的基本属性

在 VBA 程序中，我们经常会使用工作簿的一些属性，用来获取工作簿的基本信息，例如 [Name 属性](#)、[FullName 属性](#)、[Path 属性](#)、[CodeName 属性](#)、[FileFormat 属性](#)、[ReadOnly 属性](#)、[Saved 属性](#)，等等。下面我们就来分别介绍这些属性。

Name 属性

返回工作簿的名称。其语法为：

Workbook 对象.Name

如下图 6.1 所示，返回代码所在工作簿或者当前工作簿的名称。

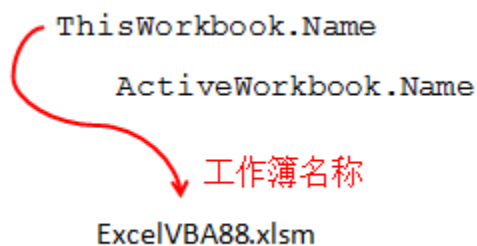


图 6.1

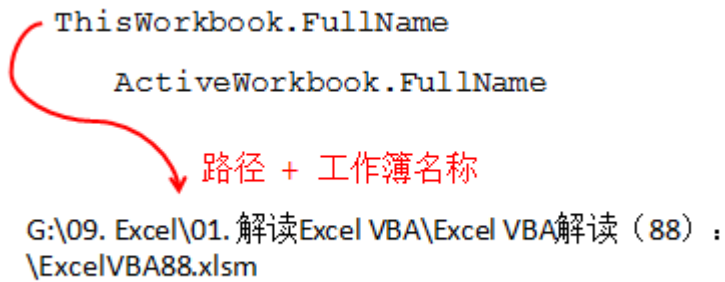


FullName 属性

返回工作簿对象的完整路径和名称，只读。其语法为：

Workbook 对象.FullName

如下图 6.2 所示，返回代码所在工作簿或者当前工作簿的完整路径和名称。



The diagram shows two code snippets: `ThisWorkbook.FullName` and `ActiveWorkbook.FullName`. A red curved arrow points from both to the text "路径 + 工作簿名称" (Path + Workbook Name). Below this, the full file path is displayed: `G:\09. Excel\01. 解读Excel VBA\Excel VBA解读 (88) : \ExcelVBA88.xlsm`.

图 6.2

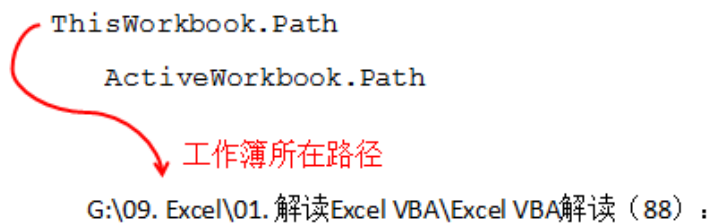
注意，如果代码所在工作簿没有保存，则只返回工作簿名。

Path 属性

返回工作簿的完整路径。其语法为：

Workbooks 对象.Path

如下图 6.3 所示，返回代码所在工作簿或者当前工作簿所在文件夹的完整路径。



The diagram shows two code snippets: `ThisWorkbook.Path` and `ActiveWorkbook.Path`. A red curved arrow points from both to the text "工作簿所在路径" (Workbook location path). Below this, the folder path is displayed: `G:\09. Excel\01. 解读Excel VBA\Excel VBA解读 (88) :`

图 6.3



如果工作簿没有被保存过，则该属性返回空字符串（""）。

CodeName 属性

返回对象的代码名称，如下图 6.4 红色框里所示为当前工作簿的代码名称。

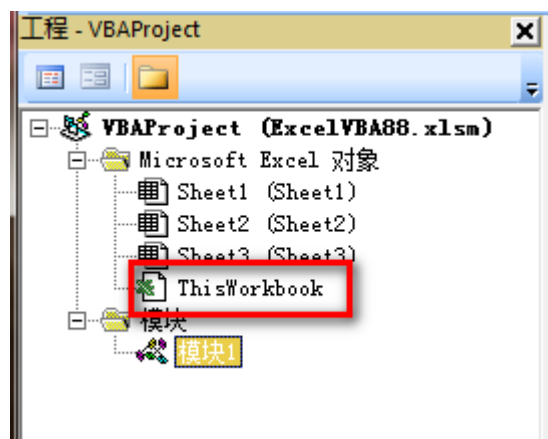


图 6.4

其语法为：

```
Workbooks 对象.CodeName
```

该属性只读，可以在工程资源管理器属性窗口修改工作簿对象的代码名称，但不能在运行时修改。

FileFormat 属性

返回工作簿的文件格式和/或类型，只读，xlFileFormat 常量。其语法为：

```
Workbook 对象.FileFormat
```



示例：将含有代码的工作簿保存为启用宏的工作簿

在 Excel 2007 及以后的版本中，如果工作簿含有宏，必须将其保存为启用宏的工作簿。因此，如果原工作簿为普通工作簿，后来因为需要在工作簿中添加了代码，此时若仍保存为普通工作簿，则会丢失代码。我们可以检查含有代码的工作簿是否是普通工作簿，如果是则保存为启用宏的工作簿。

```
If ActiveWorkbook.FileFormat = xlOpenXMLWorkbook Then
    ActiveWorkbook.SaveAs _
        FileFormat:=xlOpenXMLWorkbookMacroEnabled
End If
```

此外，Workbook 对象还有一个 `HasVBProject` 属性可以判断是否将工作簿保存为启用宏的工作簿。

ReadOnly 属性

其语法为：

Workbook 对象.ReadOnly

如果工作簿只读，则返回 True。

下面的代码当活动工作簿为只读时，将其另存为新的名称。

```
If ActiveWorkbook.ReadOnly Then
    ActiveWorkbook.SaveAs Filename:=ActiveWorkbook.Name &
        "副本.xlsm"
End If
```

Saved 属性

如果在保存之后又对工作簿进行了修改，那么该属性返回 True。其语法为：



Workbook 对象.Saved

如果想要关闭修改过的工作簿而不保存修改或者出现保存修改的提示，那么可以将该属性设置为 True。

```
ActiveWorkbook.Saved = True
```

示例 1：获取工作簿基本信息

下面的代码获取工作簿基本信息。

```
Sub WBInfo(wb As Workbook)
    Dim str As String

    str = "文件名称：" & wb.Name & vbCrLf & _
        "文件路径和名称：" & wb.FullName & vbCrLf & _
        "代码名称：" & wb.CodeName & vbCrLf & _
        "文件路径：" & wb.Path & vbCrLf & _
        "文件格式：" & wb.FileFormat

    If wb.ReadOnly Then
        str = str & vbCrLf & "工作簿只读。"
    Else
        str = str & vbCrLf & "工作簿可读写。"
    End If

    If wb.Saved Then
        str = str & vbCrLf & "工作簿已经保存。"
    Else
        str = str & vbCrLf & "工作簿需要保存。"
    End If
End Sub
```



```
MsgBox Prompt:=str, Title:="工作簿基本信息"  
End Sub
```

使用下面的代码进行测试：

```
Sub testWBInfo()  
    WBInfo ActiveWorkbook  
End Sub
```

结果如图 6.5 所示。

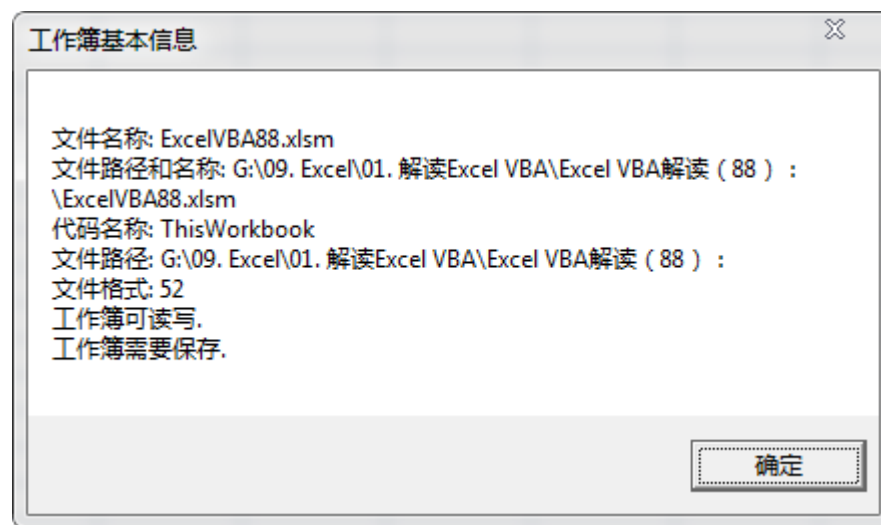


图 6.5

示例 2：获取当前工作簿名称

下面是一个自定义函数：

```
Function ThisWBName() As String  
    ThisWBName = ThisWorkbook.Name  
End Function
```

在工作表中使用该函数可以获取当前工作簿名称，如图 6.6 所示。




	A	B	C	D	E	F
1						
2		=This				
3		 ThisWBName				
4						
5						
6						
7						
8						
9						
10						

图 6.6

示例 3：获取当前工作簿所在文件路径和名称

下面是一个自定义函数：

```
Function ThisWBFullName() As String
    ThisWBFullName = ThisWorkbook.FullName
End Function
```

在工作表中使用该函数可以获取当前工作簿所在文件路径和名称，如图 6.7 所示。

	A	B	C	D	E	F	G	H	I
1									
2		=							
3									
4									
5									
6									
7									
8									
9									
10									

图 6.7





本章内容 2018 年 4 月 6 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读（89）：工作簿中的链接](#)

7. 工作簿中的链接

本章重点讲解 Workbook 对象返回链接信息的一些方法和属性，如下图 7.1 和图 7.2 所示。

方法/属性	作用
SaveLinkValues 属性	设置 Excel 是否保存工作簿的外部链接，可读写，布尔型值。 语句： <code>' 保存当前工作簿中的外部链接</code> <code>ActiveWorkbook.SaveLinkValues=True</code>
UpdateLinks 属性	返回或者设置 xlUpdateLink 常量，指定工作簿更新嵌入的 OLE 链接的设置，可读写。 详见示例：获取当前工作簿的链接更新设置信息

