

Excel VBA 技术系列

每日Excel
帮你提升技能

那些富含营养的文章



Excel VBA 解读 • Worksheet 对象篇

完美 Excel 出品
微信公众号: *excelperfect*



我们需要在 Excel 工作表中对数据进行处理和分析。工作表是使用 Excel 的入口，我们在工作表中进行有价值的工作。

本电子书《Excel VBA 解读 Worksheet 对象篇》讲解了 Worksheet 对象的常用属性、方法和事件。

坚持分享最好的Excel与VBA技术知识

微信公众号: *excelperfect*

知嗒知识号: *完美Excel*



目录

主要内容.....	1
1. 开始探索 Worksheet 对象.....	1
2. Worksheets 与 Sheets.....	3
示例：列出工作簿中的工作表数和图表工作表数.....	4
3. 工作表名称——Name 属性.....	7
理解工作表名称.....	7
Name 属性.....	8
示例 1：获取当前工作簿所有工作表的名称	9
示例 2：以当天日期命名当前工作表名称	10
示例 3：设置工作表标签颜色	11
示例 4：检查工作表是否已存在	11
CodeName 属性.....	13
示例 5：更改工作表对象名称	13
示例 6：检查工作表对象名称是否已存在	15
4. 在代码中引用工作表	17
3 种引用工作表方法的优缺点	19
重大发现（~~）	19
示例 1：将数据从一个工作表复制到另一个工作表	19
示例 2：依次列出工作簿中的工作表名	20
5. Activate 方法与 Select 方法	21
Activate 方法与 Select 方法的区别.....	24
6. 隐藏工作表——Visible 属性.....	25

示例 1: 统计工作簿中可见工作表数量	27
示例 2: 隐藏指定的工作表	29
示例 3: 使用工作簿中的工作表全部可见	29
7. 添加和删除工作表——Add 方法和 Delete 方法	31
Add 方法	32
示例 1: 添加并命名新工作表	33
Delete 方法	34
示例 2: 删除除指定工作表名外的所有工作表	34
示例 3: 安全地删除工作表	35
8. 移动或复制工作表——Move 方法和 Copy 方法	39
Copy 方法	41
Move 方法	42
示例 1: 将工作表移动到最后	42
9. 保护工作表——Protect 方法	43
Protect 方法	44
示例 1: 使用 Protect 方法保护工作表	44
Unprotect 方法	45
示例 2: 使用 Unprotect 方法取消工作表保护	45
示例 3: 设置与取消工作表保护的通用代码	46
10. 打印工作表——PrintOut 方法	49
示例 1: 实现逐行批量打印	50
示例 2: 只打印奇数页	51
11. Evaluate 方法	52
示例 1: 输入值并设置单元格格式	53
示例 2: 获取单元格的值	54
示例 3: 获取另一工作簿中单元格的值	55
12. 设置工作表页面——PageSetup 属性	56
示例: 设置当前工作表页面	58
13. 让 Excel 动起来——认识事件过程	60
14. 看看工作表会自动响应哪些操作——认识工作表事件	65

Activate 事件	66
Deactivate 事件	66
示例 1: 激活当前工作表与转移到其他工作表时的事件响应	66
BeforeDoubleClick 事件	67
示例 2: 双击单元格自动添加背景色	67
BeforeRightClick 事件	68
示例 3: 阻止显示缺省的快捷菜单	69
Calculate 事件	69
示例 4: 根据计算结果输入数值并设置格式	70
Change 事件	71
示例 5: 提示用户不要修改数据	71
FollowHyperlink 事件	72
PivotTableUpdate 事件	72
SelectionChange 事件	72
示例 6: 高亮显示单元格所在的行列	72
15. 工作表事件示例	75
示例 1: 阻止用户修改指定单元格区域的数据	75
示例 2: 让不同的工作表有不同的快捷菜单	76
示例 3: 双击单元格时显示输入框	78
示例 4: 根据比较结果自动显示上升或下降的箭头	80
关于完美 Excel	82
完美 Excel 微信公众号使用指南	83

主要内容

工作表，是使用 Excel 的入口。应该说，我们做的有价值的工作，都是在工作表中进行的。我们可能要知道工作表的名字，可能要为特定的工作表取一个特定的名字，可能要使用特定的工作表，可能要添加或者删除、移动或者复制工作表，可能要保护工作表不受用户的影响，...等等。这些都是我们经常对工作表进行的一些操作。

工作表还能自动响应用户的一些操作，例如激活工作表时、在工作表单元格中移动或者单元格内容发生改变时，这称之为工作表事件。可以利用工作表事件来“监视”对工作表所进行的操作。

本书详细讲解了 Worksheet 对象的主要属性、方法和事件。总体来说，Worksheet 对象的属性和方法、事件并不是太多，因此多加练习，应该能熟练掌握它们。

1.开始探索 Worksheet 对象

简要介绍 Worksheet 对象的常用属性、方法。

2.Worksheets 与 Sheets

详细讲解两个与工作表相关的相似的集合对象：Worksheets 集合和 Sheets 集合，区别及示例。示例：[列出工作簿中的工作表数和图表工作表数](#)。

3.工作表名称——Name 属性

讲解了工作表名称和工作表对象名称是什么，分别对应的 Name 属性和 CodeName 属性的用法。6 个示例：[①获取当前工作簿所有工作表的名称](#)；[②以当天日期命名当前工作表名称](#)；[③设置工作表标签颜色](#)；[④删除筛选出的数据](#)

(Tab 属性); ⑤更改工作表对象名称; ⑥检查工作表对象名称是否已存在。

4.在代码中引用工作表

在代码中引用工作表的 3 种方法: 使用工作表名称、使用索引值、使用工作表对象名称, 及优缺点。2 个示例: ①将数据从一个工作表复制到另一个工作表; ②依次列出工作簿中的工作表名。

5.Activate 方法与 Select 方法

详细讲解激活工作表的 2 个方法及区别。

6.隐藏工作表——Visible 属性

详细讲解了用来设置或获取工作表隐藏的 Visible 属性。3 个示例: ①统计工作簿中可见工作表数量; ②隐藏指定的工作表; ③使工作簿中的工作表全部可见。

7.添加和删除工作表——Add 方法和 Delete 方法

详细讲解了用于添加工作表的 Add 方法和用于删除工作表的 Delete 方法的语法及示例。3 个示例: ①添加并命名新工作表; ②删除指定工作表名外的所有工作表; ③安全地删除工作表。

8.移动或复制工作表——Move 方法和 Copy 方法

详细讲解了移动工作表的 Move 方法和复制工作表的 Copy 方法的语法、区别及示例。示例: 将工作表移动到最后。

9.保护工作表——Protect 方法

详细讲解了如何使用 VBA 实现保护工作表的 Protect 方法, 以及取消工作表保护的 Unprotect 方法。3 个示例: ①使用 Protect 方法保护工作表; ②使用 Unprotect 方法取消工作表保护; ③设置与取消工作表保护的通用代码。

10.打印工作表——PrintOut 方法

详细讲解了 VBA 中用于打印工作表的 `PrintOut` 方法的语法及说明。2 个示例：
①实现逐行批量打印；②只打印奇数页。

11.Evaluate 方法

简单地介绍工作表对象的 `Evaluate` 方法的语法及示例。3 个示例：①输入值并设置单元格格式；②获取单元格的值；③获取另一工作簿中单元格的值。

12.设置工作表页面——PageSetup 属性

详细讲解了返回设置工作表页面对象的 `PageSetup` 属性的语法及与“页面设置”对话框内容对应的属性。示例：设置当前工作表页面。

13.让 Excel 动起来——认识事件过程

讲解 Excel 事件的基本知识，列出了 Excel 中常使用的事件类型。

14.看看工作表会自动响应哪些操作——认识工作表事件

详细讲解工作表 `Worksheet` 对象的 9 个事件及触发的条件和示例。①激活当前工作表与转移到其他工作表时的事件响应；②双击单元格自动添加背景色；③阻止显示缺省的快捷菜单；④根据计算结果输入数值并设置格式；⑤提示用户不要修改数据；⑥高亮显示单元格所在的行列。

15.工作表事件示例

列举了一些工作表事件示例，让读者加深对工作表事件应用的理解。4 个示例：
①阻用户修改指定单元格区域的数据；②让不同的工作表有不同的快捷菜单；
③双击单元格时显示输入框；④根据比较结果自动显示上升或下降的箭头。

在完美 Excel 微信公众号中发送消息：工作表对象，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。

1. 开始探索 Worksheet 对象

我们在用的很多东西，习以为常，并没有意识到他们在为我所用。有些东西，默默地为我们服务，我们却认为理所当然，自然接受。然而，当有人给你指点一下时，或者有一天突然醒悟时，才会感受到他们的伟大。

在我眼里，Excel 工作表就是这样！

我们在工作表的单元格中输入、整理、分析数据，我们在不同的工作表间“穿梭”，一切似乎很自然！

工作表就是那个为我们默默的服务者之一。

那么，好吧！就让我们从现在开始，来深入探索 **Worksheet 对象**。

Worksheet 对象本应该是使用最频繁的对象，因为我们操作的单元格（**Range 对象**）都要涉及到某个 **Worksheet 对象**，只不过很多情形下我们将其省略罢了，就像我们这个系列之前所有文章中绝大部分示例一样，我们都是假设在当前工作表中在进行我们的操作。

在《**Excel VBA 解读基础入门篇**》的第 7 章.看看 Excel 的那些常用对象（续 2）中，我们初步接触了 **Worksheet 对象**，知道了 **Worksheet 对象**代表工作表，所有的 **Worksheet 对象**构成 **Worksheets 集合**，可以使用工作表名、工作表索引值、工作表对象名称在代码中引用工作表。另外，**ActiveSheet 属性**表示当前工作表。

Worksheet 对象的 **Name 属性**可以获取或者修改工作表名称，**CodeName 属性**可以获取工作表对象名称。**Index 属性**返回工作表的索引值。通过设置 **Visible 属性**，可以让工作表隐藏起来。

在《**Excel VBA 解读 Range 对象篇**》中已经介绍过的 **UsedRange 属性**也属于

Worksheet 对象，代表工作表中已使用的单元格区域。

Worksheet 对象的 **Activate 方法**将使指定的工作表成为活动工作表，**Delete 方法**用于删除工作表，**Copy 方法**用于复制工作表的一份副本到指定的工作表之前或之后的位置，**Move 方法**将移动工作表到指定的工作表之前或之后的位置。

Paste 方法将剪贴板中的内容粘贴到工作表中指定的单元格，**PasteSpecial 方法**使用指定的格式将剪贴板中的内容粘贴到工作表。

Protect 方法用于保护工作表，避免工作表被无故修改。**Unprotect 方法**取消工作表保护。

SaveAs 方法用于保存工作表。

PrintOut 方法打印工作表中的全部或者部分内容，**PrintPreview 方法**用于打印前的预览。

Worksheets 对象的 **Count 属性**将返回工作簿中工作表数。

Worksheets 对象的 **Add 方法**创建新的工作表，**FillAcrossSheets 方法**将复制某个单元格区域到指定的所有工作表的同一区域中。

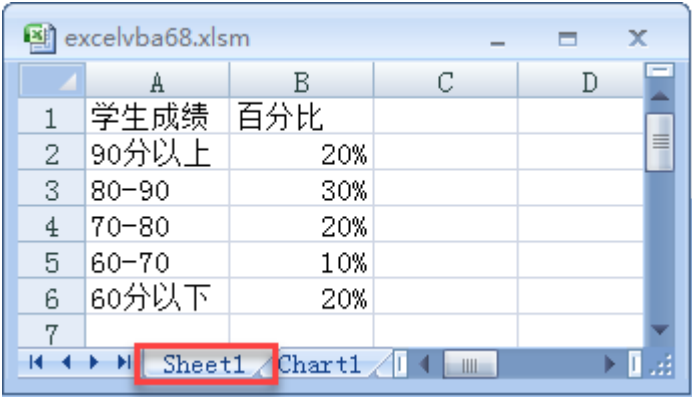
此外，与 **Range 对象**不同，可以使用工作表事件来响应一些操作，使我们的操作更加自动化，这也是我们接下来要详细讲解的内容。

在接下来的内容中，我们将详细介绍与 **Worksheet 对象**（**Worksheets 对象**）相关的常用属性、方法和事件，也会看到 **Range 对象**中与工作表相关的属性和方法的介绍示例。

2. Worksheets 与 Sheets

在往下面介绍之前，让我们先看看两个相似的集合对象：Worksheets 集合与 Sheets 集合。

通常，我们所指的工作表是这样的：



The image shows an Excel spreadsheet window titled 'excelvba68.xlsm'. The spreadsheet has columns A, B, C, and D, and rows 1 through 7. The data is as follows:

	A	B	C	D
1	学生成绩	百分比		
2	90分以上	20%		
3	80-90	30%		
4	70-80	20%		
5	60-70	10%		
6	60分以下	20%		
7				

The 'Sheet1' tab in the bottom-left corner is highlighted with a red box.

图 2.1

但是，还有这样的工作表：

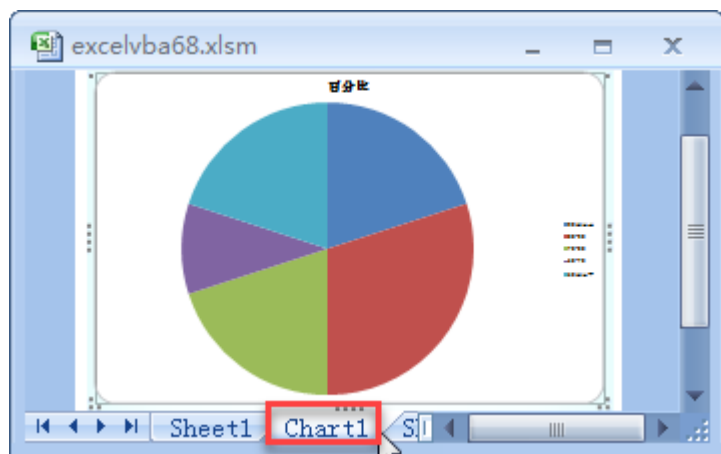


图 2.2

也就是图表工作表。

Worksheets 集合包含工作簿中所有的 Worksheet 对象代表的工作表，即图 2.1 所示的工作表(每个这样的工作表即一个 Worksheet 对象)。然而，Sheets 集合不仅包含工作簿中所有的 Worksheet 对象，而且也包含 Chart 对象，即图 2.2 所示的图表工作表。(其实，还包括以前使用的对话框工作表、宏表，如下图 2.3 给出了这 4 种类型的工作表图例)

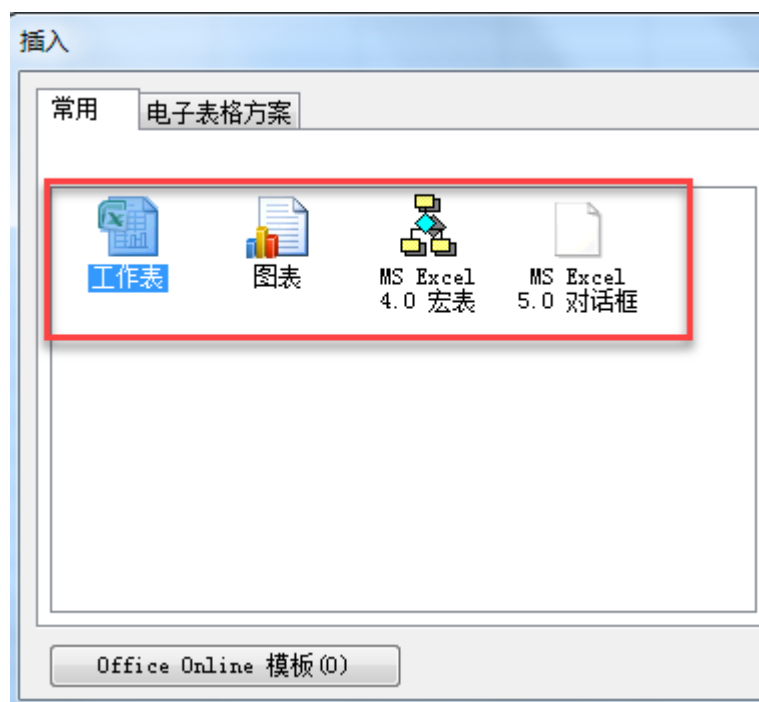


图 2.3

(注：本书中，我们所说的工作表即为图 2.1 所示的工作表，而图表工作表即为图 2.2 所示的工作表)

示例：列出工作簿中的工作表数和图表工作表数

如下图 2.4 所示的工作表：

	A	B	C	D	E
1	学生成绩	百分比			
2	90分以上	20%			
3	80-90	30%			
4	70-80	20%			
5	60-70	10%			
6	60分以下	20%			
7					
8					

图 2.4

下面的代码列出了总的工作表数和工作表数（有点拗口，慢慢理解哈）：

```
Sub SheetsCount ()
    MsgBox "本工作簿中工作表（包括图表工作表和工作表）的总数为：" & _
    Sheets.Count & vbCrLf & _
    "本工作簿中工作表的个数为：" & _
    Worksheets.Count
End Sub
```

说明：

- Count 属性统计工作表数量，且分别用于 Sheets 集合和 Worksheets 集合，结果也会不同。
- 如果工作簿中没有图表工作表或其它类型的工作表，那么 Worksheets 集合等同于 Sheets 集合。
- 运行代码后的结果如图 2.5 所示。

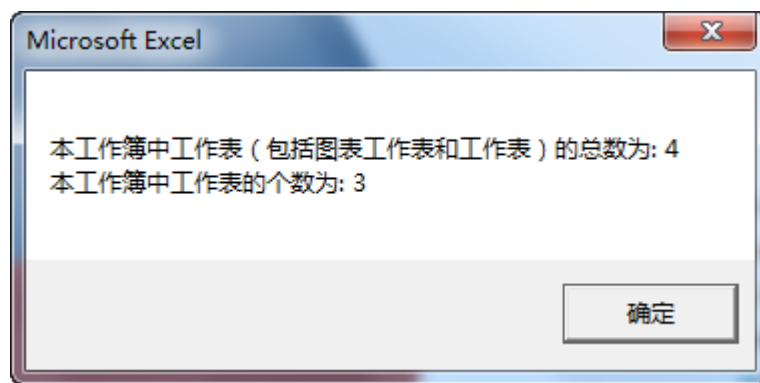


图 2.5

本章内容 2017 年 7 月 21 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读\(69\): 工作表名称——Name 属性](#)

