完美Excel技术系列(8)

Excel VBA 解读 • Application 对象篇

完美 Excel 出品
微信公众号：*excelperfect*

内容提要

Application 对象位于 Excel 对象模型层次结构的最顶端，在 Excel 中的其他对象都源自于 **Application 对象**，**Application 对象**的许多属性、方法和事件可以用来设置或者控制 Excel 应用程序、可以获取计算机配置信息，其事件是应用程序级别的，会影响到 Excel 会话中所有打开的工作簿，实现应用程序内的自动或交互功能。

本电子书《**Excel VBA 解读：Application 对象篇**》详细讲解了 **Application 对象**的常用属性、方法，以及 **Application 对象**事件的使用，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

Application 对象位于 Excel 对象模型层次结构的最顶端，在 Excel 中的其他对象都源自于 Application 对象，Application 对象的许多属性、方法和事件可以用来设置或者控制 Excel 应用程序、可以获取计算机配置信息，其事件是应用程序级别的，会影响到 Excel 会话中所有打开的工作簿，实现应用程序内的自动或交互功能。

本书详细讲解了 Application 对象的常用属性、方法，并介绍了 Application 对象事件的使用，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.Application 对象揭秘

概要介绍了 Application 对象的一些属性、方法以及事件。

2.处理屏幕刷新——ScreenUpdating 属性

Application 对象的 ScreenUpdating 属性设置屏幕刷新，可以设置或者获取该属性的值。文中的示例：关闭屏幕更新时运行时间对比。

3.屏蔽警告框——DisplayAlerts 属性

Application 对象的 DisplayAlerts 属性可用来屏幕代码操作使 Excel 弹出的大多数警告框。文中的示例：关闭而不保存工作簿的修改。

4.控制状态栏——DisplayStatusBar 属性和 StatusBar 属性

使用 Application 对象的 DisplayStatusBar 属性可以控制 Excel 的状态

栏的显示，包括显示或者关闭状态栏。而 StatusBar 属性则可以设置或者返回在状态栏中显示的内容。文中的示例：①在状态栏中显示正在处理的行号；②在状态栏中显示进度。

5.获取文件名——GetOpenFilename 方法

使用 Application 对象的 GetOpenFilename 方法，可以显示标准的“打开”对话框。用户可以从该对话框中获取文件名，但是不会真正打开文件。文中的示例：①显示打开对话框和选择的文件名；②获取单个或者多个工作簿文件名；③列出供选择的多个文件类型并允许选择多个文件。

6.指定文件名——GetSaveAsFilename 方法

使用 Application 对象的 GetSaveAsFilename 方法，可以显示标准的“另存为”对话框。用户可以在该对话框中指定或者选择文件名，但是不会真正保存文件。文中的示例：①显示另存为对话框并指定文件名；②指定并显示文件名；③拆分指定的文件名。

7.打开文件对话框——FileDialog 属性

Application 对象的 FileDialog 属性，返回一个代表文件对话框实例的 FileDialog 对象。文中的示例：①显示打开文件对话框并显示文件名；②指定默认文件路径；③显示文件选取对话框并显示选择的文件。

8.获取用户输入——InputBox 方法

讲解用于获取用户输入信息的 InputBox 方法。示例：①获取用户输入的数据；②求所选择的单元格区域中的数值之积。

9.让程序在指定的时间运行——OnTime 方法

Application 对象的 OnTime 方法可以实现在指定的时间运行某个程序。①设置闹钟；②定时刷新数据。

10.自定义快捷键运行宏——OnKey 方法

Application 对象的 OnKey 方法可以将程序赋给单个键或键组合，通过按下

赋给的键或键组合来运行该程序。同时，也可以使用这个方法来禁用组合键。文中的示例：①禁用已有的快捷键；②自定义快捷键运行特定的操作。

11.向应用程序发送键盘命令——SendKeys 方法

SendKeys 方法允许发送按键到当前应用程序窗口。文中的示例：①打开计算器并设置计算模式；②清除立即窗口中的内容。

12.在代码中使用工作表函数——WorksheetFunction 属性

在 VBA 代码中，使用 WorksheetFunction 属性，将允许调用工作表函数。文中的示例：①替换单元格中指定的字符；②查找数据列中的最大值；③显示所选单元格区域的汇总信息；④清理数据。

13.设置或获取 Excel 窗口大小的属性

使用 Application 对象的 WindowState 属性，可以设置或者返回 Excel 窗口的状态；使用 Application 对象的 UsableHeight 属性和 UsableWidth 属性，返回工作簿窗口可以在 Excel 应用程序窗口区中占据空间的最大高度和宽度，以磅为单位；Application 对象的 Width 属性返回或设置一个 Double 类型的值，代表从 Excel 应用程序窗口左边至其右边的距离，以磅为单位；Application 对象的 Height 属性返回或设置一个 Double 类型的值，代表 Excel 应用程序窗口的高度，以磅为单位。文中的示例：获取窗口状态信息。

14.在 Excel 中使用 Application 事件

主要讲解如何使用类模块来钩挂 Application 对象的事件。Application 对象事件将会影响 Excel 会话中所有打开的工作簿。

15.Application 对象常用事件概览

列出了 Application 对象的一系列常用事件及说明，供需要时查阅。

16.Application 对象事件示例

列举了 3 个 Application 对象示例：①总是将 Excel 中新建工作表放到最后；②在打印的工作表页脚放置指定的文本；③强迫用户另存为启用宏的工作簿。

温馨提示：

在完美 Excel 微信公众号中发送消息：**Excel 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 2018 年 6 月 9 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（101）：Application 对象揭秘

1 . Application 对象揭秘

在《**Excel VBA 解读：基础入门篇**》中，我们简单介绍了 Excel 对象模型。如图 1.1 所示，处于 Excel 对象模型顶端的就是 Application 对象，代表 Excel 应用程序本身。

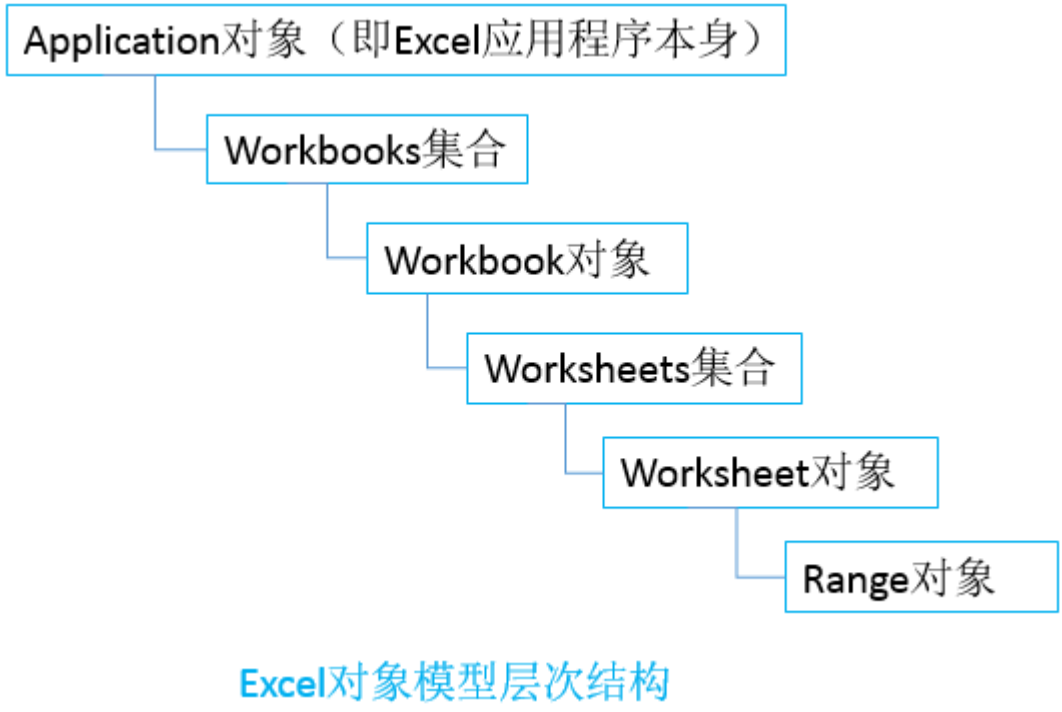


图 1.1

在 Excel VBA 中操作的对象都源于 Application 对象。虽然我们在引用对象时没有明确指定 Application 对象，但是很多情形下默认限定就是 Application 对象。