图 7.1

方法/属性	作用
BreakLink 方法	将链接到其他 Excel 工作簿或 OLE 源的公式转换成值。 详见示例：将当前工作簿中第 1 个链接公式转换成值
ChangeLink 方法	将链接从一个文档修改为另一个文档。 语句： <code>ActiveWorkbook.ChangeLink "E:\完美Excel\Book1.xlsx", _</code> <code>"E:\完美Excel\excelperfect.xlsx"</code> 修改 Excel 工作簿中的链接，假设当前工作簿中至少存在一个链接到另一个 Excel 源的公式。 详见示例：更新链接
FollowHyperlink 方法	设置到网页或者其他文档的链接。 语句： <code>ActiveWorkbook.FollowHyperlink Address:="http://office365.com"</code> 在浏览器中打开指定网页。
LinkInfo 方法	返回链接日期和更新状态。 详见示例：查看链接状态
LinkSources 方法	返回在工作簿中的链接数组。数组的名字包括链接文档、版本或者 DDE 或 OLE 服务。 如果没有链接则返回 Empty。 详见示例：获取工作簿中的链接信息
OpenLinks 方法	打开链接指向的文档。 详见示例：打开当前工作簿链接到的所有工作簿
SetLinkOnData 方法	设置在更新 DDE 链接时运行的过程名称。
UpdateLink 方法	更新 Excel、DDE 或 OLE 链接。 语句： <code>ActiveWorkbook.UpdateLink Name:=ActiveWorkbook.LinkSources</code> 更新当前工作簿中的所有链接。

图 7.2



下面，通过示例来讲解这些属性和方法的具体应用。

示例 1：获取当前工作簿的链接更新设置信息

下面的代码确定当前工作簿更新链接的设置并显示该信息。

```
Sub GetUpdateLinksInfo()  
    Select Case ActiveWorkbook.UpdateLinks  
        Case xlUpdateLinksAlways  
            MsgBox "当前工作簿的链接将总会更新。"  
        Case xlUpdateLinksNever  
            MsgBox "当前工作簿的链接将不会更新。"  
        Case xlUpdateLinksUserSetting  
            MsgBox "根据用户设置来更新当前工作簿的链接"  
    End Select  
End Sub
```

UpdateLinks 属性返回下列 xlUpdateLinks 常量之一：

- xlUpdateLinksAlways：对于指定工作簿来说，嵌入的 OLE 链接总会更新。
- xlUpdateLinksNever：对于指定的工作簿来说，嵌入的 OLE 链接从不更新。
- xlUpdateLinksUserSetting：对于指定的工作簿来说，根据用户设置来更新嵌入的 OLE 链接。

示例 2：将当前工作簿中第 1 个链接公式转换成值

下面的代码将当前工作簿中第 1 个链接转换成值。注意，要使代码正常运行，当前工作簿中至少存在 1 个链接到其他 Excel 工作簿中的公式。

```
Sub BreakLinkSample()  
    Dim varLinks As Variant
```



```

'将变量定义为 Excel 链接类型
varLinks = ActiveWorkbook.LinkSources _
    (Type:=xlLinkTypeExcelLinks)

'中断当前工作簿中的第一个链接
ActiveWorkbook.BreakLink _
    Name:=varLinks(1), _
    Type:=xlLinkTypeExcelLinks
End Sub

```

`BreakLink` 方法带有两个必需的参数。参数 `Name` 为字符串，指定链接的名称。参数 `Type` 指定链接的类型，为 `xlLinkType` 常量：
`xlLinkTypeExcelLinks` 表示链接到 Excel 源，`xlLinkTypeOLELinks` 表示链接到 OLE 源。

示例 3：更新链接

下面的代码使用新链接更新指定工作簿中的链接。

```

Sub FixLinks(wb As Workbook, strOldLink As String,
strNewLink As String)

    Dim varLinks As Variant
    Dim iIndex As Integer

    '获取链接源列表
    varLinks = wb.LinkSources(xlExcelLinks)

    '如果存在链接源,则检查是否有 strOldLink
    If Not IsEmpty(varLinks) Then
        For iIndex = 1 To UBound(varLinks)
            If StrComp(varLinks(iIndex), strOldLink,
vbTextCompare) = 0 Then
                wb.ChangeLink strOldLink, strNewLink,
xlLinkTypeExcelLinks
            End If
        Next iIndex
    End If
End Sub

```



```

        Exit For
    End If
Next iIndex
End If
End Sub

```

示例 4：查看链接状态

下面的代码获取工作簿中链接的状态。

```

Function GetLinkStatus(wb As Workbook, strLink As String)
As String

    Dim varLinks As Variant
    Dim iIndex As Integer
    Dim strResult As String
    Dim iStatus As Integer

    ' 获取链接源列表
    varLinks = wb.LinkSources(xlExcelLinks)

    ' 确保工作簿中有链接
    If IsEmpty(varLinks) Then
        GetLinkStatus = "工作簿中没有链接。"
        Exit Function
    End If

    ' 在 Case 中没有发现链接的默认结果
    strResult = "链接没找到!"

    For iIndex = 1 To UBound(varLinks)
        If StrComp(varLinks(iIndex), strLink,
vbTextCompare) = 0 Then
            iStatus = wb.LinkInfo(strLink,

```



```

xlLinkInfoStatus)
    Select Case iStatus
        Case xlLinkStatusCopiedValues
            strResult = "复制的值"
        Case xlLinkStatusIndeterminate
            strResult = "不确定"
        Case xlLinkStatusInvalidName
            strResult = "无效的名称"
        Case xlLinkStatusMissingFile
            strResult = "文件丢失"
        Case xlLinkStatusMissingSheet
            strResult = "工作表丢失"
        Case xlLinkStatusNotStarted
            strResult = "没开启"
        Case xlLinkStatusOK
            strResult = "确定"
        Case xlLinkStatusOld
            strResult = "旧的"
        Case xlLinkStatusSourceNotCalculated
            strResult = "源未计算"
        Case xlLinkStatusSourceNotOpen
            strResult = "源未打开"
        Case xlLinkStatusSourceOpen
            strResult = "源打开"
        Case Else
            strResult = "未知的状态码"
    End Select
    Exit For
End If
Next iIndex
GetLinkStatus = strResult

```



```
End Function
```

示例 5：获取工作簿中的链接信息

下面的示例获取并显示指定工作簿中的链接到其他工作簿的链接信息列表。

```
Sub LinkInfoSample(wb As Workbook)
    Dim varLinks As Variant
    Dim iIndex As Integer

    '获取基于 Excel 链接源的列表
    varLinks = wb.LinkSources(xlExcelLinks)

    If Not IsEmpty(varLinks) Then
        '遍历链接源
        For iIndex = 1 To UBound(varLinks)
            MsgBox "发现链接到 '" & varLinks(iIndex) & "'"
        Next iIndex
    Else
        MsgBox "工作簿" & wb.Name & "没有任何链接。"
    End If
End Sub
```

示例 6：打开当前工作簿链接到的所有工作簿

下面的代码获取当前工作簿中所有对其他工作簿的链接信息，并打开这些工作簿。

```
Sub OpenAllLinks()
    Dim varLinks As Variant
    Dim iIndex As Integer
```




```

varLinks = ActiveWorkbook.LinkSources(xlExcelLinks)

If Not IsEmpty(varLinks) Then
    For iIndex = LBound(varLinks) To UBound(varLinks)
        ActiveWorkbook.OpenLinks varLinks(iIndex)
    Next iIndex
Else
    MsgBox "当前工作簿没有包含外部链接。"
End If
End Sub

```

示例 7：获取指定工作簿中所有链接的状态

下面的代码获取参数 wb 代表的工作簿中所有链接的状态。

```

Sub CheckAllLinks(wb As Workbook)
    Dim varLinks As Variant
    Dim iIndex As Integer
    Dim str As String

    varLinks = wb.LinkSources(xlExcelLinks)

    If IsEmpty(varLinks) Then
        MsgBox wb.Name & "中没有任何链接。"
    Else
        For iIndex = 1 To UBound(varLinks)
            MsgBox "工作簿名称： " & wb.Name & vbCrLf & _
                "链接源： " & varLinks(iIndex) & _
                "状态： " & GetLinkStatus(wb, CStr(varLinks(iIndex)))
        Next iIndex
    End If
End Sub

```



```
End Sub
```

代码中，使用了上文查看链接状态示例中的自定义函数 GetLinkStatus 函数。使用下面的代码查看当前工作簿中链接的状态：

```
Sub test()  
    CheckAllLinks ActiveWorkbook  
End Sub
```

运行后的结果如图 7.3 所示。

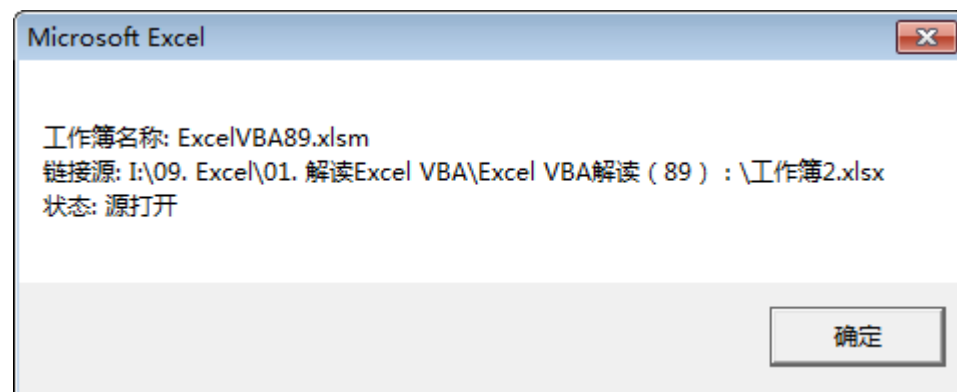


图 7.3



本章内容 2018 年 4 月 12 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读 \(90\): 文档属性](#)