3. 工作表名称——Name 属性

工作表名称让我们能够方便地识别工作表,可以给工作表取一个具有特定意义的名称。在 VBA 代码中,还可以使用工作表对象名称。

理解工作表名称

在下图 3.1 中,我们示意出了工作表名称和工作表对象名称。(仔细研读过[完美 Excel](#) 微信公众号的 Excel VBA 解读系列文章的朋友肯定对这张图有印象,它就是我们在《[Excel VBA 解读 \(7\): 看看 Excel 的那些常用对象 \(续 1\)](#)》中介绍 Worksheet 对象时使用的图)

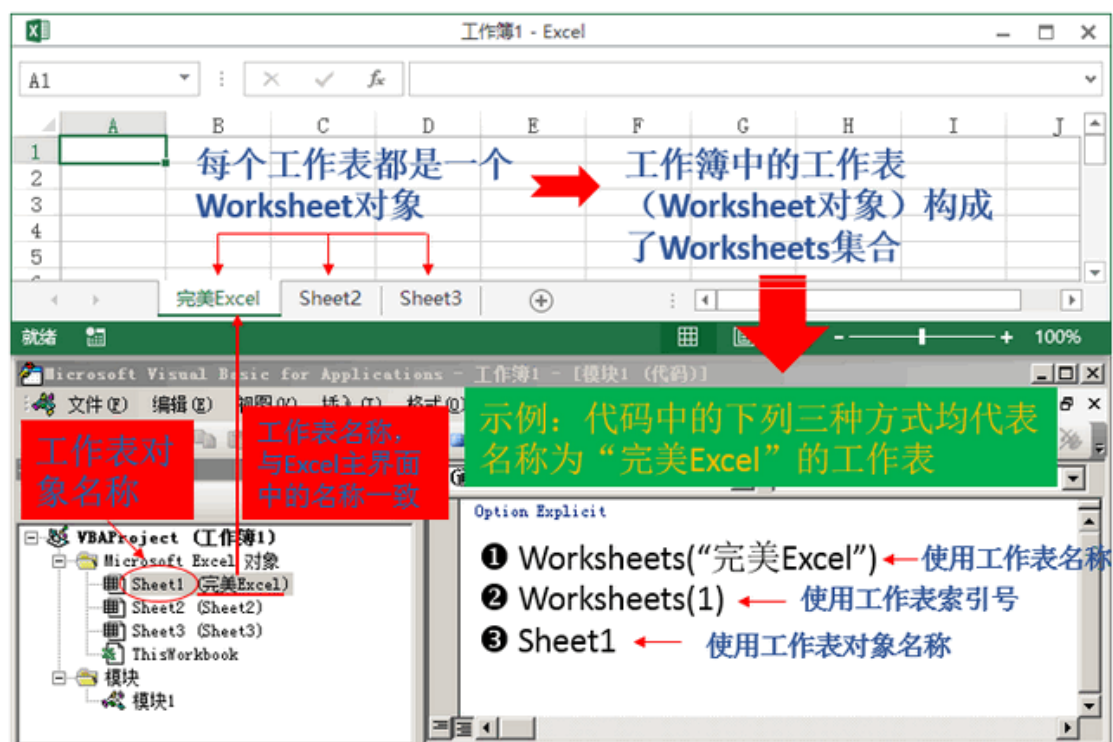


图 3.1

工作表名称就是我们在工作表界面下方看到的工作表标签，可以在工作表标签中单击右键，在弹出的菜单中选择“重命名”来修改工作表名称，也可以在工作表标签中双击来修改工作表名称。在 VBE 界面中，我们可以在其左侧的工程资源管理器中看到在括号中的工作表名称，如图 3.1 中红色横线所示，在工作表界面中修改工作表名称后，该处也随之改变为一致，如图中的“完美 Excel”。

工作表对象名称在 VBE 界面工程资源管理器工作表名称左侧，如上图 3.1 中椭圆圈所示。可以选择工作表对象后，在属性窗口修改工作表对象名称，例如下图 3.2 所示，将“Sheet1”修改为“excelperfect”。

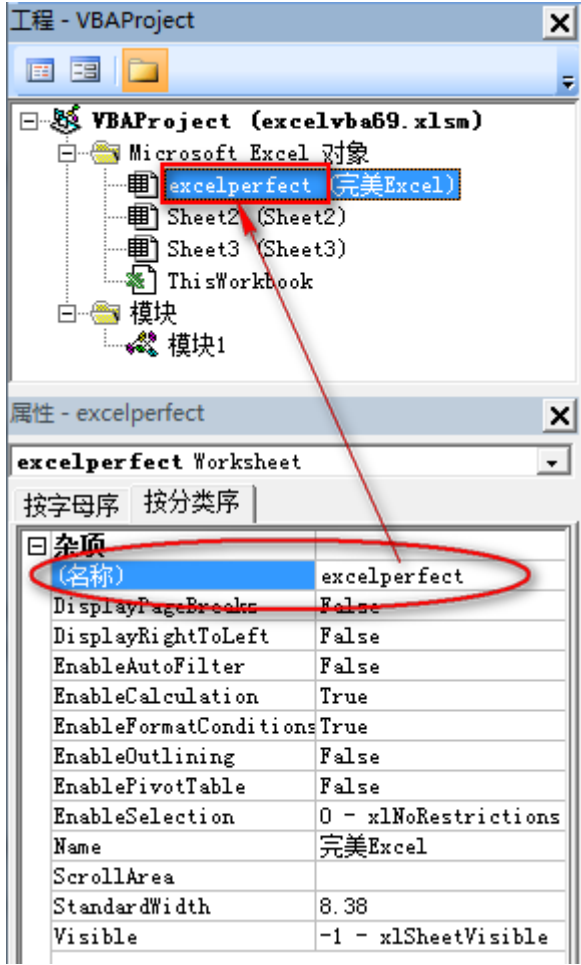


图 3.2

本文将介绍这两种代表工作表的名称在 VBA 中的表示及应用。

Name 属性

在 VBA 中，使用 Name 属性获取或者设置工作表名称，即工作簿底部工作表标

签显示的字符串 Sheet1、Sheet2、Sheet3 等，如下图 3.3 所示。

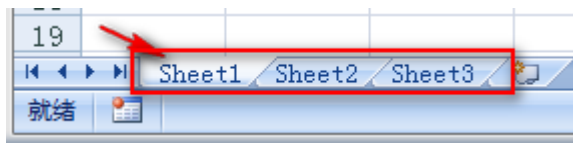


图 3.3

下面的语句：

```
Worksheets("Sheet1").Name = "完美 Excel"
```

将工作表 Sheet1 的名字修改为“完美 Excel”，如下图 3.4 所示。

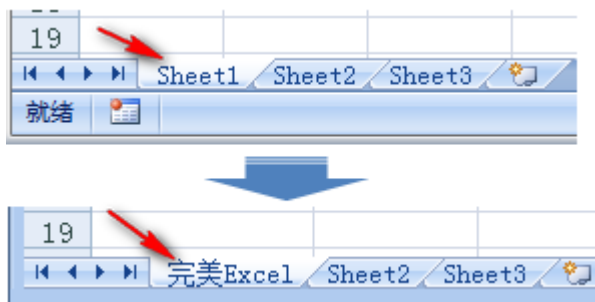


图 3.4

下面的语句获取当前工作表的名称：

```
ActiveSheet.Name
```

示例 1：获取当前工作簿所有工作表的名称

下面的代码获取并显示上文所示工作簿中所有工作表名称：

```
Sub GetWorksheetName()  
    Dim ws As Worksheet  
    Dim str As String  
  
    '遍历工作表并获取其名称  
    For Each ws In Worksheets  
        str = str & """" & ws.Name & """"  
    Next ws
```

```
MsgBox "当前工作簿中工作表名称分别为" & str  
End Sub
```

说明:

- Name 属性获取工作表名称。
- 语句"""" & ws.Name & """"中各 4 个双引号设置在显示时显示双引号。

运行代码后的效果如下图 3.5 所示:

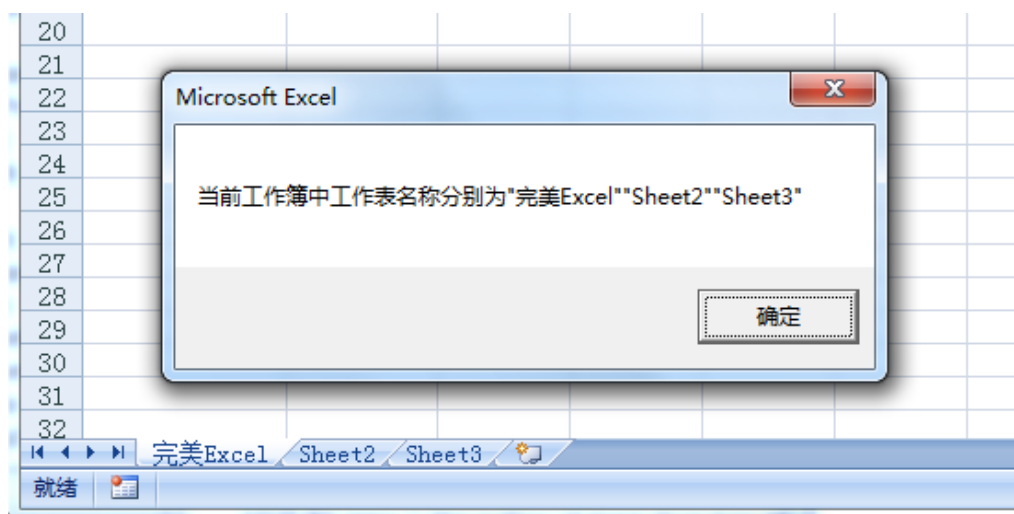


图 3.5

示例 2：以当天日期命名当前工作表名称

下面是 VBA 帮助文档中的示例代码，设置当前工作表名称为当天的日期。

```
Sub NameWorksheetByDate()  
    Range("D5").Select  
    '在单元格 D5 中输入公式, 获取今天的日期  
    Selection.Formula = "=text(now(), ""mmm ddd yyyy"")"  
    '复制文本并粘贴值  
    Selection.Copy  
    Selection.PasteSpecial Paste:=xlValues
```

```

'消除单元格周边虚框
Application.CutCopyMode = False

'以单元格 D5 中的值命名当前工作表
ActiveSheet.Name = Range("D5").Value

'清除单元格 D5 中的值
Range("D5").Value = ""

End Sub

```

说明:

- 代码首先在当前工作表单元格中使用函数获取当前的日期并将其转换成值，然后使用这个值命名当前工作表。
- 一些程序在执行过程中，先利用工作表临时存储数据，在使用完后将其清除，就好像什么都没发生一样。这是一个值得借鉴的技巧。

示例 3：设置工作表标签颜色

下面的代码设置工作表“完美 Excel”标签的颜色为绿色：

```

Sub SetWorksheetTabColor()
    Sheets("完美 Excel").Tab.Color = vbGreen
End Sub

```

说明:

- Tab 属性返回代表工作表标签的 Tab 对象。

运行代码后的效果如下图 3.6 所示：

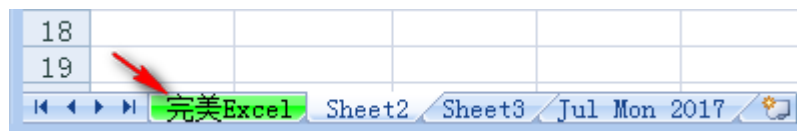


图 3.6

示例 4：检查工作表是否已存在

在代码中操作指定的工作表之前，我们可以先检查该工作表是否存在，以确保操

作的有效性。

代码如下：

```
Function WorksheetIsExists(strName As String) As Boolean
    Dim str As String

    On Error GoTo ErrHandle

    '获取变量 strName 表示的工作表名称
    '如果变量 strName 表示的名字的工作表存在,则将其名称赋给变量 str
    '否则,导致错误.跳转至 ErrHandle 语句
    str = Worksheets(strName).Name
    WorksheetIsExists = True
    Exit Function

ErrHandle:
    WorksheetIsExists = False
End Function
```

下面的代码使用上述自定义函数检查指定名称的工作表是否存在,然后进行操作:

```
Sub testWorksheetIsExists()
    Dim ws As Worksheet
    Dim str As String

    str = "Sheet3" '指定工作表名称

    '如果工作表存在,则将该工作表赋值给变量
    If WorksheetIsExists(str) Then
        Set ws = Worksheets(str)
    Else
        Set ws = Nothing
    End If

    '如果工作表存在,则在该工作表单元格 A1 中输入值
```

```
If Not ws Is Nothing Then
    ws.Range("A1").Value = "完美 Excel"
End If
End Sub
```

CodeName 属性

CodeName 属性返回工作表对象名称，例如语句：

```
Worksheets("Sheet3").CodeName
```

将获取工作表 Sheet3 的对象名称。

CodeName 属性为只读属性，不能使用该属性修改工作表对象名称。

一般情况下，Excel 默认工作表名称和工作表对象名称相同。这两个名称可以各自修改，但修改一个名称不会自动修改另一个名称。

示例 5：更改工作表对象名称

除了在属性窗口中手工修改工作表对象名称外，还可以使用代码来修改。此时，要使用 VBE 对象模型。

首先，在 Excel 中选择“开发工具”选项卡中的“宏安全性”，在“信任中心”的“宏设置”下选取“信任对 VBA 工程对象模型的访问”，如下图 3.7 所示。

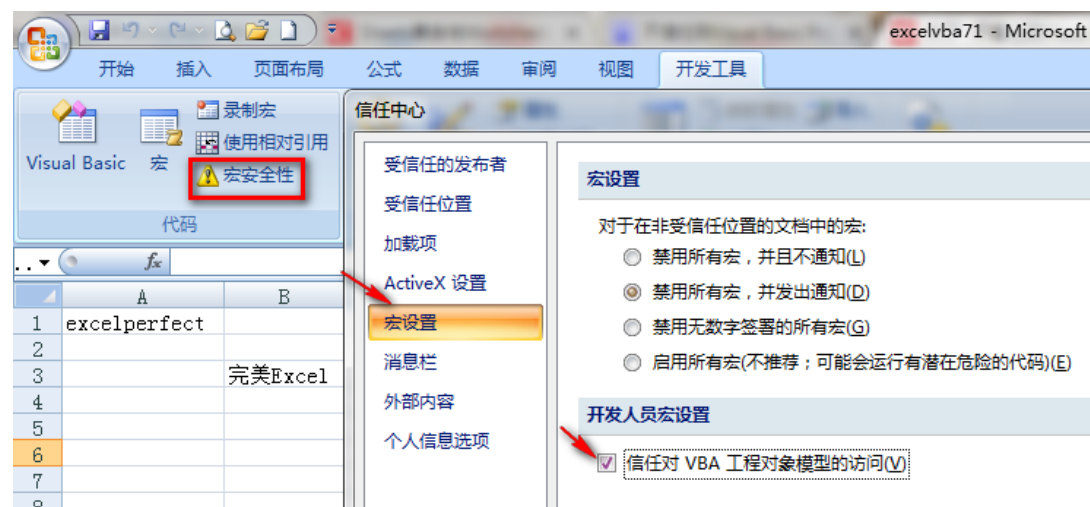


图 3.7

然后, 在 VBE 中, 选择“工具”——“引用”, 在“引用”对话框中, 选取“Microsoft Visual Basic for Applications Extensibility 5.3”, 单击“确定”, 如下图 3.8 所示。

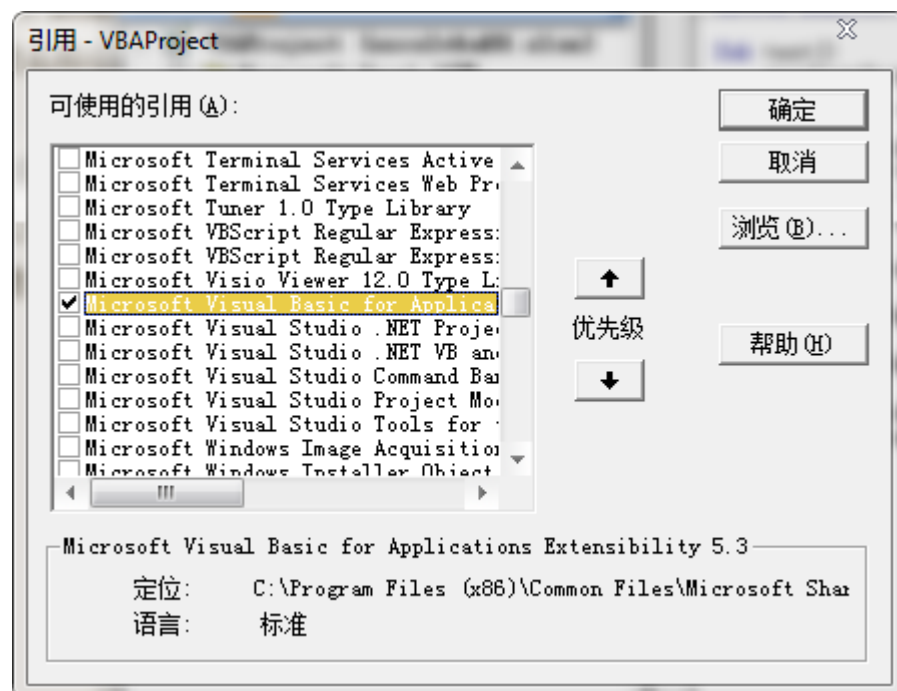


图 3.8

下面的代码将工作表“Sheet2”的对象名称修改为“完美 Excel”:

```
Sub ChangeWorksheetObjectName()  
    Dim ws As Worksheet  
    Dim strOldCodeName As String  
    Dim strNewCodeName As String  
  
    '将要修改名称的工作表赋值给变量  
    Set ws = Worksheets("Sheet2")  
  
    '获取现在的工作表对象名称  
    strOldCodeName = ws.CodeName  
  
    '设置新名称  
    strNewCodeName = "完美 Excel"
```


·修改工作表对象名称为新名称

```
ThisWorkbook.VBProject. _  
    VBComponents(strOldCodeName). _  
        Properties("_CodeName") = strNewCodeName  
End Sub
```