Application 对象有许多属性、方法和事件，用来设置或控制 Excel 应用程序、获取计算机配置信息，等等。

Application 对象提供了可以快捷访问一些 Excel 对象的属性，例如：

- ActiveCell 代表当前单元格
- Cells 属性代表当前工作表中的所有单元格
- Selection 代表当前工作表中所选择的区域
- Rows 属性代表当前工作表的所有行
- Columns 属性代表当前工作表的所有列
- Names 属性代表当前工作簿中所有名称的集合
- ActiveChart 属性代表当前活动图表
- ActiveSheet 属性代表当前工作表
- ActiveWorkbook 属性代表当前工作簿
- ThisWorkbook 属性代表当前代码所在的工作簿
- Sheets 属性代表工作簿中的工作表集合
- Worksheets 属性代表当前工作簿中的工作表集合
- Charts 属性代表工作簿中所有图表工作表的集合
- Workbooks 属性代表当前打开的所有工作簿的集合

Application 对象提供了很多设置显示的属性，例如：

- ScreenUpdating 属性用来控制屏幕更新
- StatusBar 属性用来获取或者设置状态栏中的文本
- Cursor 属性返回或者设置光标形状
- Height 属性返回或者设置 Excel 应用程序的高度
- Width 属性返回或者设置 Excel 应用程序的宽度
- UsableHeight 属性返回窗口在主应用程序窗口中占有的空间的最大高度
- UsableWidth 属性返回窗口在主应用程序窗口中占有的空间的最大宽度

使用 Application 对象的一些属性还可以获取计算机信息，例如：

- OperatingSystem 属性返回当前操作系统的名称和版本号
- OrganizationName 属性获取注册的组织名称
- UserName 属性返回或者设置当前用户名

- Version 属性返回当前使用的 Excel 版本
- MemoryUsed 属性返回 Excel 当前使用的内存数

Application 对象还提供了获取文件名或者指定要保存文件的名字的方法：

- GetOpenFilename 方法显示“打开文件”对话框，让用户选取文件名
- GetSaveAsFilename 方法显示“另存为”对话框，让用户指定文件名及路径

Application 对象还提供了很多有用的方法，例如我们在前面的系列中介绍过的 Evaluate 方法、Union 方法、Intersect 方法，还有后面的系列文章中将要介绍的 OnTime 方法、OnKey 方法、SendKeys 方法、InputBox 方法，等等。

此外，Application 对象的事件是应用程序级别的，会影响到 Excel 会话中所有打开的工作簿，可以实现很多自动或交互的功能。

接下来的章节中，我们将给大家详细讲解 Application 对象的很多有用的属性、方法和事件。

本章内容 2018 年 7 月 3 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（102）：处理屏幕刷新——ScreenUpdating 属性

2.处理屏幕刷新—— ScreenUpdating 属性

Application 对象的 ScreenUpdating 属性设置屏幕刷新，可以设置或者获取该属性的值。默认情形下，ScreenUpdating 属性的值为 True，这意味着 VBA 代码在执行一些操作时，Excel 屏幕界面会不断更新甚至闪烁，这会带给用户一些不好的体验，代码也会运行得更慢。

此时，如果将 ScreenUpdating 属性设置为 False，在代码运行时关闭屏幕更新，不仅避免运行代码时带来的屏幕闪烁，而且代码运行得更快。

代码：

```
Application.ScreenUpdating = False
```

关闭屏幕刷新。

在代码执行完毕前，恢复屏幕刷新：

```
Application.ScreenUpdating = True
```

然而，如果在代码运行过程中，需要显示用户窗体或者内置的对话框时，应先恢复屏幕刷新：

```
Application.ScreenUpdating = True
```

否则，在拖动用户窗体或对话框时，会在屏幕上产生橡皮擦的效果。在显示完用户窗体或对话框后，再重新关闭屏幕刷新。

下面的代码演示关闭屏幕更新能够使代码运行更快。代码跟踪记录隐藏工作表 Sheet1 中全部偶数列的时间，第一次是开启屏幕刷新时的用时，第二次是关闭屏幕刷新时的用时。

```
Sub testScreenUpdating()
    Dim elapsedTime(2), i, c, startTime, stopTime
    Application.ScreenUpdating = True
    For i = 1 To 2
        If i = 2 Then Application.ScreenUpdating = False
        startTime = Time
        Worksheets("Sheet1").Activate
        For Each c In ActiveSheet.Columns
            If c.Column Mod 2 = 0 Then
                c.Hidden = True
            End If
        Next c
        stopTime = Time
        elapsedTime(i) = (stopTime - startTime) * 24 * 60 *
60
    Next i
    Application.ScreenUpdating = True
    MsgBox "屏幕更新开启时,共用时: " & elapsedTime(1) & " 秒."
_
    & Chr(13) & "屏幕更新关闭时,共用时: " & elapsedTime(2) &
" 秒."
End Sub
```