8. 文档属性

本章重点讲解 Workbook 对象的内置文档属性和自定义文档属性。

BuiltinDocumentProperties 属性

返回 `DocumentProperties` 集合，代表指定工作簿的所有内置文档属性，只读。其语法为：

Workbook 对象.BuiltinDocumentProperties

该属性返回内置文档属性的完整的集合，使用 `Item` 方法指定属性的名称或索引值来返回集合中单个的成员（即 `DocumentProperty` 对象）。

下面的代码列出了内置文档属性的名称：

```
Sub ListBuiltinDocProperties()  
    Dim lngRow As Long, lngCol As Long  
    Dim DocProperty As DocumentProperty  
  
    lngRow = 1  
    lngCol = 1  
  
    For Each DocProperty In  
ActiveWorkbook.BuiltinDocumentProperties  
        Cells(lngRow, lngCol).Value = DocProperty.Name  
        lngRow = lngRow + 1  
    End For  
End Sub
```



```

        If lngRow = 16 Then
            lngCol = lngCol + 1
            lngRow = 1
        End If
    Next DocProperty

    Cells.EntireColumn.AutoFit
End Sub

```

在 Excel 2007 工作簿中运行的结果如图 8.1 所示。

	A	B	C
1	Title	Number of characters	Content type
2	Subject	Security	Content status
3	Author	Category	Language
4	Keywords	Format	Document version
5	Comments	Manager	
6	Template	Company	
7	Last author	Number of bytes	
8	Revision number	Number of lines	
9	Application name	Number of paragraphs	
10	Last print date	Number of slides	
11	Creation date	Number of notes	
12	Last save time	Number of hidden Slides	
13	Total editing time	Number of multimedia clips	
14	Number of pages	Hyperlink base	
15	Number of words	Number of characters (with spaces)	
16			

图 8.1

下面的语句获取当前工作簿作者名称：

```
ActiveWorkbook.BuiltinDocumentProperties("Author").Value
```

CustomDocumentProperties 属性

返回或者设置 `DocumentProperties` 集合，代表指定工作簿的所有自定义文档属性。其语法为：

```
Workbook 对象.CustomDocumentProperties
```

自定义属性对应的对话框如图 8.2 所示。



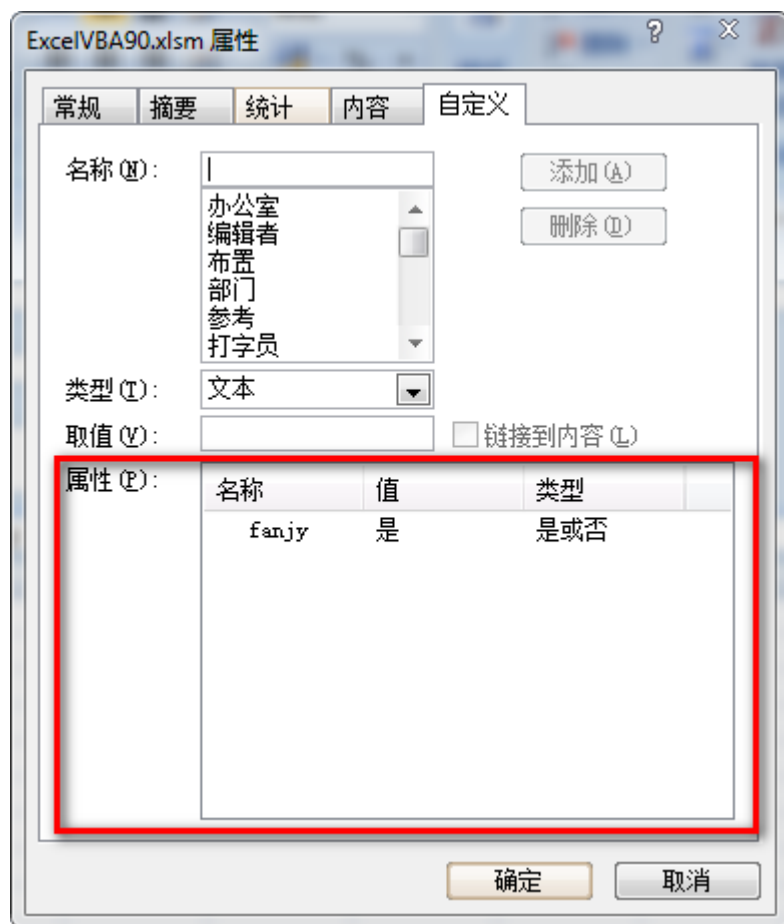


图 8.2

该属性返回自定义文档属性的完整的集合，使用 [Item 方法](#) 指定属性的名称或索引值来返回集合中单个的成员（即 [DocumentProperty 对象](#)）。

下面的代码列出了自定义文档属性的名称和值：

```
Sub ListCustomDocumentProperty()
    Dim lngRow As Long
    Dim DocProperty

    lngRow = 1

    For Each DocProperty In
ActiveWorkbook.CustomDocumentProperties
        Cells(lngRow, 1).Value = DocProperty.Name
        Cells(lngRow, 2).Value = DocProperty.Value
    
```



```

        lngRow = lngRow + 1
    Next DocProperty
End Sub

```

下面的代码，给当前工作簿添加自定义属性：

```

Sub AddCustomProperty()
    With ActiveWorkbook.CustomDocumentProperties
        .Add Name:="LastModifiedBy", _
            LinkToContent:=True, _
            Type:=msoPropertyTypeString, _
            LinkSource:="Last author"
        .Add Name:="CustomNumber", _
            LinkToContent:=False, _
            Type:=msoPropertyTypeNumber, _
            Value:=1000
        .Add Name:="CustomString", _
            LinkToContent:=False, _
            Type:=msoPropertyTypeString, _
            Value:="This is a custom property."
        .Add Name:="CustomDate", _
            LinkToContent:=False, _
            Type:=msoPropertyTypeDate, _
            Value:=Date
    End With
End Sub

```

自定义文档属性是存储要在代码中使用的相关工作簿信息的好地方。例如，如图 8.2 所示，在工作簿中创建了自定义属性“fanjy”，另外有一个用于创建自定义工具栏的加载项。如果激活含有自定义属性“fanjy”的工作簿，那么应用自定义工具栏。

在标准模块中，设置常量来识别工具栏名称和属性名称：

```

Public Const gstrISFanjy As String = "fanjy"

```



```
public Const gstrGTool As String = "MyTool"
```

在类模块中，创建应用级事件基于自定义属性来显示和隐藏工具栏：

```
Private Sub xlApp_WindowActivate(ByVal wb As Workbook, _  
                                ByVal wn As Window)  
  
    Dim blnFanjy As Boolean  
    On Error Resume Next  
    blnFanjy = wb.CustomDocumentProperties(gstrISFanjy)  
    On Error GoTo 0  
    If blnFanjy Then  
        Application.CommandBars(gstrGTool).Visible = True  
    End If  
End Sub  
  
Private Sub xlApp_WindowDeactivate(ByVal wb As Workbook,  
_                                     ByVal wn As Window)  
  
    On Error Resume Next  
    Application.CommandBars(gstrGTool).Visible = False  
End Sub
```

示例：在自定义文档属性中存储值

有时，可以在自定义文档属性中存储值为以后使用。本例中，每次调用代码时，存储在自定义属性中的值都会增加 1。

```
Sub SaveValueInCustomDocProperty()  
    Const strVer As String = "MyWbVer"  
  
    '如果名字不存在,那么创建并设置初始值为 1  
    On Error Resume Next  
    Dim DocProperty As DocumentProperty  
    Set DocProperty =
```



```

ThisWorkbook.CustomDocumentProperties(strVer)
    If Err.Number > 0 Then
        ThisWorkbook.CustomDocumentProperties.Add _
            Name:=strVer, _
            LinkToContent:=False, _
            Type:=msoPropertyTypeNumber, _
            Value:=1
    Else
        '如果名字存在,需要将其值加 1
        Dim strDocVal As String
        strDocVal =
ThisWorkbook.CustomDocumentProperties(strVer).Value
        '重设名称为新值
        ThisWorkbook.CustomDocumentProperties(strVer).Value
= CLng(strDocVal) + 1
    End If
    Set DocProperty = Nothing
End Sub

```



本章内容 2018 年 4 月 16 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
[Excel VBA 解读 \(91\): 看看工作簿中有哪些事件](#)

9. 看看工作簿中有哪些事件

事件就是发生在对象上的事情。例如，对于正在运行的机器来说，突然停止就是一个事件，这个事件会引发一系列的响应，包括检查机器、分析原因、处理故障等。

Excel 为对象预先定义好了一系列事件，允许我们创建程序来响应各种由于程序或者用户行为而发生的动作。

我们在《[Excel VBA 解读: Worksheet 对象篇](#)》中讲解工作表事件的内容中，简单介绍过 Excel 事件并列举了一些工作表事件应用示例。

Excel 也为工作簿对象预定义了一系列事件，本书接下来就将详细讲解这方面的内容。

Workbook 对象相关事件

下表列出了 Workbook 对象相关事件及发生的情形。



事件	发生情形
Activate	当激活工作簿时
AddinInstall	当安装作为加载项的工作簿时
AddinUninstall	当卸载作为加载项的工作簿时
AfterSave	在保存工作簿后
AfterXmlExport	在Microsoft Excel保存或导出指定工作簿中的XML数据后
AfterXmlImport	在刷新现有的XML数据连接或将新的XML数据导入指定的Microsoft Excel工作簿之后
BeforeClose	在工作簿关闭之前，或者修改工作簿后询问用户保存修改之前
BeforePrint	在打印工作簿之前
BeforeSave	在保存工作簿之前
BeforeXmlExport	在Microsoft Excel保存或导出指定工作簿中的XML数据之前
BeforeXmlImport	在刷新现有的XML数据连接或将新的XML数据导入指定的Microsoft Excel工作簿之前
Deactivate	在使当前工作簿成为非活动工作簿之前
ModelChange	在Excel数据模型修改之后
NewChart	当在工作簿中创建新图表时

(续表)