运行代码后的效果如下图 3.9 所示：

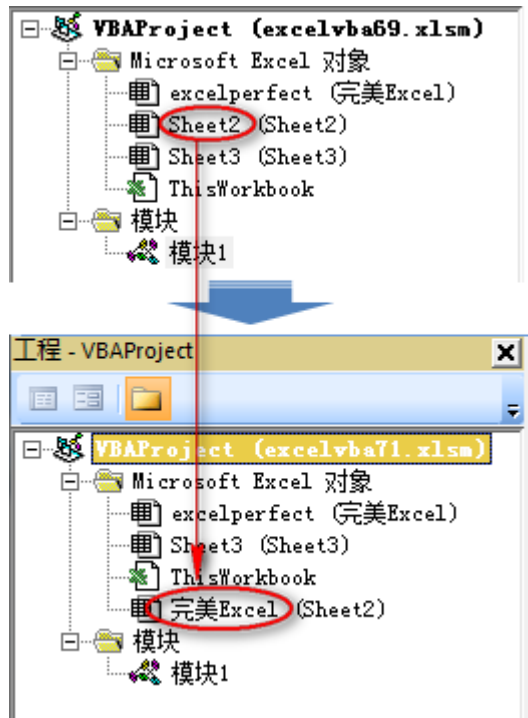


图 3.9

示例 6：检查工作表对象名称是否已存在

下面的代码检查指定的工作表对象名称是否存在：

```
Function WorksheetCodeNameIsExists(strCodeName As String) As Boolean  
  
    Dim ws As Worksheet  
  
    Dim str As String  
  
    WorksheetCodeNameIsExists = False
```

```
' 遍历工作表

For Each ws In Worksheets

    ' 比较指定的名称与已有工作表对象名称

    If StrComp(ws.CodeName, strCodeName, vbTextCompare) =
0 Then

        WorksheetCodeNameIsExists = True

    End If

Next ws

End Function
```

说明：

- StrComp 函数比较指定的字符串并返回代表比较结果的值，返回 0 表示两个字符串相同。

本章内容 2017 年 7 月 27 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读 (70): 在代码中引用工作表

4. 在代码中引用工作表

在 VBA 中，我们通常使用工作表名称或者索引值来引用工作表。如下图 4.1 所示的工作表：

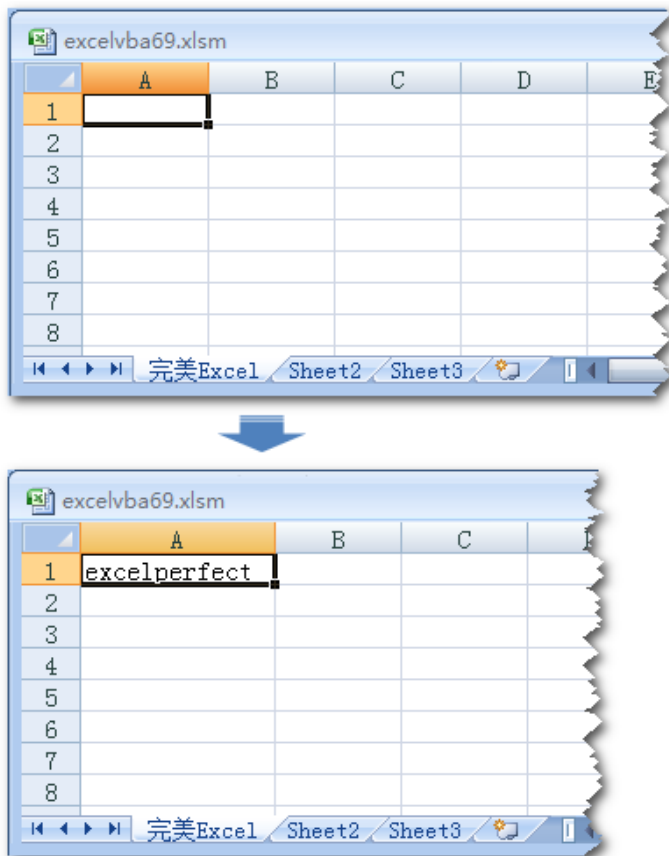


图 4.1

语句：

```
Worksheets("完美 Excel").Range("A1").Value = "excelperfect"
```

在“完美 Excel”工作表单元格 A1 中输入“excelperfect”。

在上图 4.1 所示的工作表中，我们使用语句：

```
Worksheets("Sheet2").Index
```

获取“Sheet2”工作表在工作表集合中的索引值为“2”。`Index` 属性返回指定工作表在工作表集合中的索引值，代表工作表在工作簿中的位置。Excel 会从左向右依次给工作表编号，图 1 中“Sheet2”工作表从左起排在第 2 个，其索引值为 2。

然后，使用索引值引用该工作表。例如，下面的语句：

```
Worksheets(2).Range("B2").Value = "完美 Excel"
```

在“Sheet2”工作表单元格 B2 中输入“完美 Excel”，运行代码后的结果如下图 4.2 所示。

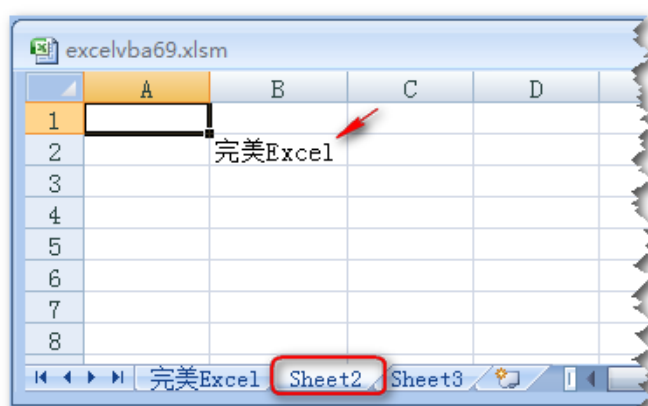


图 4.2

要引用工作表，除了上面介绍的使用工作表名称或者索引值外，还可以使用工作表对象名称。如果“完美 Excel”工作表的对象名称为“excelperfect”，那么语句：

```
excelperfect.Range("B3").Value = "完美 Excel"
```

在“完美 Excel”工作表单元格 B3 中输入“完美 Excel”，如下图 4.3 所示。

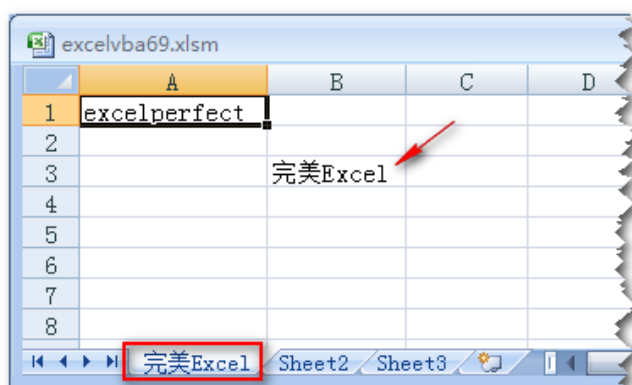


图 4.3

3 种引用工作表方法的优缺点

- 使用工作表名称引用工作表，很直观，但工作表名称容易被修改。如果用户修改了工作表名称，但代码没作相应的修改，那么运行代码就会出错。
- 使用索引值引用工作表，很简洁，但如果移动或者删除一些工作表，索引值就会相应变化，运行代码也会报错。
- 使用工作表对象名称引用工作表，不会受工作表名称或者工作表移动或删除的影响，是一种较为安全的方法。但是，想要将代码应用到其他工作簿中时，因为无法事先知道或者设置该工作簿的工作表代码名，也会存在问题。

3 种引用工作表的方法都有优缺点，使用工作表对象名称引用工作表的方法较为安全些，但无论使用哪种方法引用工作表，都要结合工作簿实际运用情况，选择合适的引用方法。

重大发现（~~）

不知道大家发现没有，在引用工作表后，无需激活工作表和单元格，也就是说，无需将工作表置为当前工作表，即可操作所引用的工作表。

示例 1：将数据从一个工作表复制到另一个工作表

下面分别使用工作表名称和工作表对象名称将一个工作表中的数据复制到另一个工作表。

如上图 4.3 所示的工作表，现在将“完美 Excel”工作表中的数据复制到工作表 Sheet3 中。

使用工作表名称复制数据

代码如下：

```
Sub CopyDataWithSheetName()  
    '将"完美 Excel"工作表单元格 A1:B3 中的数据复制到  
    'Sheet3 工作表以单元格 A1 开始的区域中  
    Worksheets("完美 Excel").Range("A1:B3").Copy _  
        Worksheets("Sheet3").Range("A1")  
End Sub
```

使用工作表对象名称复制数据

代码如下：

```
Sub CopyDataWithCodeName()  
    '将"完美 Excel"工作表单元格 A1:B3 中的数据复制到  
    'Sheet3 工作表以单元格 A1 开始的区域中  
    excelperfect.Range("A1:B3").Copy Sheet3.Range("A1")  
End Sub
```

示例 2：依次列出工作簿中的工作表名

下面的代码使用索引值，按工作表从左到右的顺序列出工作簿中的工作表名。

```
Sub SheetNameList()  
    Dim iIndex As Integer  
    Dim str As String  
  
    For iIndex = 1 To Worksheets.Count  
        str = str & Worksheets(iIndex).Name & vbCrLf  
    Next iIndex  
  
    MsgBox "本工作簿中的工作表名依次为：" & vbCrLf & str  
End Sub
```

说明：

- 代码中使用常量 `vbCrLf` 来对获得的结果换行。结果如下图 4.4 所示：

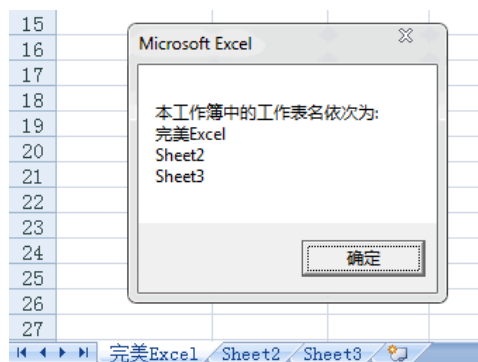


图 4.4

5. Activate 方法与 Select 方法

通常情况下，正如上一章所讲的，通过引用指定工作表后，我们无须激活该工作表即可对该工作表单元格进行操作。然而，在某些情形下，可能还是需要激活或者选择某工作表，例如，在进行一些复制后粘贴操作时。

下面，我们就来讲解 Worksheet 对象的 Activate 方法及相关的 Select 方法。

Activate 方法和 Select 方法都可以激活工作表，使其成为当前工作表。在 VBA 帮助文档中，Activate 方法属于 Worksheet 对象，而 Select 方法属于 Worksheets 集合对象和 Sheets 集合对象。

下面的代码：

```
Worksheets("Sheet2").Activate
```

使工作表 Sheet2 成为活动工作表，等价于在工作表界面中单击了 Sheet2 工作表标签。

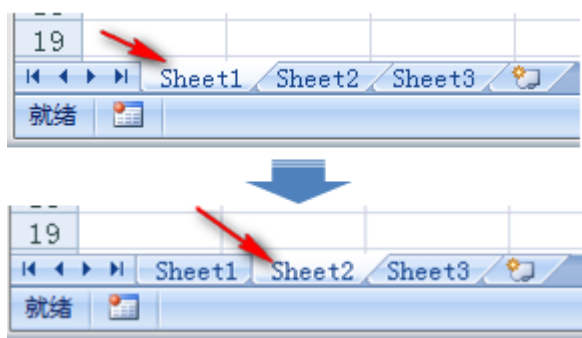


图 5.1

然而，在录制宏时，Excel 会使用 Sheets 集合的 Select 方法。例如，在打

开新工作簿时，Excel 一般默认 Sheet1 工作表为当前工作表，此时，打开宏录制器，选取 Sheet2 工作表，录制的代码如下：

```
Sub Macro1()  
'  
' Macro1 Macro  
'  
'  
  
    Sheets("Sheet2").Select  
End Sub
```

下面的语句：

```
Sheets.Select
```

选择工作表中所有的工作表，如下图 5.2 所示：

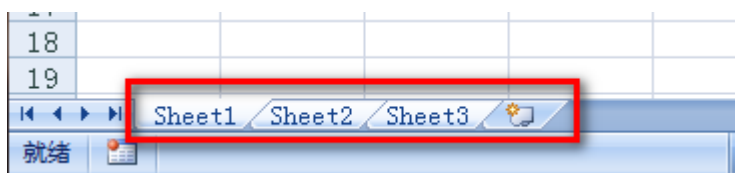


图 5.2

在工作簿中，工作表 Sheet1 为当前工作表，按住 Ctrl 键，选取工作表 Sheet2，这样同时选取工作表 Sheet1 和 Sheet2，如下图 5.3 所示：

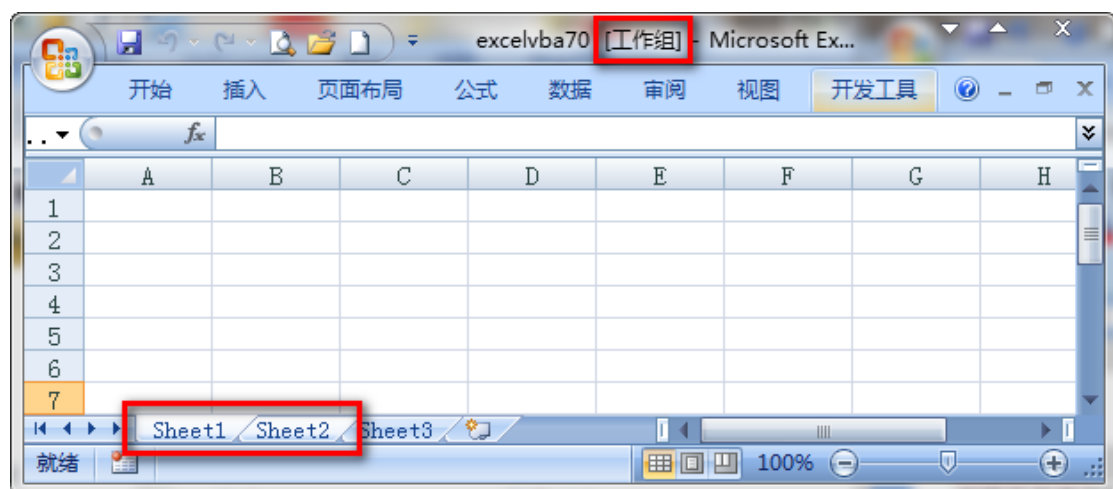


图 5.3

录制上述操作的代码如下：

```
Sub Macro2 ()  
'  
' Macro2 Macro  
'  
'  
  
    Sheets (Array ("Sheet1", "Sheet2")).Select  
    Sheets ("Sheet1").Activate  
End Sub
```

说明：

- Excel 将同时选中的工作表编成工作组，并以刚开始的活动工作表为组中的活动工作表。
- Array 函数创建一个数组。

Select 方法有一个可选的参数 Replace，当指定该参数值为 True 时，使用指定的工作表代替当前的工作表成为活动工作表。

例如，当前工作表为 Sheet2，运行下面的语句：

```
Sheets ("Sheet3").Select True
```

工作表 Sheet3 将成为活动工作表，如下图 5.4 所示：

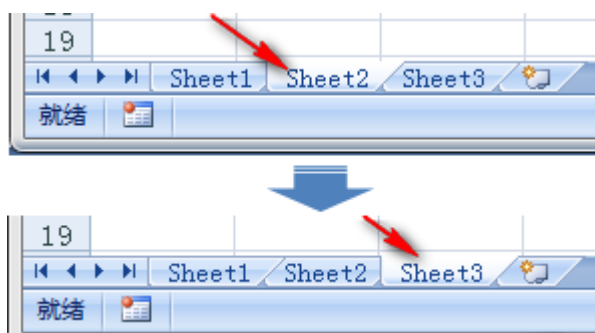


图 5.4

当然，上面的语句不带参数也会使 Sheet3 成为活动工作表。

当指定该参数值为 False 时，将扩展选定的工作表包含指定的工作表。

例如，当前工作表为 Sheet2，运行下面的语句：

```
Sheets("Sheet3").Select False
```

将扩展选定的工作表为 Sheet2 和 Sheet3，如下图 5.5 所示：

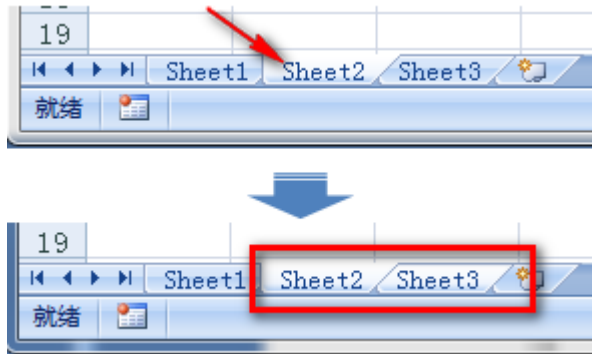


图 5.5

Activate 方法与 Select 方法的区别

- 在激活或者选取隐藏的工作表时，要使用 Activate 方法。如果使用 Select 方法，则会报错。
- 正如在上文中所看到的，Select 方法一次可以选取一个或者多个工作表，但 Activate 方法一次只能激活一个工作表。

6. 隐藏工作表——Visible 属性

我们可以隐藏工作表，让其对最终用户不可见。如下图 6.1 所示，在工作表标签 Sheet1 中单击右键，在弹出的菜单中选择“隐藏”，将工作表 Sheet1 隐藏。