图 2.1 显示了代码运行时的效果。

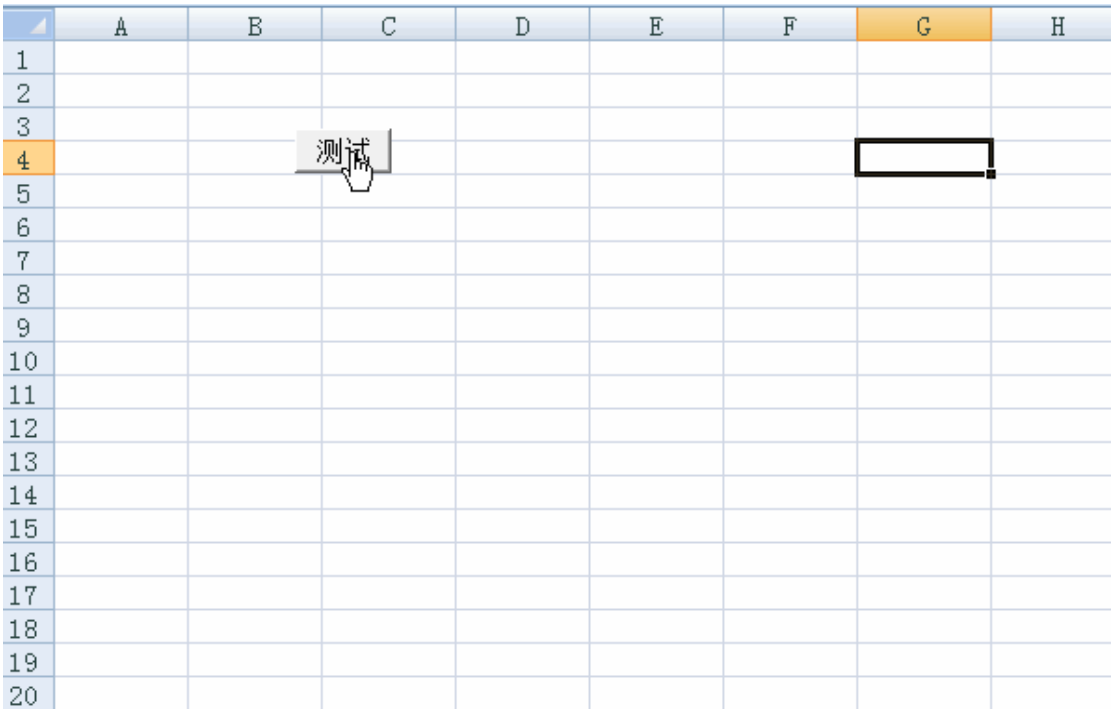


图 2.1（演示视频截图，详细视频见完美 Excel 微信公众号）

代码运行完后，会显示开启和关闭屏幕更新所用时间的对比，如图 2.2 所示。很明显，关闭屏幕刷新时，代码运行速度大大加快，且没有屏幕闪烁现象。

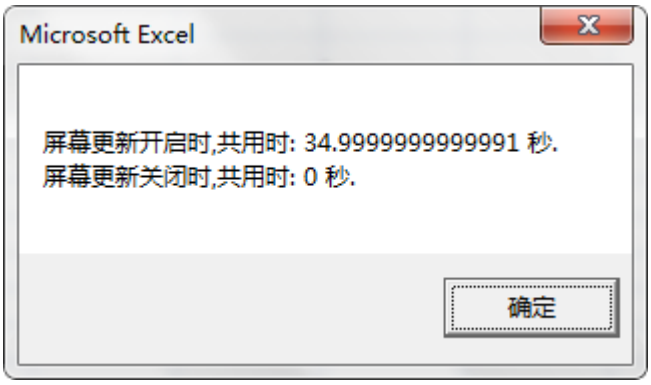
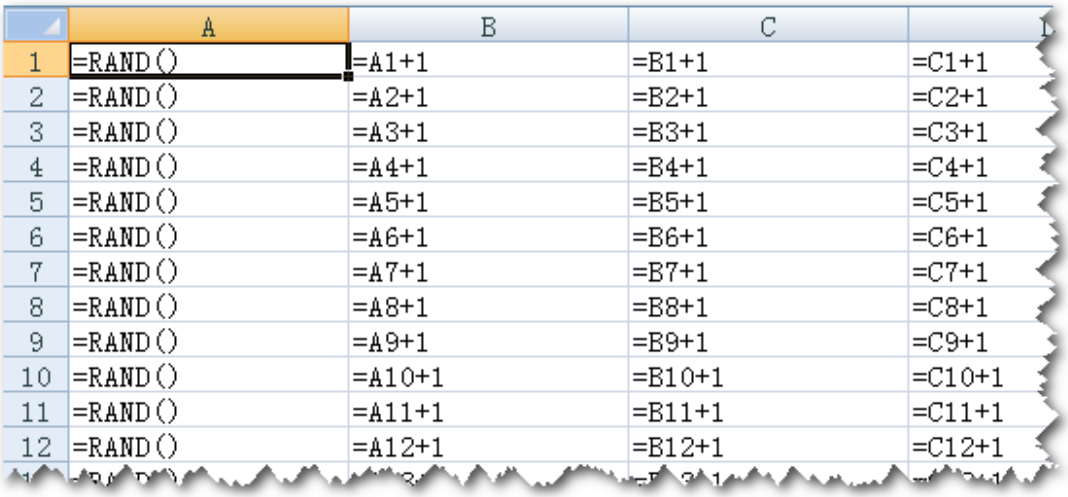


图 2.2

再列举一个测试用例，如图 2.3 所示，在列 A 中放置 28 个公式=RAND()，在列 B 至列 V 中放置简单的公式，依次比前一列中相应单元格值加 1。



	A	B	C	D
1	=RAND()	=A1+1	=B1+1	=C1+1
2	=RAND()	=A2+1	=B2+1	=C2+1
3	=RAND()	=A3+1	=B3+1	=C3+1
4	=RAND()	=A4+1	=B4+1	=C4+1
5	=RAND()	=A5+1	=B5+1	=C5+1
6	=RAND()	=A6+1	=B6+1	=C6+1
7	=RAND()	=A7+1	=B7+1	=C7+1
8	=RAND()	=A8+1	=B8+1	=C8+1
9	=RAND()	=A9+1	=B9+1	=C9+1
10	=RAND()	=A10+1	=B10+1	=C10+1
11	=RAND()	=A11+1	=B11+1	=C11+1
12	=RAND()	=A12+1	=B12+1	=C12+1

图 2.3

这将有 616 个单元格要在每次计算时都会发生变化。下面的代码将测试屏幕更新开或关时执行 10000 次 616 个单元格的计算所花的时间。

```
Sub testScreenUpdating2()
    Dim i As Long
    Dim tStart, tEnd, tScreenOn, tScreenOff

    With Application
        .Calculation = xlCalculationManual
        .ScreenUpdating = False
        tStart = Time
        For i = 1 To 10000
            .Calculate
        Next i
        tEnd = Time
        tScreenOff = (tEnd - tStart) * 24 * 60 * 60
        .ScreenUpdating = True
        tStart = Time
        For i = 1 To 10000
            .Calculate
        Next i
        tEnd = Time
```

```
        tScreenOn = (tEnd - tStart) * 24 * 60 * 60
    End With


    MsgBox "屏幕更新关闭时,共用时: " & tScreenOff & " 秒." & _
           vbCrLf & "屏幕更新打开时,共用时: " & tScreenOn & "
秒."
End Sub
```

运行上述代码时的效果如图 2.4 所示。

	A	B	C	D	E	F	G	H	I	J
1	0.64222	1.64222	2.64222	3.64222	4.64222	5.64222	6.64222	7.64222	8.64222	9.64222
2	0.461388	1.461388	2.461388	3.461388	4.461388	5.461388	6.461388	7.461388	8.461388	9.461388
3	0.640483	1.640483	2.640483	3.640483	4.640483	5.640483	6.640483	7.640483	8.640483	9.640483
4	0.141522	1.141522	2.141522	3.141522	4.141522	5.141522	6.141522	7.141522	8.141522	9.141522
5	0.256717	1.256717	2.256717	3.256717	4.256717	5.256717	6.256717	7.256717	8.256717	9.256717
6	0.441878	1.441878	2.441878	3.441878	4.441878	5.441878	6.441878	7.441878	8.441878	9.441878
7	0.265163	1.265163	2.265163	3.265163	4.265163	5.265163	6.265163	7.265163	8.265163	9.265163
8	0.427239	1.427239	2.427239	3.427239	4.427239	5.427239	6.427239	7.427239	8.427239	9.427239
9	0.124518	1.124518	2.124518	3.124518	4.124518	5.124518	6.124518	7.124518	8.124518	9.124518
10	0.071921	1.071921	2.071921	3.071921	4.071921	5.071921	6.071921	7.071921	8.071921	9.071921
11	0.173116	1.173116	2.173116	3.173116	4.173116	5.173116	6.173116	7.173116	8.173116	9.173116
12	0.321869	1.321869	2.321869	3.321869	4.321869	5.321869	6.321869	7.321869	8.321869	9.321869
13	0.182879	1.182879	2.182879	3.182879	4.182879	5.182879	6.182879	7.182879	8.182879	9.182879
14	0.170657	1.170657	2.170657	3.170657	4.170657	5.170657	6.170657	7.170657	8.170657	9.170657
15	0.119348	1.119348	2.119348	3.119348	4.119348	5.119348	6.119348	7.119348	8.119348	9.119348
16	0.480614	1.480614	2.480614	3.480614	4.480614	5.480614	6.480614	7.480614	8.480614	9.480614
17	0.520311	1.520311	2.520311	3.520311	4.520311	5.520311	6.520311	7.520311	8.520311	9.520311
18	0.494984	1.494984	2.494984	3.494984	4.494984	5.494984	6.494984	7.494984	8.494984	9.494984
19	0.848772	1.848772	2.848772	3.848772	4.848772	5.848772	6.848772	7.848772	8.848772	9.848772
20	0.053304	1.053304	2.053304	3.053304	4.053304	5.053304	6.053304	7.053304	8.053304	9.053304
21	0.592755	1.592755	2.592755	3.592755	4.592755	5.592755	6.592755	7.592755	8.592755	9.592755
22	0.103845	1.103845	2.103845	3.103845	4.103845	5.103845	6.103845	7.103845	8.103845	9.103845
23	0.65546	1.65546	2.65546	3.65546	4.65546	5.65546	6.65546	7.65546	8.65546	9.65546
24	0.827747	1.827747	2.827747	3.827747	4.827747	5.827747	6.827747	7.827747	8.827747	9.827747
25	0.01361	1.01361	2.01361	3.01361	4.01361	5.01361	6.01361	7.01361	8.01361	9.01361

图 2.4 (演示视频截图，详细视频见完美 Excel 微信公众号)