事件	发生情形
NewSheet	当在工作簿中创建新工作表时
Open	当打开工作簿时
PivotTableCloseConnection	在数据透视表报告关闭与其连接的数据源之后
PivotTableOpenConnection	在数据透视表报告打开与其连接的数据源之后
RowsetComplete	当用户在钻取整个记录集或调用OLAP数据透视表上的行集操作时
SheetActivate	当激活工作簿中任一工作表时
SheetBeforeDelete	当删除工作簿中任一工作表时
SheetBeforeDoubleClick	当双击工作簿中任一工作表时，在默认的双击操作之前
SheetBeforeRightClick	当右击工作簿中任一工作表时，在默认的右击操作之前
SheetCalculate	在重新计算工作簿中任一工作表之后，或者任何修改的数据绘制在图表中之后
SheetChange	当用户或者外部链接改变任意工作表中的单元格时
SheetDeactive	当工作簿中任一工作表成为非活动工作表时
SheetFollowHyperlink	当单击Excel中的任意超链接时
SheetLensGalleryRenderComplete	当标注图库的图标（动态和静态）已完成工作表呈现时



(续表)

事件	发生情形
SheetPivotTableAfterValueChange	在数据透视表里面的单元格或单元格区域被编辑或者重新计算（包含公式的单元格）之后
SheetPivotTableBeforeAllocateChanges	在更改被应用到数据透视表之前
SheetPivotTableBeforeCommitChanges	在针对数据透视表的OLAP数据源提交更改之前
SheetPivotTableBeforeDiscardChanges	在放弃对数据透视表的修改之前
SheetPivotTableChangeSync	在对数据透视表修改之后
SheetPivotTableUpdate	在已更新数据透视表报告的工作表之后
SheetSelectionChange	当任意工作表中的选择区域更改时（不包括图表工作表中的选区更改）
SheetTableUpdate	在更新工作表中的表之后
Sync	已弃用
WindowActivate	当激活任一工作簿窗口时
WindowDeactivate	当使任一工作簿窗口成为非活动窗口时
WindowResize	当调整任一工作簿窗口大小时

Workbook 对象相关事件的位置

在 ThisWorkbook 代码模块中处理当前工作簿对象的事件，如图 9.1 所示。

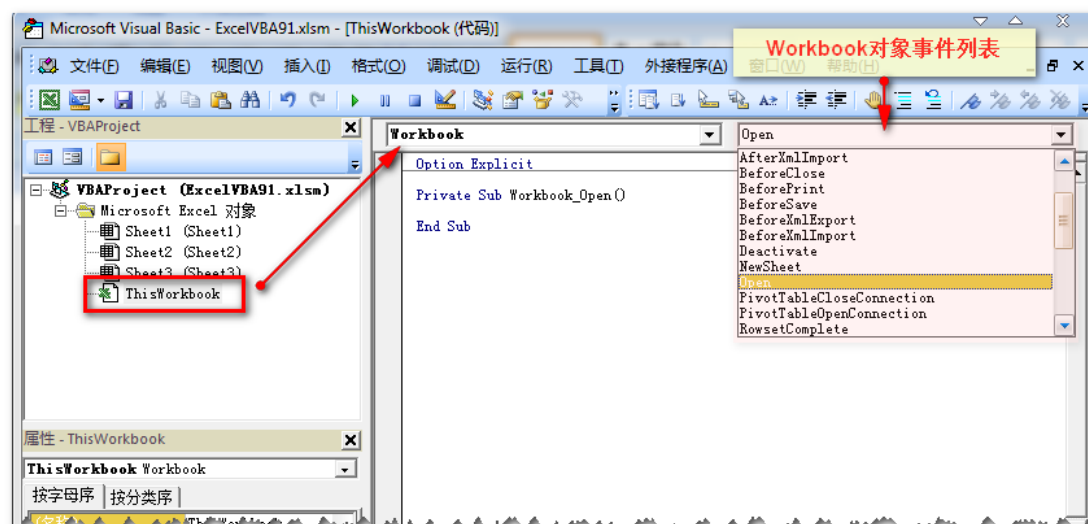


图 9.1

在 VBE 中，双击左侧工程资源管理器中的 ThisWorkbook 对象，将在右侧显



示其代码窗口，在该窗口顶部左侧的下拉列表中选择“Workbook”，在右侧的下拉列表中选择相应的事件，Excel 会在代码窗口中自动插入过程名。默认事件过程代码为：

```
Private Sub Workbook_Open()  
    '在这里输入想要在打开工作簿时运行的代码  
End Sub
```

在本书后面的内容中，我们将通过示例来讲解其中比较常用的一部分 Workbook 对象事件。



本章内容 2018 年 4 月 23 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读（92）：Workbook 对象的
Open 事件和 BeforeClose 事件](#)

10. Workbook 对象的 Open 事件 和 BeforeClose 事件

本章将详细讲解 Workbook 对象常用的两个事件：

- Open 事件
- BeforeClose 事件

Workbook_Open 事件

当打开工作簿时发生 Workbook_Open 事件。

在 ThisWorkbook 代码模块中输入下面的代码，使工作簿打开时弹出一个消息框：

```
Private Sub Workbook_Open()  
    MsgBox "欢迎来到[完美 Excel]微信公从号!"  
End Sub
```

在打开工作簿时，弹出如图 10.1 所示的消息框。

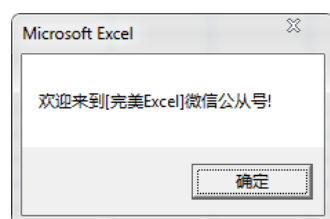


图 10.1



Workbook_BeforeClose 事件

在关闭工作簿之前发生该事件。如果修改了工作簿但没有保存修改，那么该事件发生在询问用户保存修改之前。

其语法为：

```
Workbook_BeforeClose(Cancel As Boolean)
```

说明：

- 参数 Cancel 为布尔值，必需。事件发生时为 False；如果事件过程将此参数设置为 True，则停止关闭工作簿并且工作簿保持打开状态。

下面的代码在关闭工作簿时总是会保存对该工作簿的修改，而不会弹出询问是否保存修改的消息框。

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)
    If Me.Saved = False Then Me.Save
End Sub
```

Workbook_Open 事件可用于在打开工作簿时初始化工作簿，设置计算模式、设置屏幕、添加自定义菜单、为工作表中的组合框或列表框添加数据，等等。Workbook_BeforeClose 事件可用于恢复工作簿的初始设置、阻止用户关闭工作簿，等等。

示例 1：设定特定用户才能操作工作表

下面的代码在打开工作簿时检查用户名是否为“完美 Excel”，如果不是，则保护工作表，防止其他用户对其进行修改。

```
Private Sub Workbook_Open()
    Dim wks As Worksheet
    If Application.UserName <> "完美 Excel" Then
        For Each wks In Worksheets
            wks.Protect UserInterfaceOnly:=True
        Next wks
    End If
End Sub
```



```
Next wks  
End If  
End Sub
```

示例 2：要求用户输入指定值

只有当工作簿中工作表 Sheet1 的单元格 A1 中的值为“完美 Excel”时，才能关闭该工作簿。代码如下：

```
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    If Worksheets("Sheet1").Range("A1") <> "完美 Excel"  
Then  
        MsgBox "请在工作表 Sheet1 的单元格中输入""完美 Excel""  
        Cancel = True  
    End If  
End Sub
```

关闭工作簿时，如果工作表 Sheet1 的单元格 A1 中的值不是“完美 Excel”，则会弹出如下图 10.2 所示的提示消息。

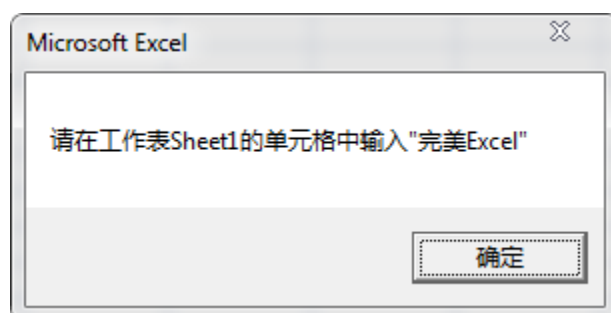


图 10.2

示例 3：添加/删除自定义快捷菜单

当仅需要在特定的工作簿中添加自定义快捷菜单时，我们可以在 Workbook_Open 事件中添加自定义快捷菜单，在 Workbook_BeforeClose 事件中删除该菜单。这是经常使用的一种技术。



下面的代码在打开工作簿时，在单元格右键菜单中添加一个名为“完美 Excel”的按钮，关闭工作簿时将其删除。

```
Private Sub Workbook_Open()  
    Dim cmb As CommandBarControl  
    On Error Resume Next  
    Application.CommandBars("Cell").Controls("完美  
Excel").Delete  
    Set cmb = Application.CommandBars("Cell").Controls.Add  
    —  
        (Type:=msoControlButton, Temporary:=True)  
    cmb.Caption = "完美 Excel"  
    cmb.OnAction = "excelperfect"  
End Sub  
  
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    Dim strMsg As String  
    Dim Response  
  
    If Not ThisWorkbook.Saved Then  
        strMsg = "是否想要保存所做的修改?"  
        Response = MsgBox(strMsg, vbQuestion +  
vbYesNoCancel)  
        Select Case Response  
            Case vbYes  
                ThisWorkbook.Save  
            Case vbNo  
                ThisWorkbook.Saved = True  
            Case vbCancel  
                Cancel = True  
            Exit Sub  
        End Select  
    End If  
    On Error Resume Next
```




```
Application.CommandBars("Cell").Controls("完美Excel").Delete  
End Sub
```

代码中，语句：