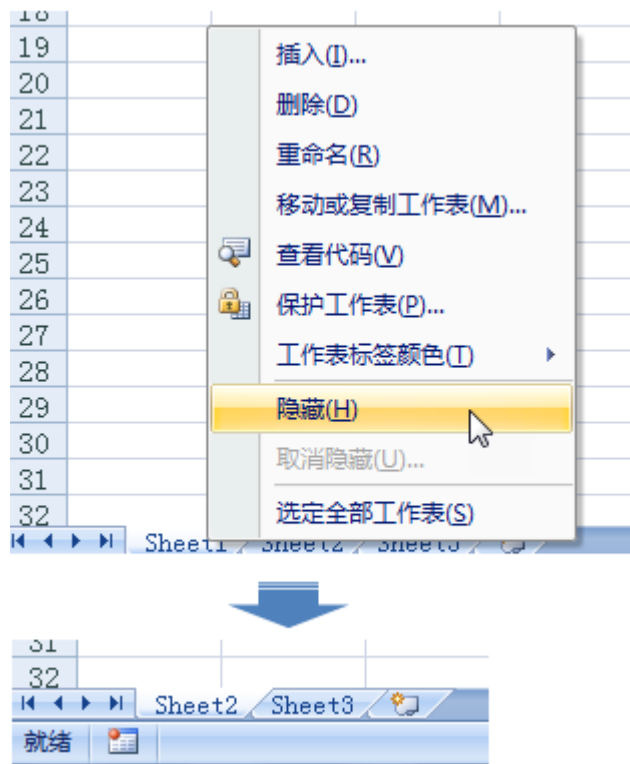


图 6.1

上述操作由 Excel 宏录制器录制的代码如下：

```
Sub Macro1()  
'  
' Macro1 Macro  
' 由完美 Excel 创建
```

```
Sub  
  
    Sheets("Sheet1").Select  
    ActiveWindow.SelectedSheets.Visible = False  
End Sub
```

Visible 属性用于设置工作表的可见状态，例如上面操作中设置工作表 Sheet1 的 Visible 属性为 False，使其隐藏。接下来，如果运行下面的语句：

```
Sheets("Sheet1").Visible = True
```

工作表 Sheet1 将重新可见。

Visible 属性也可返回指定工作表的可见状态。语句：

```
Sheets("Sheet1").Visible
```

返回的值表明工作表的可见状态。其返回值是一个 xlSheetVisibility 常量：

- -1—xlSheetVisible，代表正常可见状态
- 0—xlSheetHidden，代表工作表隐藏状态，等同于在工作表标签右键菜单中选择“隐藏”命令；隐藏后，可在工作表标签右键菜单中选择“取消隐藏”来使恢复工作表可见。
- 2—xlSheetVeryHidden，代表工作表隐藏状态，此时在工作表标签右键菜单中不能取消隐藏。

注意，工作簿中必须至少有一个可见工作表。

可以在 VBE 编辑器的属性窗口设置相应工作表的可见状态，如下图 6.2 所示。

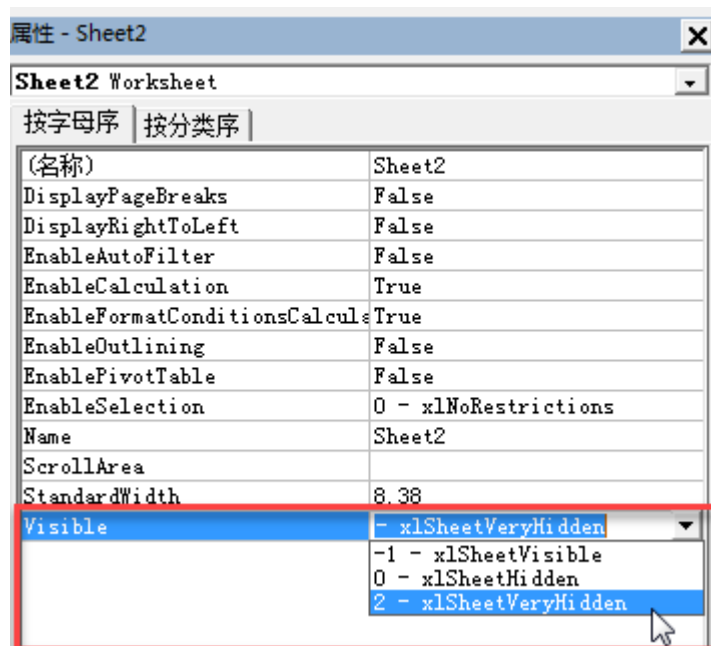


图 6.2

当然，除本文开头设置 `Visible` 属性值为 `False` 或者 `True` 来使工作表隐藏/可见外，还可以设置上述常量值来使工作表隐藏或者可见。例如，下面的语句：

```
Worksheets("Sheet2").Visible = xlSheetVeryHidden
```

使工作表 `Sheet2` 隐藏，并且在 Excel 界面的工作表标签中单击右键时不能使其恢复可见。此时，必须在 VBE 界面设置该工作表的 `Visible` 属性或者使用代码设置 `Visible` 属性使该工作表可见。

示例 1：统计工作簿中可见工作表数量

下面的程序将统计工作簿中可见工作表的数量并显示：

```
Sub CountVisibleWorksheets()  
    Dim i As Long  
    Dim lngNum As Long  
  
    lngNum = 0  
  
    '遍历工作表
```

```

For i = 1 To Sheets.Count
    '如果工作表可见则统计
    If Sheets(i).Visible = xlSheetVisible Then
        lngNum = lngNum + 1
    End If
Next

MsgBox "工作簿中可见工作表的数量是：" & lngNum
End Sub

```

对于默认带有 3 个工作表的工作簿，如果使用上文中的代码将工作表 Sheet2 隐藏后，统计的可见工作表数为：

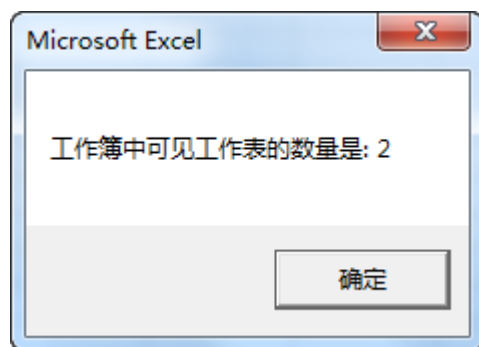


图 6.3

可以将上述代码转换为通用的函数过程，供其它过程在需要时调用：

```

'统计工作簿中所有可见工作表的数量
Function VisibleSheetsNum() As Long
    Dim i As Long
    Dim lngNum As Long

    lngNum = 0

    '遍历工作表
    For i = 1 To Sheets.Count
        '如果工作表可见则统计
        If Sheets(i).Visible = xlSheetVisible Then

```

```

        lngNum = lngNum + 1
    End If
Next

    VisibleSheetsNum = lngNum
End Function

```

示例 2：隐藏指定的工作表

下面的代码将隐藏指定的工作表，并指定了隐藏方式。

```

Sub HideWorksheet(strName As String, blnVeryHidden As Boolean)
    '如果 blnVeryHidden 值为 True, 则深度隐藏工作表
    If blnVeryHidden Then
        Worksheets(strName).Visible = xlSheetVeryHidden
    Else
        Worksheets(strName).Visible = xlSheetHidden
    End If
End Sub

```

下面的语句代码调用 HideWorksheet 过程，隐藏工作表 Sheet1：

```
HideWorksheet "Sheet1", True
```

在隐藏指定的工作表之前，我们可以使用 [3. 工作表名称—Name 属性](#) 中 [示例 4：检查工作表是否已存在](#) 来检查指定名称的工作表是否存在，然后进行下一步操作。

示例 3：使用工作簿中的工作表全部可见

有时，工作簿中有多个隐藏的工作表，可以遍历工作表，使所有工作表都可见。

下面的代码取消隐藏工作簿中的所有工作表：

```
Sub UnhideAllWorksheets()
```

```
Dim ws As Worksheet

For Each ws In Worksheets
    ws.Visible = xlSheetVisible
Next ws
End Sub
```


本章内容 2017 年 8 月 17 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读 \(73\): 添加和删除工作表——Add 方法和 Delete 方法](#)

7. 添加和删除工作表——Add 方法和 Delete 方法

一般情况下，开启工作簿时，Excel 默认自带 3 个工作表。如果想要更多的工作表，可以单击工作表界面底部工作表标签右侧的“插入工作表”标签或者按 Alt+F11 组合键快速插入一个新工作表。也可以在工作表标签中单击右键，在弹出的菜单中单击“插入”，选择插入工作表的类型后插入一个新工作表。

下面是单击“插入工作表”标签后，Excel 录制的代码：

```
Sub Macro1()  
'  
' Macro1 Macro  
' 由完美 Excel 创建  
'  
  
    Sheets.Add After:=Sheets(Sheets.Count)  
End Sub
```

在我的工作簿中，Excel 插入了一个名为“Sheet4”的工作表，并且该工作表位于所有工作表之后，这是由参数 After 指定的。

下面是先选取工作表 Sheet1，然后单击右键，在弹出的菜单中单击“插入”后插入一个新工作表的操作，由 Excel 录制的代码：

```
Sub Macro2()  
'  
' Macro2 Macro
```

```
' 由完美 Excel 创建  
  
'  
  
    Sheets("Sheet1").Select  
    Sheets.Add  
  
End Sub
```

Excel 将插入的工作表默认放置在当前工作表之前。

从上面的代码可以看出，Excel VBA 使用 Add 方法添加新工作表。

Add 方法

Add 方法创建一个新工作表并使其成为活动工作表，其语法如下：

工作表对象.Add(Before, After, Count, Type)

说明：

- 所有参数均可选。
- 参数 Before 指定一个工作表，新添加的工作表放置在该工作表之前。
- 参数 After 指定一个工作表，新添加的工作表放置在该工作表之后。
- 参数 Before 和参数 After 只能二选一。
- 参数 Count 指定要添加的工作表数量，默认值为 1。
- 参数 Type 指定添加的工作表类型，可以是下列 xlSheetType 常量之一：
xlWorksheet（工作表）、xlChart（图表工作表）、
xlExcel4MacroSheet（宏表）、xlExcel4IntlMacroSheet。如果
基于已有模板插入工作表，那么指定该模板的路径。默认值是
xlWorksheet。
- 如果没有指定参数 Before 和参数 After，那么在当前工作表的前面插入新工作表。
- 该方法返回一个代表新工作表的对象。

示例 1：添加并命名新工作表

下面的代码在工作簿中所有工作表之后添加一个新工作表，并将其命名为“完美 Excel”。

```
Sub AddNewSheetPlaceInLast()  
    Worksheets.Add after:=Worksheets(Worksheets.Count)  
    ActiveSheet.Name = "完美 Excel"  
End Sub
```

说明：

- 新添加的工作表成为活动工作表，因此使用 `ActiveSheet` 属性来返回当前工作表并使用 `Name` 属性来给工作表命名。

也可以删除工作表。在 Excel 界面底部的工作表标签中，选择要删除的工作表，单击右键，在弹出的菜单中选择“删除”，Excel 会弹出一个提示工作表可能存在数据的警告消息框（如下图 7.1 所示），单击“删除”按钮，即可删除工作表。

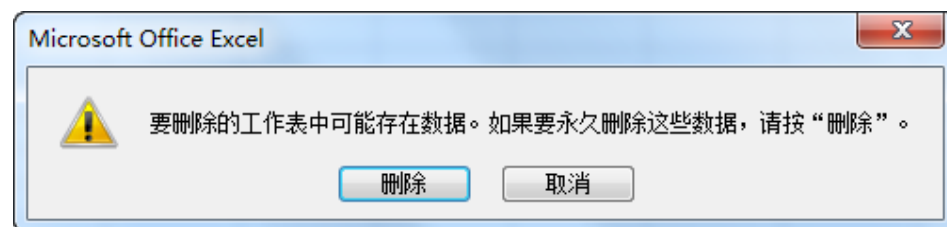


图 7.1

上述操作由 Excel 录制的宏代码为：

```
Sub Macro3()  
'  
' Macro3 Macro  
' 由完美 Excel 创建  
'  
'
```

```
Sheets("Sheet6").Select  
ActiveWindow.SelectedSheets.Delete  
End Sub
```

从上面的代码可以看出，Excel VBA 使用 Delete 方法删除工作表。

Delete 方法

删除工作表，其语法如下：

工作表对象.Delete

说明：

- 该方法返回一个 Boolean 值。
- 删除工作表时，Excel 会默认显示一个提示用户来确认删除操作的对话框。如果用户单击“取消”按钮，则返回 False；如果单击“删除”按钮，则返回 True。

示例 2：删除除指定工作表名外的所有工作表

下面的代码删除工作簿中除“完美 Excel”工作表外的所有工作表：

```
Sub DeleteWorksheet()  
    Dim ws As Worksheet  
  
    '关闭警告消息  
    Application.DisplayAlerts = False  
  
    '遍历工作表并删除工作表名不是"完美 Excel"的工作表  
    For Each ws In Worksheets  
        If ws.Name <> "完美 Excel" Then ws.Delete  
    Next ws
```

'恢复警告消息

```
Application.DisplayAlerts = True  
End Sub
```

说明:

- 代码中设置 DisplayAlerts 属性值为 False 来禁止显示警告消息框，避免代码在删除工作表时弹出警告消息框，影响用户体验。代码最后将该属性值设置为 True，以恢复 Excel 的默认设置。
- Delete 方法也将隐藏的工作表删除。

示例 3：安全地删除工作表

Excel 必须保证工作簿中至少有一个可见工作表，因此在删除工作表时，如果删除工作簿中仅有的一个可见工作表，就会导致运行时错误。

下面的代码保证工作簿中至少有一个可见工作表，然后才进行删除操作。代码使用了 **6. 隐藏工作表—Visible 属性** 示例 1 中用于统计工作簿中可见工作表数量的 VisibleSheetsNum 函数过程。

```
Sub DeleteSheet()  
    Dim strName As String  
  
    '指定要删除的工作表名  
    strName = "Sheet1"  
  
    If VisibleSheetsNum > 1 Then  
        Application.DisplayAlerts = False  
        Worksheets(strName).Delete  
    Else  
        MsgBox "工作簿中已没有可供删除的工作表!"  
    End If  
End Sub
```

下面是 Steven M.Hansen 给出的安全删除工作表的较完善的代码,辑录于此,供学习与研究。

```
'删除参数 ws 指定的工作表
'bQuiet 指定是否显示警告消息

Function DeleteSheet(ws As Worksheet, bQuiet As Boolean) As Boolean

    Dim bDeleted As Boolean

    On Error GoTo ErrHandler

    bDeleted = False

    If CountVisibleSheets(ws.Parent) > 1 Then
        '决定是否显示警告消息
        If bQuiet Then Application.DisplayAlerts = False

        bDeleted = ws.Parent.Worksheets(ws.Name).Delete
    Else
        '工作簿中需要至少一个工作表
    End If

ExitPoint:
    '恢复显示警告消息
    Application.DisplayAlerts = True
    DeleteSheet = bDeleted
    Exit Function
ErrHandler:
    bDeleted = False
    Resume ExitPoint
End Function

'统计工作簿 wb 中所有可见工作表的数量
```

```

Function CountVisibleSheets(wb As Workbook) As Integer
    Dim nSheetIndex As Integer
    Dim nCount As Integer
    nCount = 0
    For nSheetIndex = 1 To wb.Sheets.Count
        If wb.Sheets(nSheetIndex).Visible = xlSheetVisible
Then
            nCount = nCount + 1
        End If
    Next
    CountVisibleSheets = nCount
End Function

```

说明:

- DeleteSheet 函数过程需要两个变量，变量 ws 表示要删除的工作表，变量 bQuiet 设置在删除工作表时是否显示警告消息。
- 代码中的 ws.Parent 返回工作表的父对象，即工作表所在的工作簿。
- 变量 bDeleted 用于跟踪工作表是否删除。
- 过程代码使用了错误处理技术，在代码运行过程中发生错误时会直接执行错误处理代码。在后续文章中会详细介绍有关错误处理技术。
- 在工作簿中，至少要有一个可见工作表（包括图表工作表等），因此使用 CountVisibleSheets 函数过程统计工作簿中的可见工作表数。
- ws.Parent.Worksheets(ws.Name).Delete 可获取一个 Boolean 值，代表工作表是否被删除。此外，如果直接使用 ws.Delete，则不能够捕捉指定工作表实际上是否已删除。

8. 移动或复制工作表——Move 方法和 Copy 方法

有时候，我们可能想复制工作表，保留一份工作表的副本，以免误操作打乱工作表后无法恢复。有时候，我们也可能想移动工作表，调整工作表顺序，将工作表重新排列，以方便工作表的布置。本文介绍在 VBA 中实现这两种操作。

在工作簿底部的工作表标签中单击右键，选择菜单中的“移动或复制工作表”，如下图 8.1 所示。

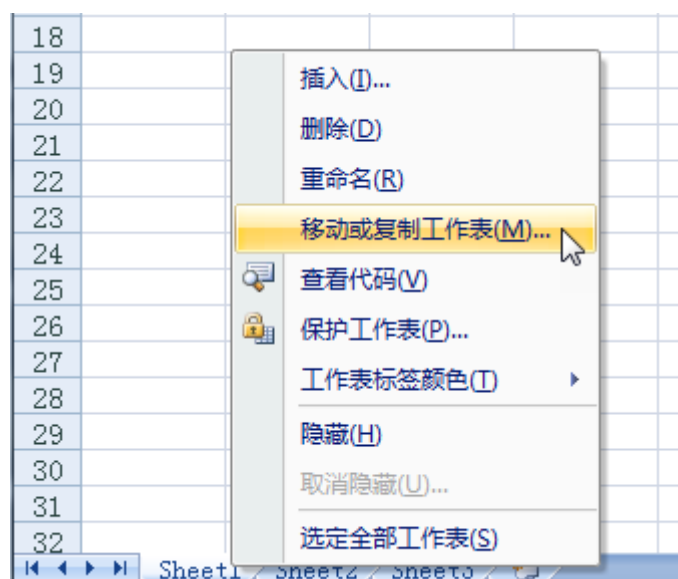


图 8.1

在出现的对话框中选中“建立副本”，如下图 8.2 所示。

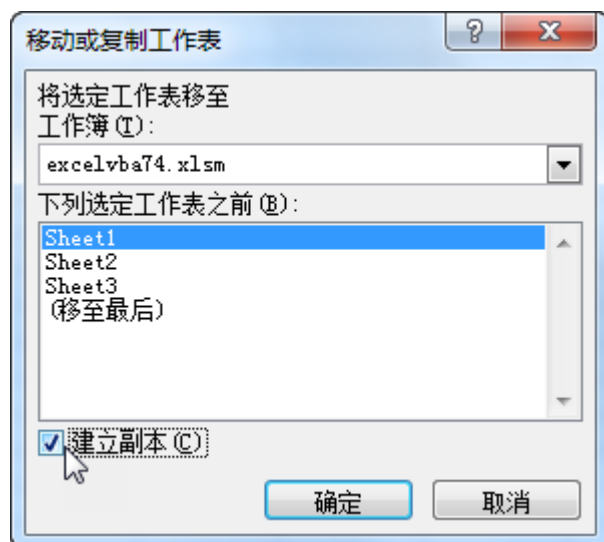


图 8.2

单击“确定”按钮，复制得到工作表 Sheet1 的一份副本，Excel 将其自动命名为工作表 Sheet1 (2)，如下图 8.3 所示。



图 8.3

技巧：选择要复制的工作表标签后，按住 Ctrl 键，拖动工作表即可完成复制工作表操作。

使用宏录制器录制上述操作的代码为：

```
Sub Macro1 ()
'
' Macro1 Macro
' 由完美 Excel 创建
'

    Sheets("Sheet1").Select
    Sheets("Sheet1").Copy Before:=Sheets(1)
End Sub
```