屏幕刷新关闭和开启用时如图 2.5 所示。

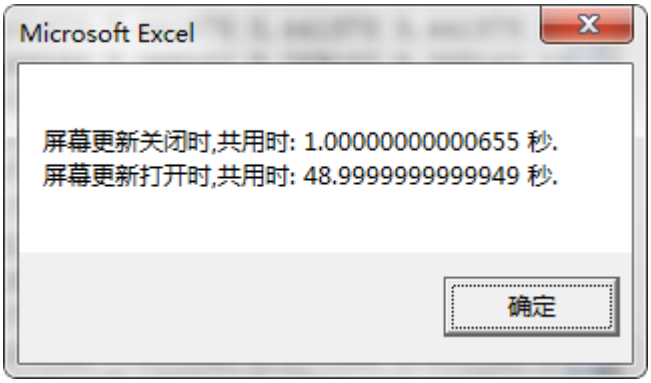


图 2.5

本章内容 2018 年 7 月 11 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（103）：屏蔽警告框——DisplayAlerts 属性

3.屏蔽警告框 — DisplayAlerts 属性

有时候，在执行代码的过程中，一些代码的操作会导致 Excel 给出警告框，此时，必须手动响应警告框，才能够继续执行代码。例如，如果程序中有一句代码删除某个工作表，Excel 将会弹出一个警告，如图 3.1 所示。

图 3.1

这种情况下，如果我们希望响应警告框的默认操作（例如图 3.1 中的“删除”按钮）而不中断代码的执行，那么可以使用 Application 对象的 DisplayAlerts 属性。

代码：

```
Application.DisplayAlerts = False
```

将自动屏蔽掉因代码操作使 Excel 弹出的大多数警告框，自动执行该对话框中默认按钮相关联的操作。

例如，代码：

```
Application.DisplayAlerts = False
ActiveSheet.Delete
```

将自动执行如图 3.1 中的“删除”按钮而不会显示该对话框。

DisplayAlerts 属性默认值是 True，即程序执行时，会显示一些代码触发

系统的警告框。一般，在需要屏蔽警告框的代码前将 DisplayAlerts 属性设置为 False，达到屏蔽目的后，再将其恢复为默认值 True。

示例：关闭而不保存工作簿的修改

下面的代码将关闭工作簿 ExcelVBA103，不会保存对其所作的任何修改。

```
Sub NotSavedChange()
    Application.DisplayAlerts = False
    Workbooks("ExcelVBA10301.xlsm").Close
    Application.DisplayAlerts = True
End Sub
```

本章内容 2018 年 7 月 23 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（104）：控制状态栏——DisplayStatusBar 属性和 StatusBar 属性

4.控制状态栏——DisplayStatusBar 属性和 StatusBar 属性

Excel 状态栏位于界面底部，可以在状态栏中显示当前信息，在对用户不造成干扰的前提下让用户能够实时了解相关信息。

使用 Application 对象的 DisplayStatusBar 属性可以控制 Excel 的状态栏的显示，包括显示或者关闭状态栏。而 StatusBar 属性则可以设置或者返回在状态栏中显示的内容。

如图 4.1 所示，在状态栏中显示自定义的信息。

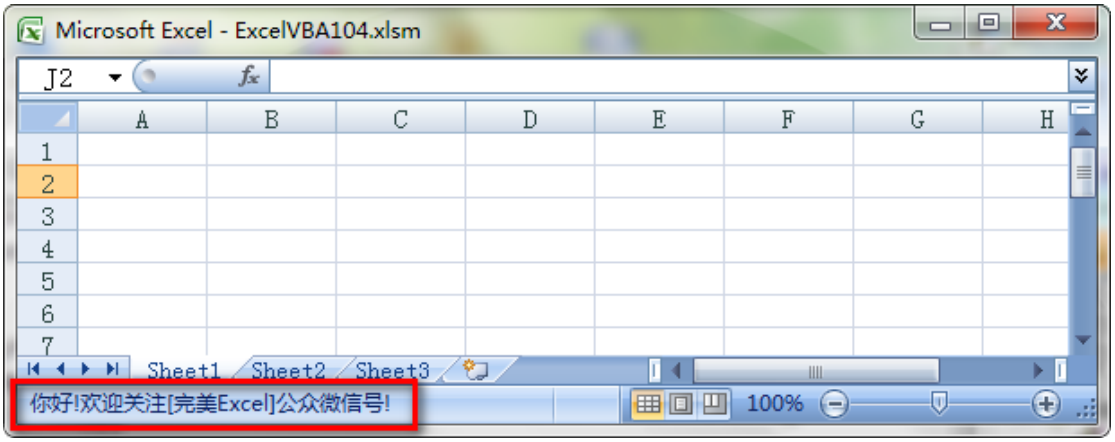


图 4.1

可以首先在程序开始处获取当前状态栏的设置：

```
blnStatusBar = Application.DisplayStatusBar
```

然后对状态栏进行设置：

```
Application.DisplayStatusBar = True
Application.StatusBar = "你好!欢迎关注[完美 Excel]公众微信号!"
```

在程序最后恢复初始的状态栏设置：

```
Application.StatusBar = False
Application.DisplayStatusBar = blnStatusBar
```

注意，要恢复状态栏中默认的内容，将 StatusBar 属性设置为 False，正如上面的代码所示。

示例 1：在状态栏中显示正在处理的行号

下面的代码在状态栏中显示代码正在处理的行号。

```
Sub DisplayNumINStatusBar()
    Dim lngNum As Long
    Dim TotalRows As Long

    TotalRows = ActiveSheet.Rows.Count
    For lngNum = 0 To TotalRows
        If lngNum Mod 100 = 0 Then
            Application.StatusBar = "正在处理第" & lngNum & "
行."
        End If
    Next lngNum
    Application.StatusBar = False
End Sub
```

运行代码后的结果如图 4.2 所示。

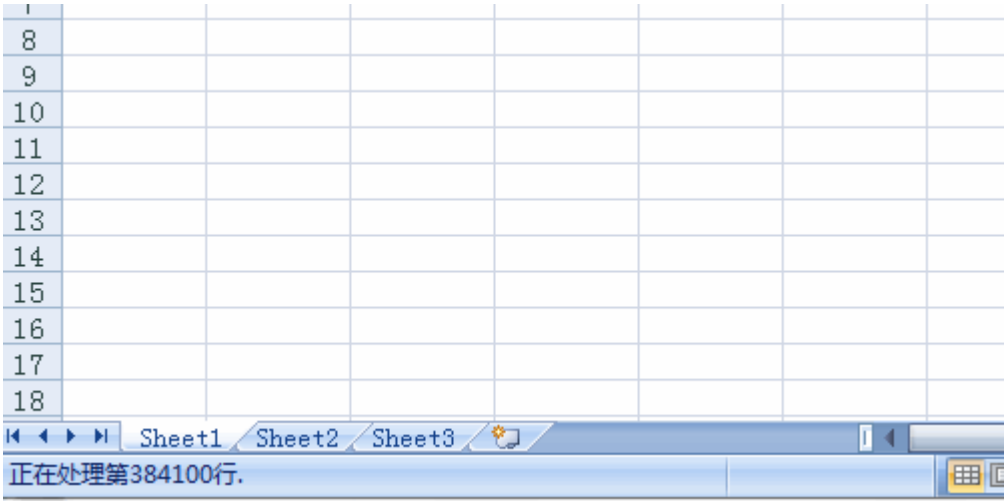


图 4.2 (演示视频截图，详细视频见完美 Excel 微信公众号)

示例 2：在状态栏中显示进度

下面的代码将在状态栏中以数字、进度条和百分比来展示程序处理的进度。

```
Sub StatusBarProgress()
    Const C_PROGRESS_CHAR = 9632 ' 进度条
    Const C_EMPTY_CHAR = 9633 ' "空条"

    Dim lngCount As Long
    Dim lngPercent As Long
    Dim lngInterval As Long
    Dim lngIterations As Long
    Dim strProgressBar As String
    Dim strMaxIterations As String

    lngInterval = 1
    lngIterations = 1000
    strMaxIterations = Format(lngIterations, "#,###")

    For lngCount = 1 To lngIterations
        If lngCount Mod lngInterval = 0 Then
```


```
            lngPercent = CLng(Round(lngCount /
lngIterations * 10, 0))

            strProgressBar = String(lngPercent,
ChrW(C_PROGRESS_CHAR)) & String(10 - lngPercent,
ChrW(C_EMPTY_CHAR))

            Application.StatusBar = "循环 " &
strMaxIterations & " 中的 " & Format(lngCount, "#,###") &
" " & _
                                    strProgressBar & " (" &
Format(lngPercent / 10, "0%") & ")"

            DoEvents

        End If

    Next lngCount


    Application.StatusBar = False

End Sub
```