```
ThisWorkbook.Save
```

将保存工作簿的修改而不会出现提示信息。

语句：

```
ThisWorkbook.Saved = True
```

放弃对工作簿所作的修改而不出现提示信息。

打开该工作簿时的右键菜单如图 10.3 所示。

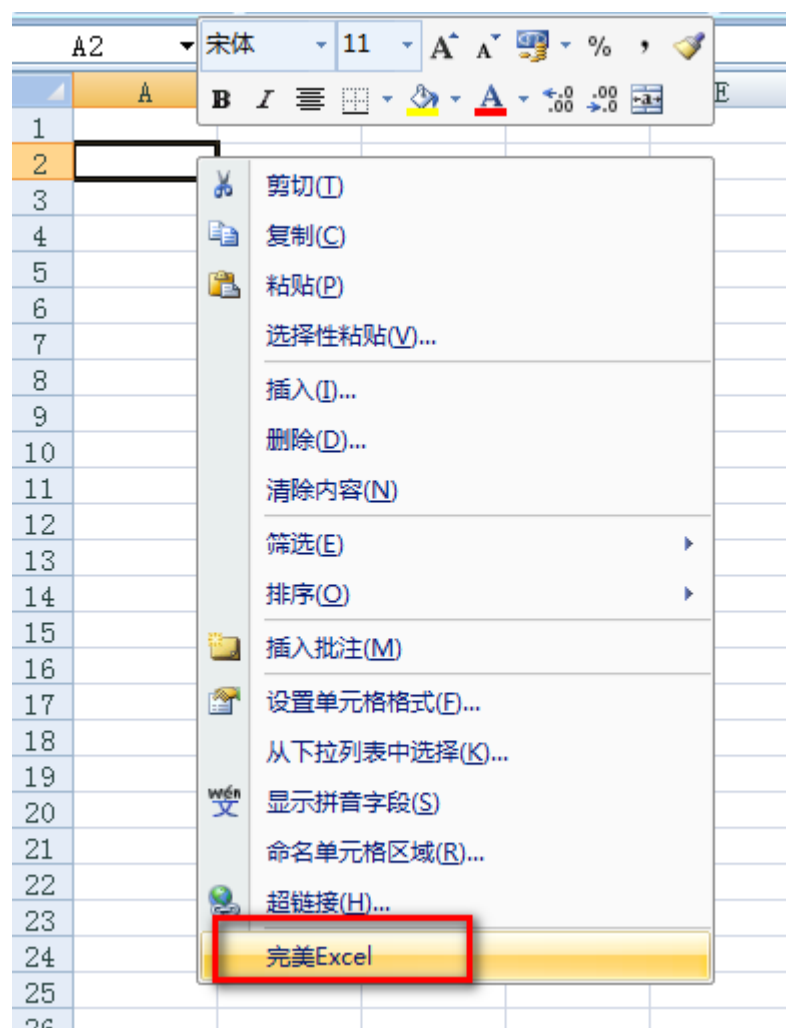


图 10.3



在 BeforeClose 事件中，我们创建了自己的“保存”对话框。之所以这样做，是因为如果在修改了工作簿而没有保存时，Excel 会弹出是否保存修改对话框，这个对话框发生在 BeforeClose 事件之后，此时，如果用户选择“取消”而回到工作簿，那么菜单中的自定义按钮已被删除。



本章内容 2018 年 5 月 2 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为

Excel VBA 解读 (93): Workbook 对象的
SheetActivate 事件、SheetDeactivate 和
SheetSelectionChange 事件

11. Workbook 对象的 SheetActivate 事件、 SheetDeactivate 和 SheetSelectionChange 事件

本章重点详细讲解 Workbook 对象的 3 个事件：

- SheetActivate 事件
- SheetDeactivate 事件
- SheetSelectionChange 事件。

Workbook_SheetActivate 事件

当激活任意工作表时发生 Workbook_SheetActivate 事件。其语法为：

```
Workbook_SheetActivate (ByVal Sh As Object)
```

说明：

- 参数 Sh，必需，表示被激活的工作表，可以是图表工作表或标准工作表。

在 ThisWorkbook 代码模块中输入下面的代码，当激活工作表时弹出一个显示该工作表名字的消息框：

```
Private Sub Workbook_SheetActivate (ByVal Sh As Object)  
    MsgBox "当前工作表是：" & Sh.Name  
End Sub
```



Workbook_SheetDeactivate 事件

当使工作表变为非活动工作表时发生 Workbook_SheetDeactivate 事件。其语法为：

```
Workbook_SheetDeactivate (ByVal Sh As Object)
```

说明：

- 参数 Sh，必需，表示变为非活动工作表的工作表，可以是图表工作表或标准工作表。

在 ThisWorkbook 代码模块中输入下面的代码，当使工作表变为非活动工作表时弹出一个显示该工作表名字的消息框：

```
Private Sub Workbook_SheetDeactivate (ByVal Sh As Object)  
    MsgBox "走了,工作表:" & Sh.Name  
End Sub
```

当 SheetActivate 事件和 SheetDeactivate 事件都存在时，先发生 SheetDeactivate 事件，再发生 SheetActivate 事件。

Workbook_SheetSelectionChange 事件

当改变任意工作表（图表工作表除外）的单元格选择时发生 Workbook_SheetSelectionChange 事件。其语法为：

```
Workbook_SheetSelectionChange (ByVal Sh As Object, ByVal  
Target As Range)
```

说明：

- 参数 Sh，必需，包含新的单元格选区的工作表。
- 参数 Target，必需，新选择的单元格区域。

在 ThisWorkbook 代码模块中输入下面的代码，将在工作簿状态栏中显示当



前工作表名称和所选区域的地址：

```
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As Range)
    Application.StatusBar = Sh.Name & ":" & Target.Address
End Sub
```

示例 1：只允许访问指定的工作表

下面的代码总是使工作簿中的第 1 个工作表为当前工作表，限制对其他工作表的访问。

```
Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    '将 1 修改为允许访问的工作表索引号或名称
    Worksheets(1).Activate
End Sub
```

示例 2：限制用户必须包含指定内容

下面的代码确定工作表 Sheet1 中单元格 A1 的内容为“完美 Excel”，否则不能访问其他工作表。

```
Private Sub Workbook_SheetDeactivate(ByVal Sh As Object)
    If Sheet1.Range("A1") <> "完美 Excel" Then
        MsgBox "Sheet1 工作表的单元格 A1 的内容必须是""完美 Excel""
        Sheet1.Activate
    End If
End Sub
```

示例 3：限制用户必须在指定区域中操作

下面的代码限制用户必须在工作表 Sheet1 的单元格区域“A1:D3”中进行操作：



```
Private Sub Workbook_SheetSelectionChange(ByVal Sh As  
Object, ByVal Target As Range)  
    Dim rng As Range  
  
    Set rng = Sheet1.Range("A1:D3")  
  
    If Sh.Name = "Sheet1" Then  
        If Intersect(Target, rng) Is Nothing Then  
            Sheet1.Range("A1").Select  
        End If  
    End If  
End Sub
```

在工作表 Sheet1 中，选择单元格区域 A1:D3 之外的单元格时，会自动跳到单元格 A1，如下图 11.1 所示。

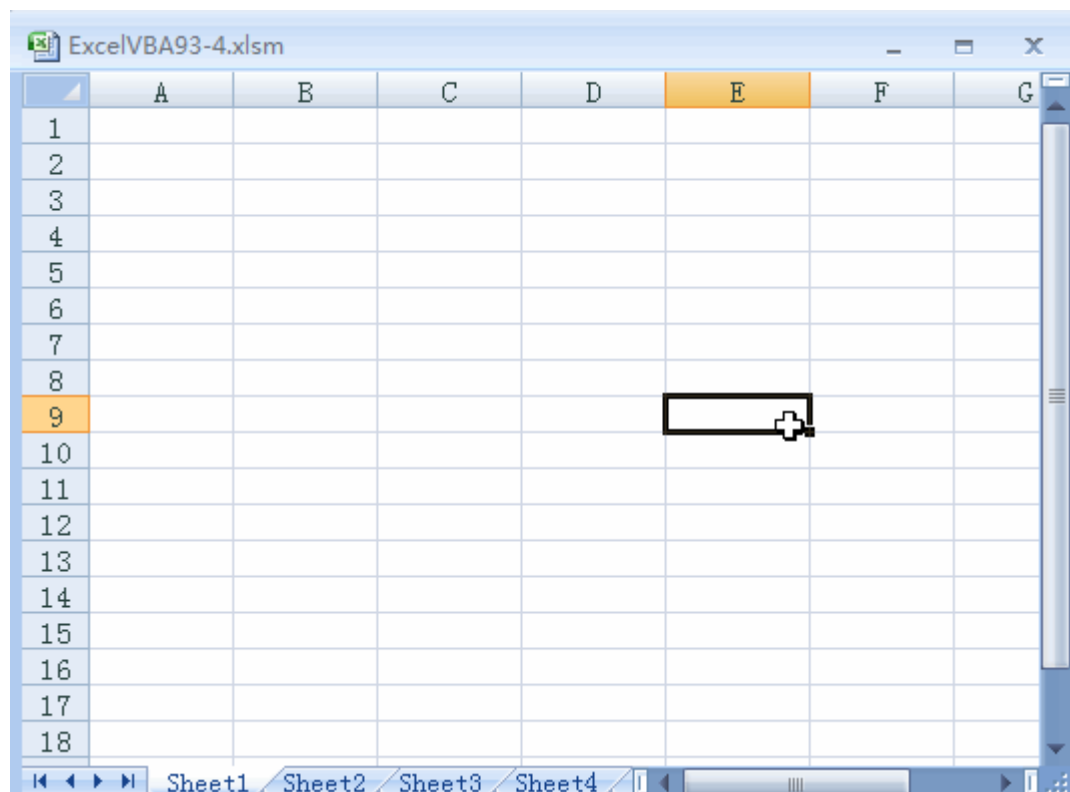


图 11.1 (演示视频截图，详细视频见完美 Excel 微信公众号)



示例 4：阻止用户修改工作表名称

Excel 中没有能够阻止用户修改工作表名称的事件，因此只能联合已有的事件实现这一功能。下面是 ThisWorkbook 代码模块中的代码：

```
Private mstr_ActiveSheetPreviousName As String
Private lng_DeactivatedSheetIndex As Long
Private mstr_ActiveSheetName As String

Private Sub Workbook_Open()
    mstr_ActiveSheetPreviousName =
ThisWorkbook.ActiveSheet.Name
End Sub

Private Sub Workbook_SheetDeactivate(ByVal Sh As Object)
    lng_DeactivatedSheetIndex = Sh.Index
    SheetNameChange 0
End Sub

Private Sub Workbook_SheetActivate(ByVal Sh As Object)
    SheetNameChange 1
End Sub

Private Sub Workbook_SheetSelectionChange(ByVal Sh As
Object, ByVal Target As Range)
    SheetNameChange 2
End Sub

Sub SheetNameChange(ByVal lngCaller As Long)
    Select Case lngCaller
        Case 0
            If lng_DeactivatedSheetIndex <>
ThisWorkbook.ActiveSheet.Index Then
                If
ThisWorkbook.Sheets(lng_DeactivatedSheetIndex).Name <>
```



```

mstr_ActiveSheetPreviousName Then

ThisWorkbook.Sheets(lng_DeactivatedSheetIndex).Name =
mstr_ActiveSheetPreviousName

        mstr_ActiveSheetPreviousName =
mstr_ActiveSheetName

        NameChanged

    End If

End If

Case 1

    mstr_ActiveSheetPreviousName =
ThisWorkbook.ActiveSheet.Name

    mstr_ActiveSheetName =
ThisWorkbook.ActiveSheet.Name

Case 2

    mstr_ActiveSheetName =
ThisWorkbook.ActiveSheet.Name

    If mstr_ActiveSheetName <>
mstr_ActiveSheetPreviousName Then

        ThisWorkbook.ActiveSheet.Name =
mstr_ActiveSheetPreviousName

        NameChanged

    End If

End Select

End Sub

Sub NameChanged()

    MsgBox "您不能修改工作表名称!"