Copy 方法

工作表对象的 Copy 方法复制指定工作表到工作簿中指定的位置。其语法为：

工作表对象.Copy (Before, After)

说明：

- 参数 Before 和参数 After 用于指定工作表，复制的工作表将放置在该工作表之前或者之后。二者只能同时选一。
- 如果没有指定参数，Excel 将创建一个包含所复制工作表的新工作簿。

下面接着来看移动工作表的操作。

在上图 8.2 所示的对话框中，我们不选取“建立副本”复选框，这样将会移动所选择的工作表在对话框中指定的某工作表之前。或者在 Excel 工作簿中，选择要移动的工作表标签后，直接拖动到想要放置的位置。如下图 8.4 所示，将工作表 Sheet1 (2) 移动至工作表 Sheet3 之前。

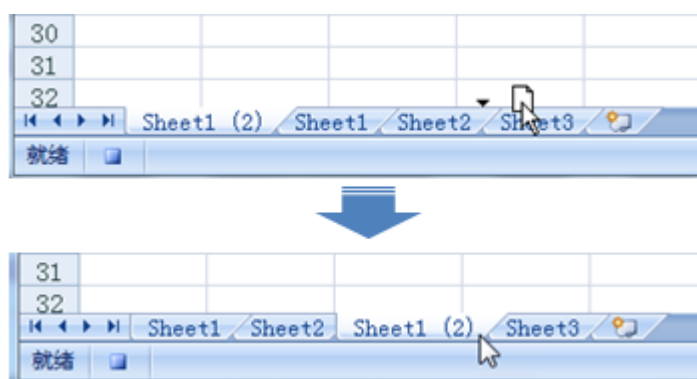


图 8.4

所录制的代码如下：

```
Sub Macro2 ()  
'  
' Macro2 Macro  
' 由完美 Excel 创建  
'  
  
Sheets ("Sheet1 (2)").Select  
Sheets ("Sheet1 (2)").Move After:=Sheets (3)
```

```
End Sub
```

Move 方法

工作表对象的 Move 方法移动指定工作表到工作簿中指定的位置。其语法为：

```
工作表对象.Move (Before, After)
```

说明：

- 参数 Before 和参数 After 用于指定工作表，复制的工作表将放置在该工作表之前或者之后。二者只能同时选一。
- 如果没有指定参数，Excel 将创建一个包含所复制工作表的新工作簿。

对比一下，Copy 方法与 Move 方法似乎完全相同，只是 Copy 方法创建了工作表的一份副本，而 Move 方法只是移动了工作表的位置。

下面的语句将工作簿中第 3 个工作表放置到一个新工作簿中：

```
Worksheets(3).Copy
```

或者：

```
Worksheets(3).Move
```

示例 1：将工作表移动到最后

下面的代码将名为“完美 Excel”的工作表移动到工作簿工作表的最后。

```
Sub MoveWorksheets()  
    Worksheets("完美 Excel").Move _  
        After:=Worksheets(Worksheets.Count)  
End Sub
```

9. 保护工作表——Protect 方法

在 Excel 中，我们可以保护工作表，避免用户对工作表进行不必要的修改。

要想保护工作表，可以单击“审阅”选项卡中的“保护工作表”按钮，如下图 9.1 所示。

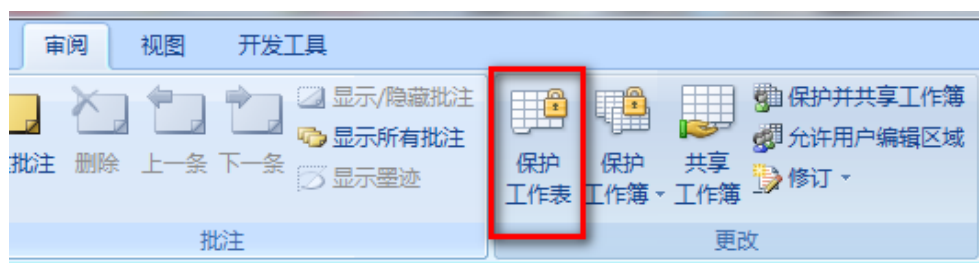


图 9.1

在“保护工作表”对话框中输入密码（当然也可以为空，即不设置密码），进行相应的选取设置后，即对工作表设置了保护。此时，用户不能随意在工作表中编辑单元格。

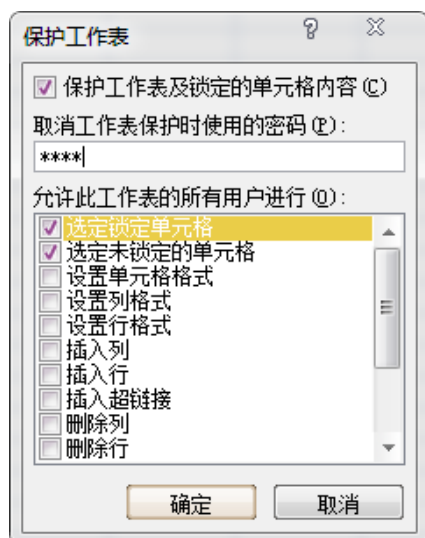


图 9.2

Excel 宏录制器为上述操作录制的代码为：

```
Sub Macro1()  
'  
' Macro1 Macro  
' 由完美 Excel 录制  
'  
  
    ActiveSheet.Protect DrawingObjects:=True,  
Contents:=True, Scenarios:=True  
End Sub
```

可以看出，Excel VBA 使用 Protect 方法保护工作表，虽然在操作中我们为保护工作表设置了密码，但 Excel 并没有为我们录制设置的密码项。然而，我们可以方便地使用 Protect 方法通过代码为保护工作表设置密码。

Protect 方法

工作表对象的 Protect 方法用于保护工作表，使之不能够被修改。Protect 方法有很多参数，多达 16 个，但均为可选参数，我们可以根据需要有选择地使用这些参数以达到目的，例如，虽然对工作表进行了保护，但仍允许用户对限定的区域进行修改。

示例 1：使用 Protect 方法保护工作表

下面的代码保护当前工作表，并设置密码为“test”。

```
Sub ProtectActiveSheet()  
    If Not ActiveSheet.ProtectContents Then  
        ActiveSheet.Protect Password:="test", _  
                                DrawingObjects:=True, _  
                                Contents:=True, _  
                                Scenarios:=True  
    End If  
End Sub
```

```
End If  
End Sub
```

说明:

- 参数 Password 用来设置密码，区分大小写。如果忽略该参数，那么在取消工作表保护时不需要密码。
- 参数 DrawingObject、Contents、Scenarios 分别设置是否保护工作表中的形状、内容和方案。它们的默认值都是 True。
- ProtectContents 属性返回一个 Boolean 值，表明工作表是否已保护。上述代码运行时，如果工作表已受保护，则不执行任何操作。

Unprotect 方法

要想通过代码修改受保护的工作表，必须先取消工作表保护，才能进行修改。Unprotect 方法用于取消工作表保护。

示例 2：使用 Unprotect 方法取消工作表保护

下面的代码取消对设置了密码“test”的当前工作表的保护。

```
Sub UnprotectActiveSheet()  
    If ActiveSheet.ProtectContents Then  
        ActiveSheet.Unprotect Password:="test"  
    End If  
End Sub
```

说明:

- Unprotect 方法只有一个可选参数 Password，用来提供保护工作表时所设置的密码。
- 如果保护工作表时没有设置密码，可以忽略参数 Password。

- 如果提供的密码不正确，那么 Excel 会弹出一个错误提示框，如下图

9.3 所示。

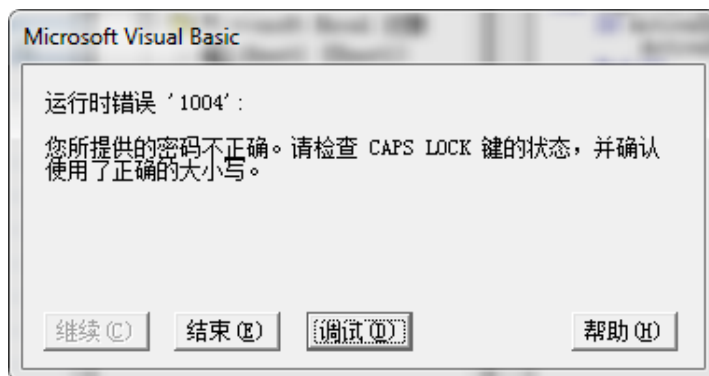


图 9.3

示例 3：设置与取消工作表保护的通用代码

下面是 Steven M. Hansen 编写的一段代码，用于设置与取消工作表保护：

```
'测试设置和取消保护工作表的函数
'可以以此代码为框架,运用到自己的代码中
Sub test()
    Dim ws As Worksheet

    '设置要保护的工作表
    Set ws = Worksheets("Sheet3")

    '使用 ProtectWorksheet 函数设置保护工作表
    If Not ProtectWorksheet(ws, "Test") Then
        MsgBox ws.Name & "工作表没有被保护."
    Else
        MsgBox ws.Name & "工作表已受保护."
    End If
```



```

'使用 UnprotectWorksheet 函数取消工作表保护

If UnprotectWorksheet(ws, "Test") Then
    MsgBox ws.Name & "工作表已取消保护."
Else
    MsgBox ws.Name & "工作表仍受保护."
End If

Set ws = Nothing
End Sub

'返回是否设置了保护工作表

Function ProtectWorksheet(ws As Worksheet, strPassword As
String) As Boolean
    On Error GoTo ErrHandler
    If Not ws.ProtectContents Then
        ws.Protect strPassword, True, True, True
    End If
    ProtectWorksheet = True
    Exit Function
ErrHandler:
    ProtectWorksheet = False
End Function

'返回是否取消工作表保护

Function UnprotectWorksheet(ws As Worksheet, strPassword As
String) As Boolean
    On Error GoTo ErrHandler
    If ws.ProtectContents Then
        ws.Unprotect strPassword
    End If
    UnprotectWorksheet = True
    Exit Function
ErrHandler:

```

```
UnprotectWorksheet = False  
End Function
```

说明：

- 这段代码可以沿用到自己的代码中，用于需要在代码运行过程中设置工作表保护，或者需要修改受保护工作表中的内容而临时取消工作表保护的情形。
- 代码中使用了错误处理语句，以及带参数的函数过程。在[完美 Excel](#) 微信公众号后续的文章中会详细讲解这些技术。

10. 打印工作表——PrintOut 方法

在 VBA 中，使用 PrintOut 方法来打印工作表。例如，语句：

```
ActiveSheet.PrintOut
```

将打印当前工作表。

PrintOut 方法的语法如下：

```
对象.PrintOut (From, To, Copies, Preview, ActivePrinter, PrintToFile, Collate, PrToFile, IgnorePrintAreas)
```

说明：

- 所有参数均可选。使用适当的参数指定打印机、份数、逐份打印以及是否需要打印预览。使用参数 PrintToFile 和参数 PrToFile 将工作表打印到文件。参数 From 和参数 To 用于指定打印的页码范围。
- 参数 From 指定开始打印的页码。如果忽略，则从头开始打印。
- 参数 To 指定最后打印的页码。如果忽略，则打印到最后一页。
- 参数 Copies 指定要打印的份数。如果忽略，则只打印 1 份。
- 参数 Preview 指定打印前是否要预览打印效果。设置为 True 则打印预览；设置为 False（默认值）则直接打印。
- 参数 ActivePrinter 设置当前打印机的名称。
- 参数 PrintToFile 设置为 True，将打印到文件。如果没有指定参数 PrToFile，将提示用户输入要输出的文件名。

- 参数 Collate 设置为 True 将逐份打印。
- 参数 PrToFileName 在参数 PrintToFile 设置为 True 时指定想要打印到文件的名称。
- 参数 IgnorePrintAreas 设置为 True 将忽略打印区域，打印整份文档。

示例 1：实现逐行批量打印

本示例是[完美 Excel](#) 微信公众号中《问与答 9：如何实现逐条批量打印》的例子。

如下图 10.1 所示的工作表，除表头的第 1 行不变外，从第 2 行开始逐条打印记录。

	A	B	C	D	E	F	G
1	编号	姓名	性别	基本工资	绩效工资	加班费	应发合计
2	001	郭靖	男	2000	3000	1000	6000
3	002	黄蓉	女	2200	3000	800	6000
4	003	杨康	男	2100	2900	900	5900
5	004	穆念慈	女	2200	2800	1000	6000
6	005	杨过	男	2200	3100	1000	6300
7	006	黄老邪	男	2500	3200	600	6300
8	007	洪七公	男	2600	3300	500	6400
9	008	花无缺	男	2100	2900	500	5500
10	009	江小鱼	男	2100	2800	600	5500
11							

图 10.1

代码如下：

```

Sub testPrintOfRow()
    Dim lngLastRow As Long
    Dim i As Long

    '找到工作表最后一行
    lngLastRow = Range("A" & Rows.Count).End(xlUp).Row

    '隐藏数据行
    Range("A2:A" & lngLastRow).EntireRow.Hidden = True

    '逐条显示数据行并打印
    For i = 2 To lngLastRow
        '显示相应的数据行
        Range("A" & i).EntireRow.Hidden = False
        '打印
        ActiveSheet.PrintOut
        '隐藏数据行
        Range("A2:A" & lngLastRow).EntireRow.Hidden = True
    Next i
End Sub

```

说明:

- PrintOut 方法只打印工作表中的可见行。因此，将要打印的行逐条显示，同时隐藏不需要打印的行，从而实现逐条打印。

示例 2：只打印奇数页

如果工作表共 800 页，只想打印奇数页，该如何实现呢？

下面是一段简单的小程序，可以只打印奇数页。

```

Sub PrintOddPages()
    Dim k As Integer
    k = 1
    Do While (k < 800)
        ActiveSheet.PrintOut From:=k, to:=k
        k = k + 2
    Loop
End Sub

```

11. Evaluate 方法

Worksheet 对象的 Evaluate 方法，有时会非常有用。

但是在这里，只是简单地介绍该方法。在专题文章中，我会对该方法进行详细剖析。

Evaluate 方法将 Excel 名称转换成对象或值。

其语法为：

工作表对象.Evaluate (Name)

说明：

- 参数 Name，必需，使用 Excel 命名约定的对象的名称。
- 可以使用下列类型的名称：
 - ✧ A1 样式引用，引用被认为是绝对引用。
 - ✧ 单元格区域。可以使用冒号、空格和逗号操作符所构成的单元格区域、交叉区域和联合区域的引用。
 - ✧ 定义的名称。
 - ✧ 外部引用。可以使用!操作符来引用另一工作簿中的单元格或者命名区域。
 - ✧ 图表对象。可以指定任何图表对象名，例如“图例”、“绘图区”或者“系列 1”，来访问该对象的属性和方法。
 - ✧ 当参数指定为字符串时，使用方括号（[]）与调用 Evaluate 方法相同。然而，如果引用中含有变量，就必须使用 Evaluate 方法。

示例 1：输入值并设置单元格格式

如下图 11.1 所示的工作表，将单元格区域 A1:A7 命名为“Data”，使用含有 Evaluate 方法的代码在该区域输入数据并设置为加粗。

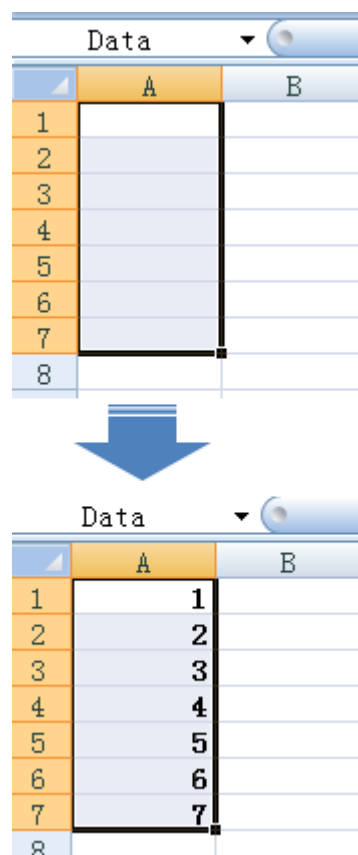


图 11.1

代码如下：

```
Sub testEvaluate()  
    Dim rng As Range  
    Dim i As Integer  
  
    i = 1  
  
    For Each rng In Evaluate("Data")  
        rng.Value = i  
        i = i + 1  
    Next rng
```

```
    Evaluate("Data").Font.Bold = True
End Sub
```

可以看出，Evaluate("Data") 等价于 Range("A1:A7")。

可以将上述代码中的 Evaluate("Data") 修改为 [Data]，效果相同，但更简洁。

```
Sub testEvaluate()
    Dim rng As Range
    Dim i As Integer

    i = 1

    For Each rng In [Data]
        rng.Value = i
        i = i + 1
    Next rng

    [Data].Font.Bold = True
End Sub
```

示例 2：获取单元格的值

下面的代码获取当前工作表单元格 A1 中的值：

```
Sub GetValueWithEvaluate()
    MsgBox Evaluate("A1").Value
End Sub
```