代码运行的效果如图 4.3 所示。

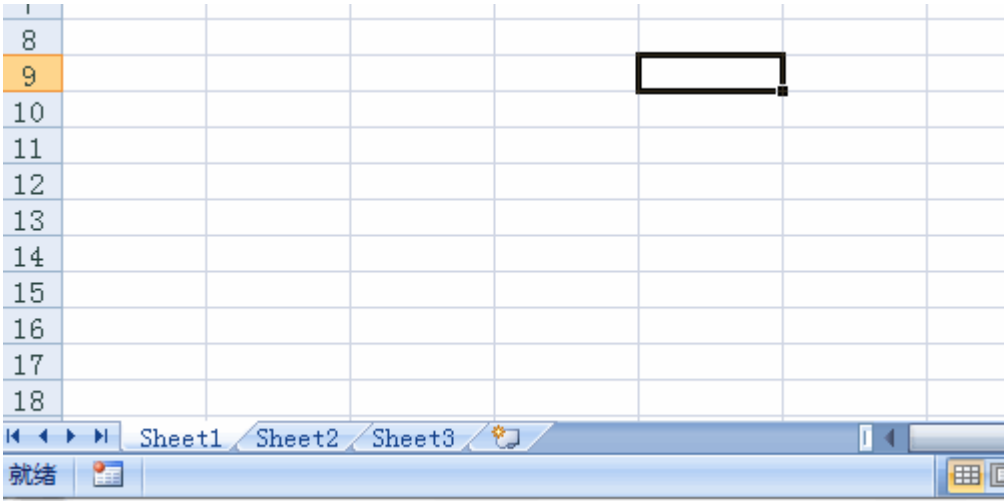


图 4.3(演示视频截图，详细视频见完美 Excel 微信公众号)

本章内容 2018 年 7 月 30 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（105）：获取文件名——GetOpenFilename 方法

5.获取文件名——GetOpenFilename 方法

使用 Application 对象的 GetOpenFilename 方法，可以显示标准的“打开”对话框。用户可以从该对话框中获取文件名，但是不会真正打开文件。

GetOpenFilename 方法的语法：

```
Application.GetOpenFilename(FileFilter,FilterIndex,Title,
ButtonText,MultiSelect)
```

其中：

- 参数 FileFilter，字符串，指定文件筛选条件，即出现在“打开”对话框中“文件类型”下拉列表中的内容，由文件筛选字符串和通配符表示的文件筛选规则说明组成，其中每一部分和每一对都用逗号隔开。“打开”对话框只显示与筛选条件相匹配的文件，默认为“所有文件（*.*）,*.*”，该字符串的第一部分（所有文件（*.*））是显示在“文件类型”下拉列表中的文本，第二部分（*.*）实际上确定要显示哪些文件。
- 参数 FilterIndex，指定默认文件筛选条件的索引值，即在“文件类型”框中显示的文件类型，索引值从 1 至在参数 FileFilter 中指定的筛选条件数。默认情况下，使用第一个文件筛选条件。
- 参数 Title，指定对话框的标题。默认情况下，标题为“打开”。
- 参数 ButtonText，仅适用 Macintosh。
- 参数 MultiSelect，设置为 True 允许选择多个文件名称；设置为 False 只允许选择一个文件名称。默认情况下为 False，只允许选择一个文件名。

GetOpenFilename 方法返回所选择的文件名或者用户输入的文件名，返回的名称包含文件路径。如果参数 MultiSelect 为 True，则返回由所选文件名组成的数组（即使只选择了一个文件名）。如果用户单击“取消”关闭对话框，则返回 False。

示例 1：显示打开对话框和选择的文件名

下面的代码显示“打开”对话框，并列出文件夹中的文本文件。用户选择某文本文件后，弹出消息框显示该文件名称和路径。

```
Sub testGetOpenFilename()
    Dim FileToOpen As Variant

    FileToOpen = Application.GetOpenFilename("文本文件(*.txt),*.txt")
    If FileToOpen <> False Then
        MsgBox "想要打开文件: " & FileToOpen
    End If
End Sub
```

运行代码后的效果如图 5.1 所示。

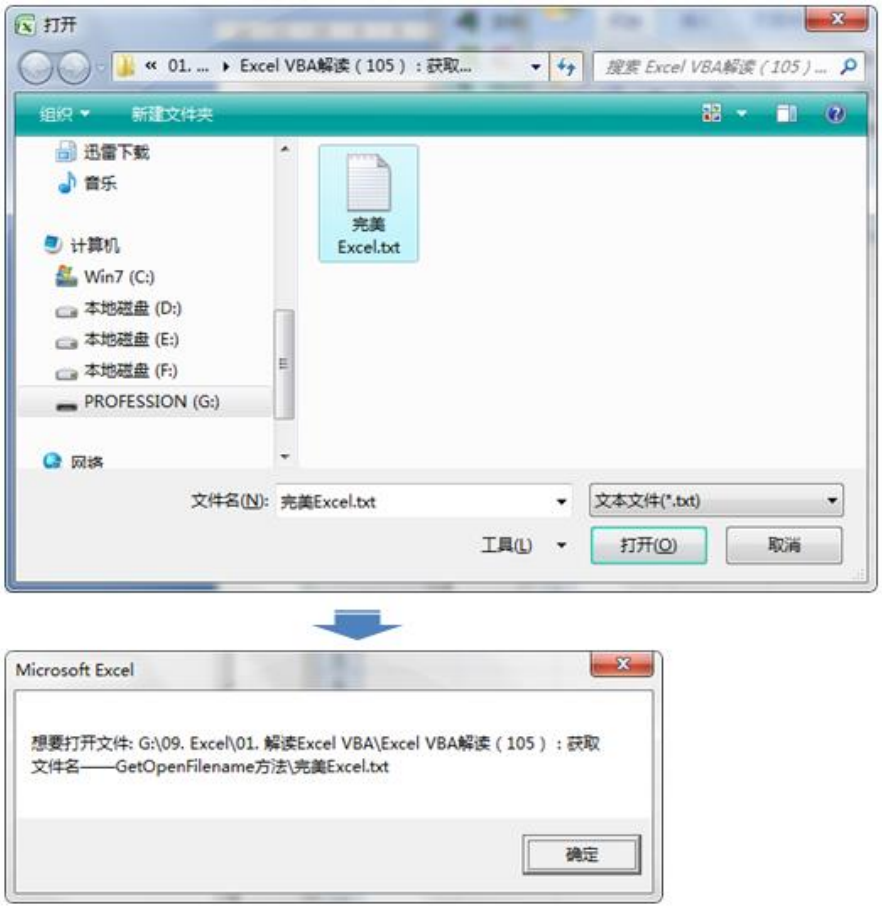


图 5.1

示例 2：获取单个或者多个工作簿文件名

下面的代码示例演示了如何获取单个工作簿或者多个工作簿文件名。

```
Sub TestGetExcelFile()
    Dim strFile As String
    Dim nIndex As Integer
    Dim varFiles As Variant
    Dim strFiles As String

    '获取单个工作簿文件
    strFile = GetExcelFilename("选择单个工作簿")
    If strFile = "False" Then
        MsgBox "没有选择文件!"
    Else
        MsgBox "选择了工作簿: " & strFile
    End If

    '获取多个工作簿文件
    varFiles = GetExcelFilename("选择多个工作簿", True)
    If Not IsArray(varFiles) Then
        MsgBox "没有选择文件!"
    Else
        For nIndex = 1 To UBound(varFiles)
            strFiles = strFiles & varFiles(nIndex) & vbCrLf
        Next nIndex
        MsgBox "所选择的文件分别为:" & strFiles
    End If
End Sub

Function GetExcelFilename(strTitle As String, _
```


```
        Optional blnSelect As Boolean = False) As Variant
    Dim strFilter As String


    strFilter = "工作簿(*.xls*),*.xls*"


    GetExcelFilename =
Application.GetOpenFilename(FileFilter:=strFilter, _
          Title:=strTitle, MultiSelect:=blnSelect)
End Function
```