End Sub

```

此时，当用户修改任意工作表名称时，工作表名称会恢复原名，并弹出“您不能修改工作表名称”的提示信息。



本章内容 2018 年 5 月 8 日首发于
[完美 Excel] 微信公众号 *excelperfect*
原标题为
**Excel VBA 解读 (94): Workbook 对象的
BeforePrint 事件**

12 . Workbook 对象的 BeforePrint 事件

在打印工作簿或其中的任意内容时发生 Workbook_BeforePrint 事件。其语法为:

```
Workbook_BeforePrint(Cancel As Boolean)
```

说明:

- 参数 Cancel 设置为 True 时将阻止打印工作簿。

示例 1 : 打印前重新计算工作表

下面的代码在任何打印操作前重新计算当前工作簿中的所有工作表。

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)  
    Dim wks As Worksheet  
    For Each wks In Worksheets  
        wks.Calculate  
    Next wks  
End Sub
```



示例 2：打印前添加页眉和页脚

BeforePrint 事件可用于在工作表打印前添加页眉和页脚。例如，下面的代码在打印当前工作表时在打印页的页脚中添加工作簿路径和名称：

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
    ActiveSheet.PageSetup.RightFooter =
ActiveSheet.FullName
End Sub
```

示例 3：跟踪工作簿打印情况

下面的代码跟踪每次打印工作簿中的工作表时的情况，在工作表“打印日志”中记录了每次打印的日期和时间、用户名和打印的工作表名。

```
Private Sub Workbook_BeforePrint(Cancel As Boolean)
    Dim lngLastRow As Long
    Dim wksPrintLog As Worksheet
    Set wksPrintLog = Worksheets("打印日志")
    lngLastRow = wksPrintLog.Range("A" &
Rows.Count).End(xlUp).Row + 1
    With wksPrintLog
        .Cells(lngLastRow, 1).Value = Now
        .Cells(lngLastRow, 2).Value = Application.UserName
        .Cells(lngLastRow, 3).Value = ActiveSheet.Name
    End With
End Sub
```



本章内容 2018 年 5 月 14 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
**Excel VBA 解读 (95): Workbook 对象的
WindowResize 事件**

13. Workbook 对象的 WindowResize 事件

当调整工作簿窗口的大小时，发生 `Workbook_WindowResize` 事件。其语法为：

```
Workbook_WindowResize (ByVal Wn As Window)
```

说明：

- 参数 `Wn` 代表被调整大小的工作簿窗口。

示例 1：禁止调整工作簿窗口大小

下面的代码禁止用户调整当前工作簿窗口的大小：

```
Private Sub Workbook_WindowResize (ByVal Wn As Window)  
    Wn.EnableResize = False  
End Sub
```

此时，工作簿窗口中的最大化和最小化按钮被移走，你无法调整工作簿大小，如图 13.1 所示。



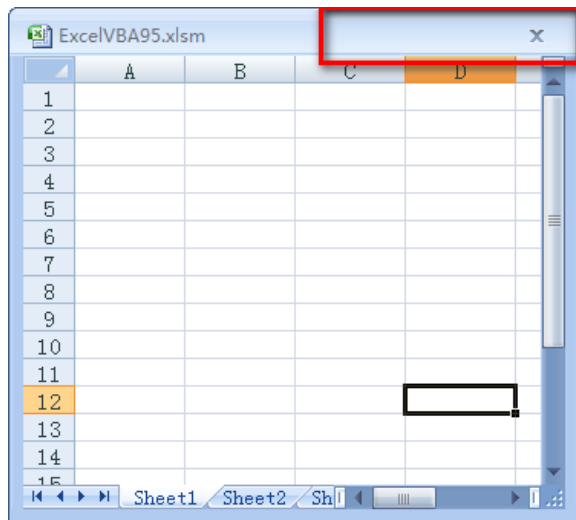


图 13.1

示例 2：在状态栏中显示正在调整窗口大小的工作簿名称

下面的代码在 Excel 状态栏中显示正在调整窗口大小的工作簿名称：

```
Private Sub Workbook_WindowResize(ByVal Wn As Window)
    Application.StatusBar = "你正在调整工作簿 " & _
        Wn.Caption & " 的窗口大小."
End Sub
```

此时，在调整工作簿窗口大小时，Excel 状态栏中的信息如图 13.2 所示。

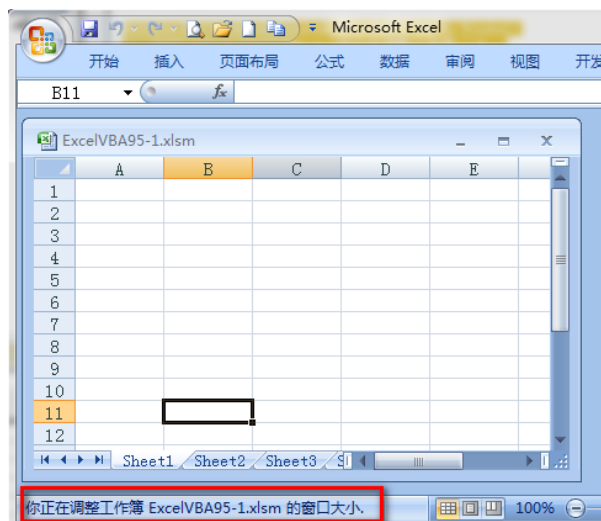


图 13.2



本章内容 2018 年 5 月 18 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读 \(96\): Workbook 对象的 BeforeSave 事件](#)

14.Workbook 对象的 BeforeSave 事件

在保存工作簿之前，发生 Workbook_BeforeSave 事件。其语法为：

```
Workbook_BeforeSave (ByVal SaveAsUI As Boolean, Cancel As Boolean)
```

说明：

- 参数 SaveAsUI 为布尔值，如果设置为 True，将打开“另存为”对话框。
- 参数 Cancel 为布尔值，当该事件发生时为 False。如果将该参数设置为 True，则不会保存工作簿。

示例 1：让用户决定是否保存工作簿

下面的代码在保存工作簿前，给出提示信息，让用户决定是否保存工作簿。

```
Private Sub Workbook_BeforeSave (ByVal SaveAsUI As Boolean, Cancel As Boolean)  
    Dim str  
    str = MsgBox("希望保存工作簿吗?", vbYesNo)  
    If str = vbNo Then Cancel = True  
End Sub
```

在保存工作簿时，弹出如图 14.1 所示的消息框，供用户选择是否保存该工作簿。



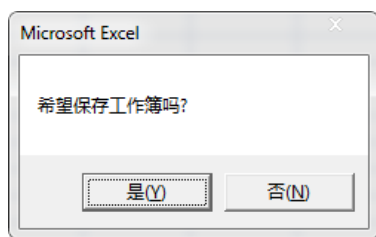


图 14.1

示例 2：限定用户必须在指定的单元格中输入数据

在工作簿保存前，检查指定的单元格中是否包含数据。如果这些单元格中没有全部输入数据，那么不会保存该工作簿。也就是说，关闭该工作簿时，不会保存对该工作簿所做的修改。

```
Private Sub Workbook_BeforeSave(ByVal SaveAsUI As
Boolean, Cancel As Boolean)

    If WorksheetFunction.CountA(Worksheets("Sheet1"). _
        Range("A1,A3,C2,D1")) < 4 Then

        MsgBox "不能保存本工作簿!!!" & vbCrLf & _
            "因为单元格 A1,A3,C2,D1 没有输入数据!"