运行后的结果如图 11.2。

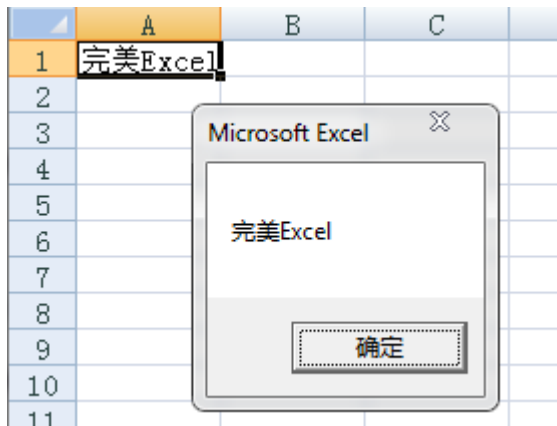


图 11.2

上面代码中的 `Evaluate("A1")` 可以修改为 `[A1]`。

示例 3：获取另一工作簿中单元格的值

下面的代码将工作簿 Book2 的工作表 Sheet1 中单元格区域 A1:A5 的值输入到当前工作簿的工作表 Sheet3 中。

```
Sub GetValueFromAnotherWB()  
    Sheet3.[A1:A5] =  
    Evaluate("[Book2.xlsx]Sheet1!A1:A5").Value  
End Sub
```

或者：

```
Sub GetValueFromAnotherWB()  
    Sheet3.[A1:A5] = [[Book2.xlsx]Sheet1!A1:A5].Value  
End Sub
```

通过以上简单的示例可以发现，这里使用的 `Evaluate` 方法和直接使用 `Range` 对象引用单元格非常相似。

然而，为什么还要使用 `Evaluate` 方法呢？在[完美 Excel 微信公众号](#)专题文章中有详细的讲解。这里只是简单介绍一下 `Evaluate` 方法，让你对其有所了解。

12. 设置工作表页面——PageSetup 属性

在 VBA 中, 可以使用 PageSetup 属性返回的对象来设置工作表在打印时的页边距、纸张方向、纸张大小等。

PageSetup 属性的语法为:

`Worksheet 对象.PageSetup`

说明:

- 只读。返回一个 PageSetup 对象, 包含对指定工作表对象页面设置的全部内容, 其属性与“页面设置”对话框中相应内容如图 12.1~12.4 所示。

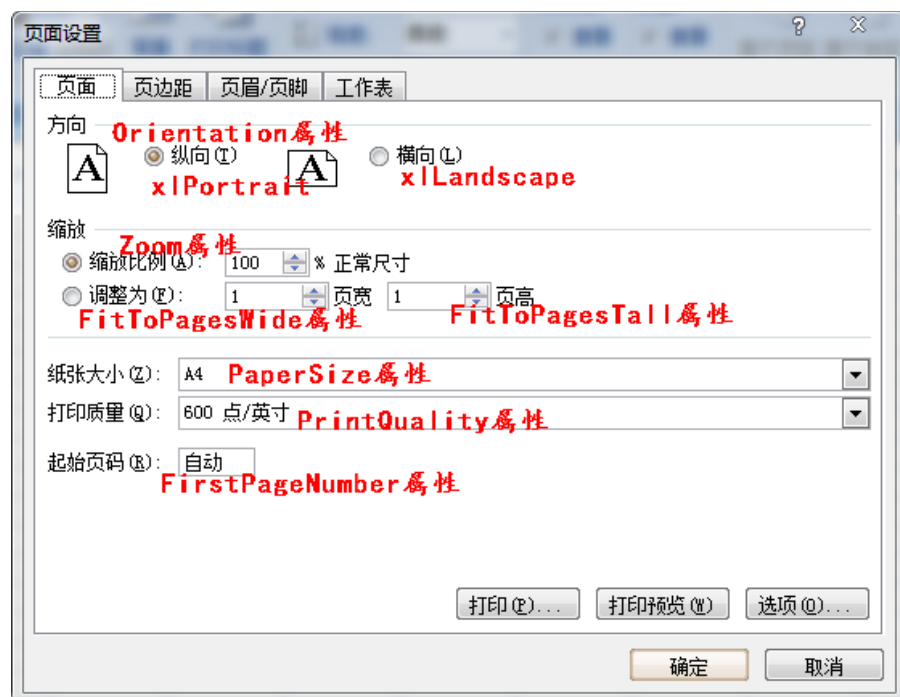


图 12.1

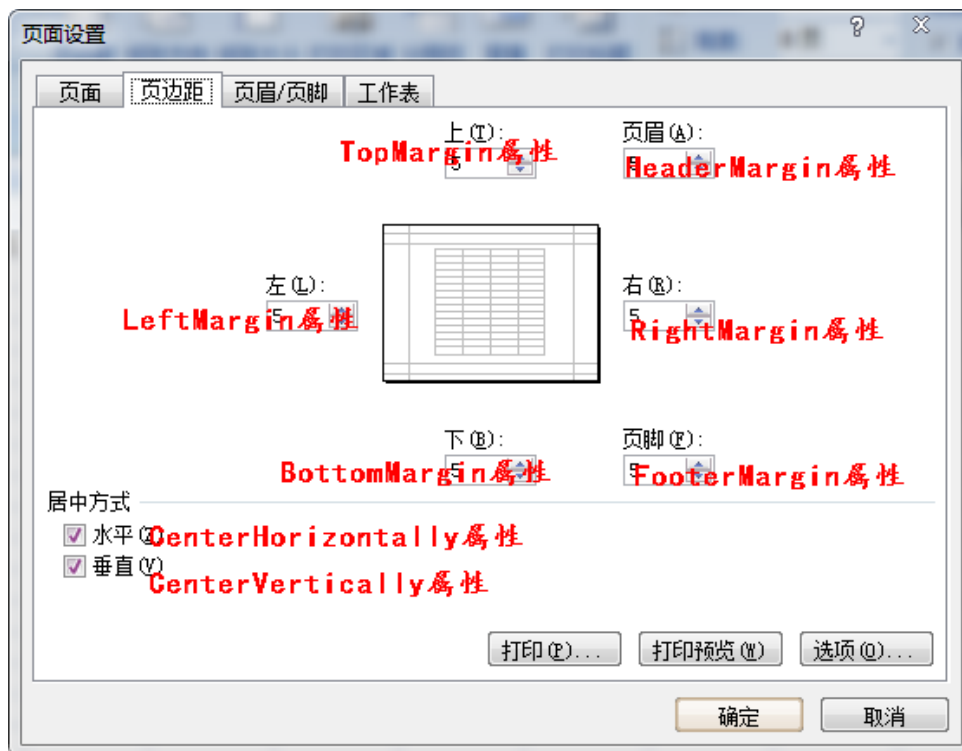


图 12.2

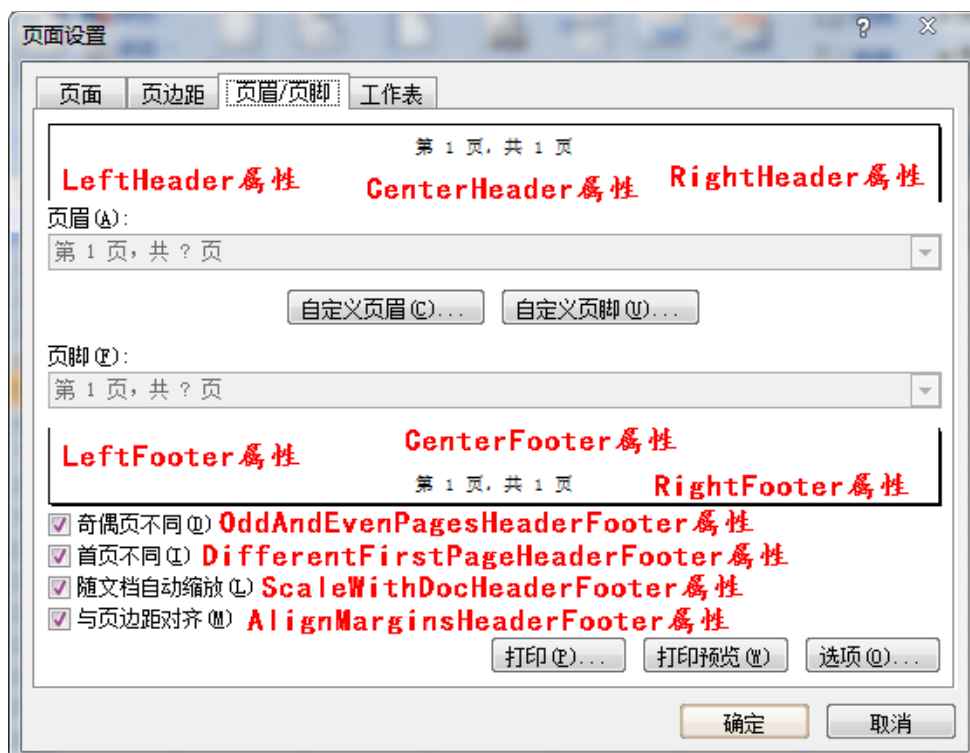


图 12.3

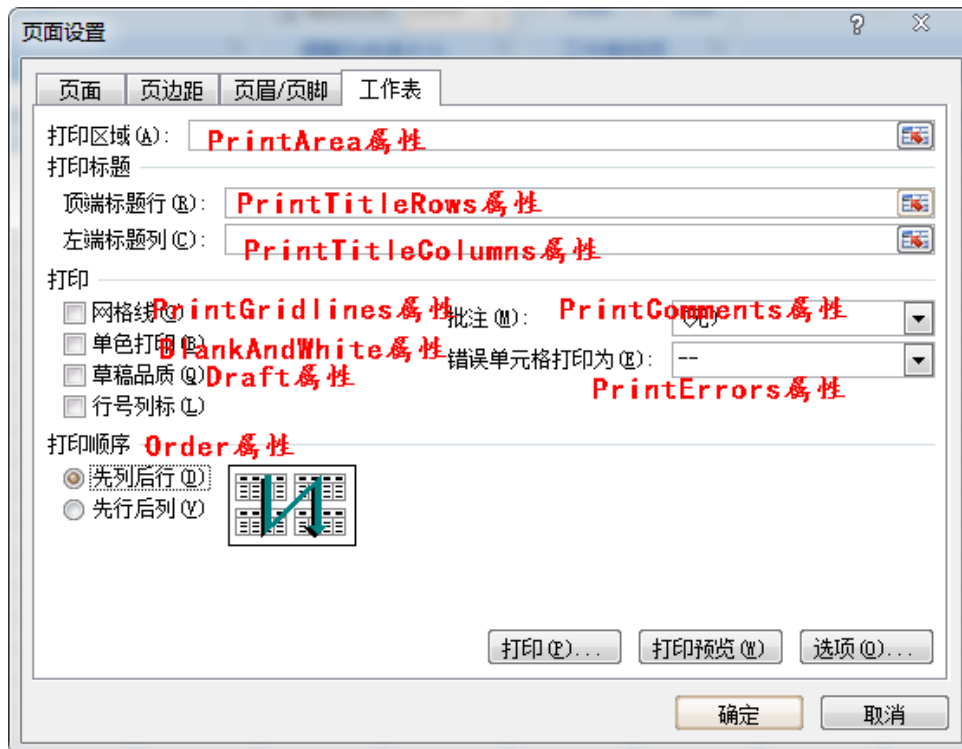


图 12.4

其中，一些属性需要设置 True 或 False，如图中复选框对应的属性；一些属性需要赋予其常量值，如图 12.1 中的方向对应的属性；一些属性需要设置数值，如图 12.2 中页边距对应的属性；一些属性需要填写内容，如图 12.4 中的打印区域。有兴趣的朋友可以自己录制设置页面的代码，或者试着设置其中一些属性的值，看看运行后的效果，以此熟悉这些属性。

示例：设置当前工作表页面

下面的代码设置当前工作表打印区域、打印标题行、顶部页边距、以及页眉文字。

```
Sub PageSetupTest()
    With ActiveSheet.PageSetup
        .PrintTitleRows = "A1"
        .PrintArea = "$A$1:$C$13"
        .TopMargin = Application.InchesToPoints(2)
        .LeftHeader = "完美 Excel"
        .CenterHeader = "第 &P 页，共 &N 页"
    End With
End Sub
```

```

        .RightHeader = "exceperfect"

    End With

    ActiveSheet.PrintPreview

End Sub

```

由于格式属性需要以磅为单位来度量，因此使用 `InchesToPoints` 函数将以英寸表示的尺寸数转换成以磅表示的尺寸。`PrintTitleRows` 设置每页的标题为单元格 A1 中的数据。运行代码后的结果如图 12.5 所示。



图 12.5

本章内容 2017 年 12 月 4 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读（79）：让 Excel 动起来——认识事件过程](#)

13. 让 Excel 动起来——认识事件过程

我们知道，在 Excel 中进行公式运算时，随着单元格数据的变化，公式单元格中的结果也会随之自动更新，在[完美 Excel](#) 微信公众号中有详细的视频演示，下面是演示截图。

	A	B	C	D	E	F
1	+	和：	15			
2	2	平均值：	3			
3	3					
4	4					
5	5					
6						
7						
8						

图 13.1

Excel VBA 中有一类称作“事件”的过程程序，能够对我们的操作自动作出相应的响应。例如，每次打开工作簿时，自动弹出一个设定好的信息框，图 13.2 为视频演示截图，演示视频在[完美 Excel](#) 微信公众号中。

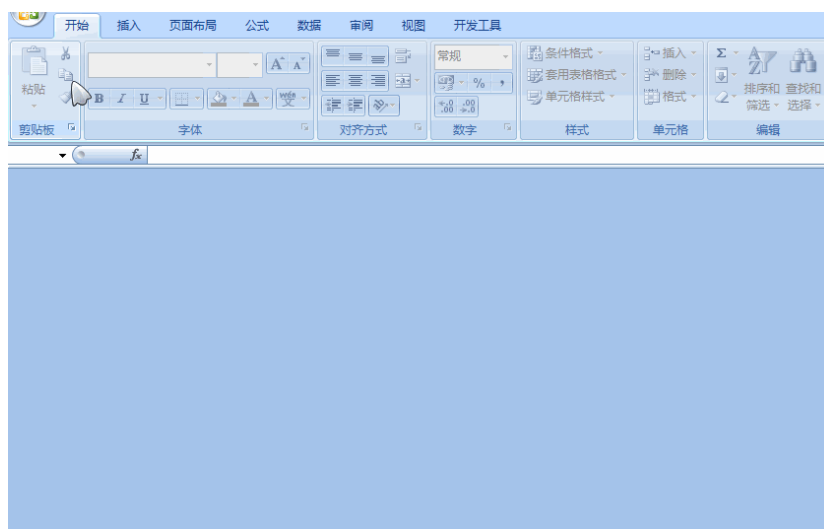


图 13.2

这是响应工作簿的 Open 事件，即在打开工作簿时 Excel 所发生的行为。

在 VBE 编辑器中，单击工作簿或者工作表模块，然后在代码输入窗口顶部左侧列表框中选择工作表对象名，在右侧列表框中会出现供选择的相应的事件。通常，在左侧列表框中选择对象名后，Excel 会自动在代码输入窗口中放置默认的事件，如工作表的 SelectionChange 事件。



图 13.3（演示视频截图，详细视频见[完美 Excel 微信公众号](#)）

事件响应所执行的操作就是我们在事件过程中放置的代码，这些代码规定了我们希望事件发生时要做什么。

在[完美 Excel 微信公众号](#)《[Excel 实战技巧 8：基于当前单元格实时显示相关数据记录](#)》中，当用户在工作表第 1 列含有数据的单元格中移动时，会自动弹出一个窗体，并显示该单元格所在行的详细信息。

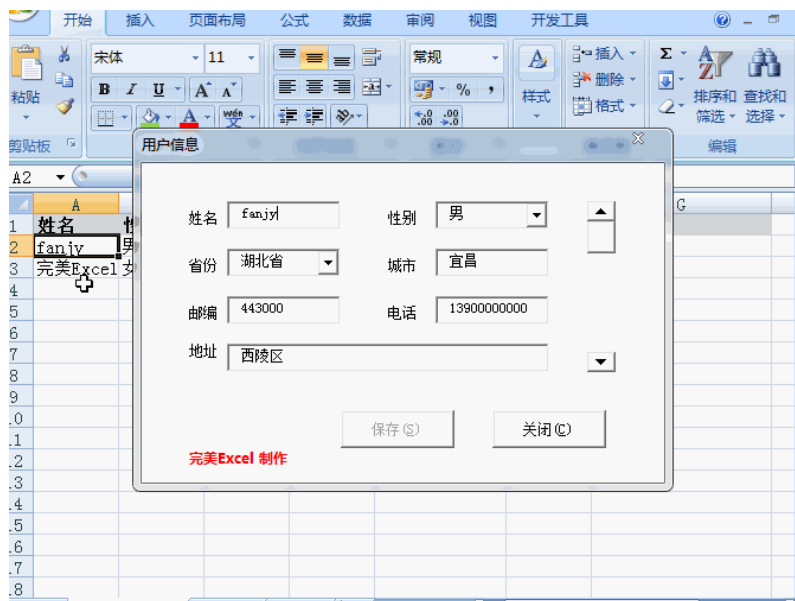


图 13.4（演示视频截图，详细视频见[完美 Excel 微信公众号](#)）

这里就是使用了工作表的 SelectionChange 事件，其代码如下：

```
Dim lngRow As Long

Private Sub Worksheet_SelectionChange (ByVal Target As Range)
    Dim rngBottom As Range

    With wksUserInfo
        ' 查找列 A 中最后使用的单元格
        Set rngBottom = .Range("A" & .Rows.Count).End(xlUp)
    End With

    ' 卸载原来的用户窗体
    If lngRow <> Target.Row Then
        Unload ufmUser
    End If

    ' 当前单元格位于第 1 列且具有数据时才显示用户窗体
    If Target.Column = 1 And Target.Row > 1 And Target.Row <=
rngBottom.Row Then
        ufmUser.Show 0 ' 无模式窗体
    End If

    ' 填充用户窗体
    ' 在工作表单元格发生改变后能实时反映到用户窗体中
    ufmUser.PopulateRecord

    ' 更新用户窗体显示
    ' 当活动单元格变化后, 显示该单元格所在行的内容
    lngRow = Target.Row
End Sub
```


注意到，事件过程含有参数，允许将相应的值通过参数传递给事件过程。例如本例中的参数 Target，将当前单元格传递到过程。

有时候，一个事件可能触发其他事件，包括该事件本身，从而带来连锁反应。例如，修改工作表单元格中的数据时，会触发 Worksheet_Change 事件，此时如果事件中的代码修改了另一个单元格中的数据，那么会再次触发该事件，再该事件会再次修改单元格，这引起事件的又一次触发，如此反复循环。这往往不是我们所需要的。