注意到，返回多个工作簿文件名的数组下标基于 1 而不是通常的基于 0。

示例 3：列出供选择的多个文件类型并允许选择多个文件

下面的示例代码创建了 5 种类型的筛选条件，并设置在“文件类型”下拉框中显示第 5 种类型，允许按 Shift 键或 Ctrl 键来选择每种类型下的多个文件。

```
Sub GetFileNames()
    Dim strFilt As String
    Dim iFilterIndex As Integer
    Dim strTitle As String
    Dim varFilename As Variant
    Dim i As Integer
    Dim str As String


    '创建文件筛选列表
    strFilt = "文本文件(*.txt),*.txt," & _
              "Excel 工作簿(*.xls*),*.xls*," & _
              "逗号分隔文件(*.csv),*.csv," & _
              "ASCII 文件(*.asc),*.asc," & _
              "所有文件(*.*),*.*"
```


```
    '默认显示的文件类型*.*
    iFilterIndex = 5

    '设置对话框标题
    strTitle = "选择要输出的文件"

    '获取文件名
    varFilename = Application.GetOpenFilename _
       (FileFilter:=strFilt, _
        FilterIndex:=iFilterIndex, _
        Title:=strTitle, _
        MultiSelect:=True)

    '如果取消对话框则退出
    If Not IsArray(varFilename) Then
        MsgBox "没有选择任何文件."
        Exit Sub
    End If

    '显示文件的完整路径和名称
    For i = LBound(varFilename) To UBound(varFilename)
        str = str & varFilename(i) & vbCrLf
    Next i

    MsgBox "你选择的文件是: " & vbCrLf & str
End Sub
```


本章内容 2018 年 8 月 4 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（106）：指定文件名——GetSaveAsFilename 方法

6.指定文件名——GetSaveAsFilename 方法

使用 Application 对象的 GetSaveAsFilename 方法，可以显示标准的“另存为”对话框。用户可以在该对话框中指定或者选择文件名，但是不会真正保存文件。

GetSaveAsFilename 方法的语法：

```
Application.GetSaveAsFilename(InitialFilename,FileFilter,
FilterIndex,Title,ButtonText)
```

其中：

- 参数 InitialFilename，指定默认使用的文件名字。如果忽略，Excel 使用当前工作簿的名字。
- 参数 FileFilter，字符串，指定文件筛选条件，即出现在对话框中“保存类型”下拉列表中的内容，由文件筛选字符串和通配符表示的文件筛选规则说明组成，其中每一部分和每一对都用逗号隔开。对话框只显示与筛选条件相匹配的文件，默认为“所有文件（*.*）,*.*”，该字符串的第一部分（所有文件（*.*））是显示在“文件类型”下拉列表中的文本，第二部分（*.*）实际上确定要显示哪些文件。
- 参数 FilterIndex，指定默认文件筛选条件的索引值，即在“保存类型”框中显示的文件类型，索引值从 1 至在参数 FileFilter 中指定的筛选条件数。默认情况下，使用第一个文件筛选条件。
- 参数 Title，指定对话框的标题。默认情况下，标题为“另存为”。
- 参数 ButtonText，仅适用 Macintosh。

GetSaveAsFilename 方法返回所指定的文件名，返回的名称包含文件路径。如果用户单击“取消”关闭对话框，则返回 False。

示例 1：显示另存为对话框并指定文件名

下面的代码显示“打开”对话框，并列出文件夹中的文本文件。用户选择某文本文件后，弹出消息框显示该文件名称和路径。

```
Sub testGetSaveAsFilename()
    Dim SaveFileNameAs Variant

    SaveFileName = Application.GetSaveAsFilename( _
        FileFilter:="文本文件(*.txt),*.txt")
    If SaveFileName<> False Then
        MsgBox "保存文件为: " &SaveFileName
    End If
End Sub
```

运行代码后的效果如图 6.1 所示。

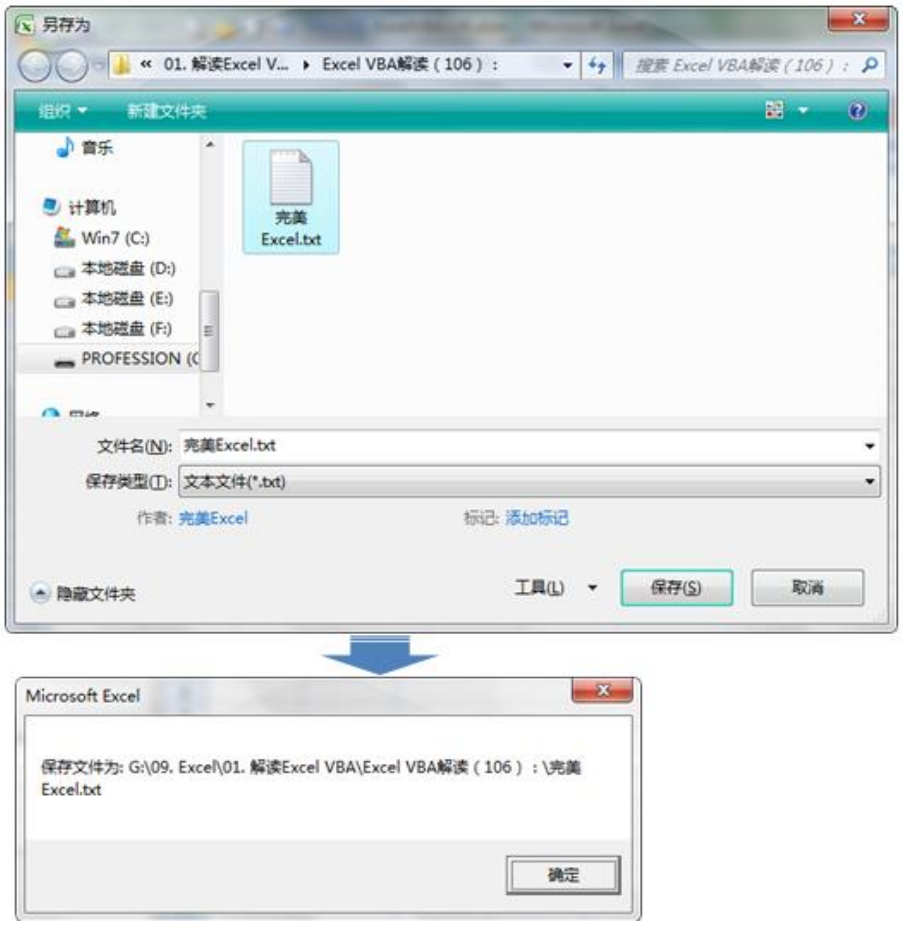


图 6.1

示例 2：指定并显示文件名

下面的代码显示用户提供的文件名，并询问是否要继续指定文件并显示文件名，如果单击“是”，则继续指定并显示文件名，单击“否”则退出。

```
Sub GetSaveAsFilenameEx()
    Dim strFile As String
    Dim lngResponse As Long
    Dim strMsg As String

    Do
        strFile = Application.GetSaveAsFilename
        strMsg = "你选择了文件: " &strFile& "." & _
            vbCrLf& "继续选择吗?"
        lngResponse = MsgBox(strMsg, vbYesNo)
    Loop While lngResponse = vbYes
End Sub
```

示例 3：拆分指定的文件名

下面的示例由 Steven M. Hansen 提供，将获得的完整文件名拆分成文件路径和文件名。

```
Sub testSplitFilename()
    Dim strPath As String
    Dim strName As String
    Dim strFileName As String
    Dim strMsg As String

    strFileName = Application.GetSaveAsFilename
    SplitFilenamestrFileName, strName, strPath
    strMsg = "这个文件的名字是: " & strName & vbCrLf
```


```
    strMsg = strMsg& "路径是: " &strPath
    MsgBoxstrMsg, vbOKOnly
End Sub

Sub SplitFilename(strFullName As String, _
ByRefstrName As String, _
ByRefstrPath As String)
    Dim nPos As Integer

    '找到文件名开始的位置
    nPos = FilenamePosition(strFullName)

    If nPos> 0 Then
     strName = Right(strFullName, Len(strFullName) - nPos)
     strPath = Left(strFullName, nPos - 1)
    Else
      MsgBox "无效的文件名!"
    End If
End Sub

'返回文件名字的第一个字符在完整文件名中的位置
'完整文件名包括路径和文件名字
Function FilenamePosition(strFullName As String) As
Integer
    Dim blnFound As Boolean
    Dim nPosition As Integer

    blnFound = False
    nPosition = Len(strFullName)

    Do While blnFound = False
        If nPosition = 0 Then Exit Do
```


```
        '从右边开始查找第一个"\"
        If Mid(strFullName, nPosition, 1) = "\" Then
            blnFound = True
        Else
            '从右到左
            nPosition = nPosition - 1
        End If
    Loop

    If blnFound = False Then
        FilenamePosition = 0
    Else
        FilenamePosition = nPosition
    End If
End Function
```