        Cancel = True

    End If

End Sub
```

将工作表 Sheet1 中的单元格 D1 留空，单击“保存”按钮，会弹出如图 14.2 所示的消息框。只有当工作表 Sheet1 中的单元格 A1、A3、C2、D1 都输入有数据时，才能够正常保存该工作簿。

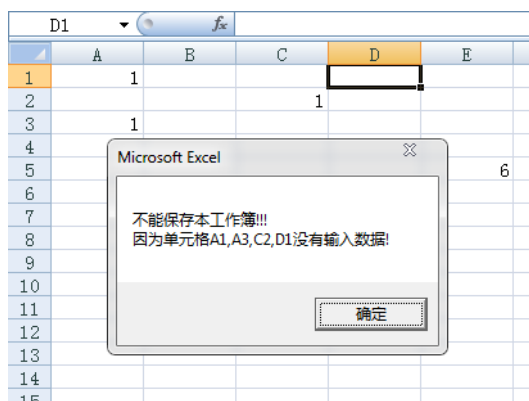


图 14.2



本章内容 2018 年 5 月 21 日首发于

[完美 Excel]微信公众号 [excelperfect](#)

原标题为

Excel VBA 解读 (97): 工作簿事件示例——

在单元格快捷菜单中添加自定义列表

15.工作簿事件示例：在单元格快捷菜单中 添加自定义列表

事件确实给 Excel VBA 编程带来了很多的奇妙，让 Excel 真正实现与用户的实时互动。在前面的系列章节中，通过讲解 Workbook 对象的几个事件，我们可以体会到事件编程给我们带来的新奇的感觉。本章及后面的内容会再列举一些实用示例，进一步体会到事件编程的妙处，以加深对工作簿事件的理解，方便大家在实际中灵活应用工作簿事件。

示例：在单元格快捷菜单中添加自定义列表

有时候，我们可以创建自定义列表，方便在单元格中输入一系列数据。例如，在自定义序列中，我们定义了一个序列：洪七公、小龙女、穆念慈、江小鱼、花无缺，如图 15.1 所示。

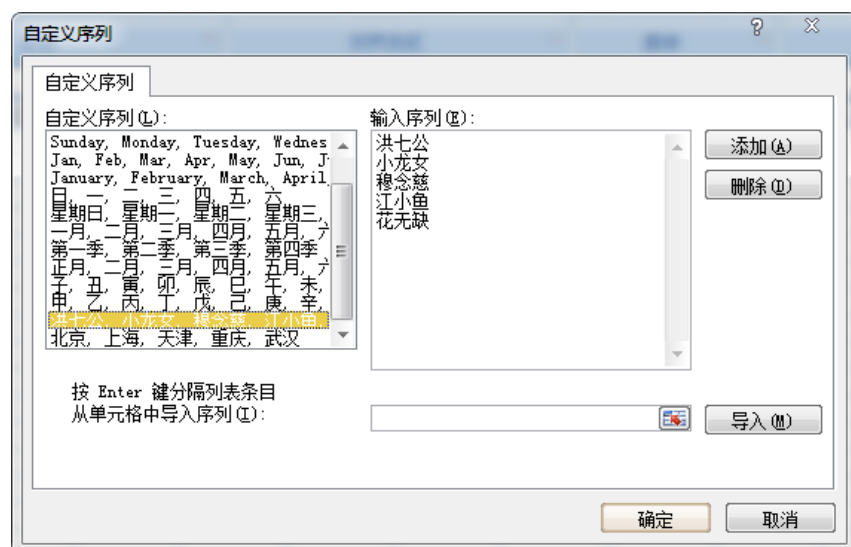


图 15.1



此时，我们可以在任意工作表单元格中输入“洪七公”，然后拖动该单元格右下角的填充句柄向下拖动到其他单元格，Excel 会根据自定义的序列在下面的单元格中自动输入“小龙女、穆念慈、...”。也就是说，我们只需输入序列中的第一个数据，余下的就不用管了！

然而，自定义序列多了或者时间长了，可能不记得序列第一个数据。我们可以将自定义的序列放置在单元格右键菜单中并显示序列第一个数据（如图 15.2 所示），这样只需单击右键，在快捷菜单中选择要输入的序列，然后向下拖动即可完成整个序列的自动输入。



图 15.2



下面是实现代码。

首先，在标准模块中输入代码：

```
Sub AddFirstList()  
    Dim strList As String  
    strList =  
Application.CommandBars.ActionControl.Caption  
    If Not strList Like "*...*" Then Exit Sub  
    ActiveCell = Left(strList, InStr(1, strList, ".",  
vbTextCompare) - 1)  
End Sub
```

在 ThisWorkbook 代码模块中，Workbook_SheetBeforeRightClick 事件的代码：

```
Private Sub Workbook_SheetBeforeRightClick(ByVal Sh As  
Object, ByVal Target As Range, Cancel As Boolean)  
    Dim cmbBtn As CommandBarButton  
    Dim lngListCount As Long  
    Dim lngCount As Long  
    Dim strList As String  
    Dim MyList  
  
    On Error Resume Next  
  
    With Application  
        lngListCount = .CustomListCount  
        For lngCount = 1 To lngListCount  
            MyList = .GetCustomListContents(lngCount)  
            strList  
= .CommandBars("Cell").Controls(MyList(1) & _  
                "... " & MyList(UBound(MyList))).Caption  
            .CommandBars("Cell").Controls(strList).Delete  
            Set cmbBtn  
= .CommandBars("Cell").Controls.Add(Temporary:=True)
```



```

        With cmbBtn
            .Caption = MyList(1) & "... " &
MyList (UBound (MyList))

            .Style = msoButtonCaption

            .OnAction = "AddFirstList"

        End With

    Next lngCount

End With

On Error GoTo 0

End Sub

```

代码输入完成后，回到工作表中，结果演示如图 15.3 所示。

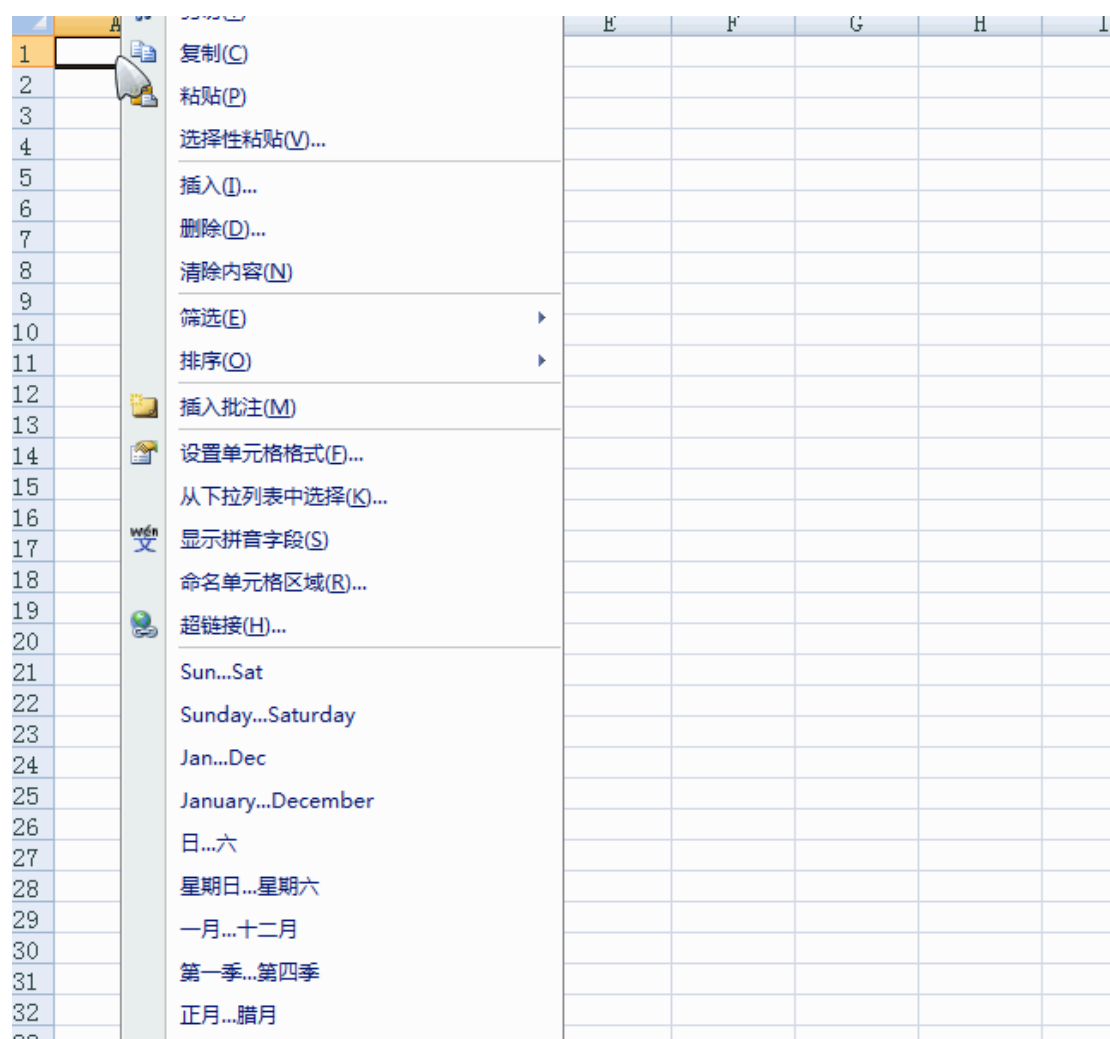


图 15.3 (演示视频截图，详细视频见完美 Excel 微信公众号)



代码运用了 SheetBeforeRightClick 事件，其语法为：

```
Workbook_SheetBeforeRightClick(ByVal Sh As Object, ByVal  
Target As Range, Cancel As Boolean)
```

说明：

- 当在工作簿的任意工作表中单击右键时发生此事件，并且事件发生在默认的右击操作之前。
- 参数 Sh，代表工作表对象。
- 参数 Target，代表单元格对象，即最接近右击时光标所在位置的单元格。
- 参数 Cancel，当事件发生时为 False。如果将该参数设置为 True，则不会执行默认的右击操作。

下面的代码禁止默认的右键单击操作：