可以使用：

```
Application.EnableEvents = False
```

临时禁止触发事件，从而防止发生这类情形。当达到目的后，再重新启用触发事件：

```
Application.EnableEvents = True
```

例如：

```
Private Sub Worksheet_Change(ByVal Target As Range)
    Application.EnableEvents = False
    If Target.Column = 1 Then
        Range("C1") = Target.Value
    End If
    Application.EnableEvents = True
End Sub
```

Excel 定义了与工作簿对象、工作表对象等相关的一系列事件，还可以使用 VBA 自定义事件。一般来说，常使用的 Excel 事件有：

- Excel 应用程序事件，即根据 Excel 本身的操作来响应的事件，例如响应新建工作簿的 NewWorkbook 事件。
- 工作簿事件，即根据操作工作簿的动作来响应的事件，例如打开工作簿的 Open 事件。
- 工作表事件，即根据操作工作表的动作来响应的事件，例如选取工作表单元格变化时的 SelectionChange 事件。
- 图表事件，即操作图表时响应的事件，例如激活图表时的 Activate 事件。

后面的内容将主要讲解工作表事件，其他事件将在后续讲到相应内容时再详细讲解。

14. 看看工作表会自动响应哪些操作—— 认识工作表事件

我们可以设置在工作表上进行操作时，工作表要做的事情，例如激活某工作表时弹出一个对话框、在单元格之间移动时高亮显示单元格所在的行列，等等。

这就要用到 Worksheet 对象的事件。

Worksheet 对象的事件并不多，共 9 个，如图 14.1 所示。

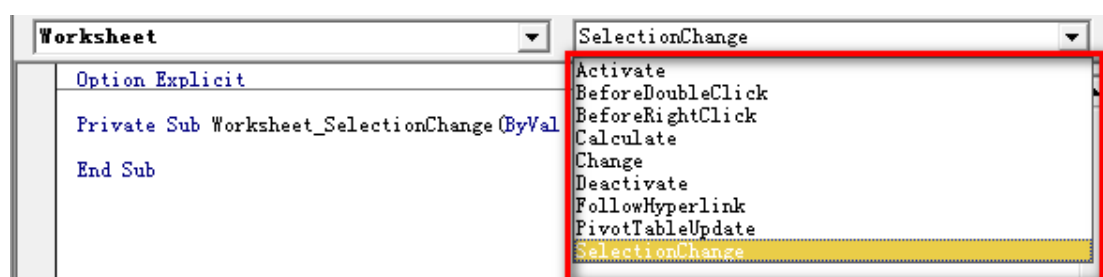


图 14.1

其中，各个事件发生条件为：

- Activate 事件发生在工作表成为当前活动工作表时
- BeforeDoubleClick 事件发生在工作表单元格中双击时发生且在默认的双击操作之前
- BeforeRightClick 事件发生在工作表单元格中右击时发生且在默认的右击操作之前
- Calculate 事件发生在重新计算工作表后

- Change 事件发生在工作表单元格被修改后
- Deactivate 事件发生在转移到并使其他工作表为活动工作表前
- FollowHyperlink 事件发生在单击工作表中的超链接时
- PivotTableUpdate 事件发生在更新工作表中数据透视表后
- SelectionChange 事件发生在改变工作表单元格选择时

下面分别详细介绍这些事件。

Activate 事件

Worksheet_Activate()

在工作表成为活动工作表时触发该事件。

Deactivate 事件

Worksheet_Deactivate()

当转移到其他工作表时触发该事件。

示例 1：激活当前工作表与转移到其他工作表时的事件响应

以工作表 Sheet2 为例，当激活使工作表 Sheet2 成为活动工作表时，以及转移到其他工作表时，分别显示相应的消息框。

```
Private Sub Worksheet_Activate()  
    MsgBox "Hi!欢迎来到【完美 Excel】.", , "excelperfect"  
End Sub
```

```
Private Sub Worksheet_Deactivate()  
    MsgBox "谢谢你的来访!", , "完美 Excel"  
End Sub
```

代码效果如图 14.2 所示。

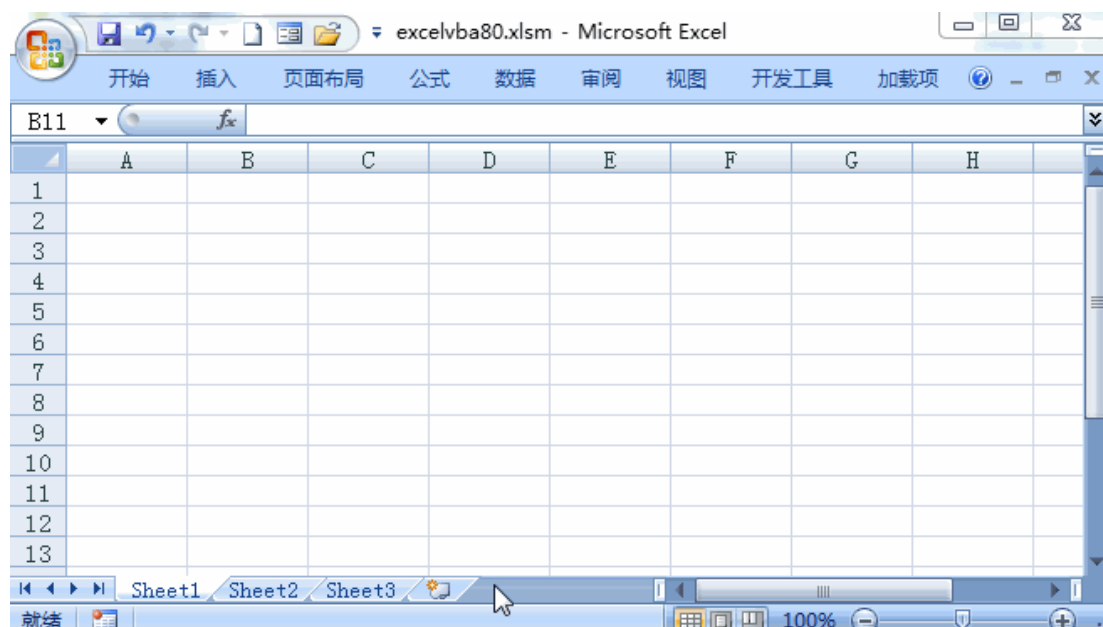


图 14.2（演示视频截图，详细视频见[完美 Excel](#) 微信公众号）

BeforeDoubleClick 事件

```
Worksheet_BeforeDoubleClick (ByVal Target As Range, Cancel As Boolean)
```

当双击工作表单元格时触发该事件。

参数 Target 代表工作表中所选取的单元格或单元格区域。

参数 Cancel 设置是否允许默认的操作，缺省值为 False。若设置为 True，则不允许按照双击单元格方式来输入数据。

示例 2：双击单元格自动添加背景色

在用户双击工作表单元格区域 A1:C3 中的任意单元格时，会自动为该单元格添加红色作为背景色。代码如下：

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As Range, Cancel As Boolean)

    If Not Intersect(Target, Range("A1:C3")) Is Nothing Then

        Cancel = True

        Target.Interior.Color = vbRed

    End If

End Sub
```

代码效果如图 14.3 所示。

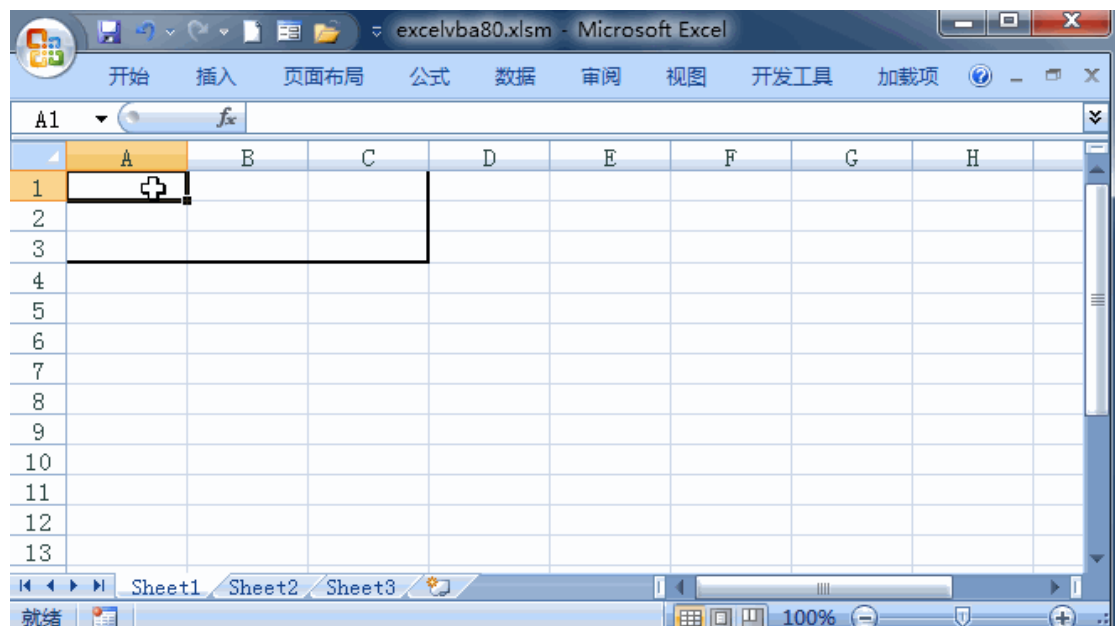


图 14.3（演示视频截图，详细视频见完美 Excel 微信公众号）

BeforeRightClick 事件

Worksheet_BeforeRightClick (ByVal Target As Range, Cancel As Boolean)

在单元格或单元格区域中单击右键时触发该事件。

参数 Target 代表工作表中所选取的单元格或单元格区域。

参数 Cancel 设置是否允许默认的操作，缺省值为 False。若设置为 True，则不显示默认的快捷菜单。

示例 3：阻止显示缺省的快捷菜单

当工作表单元格中的数据为“”时，选择该单元格后单击右键，不会出现缺省的快捷菜单。代码如下：

```
Private Sub Worksheet_BeforeRightClick (ByVal Target As Range,
Cancel As Boolean)

    If Target.Value = "完美 Excel" Then
        Cancel = True
    End If
End Sub
```

代码效果如图 14.4 所示。

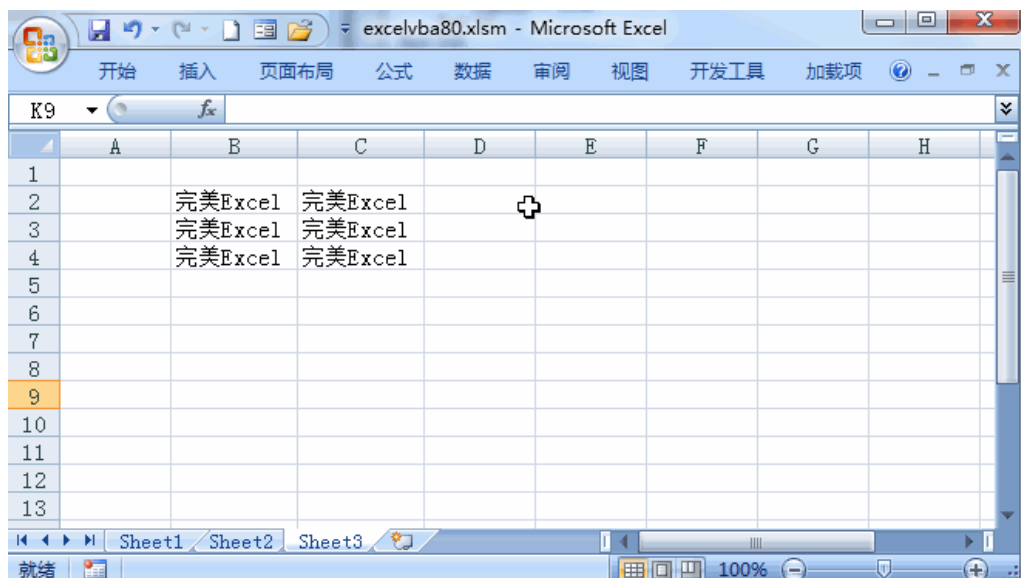


图 14.4（演示视频截图，详细视频见[完美 Excel](#) 微信公众号）

Calculate 事件

Worksheet_Calculate()

当工作表被重新计算时触发该事件。

示例 4：根据计算结果输入数值并设置格式

下面的代码演示当工作表单元格 B2 中的数值大于 B3 中的数值时，在 B4 中显示两个值的差并设置该单元格的背景色为红色。

```
Private Sub Worksheet_Calculate()  
    If Range("B2").Value > Range("B3") Then  
        Range("B4").Value = Range("B2").Value - Range("B3")  
        Range("B4").Interior.Color = vbRed  
    Else  
        Range("B4").Clear  
    End If  
End Sub
```

运行效果如图 14.5 所示。

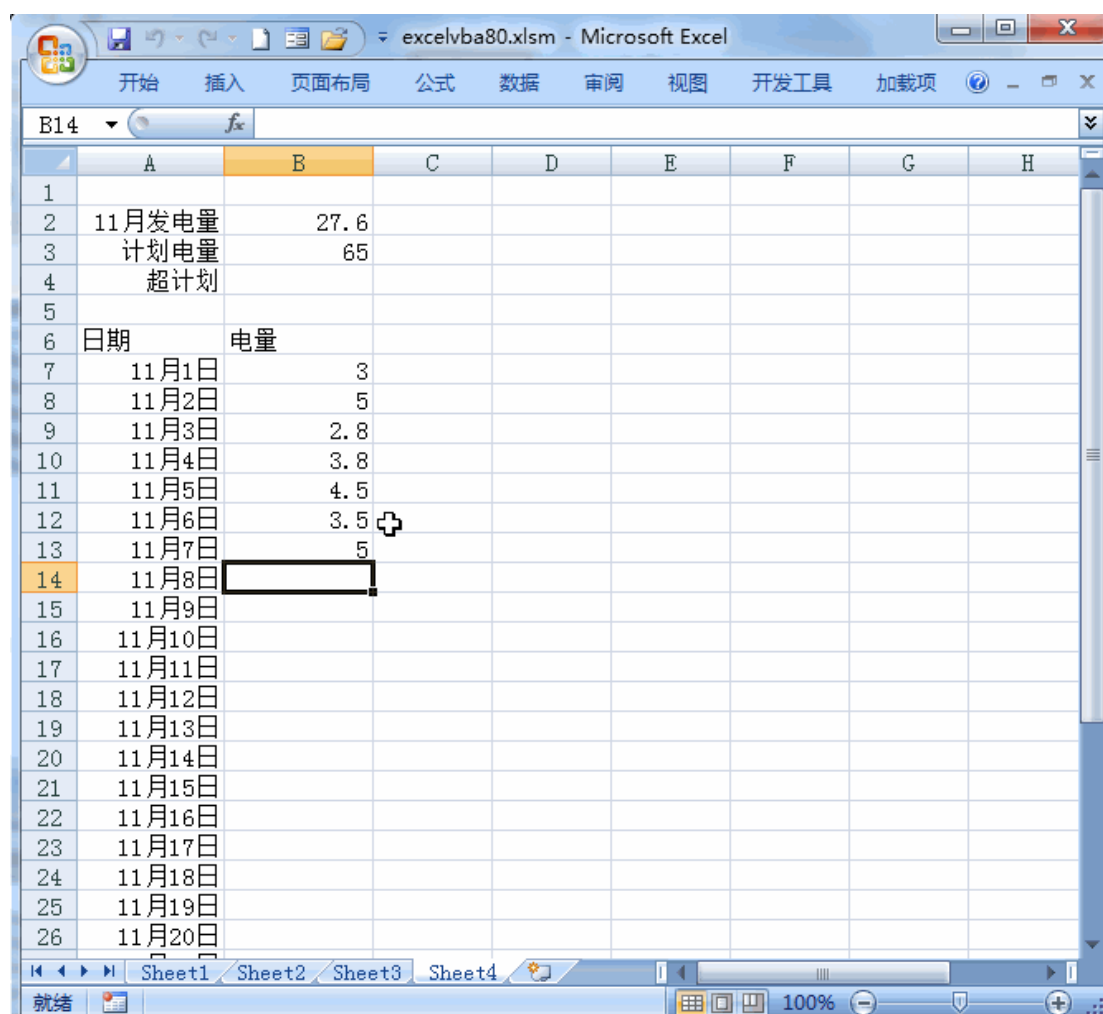


图 14.5（演示视频截图，详细视频见完美 Excel 微信公众号）

Change 事件

Worksheet_Change (ByVal Target As Range)

当修改单元格中的数据时触发该事件。

参数 Target 代表工作表中所选取的单元格或单元格区域。

示例 5：提示用户不要修改数据

下面的代码在用户修改工作表单元格区域 A1:C3 中的数据时，给出提示信息。