上面的代码在主过程 testSplitFilename 中调用了 SplitFilename 过程，而 SplitFilename 过程又调用了自定义函数过程 FilenamePosition。其中，SplitFilename 过程在参数中使用了关键字 ByRef，表示在被调用过程中修改了参数值后，调用过程中传递的变量同时改变。关于函数调用和参数传递的详细内容，在完美 **Excel** 微信公众号系列文章中将会详细讲解。

本章内容 2018 年 8 月 7 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（107）：打开文件对话框——FileDialog 属性

7 . 打开文件对话框——FileDialog 属性

Application 对象的 FileDialog 属性，返回一个代表文件对话框实例的 FileDialog 对象。

FileDialog 属性的语法：

```
Application.FileDialog(fileDialogType)
```

其中：

- 参数 fileDialogType，用于指定对话框的类型，是一个

  MsoFileDialogType 常量。有 4 种类型的 FileDialog 对象，其中：

  常量 msoFileDialogOpen 的值为 1，表示打开文件对话框，允许选择一个或多个文件，使用 Execute 方法打开文件；

  常量 msoFileDialogSaveAs 的值为 2，表示保存文件对话框，允许选择一个文件并使用 Execute 方法保存当前文件；

  常量 msoFileDialogFilePicker 的值为 3，表示文件选取对话框，允许选择一个或多个文件，并在 FileDialogSelectedItems 集合中捕获用户选择的文件路径；

  常量 msoFileDialogFolderPicker 的值为 4，表示文件夹选取对话框，并在 FileDialogSelectedItems 集合中捕获用户选择的文件路径。

示例 1：显示打开文件对话框并显示文件名

下面的代码引用自 VBA 帮助。显示“打开文件”对话框，允许用户选择一个或

多个文件，然后依次显示每个文件带路径的完整名称。

```
Sub UseFileDialogOpen()
    Dim lngCount As Long

    '打开文件对话框
    With Application.FileDialog(msoFileDialogOpen)
        .AllowMultiSelect = True
        .Show

        '显示所选择的文件的路径
        For lngCount = 1 To .SelectedItems.Count
            MsgBox .SelectedItems(lngCount)
        Next lngCount
    End With
End Sub
```

使用 Show 方法显示文件对话框。运行代码后的效果如图 7.1 所示。

图 7.1

示例 2：指定默认文件路径

下面的代码为用户提供了默认的文件路径，即 Excel 的默认文件路径作为起始目录。

```
Sub GetFile()
    With Application.FileDialog(msoFileDialogFilePicker)
        .InitialFileName = Application.DefaultFilePath &
"\"
        .Title = "请选择备份的位置"
        .Show

        If .SelectedItems.Count = 0 Then
            MsgBox "已取消!"
        Else
            MsgBox .SelectedItems(1)
        End If
    End With
End Sub
```

示例 3：显示文件选取对话框并显示选择的文件

下面的示例创建并显示文件选取（浏览）对话框，并在消息框中显示所选择的文件。

```
Sub SelectAndDisplayFiles()
    Dim fd As FileDialog
    Dim varSelectedItem As Variant

    Set fd =
Application.FileDialog(msoFileDialogFilePicker)

    With fd
        If .Show = -1 Then
```


```
            For Each varSelectedItem In .SelectedItems
                MsgBox "文件路径是: " & varSelectedItem
            Next varSelectedItem
        End If
    End With

    Set fd = Nothing
End Sub
```

代码中的 Show 方法显示文件对话框并返回一个 Long 值，如果用户按下操作按钮（如打开、保存等），则返回-1；如果用户按下取消按钮，则返回 0。

扩展：FileDialog 对象成员

方法

- Show 方法：显示文件对话框并返回代表用户操作的值。
- Execute 方法：调用 Show 方法后执行用户的操作。

属性

- AllowMultiSelect 属性：可读写，设置为 True，则允许用户从文件对话框中选择多个文件。
- ButtonName 属性：可读写，设置或获取字符串，表示文件对话框中操作按钮显示的文本。
- DialogType 属性：只读，文件对话框类型。
- FilterIndex 属性：可读写，获取或设置文件对话框的默认筛选条件。
- Filters 属性：只读，获取 FileDialogFilters 集合。

- InitialFileName 属性：可读写，设置或返回文件对话框中默认显示的文件路径和名称。
- InitialView 属性：可读写，设置或返回 MsoFileDialogView 常量，表示文件和文件夹在文件对话框中的默认显示。
- Item 属性：只读，返回与对象关联的文本。
- SelectedItems 属性：只读，返回 FileDialogSelectedItems 集合，包含用户在文件对话框中使用 Show 方法显示选择的文件名列表。
- Title 属性：可读写，设置或返回显示在文件对话框中的标题。

本章内容 2018 年 8 月 13 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（108）：获取用户输入——InputBox 方法

8.获取用户输入——InputBox 方法

通常，我们会使用用户窗体来设计与用户交互的界面，实现用户与程序的连接。其实，Excel VBA 已经为我们提供了简单方便的交互方式。在《Excel VBA 解读：基础入门篇》中，我们介绍了 VBA 的内置函数：MsgBox 函数。MsgBox 函数会让 Excel 弹出一个消息框，为用户提供相关的信息。

本章主要讲解用于获取用户输入信息的 InputBox 方法。如下图 8.1 所示的对话框，允许用户输入数据并获取用户输入。

图 8.1

InputBox 方法的语法：

```
Application.InputBox(Prompt,[Title],[Default],[Left],[Top],[HelpFile],[HelpContextID],[Type])
```

其中：

- 参数 Prompt，必需，用于指定显示在对话框中提示用户输入的文本。
- 参数 Title，可选，提供显示在对话框标题栏的文本。如果忽略，则默认为“输入”。
- 参数 Default，可选，当对话框初始化时，指定对话框中显示在文本输入框中的值。如果忽略，则文本框中为空。
- 参数 Left，可选，指定对话框相对于屏幕左上角的 x 位置，以磅为单位。

- 参数 Top，可选，指定对话框相对于屏幕左上解的 y 位置，以磅为单位。
- 参数 HelpFile，可选，为对话框提供帮助文件的名字。如果参数 HelpFile 和参数 HelpContextID 都被设置，则对话框中显示帮助按钮。
- 参数 HelpContextID，可选，在参数 HelpFile 中的帮助主题的上下文 ID 编号。
- 参数 Type，可选，指定返回的数据类型。如果忽略该参数，则对话框返回文本。下图 8.2 所示的表格中，列出了可以传递到参数 Type 中的值，可以是一个值或者多个值之和（即任意组合）。例如，如果想要输入框接受文本和数字，则可以设置参数 Type 为 1+2。

值	说明
0	公式
1	数字
2	文本（字符串）
4	逻辑值（True或False）
8	单元格引用，作为Range对象
16	错误值，例如#N/A
64	一组值

图 8.2

使用 InputBox 方法，可以是显示简单的对话框，接受用户输入的信息并可以在代码中使用这些信息。如上图 8.1 所示，对话框有“确定”和“取消”两个按钮。如果单击“确定”按钮，那么 Input 方法返回在对话框中输入的值；如果单击“取消”按钮，则返回 False。

如果参数 Type 设置为 0，那么 InputBox 方法以文本形式返回公式，例如“=2+3”。如果在公式中有任何的引用，那么作为 A1 样式的引用返回。

如果参数 Type 设置为 8，那么 InputBox 方法返回一个 Range 对象。此时，必须使用 Set 语句来将对话框的结果赋值给一个 Range 对象变量，否则，变量会被设置为单元格区域中的值而不是 Range 对象。