```
Private Sub Workbook_SheetBeforeRightClick(ByVal Sh As  
Object, _  
ByVal Target As Range, ByVal Cancel As Boolean)  
    Cancel = True  
End Sub
```





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

[Excel VBA 解读 \(98\): 工作簿事件示例——](#)

[在单元格快捷菜单中添加自定义列表](#)

16. 工作簿事件示例：强制用户必须在指定 单元格中输入数据

在第 10 章：[Workbook 对象的 Open 事件和 BeforeClose 事件](#)中，我们详细介绍了 [Workbook_BeforeClose 事件](#)。该事件在关闭工作簿之前发生。

下面的示例使用 [Workbook_BeforeClose 事件](#)强制用户必须在指定的单元格中输入数据，否则就不能关闭该工作簿。如果用户想要关闭工作簿但没有在指定的所有单元格中都输入数据，那么 Excel 会弹出提示信息，列出还没有输入数据的单元格，并将这些单元格的背景设置为黄色。

图 16.1 演示了事件的运行效果。

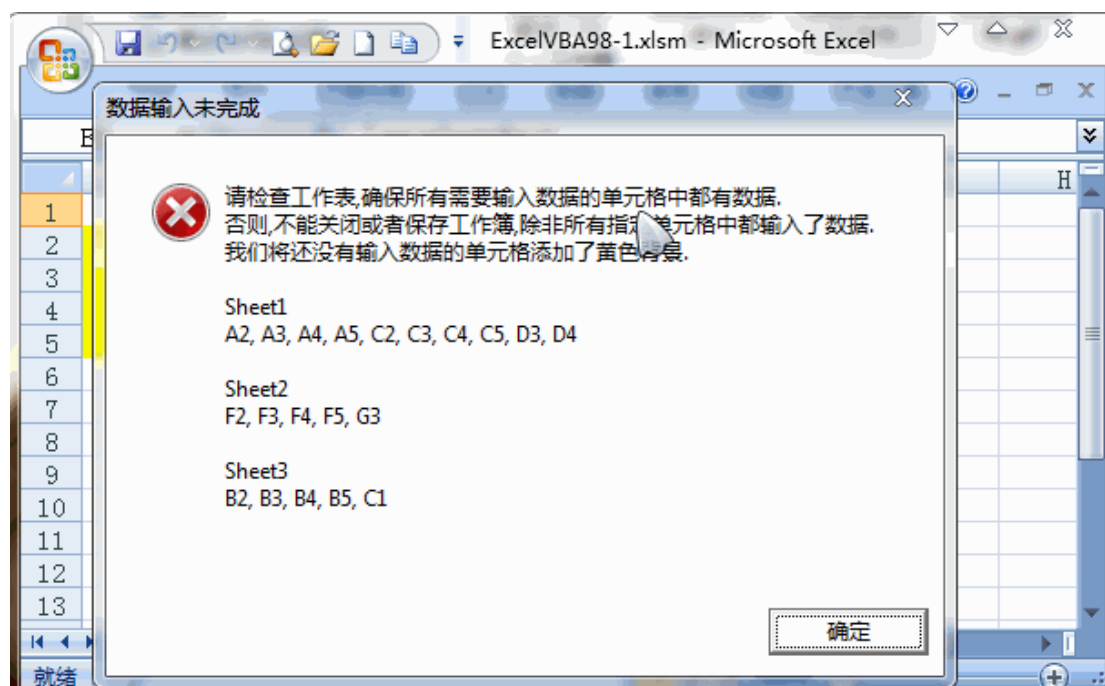


图 16.1 (演示视频截图，详细视频见[完美 Excel](#) 微信公众号)



下面是 ThisWorkbook 模块中的代码：

```
Dim strRng As String

Private Sub Workbook_BeforeClose(Cancel As Boolean)
    Dim strPrompt As String
    Dim rng1 As Range, rng2 As Range, rng3 As Range

    ' 设置必须要输入数据的单元格

    Set rng1 =
Sheets("Sheet1").Range("A2:A5,B1,C2:C5,D3:D5")
    Set rng2 = Sheets("Sheet2").Range("E1,F2:F5,G3")
    Set rng3 = Sheets("Sheet3").Range("A1,B2:B5,C1")

    strRng = ""

    strPrompt = "请检查工作表,确保所有需要输入数据的单元格中都有数"
据." & vbCrLf & _
        "否则,不能关闭或者保存工作簿,除非所有指定单元格中都输"
入了数据." & vbCrLf & _
        "我们将还没有输入数据的单元格添加了黄色背景."

    ' 调用子过程,检查单元格

    CheckCells rng1
    CheckCells rng2
    CheckCells rng3

    ' 如果存在必须输入数据的单元格没有数据,则提示用户
    ' 并且不能关闭工作簿

    If strRng <> "" Then
        MsgBox strPrompt & vbCrLf & vbCrLf & strRng,
vbCritical, "数据输入未完成"
        Cancel = True
    Else
        ThisWorkbook.Save
    End If
End Sub
```



```

        Cancel = False
    End If

    Set rng1 = Nothing
    Set rng2 = Nothing
    Set rng3 = Nothing
End Sub

'检查指定区域的单元格中是否有数据
'如果没有则添加黄色背景色
Sub CheckCells(rng As Range)
    Dim cell As Range
    Dim blnStart As Boolean

    blnStart = True

    For Each cell In rng
        If cell.Value = vbNullString Then
            cell.Interior.ColorIndex = 6 '黄色
            If blnStart Then strRng = strRng &
cell.Parent.Name & vbCrLf
            blnStart = False
            strRng = strRng & cell.Address(False, False) &
", "
        Else
            cell.Interior.ColorIndex = 0 '无颜色
        End If
    Next cell

    If strRng <> "" Then strRng = Left$(strRng,
Len(strRng) - 2) & vbCrLf & vbCrLf

End Sub

```



```
Private Sub Workbook_SheetChange(ByVal Sh As Object,  
ByVal Target As Range)  
    Target.Interior.ColorIndex = 0  
End Sub
```

在上面的代码中，还使用了 Workbook_SheetChange 事件，其语法为：

```
Workbook_SheetChange(ByVal Sh As Object, ByVal Target As  
Range)
```

说明：

- 当工作簿的任意工作表中的单元格由于用户操作或外部链接产生变化时发生此事件。
- 参数 Sh，代表工作表对象。
- 参数 Target，代表单元格对象，即发生变化的单元格。



本章内容 2018 年 5 月 28 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读 \(99\): 工作簿事件示例——
强制用户必须启用宏](#)

17.工作簿事件示例：强制用户必须启用宏

这是完美 Excel 微信公众号的一篇文章《问与答 10：如何强制用户启用宏？》中介绍过的内容，引用到这里作为讲解工作簿事件的一个典型示例，以加深对工作簿事件的理解。

有时候，我们使用 VBA 为工作簿编写了一些功能，但是如果用户在打开工作簿时不启用宏设置或者彻底禁用宏，那么这些功能就无法使用。例如，我使用 VBA 为工作簿添加了自定义菜单，但是如果用户禁用宏，那么自定义菜单就无法使用。

如何强制用户在使用工作簿时启用宏呢？

使用 VBA 代码来实现。在打开工作簿时，Excel 提示用户必须启用宏，否则工作簿中数据工作表均不可见。

首先，在工作簿中新建一个如下图 17.1 所示的工作表，并命名为<启用宏>，作为用户禁用宏时的特别提示。

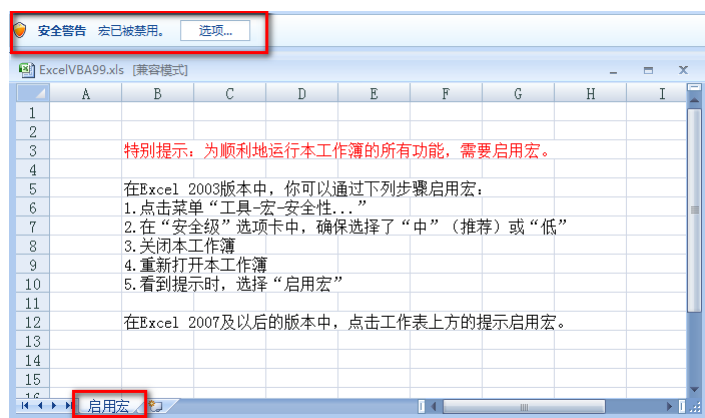


图 17.1



接着，打开 VBE 编辑器，在标准模块中输入下面的代码：

```
Sub AskUserEnabledMacros()  
  
    Dim wksInfoSheet As Worksheet  
  
    Dim objSheet As Object  
  
  
    On Error Resume Next  
  
    '引用<启用宏>工作表并判断其是否存在  
  
    Set wksInfoSheet = ThisWorkbook.Worksheets("启用宏")  
  
    If wksInfoSheet Is Nothing Then  
        MsgBox "不能够找到<启用宏>工作表", vbCritical  
        Exit Sub  
    End If  
  
    '关闭屏幕更新  
  
    Application.ScreenUpdating = False  
  
    '遍历工作簿中的所有工作表并设置所有工作表可见  
  
    For Each objSheet In ThisWorkbook.Sheets  
        objSheet.Visible = xlSheetVisible  
    Next objSheet  
  
    '隐藏<启用宏>工作表  
  
    wksInfoSheet.Visible = xlSheetVeryHidden  
  
    '隐藏想要隐藏的工作表,例如<sheet3>工作表  
  
    ThisWorkbook.Worksheets("Sheet3").Visible = xlSheetVeryHidden  
  
    '保存工作簿  
  
    ThisWorkbook.Saved = True
```



'恢复屏幕更新

Application.ScreenUpdating = True

End Sub

'隐藏除<启用宏>工作表之外的所有工作表

Sub RunOnClose()

Dim wksInfoSheet As Worksheet

Dim objSheet As Object

On Error Resume Next

'引用<启用宏>工作表并判断其是否存在

Set wksInfoSheet = ThisWorkbook.Worksheets("启用宏")

If wksInfoSheet Is Nothing Then

MsgBox "不能够找到<启用宏>工作表", vbCritical

Exit Sub

End If

'关闭屏幕更新

Application.ScreenUpdating = False

'显示<启用宏>工作表

wksInfoSheet.Visible = xlSheetVisible

'隐藏其他工作表

For Each objSheet In ThisWorkbook.Sheets

If Not objSheet Is wksInfoSheet Then

objSheet.Visible = xlSheetVeryHidden

End If

Next objSheet



```
'保存工作簿  
ThisWorkbook.Save  
End Sub
```

在 ThisWorkbook 对象模块中，输入下面的代码：

```
Private Sub Workbook_Open()  
    '当工作簿打开时运行 AskUserEnabledMacros 过程  
    AskUserEnabledMacros  
End Sub  
  
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    '隐藏除<启用宏>工作表之外的所有工作表  
    RunOnClose  
End Sub
```

当打开工作簿时，如果用户禁用了宏，那么将只显示<启用宏>工作表，提示用户要启用宏才能运用工作簿的所有功能，而其他工作表都被隐藏。并且，Excel 会在上方显示“安全警告：宏已被禁用”，如图 17.1 所示。如果单击其右侧的“选项”并启用宏，那么会隐藏<启用宏>工作表并使其他工作表可见。

如果用户启用了宏，那么在打开工作簿时，触发 [Workbook_Open 事件](#)，执行其中的代码，即调用 AskUserEnabledMacros 过程，隐藏<启用宏>工作表及用户不想让他人看到的工作表，其他工作表可见。在关闭工作簿时，触发 [Workbook_BeforeClose 事件](#)，执行其中的代码，即调用 RunOnClose 过程，隐藏除<启用宏>工作表之外的其他所有工作表。这样，再次打开该工作簿时，如果用户禁用宏，那么工作簿中就会只出现这个工作表。



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 解读中关于 Workbook 对象应用技术的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。