```
Private Sub Worksheet_Change(ByVal Target As Range)
    If Not Intersect(Target, Range("A1:C3")) Is Nothing Then
        MsgBox "该区域数据重要, 请不要修改!"
    End If
End Sub
```

运行效果如图 14.6 所示。

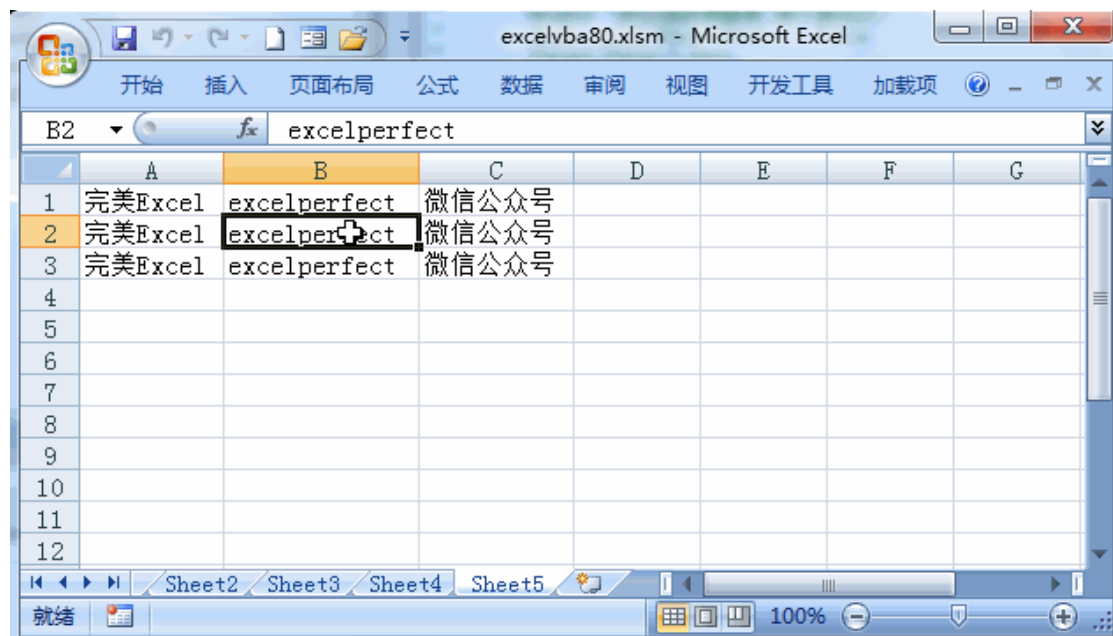


图 14.6（演示视频截图，详细视频见[完美 Excel 微信公众号](#)）

FollowHyperlink 事件

Worksheet_FollowHyperlink (ByVal Target As Hyperlink)

当单击含有超链接的单元格时触发该事件。

PivotTableUpdate 事件

Worksheet_PivotTableUpdate (ByVal Target As PivotTable)

当更新数据透视表时触发该事件。

SelectionChange 事件

Worksheet_SelectionChange (ByVal Target As Range)

当工作表中选择的单元格改变时触发该事件。

参数 Target 代表工作表中所选取的单元格或单元格区域。

示例 6：高亮显示单元格所在的行列

下面的代码高亮显示工作表单元格所在的行列：

```
Private Sub Worksheet_SelectionChange (ByVal Target As Range)
    Cells.Interior.ColorIndex = xlNone
    With Target
        .EntireRow.Interior.Color = vbRed
        .EntireColumn.Interior.Color = vbRed
    End With
End Sub
```

运行效果如图 14.7 所示。

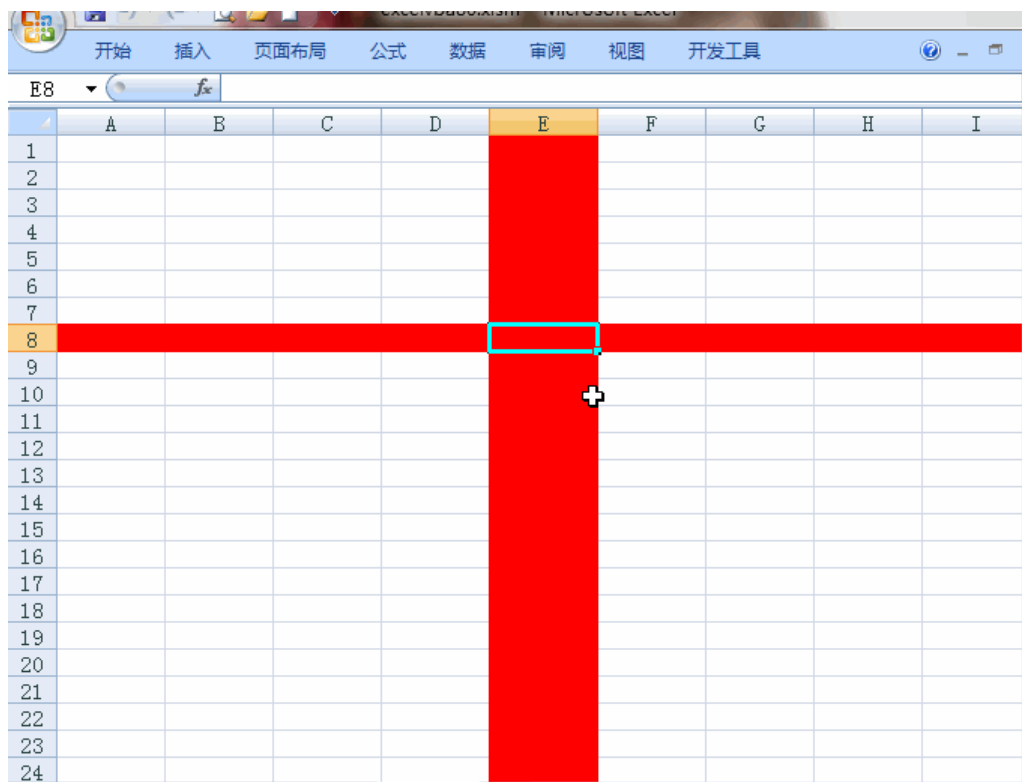


图 14.7（演示视频截图，详细视频见[完美 Excel 微信公众号](#)）

本章内容 2017 年 12 月 9 日首发于
[完美 Excel]微信公众号 [excelperfect](#)
原标题为
[Excel VBA 解读（81）：工作表事件示例](#)

15. 工作表事件示例

本章再列举一些示例，以加深对工作表事件的理解，方便应用。

示例 1：阻止用户修改指定单元格区域的数据

当用户修改工作表中指定单元格区域的数据时，给出提示信息并改回为原数据。

代码如下：

```
Dim data

Private Sub Worksheet_Change(ByVal Target As Range)
    If Not Intersect(Target, Range("A1:C3")) Is Nothing Then
        MsgBox "该区域数据重要, 请不要修改!"
        Application.EnableEvents = False
        Target.Value = data
        Application.EnableEvents = True
    End If
End Sub

Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    data = Target.Value
End Sub
```

运行效果如图 15.1 所示。

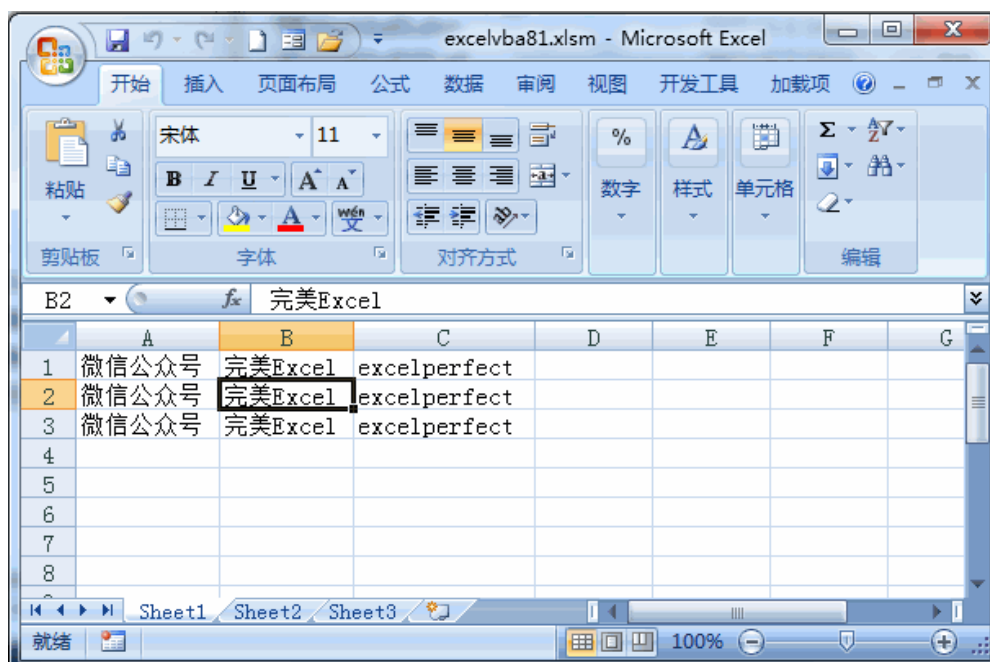


图 15.1（演示视频截图，详细视频见[完美 Excel 微信公众号](#)）

示例 2：让不同的工作表有不同的快捷菜单

在工作表“完美 Excel”中，单击右键会出现属于该工作表的快捷菜单，同样，在工作表“Data”中单击右键也会出现属于该工作表的快捷菜单。如下图 15.2 所示。

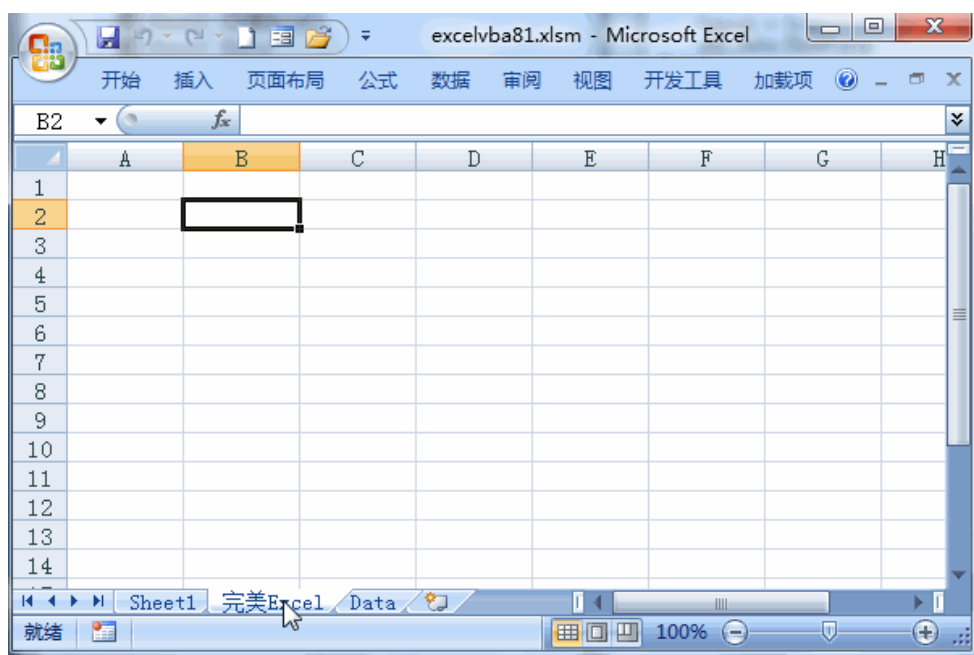


图 15.2（演示视频截图，详细视频见[完美 Excel 微信公众号](#)）

实现上述效果的代码如下。

在“完美 Excel”工作表代码模块的 BeforeRightClick 事件中，输入下面的代码：

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range,
Cancel As Boolean)

    Dim cmb_excelperfect As CommandBar

    Set cmb_excelperfect = CreateSubMenu("完美 Excel")

    Cancel = True

    cmb_excelperfect.ShowPopup

End Sub
```

在“Data”工作表代码模块的 BeforeRightClick 事件中，输入相似的代码：

```
Private Sub Worksheet_BeforeRightClick(ByVal Target As Range,
Cancel As Boolean)

    Dim cmb_data As CommandBar

    Set cmb_data = CreateSubMenu("数据处理")

    Cancel = True

    cmb_data.ShowPopup

End Sub
```

插入一个标准模块，并输入下面的代码：

```

Function CreateSubMenu(strCMB) As CommandBar
    Const CMBPREFIX = "customPopUp"
    Dim cmb As CommandBar
    Dim cmbCtl As CommandBarControl
    Dim strCMBName As String

    '自定义菜单名称
    strCMBName = CMBPREFIX & strCMB

    '删除以前的实例
    Call DeleteCommandBar(strCMBName)

    '添加快捷菜单
    Set cmb = CommandBars.Add(Name:=strCMBName, _
        Position:=msoBarPopup, _
        MenuBar:=False, _
        Temporary:=False)

    '添加控件
    Set cmbCtl = cmb.Controls.Add
    With cmbCtl
        .Caption = strCMB & "控件1"
        .OnAction = "DummyMessage"
    End With

    Set cmbCtl = cmb.Controls.Add
    With cmbCtl
        .Caption = strCMB & "控件2"
        .OnAction = "DummyMessage"
    End With

    Set CreateSubMenu = cmb
    Set cmbCtl = Nothing
    Set cmb = Nothing
End Function

Sub DeleteCommandBar(cmbName)
    On Error Resume Next
    CommandBars(cmbName).Delete
End Sub

Sub DummyMessage()
    MsgBox CommandBars.ActionControl.Caption, vbInformation + vbOKOnly, "示例消息"
End Sub

```

示例 3：双击单元格时显示输入框

在双击工作表第 1 列中的单元格时，会显示下图 15.3 所示的自定义输入框。

图 15.3

在其中输入相应的数据后，单击“输入”即将数据输入到双击的单元格所在行的第 1 至 3 列所在的单元格。如图 15.4 所示。

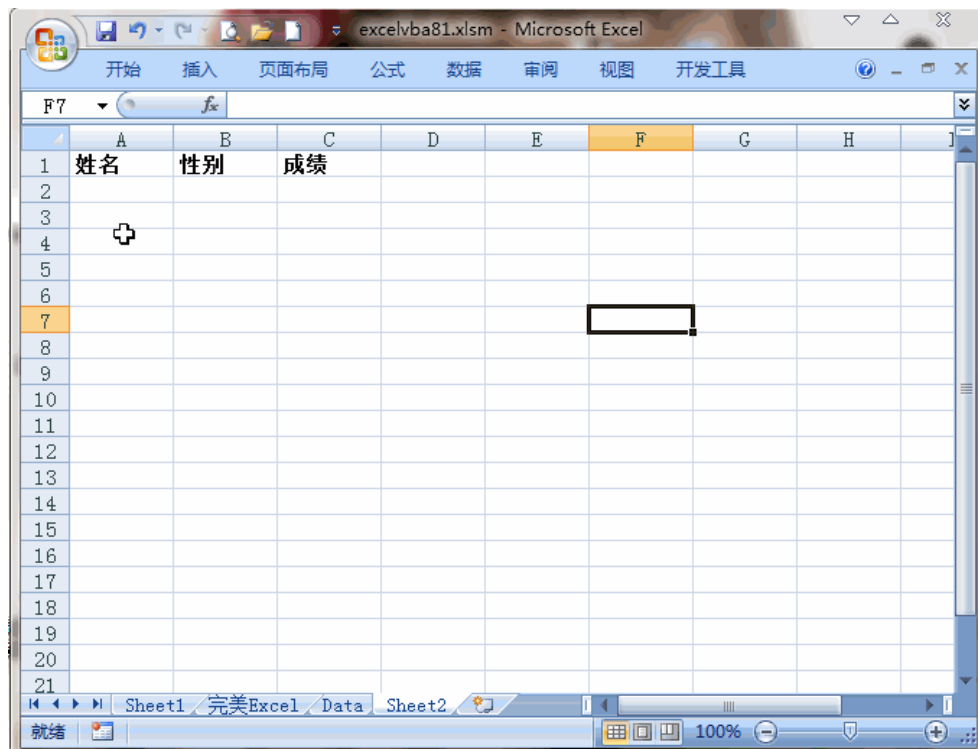


图 15.4（演示视频截图，详细视频见完美 Excel 微信公众号）

实现上述效果的代码如下。

插入一个标准模块，并声明一个全局变量。

```
Global lngRow As Long
```

创建一个如图 3 所示的用户窗体，窗体名及相应的控件名如图 3 所示。代码如下：

```
Private Sub cmdEntry_Click()  
    Cells(lngRow, 1).Value = txtXM  
    Cells(lngRow, 2).Value = txtXB  
    Cells(lngRow, 3).Value = txtCJ  
End Sub
```

在工作表代码模块中，输入下面的代码：

```
Private Sub Worksheet_BeforeDoubleClick(ByVal Target As  
Range, Cancel As Boolean)  
    If Target.Column = 1 Then
```

```

Cancel = True

lngRow = Target.Row

ufmTest.Show

End If

End Sub

```

示例 4：根据比较结果自动显示上升或下降的箭头

这是 Bill Jelen 提供的一个例子。如图 15.5 所示，比较两年同一个月份的利润。若利润上升，将在月份下面显示向上的红色箭头，如果利润下降，则在月份下面显示向下的绿色箭头。

	A	B	C	D
1				
2	五月	今年	7809	
3	↑	去年	7000	
4				
5		2016	2017	
6	一月	1200	3123	
7	二月	2300	2111	
8	三月	4500	5200	
9	四月	6000	5698	
10	五月	7000	7809	
11	六月	6500	5367	
12	七月	5300	6528	
13	八月	2308	3000	
14	九月	3398	3232	
15	十月	3500	4687	
16				

图 15.5

使用了工作表的 Calculate 事件，代码如下：

```

Private Sub Worksheet_Calculate()
    Select Case Range("C2").Value
        Case Is > Range("C3").Value
            SetArrow 10, msoShapeUpArrow
        Case Is < Range("C3").Value
            SetArrow 3, msoShapeDownArrow
    End Select
End Sub

```

```

Private Sub SetArrow(ByVal ArrowColor As Integer, ByVal ArrowDegree)
    ActiveSheet.Shapes.AddShape(ArrowDegree, 17.25, 30.5, 5, 10).Select
    With Selection.ShapeRange
        With .Fill
            .Visible = msoTrue
            .Solid
            .ForeColor.SchemeColor = ArrowColor
            .Transparency = 0#
        End With
        With .Line
            .Weight = 0.75
            .DashStyle = msoLineSolid
            .Style = msoLineSingle
            .Transparency = 0#
            .Visible = msoTrue
            .ForeColor.SchemeColor = 64
            .BackColor.RGB = RGB(255, 255, 255)
        End With
    End With
End Sub

```

在我们前面的章节中，也介绍过一些工作表事件的示例，你可以参照学习理解。

工作表事件为 Excel 自动化打开了一扇明亮的窗，有很多有用的应用，在[完美 Excel](#) 微信公众号的文章中会穿插讲解。

关于完美 Excel

完美 Excel 是我创建的一个微信公众号，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 300 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《玩转 Excel 数据有效性》系列文章整理而成的。

每天早晨，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 为完美 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“VBA 学习经验”、“玩转数据验证”、“快速学会 30 个函数”、“全部文章合集 1”等 4 个子菜单；菜单“VBA 精选”中设置有“最最基础入门”、“Range 对象详解”、“工作表对象详解”等 3 个子菜单；菜单“联系 me”中设置有“知识分享架构”、“个人声明”、“答疑解惑”、“坚持的美好”、“爱沐智养亲子中心”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。



图 5

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。

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。