如果使用 InputBox 方法要求用户输入公式，那么必须使用 FormulaLocal 属性将公式赋值给 Range 对象。

VBA 还提供了一个与 InputBox 方法同名的内置函数：InputBox 函数，只能在对话框中输入数据，如果用户什么也没有输入，则返回零长度的字符串；如果用户单击“取消”按钮，同样返回一个零长度字符串。注意，

Application.InputBox 调用 InputBox 方法，而单独的 InputBox 则是 InputBox 函数。总的来说，InputBox 方法的功能更加强大。

示例 1：获取用户输入的数据

下面的代码显示一个对话框，要求用户输入文本，然后获取并显示用户输入的文本。

```
Sub testInputBox()
    Dim varInput As Variant

    varInput = Application.InputBox( _
               Prompt:="请输入要查找的文本:", _
               Title:="查找数据")
    MsgBox "你要查找的文本是: " & vbCrLf & varInput
End Sub
```

运行代码后的效果如图 8.3 所示。

图 8.3

示例 2：求所选择的单元格区域中的数值之积

下面的代码来自 VBA 帮助。使用对话框要求用户选择包含 3 个单元格的单元格区域，并将其传递给自定义函数 MyFunction，将单元格区域中的 3 个值相乘并返回结果。

```
Sub cmb_value_select()
    Dim rng As Range

    Set rng = Application.InputBox( _
           Prompt:="选择单元格区域:", _
           Type:=8)

    If rng.Cells.Count <> 3 Then
       MsgBox "请选择 3 个单元格!"
       Exit Sub
    End If

    '调用 MyFunction 函数求值
    '并在当前单元格中放置结果
    ActiveCell.Value = MyFunction(rng)
End Sub

Function MyFunction(rng As Range) As Double
    MyFunction = rng(1) * rng(2) * rng(3)
End Function
```

本章内容 2018 年 8 月 21 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（109）：让程序在指定的时间运行——OnTime 方法

9.让程序在指定的时间运行——OnTime 方法

某些情形下，我们可能需要在指定的时间运行某个程序，Application 对象的 OnTime 方法提供了这样的功能。

OnTime 方法计划在将来指定的时间运行程序，需要指定想要程序运行的时间和程序名称。其语法为：

```
Application.OnTime(EarliestTime,Procedure,[LatestTime],[Schedule])
```

其中：

- 参数 EarliestTime，必需，指定想要程序在什么时间运行。
- 参数 Procedure，必需，要运行的程序的名字。
- 参数 LatestTime，可选，指定程序最迟运行的时间。例如，如果参数 LatestTime 设置为 EarliestTime+30，Excel 因为在运行另一个程序而在 EarliestTime 时没有在准备、复制、剪切或查找模式，那么 Excel 将等待 30 秒以便该程序运行完成。如果 Excel 在 30 秒内还不处于准备模式，那么程序将不再运行。如果忽略该参数，那么 Excel 将等待直到能够运行该程序。
- 参数 Schedule，可选，设置为 True 来计划运行一个新的 OnTime 程序；设置为 False 来清除之前设置的程序。默认值为 False。

使用 Now+TimeValue(time)来计划在特定时间（从现在开始计算）过去多少时间要运行的程序。使用 TimeValue(time)来计划某个特定时间要运行的程序。

下面的代码在 15 秒后运行程序 my_Procedure：

```
Application.OnTime Now + TimeValue("00:00:15"),
```


```
"my_Procedure"
```

下面的代码在下午 5 点运行程序 my_Procedure：

```
Application.OnTime TimeValue("17:00:00"), "my_Procedure"
```

下面的代码取消上面的 OnTime 设置：

```
Application.OnTime EarliestTime:=TimeValue("17:00:00"), _
 Procedure:="my_Procedure", Schedule:=False
```

下面的代码在 2018 年 7 月 1 日上午 6 时 30 分运行程序 my_Procedure：

```
Application.OnTime DateSerial(2018,7,1) +
TimeValue("6:30:00"), "my_Procedure"
```

与 Wait 方法不同，OnTime 方法在等待运行指定的程序时，允许进行正常的 Excel 操作，包括运行其他的程序，而 Wait 方法暂停程序的运行，所有的 Excel 交互操作都将被挂起。

注意，当使用 OnTime 方法计划在将来的某个时间运行程序时，需要确保 Excel 一直在内存中运行至预定的时间，不需要一直打开包含 OnTime 方法程序的工作簿，Excel 会在需要时自行打开该工作簿。

示例 1：设置闹钟

下面的代码在早上 6 时 30 分时，发出声音并显示一个消息框。

```
Sub SetColck()
    Application.OnTime TimeValue("6:30:00"),
"DisplayClock"
End Sub


Sub DisplayClock()
    Beep
    MsgBox "快起床啦!"
End Sub
```

示例 2：定时刷新数据

下面是 Excel 2007 VBA Programmer's Reference 中的一个示例。在 OnTime 方法中，指定程序名字为代码自身所在程序，则可以实现定时刷新数据，如下面的代码所示。

```
Dim mdteScheduledTime As Date


Sub RefreshData()
    ThisWorkbook.UpdateLink Name:="C:\Data.xlsx", _
                  Type:=xlExcelLinks
    mdteScheduledTime = Now + TimeSerial(0, 1, 0)
    Application.OnTime mdteScheduledTime, "RefreshData"
End Sub


Sub StopRefresh()
    Application.OnTime mdteScheduledTime, "RefreshData", ,
False
End Sub
```

运行程序 RefreshData 后，该程序将每分钟执行一次。由于程序中使用了 UpdateLink 方法更新外部链接，因此会定时刷新当前工作簿中的数据。

要停止该程序，运行 StopRefresh 程序，取消 RefreshData 程序的运行。

下面是 John Walkenbach 经典的 Excel Power Programming with VBA 中的一个类似的例子。在单元格 A1 中显示时间，并每隔 5 秒中更新一次，代码如下：

```
Dim NextTick As Date
Sub UpdateClock()
    '使用当前时间更新单元格 A1 中的内容
    ThisWorkbook.Sheets(1).Range("A1") = Time
    '设置更新时间间隔
    NextTick = Now + TimeValue("00:00:05")
    Application.OnTime NextTick, "UpdateClock"
End Sub
```


```
Sub StopClock()
    On Error Resume Next
    Application.OnTime NextTick, "UpdateClock", , False
End Sub
```


本章内容 2018 年 8 月 27 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（110）：自定义快捷键运行宏——OnKey 方法

10.自定义快捷键运行宏——OnKey 方法

在操作 Excel 时，我们常常会使用快捷键来方便地达到目的，例如使用经典的 Ctrl+C 键复制，再用 Ctrl+V 键粘贴。实际上，快捷键的背后就是实现相应功能的程序。在 VBA 程序代码中，我们可以自定义运行宏的快捷键，这就要使用 Application 对象的 OnKey 方法。

使用 OnKey 方法，可以将程序赋给单个键或键组合，通过按下赋给的键或键组合来运行该程序。同时，也可以使用这个方法来禁用组合键。

OnKey 方法的语法为：

```
Application.OnKey(Key,Procedure)
```

说明：

- 参数 Key，必需，指定代表所按键的字符串。
- 参数 Procedure，可选，指定当按下指定键时要运行的过程名。
- 忽略第 2 个参数将使指定键恢复正常功能；而将空字符串作为第 2 个参数并不会恢复快捷键的原有功能，而是忽略按键，即当使用指定键时不会有任何操作。
- OnKey 方法设置的快捷键对所有打开的工作簿都有效

下图 10.1 列出了 OnKey 方法中可以使用的按键代码：

键	代码
Backspace	{BACKSPACE}或{BS}
Break	{BREAK}
Caps Lock	{CAPSLOCK}
Clear	{CLEAR}
Delete	{DELETE}或{DEL}
向下箭头	{DOWN}
向左箭头	{LEFT}
向右箭头	{RIGHT}
向上箭头	{UP}
End	{END}
回车键	~ 或 {ENTER}
Escape	{ESCAPE} 或 {ESC}
Help	{HELP}
Home	{HOME}
Insert	{INSERT}
NumLock	{NUMLOCK}
Page Down	{PGDN}
Page Up	{PGUP}
Return	{RETURN}
Scroll Lock	{SCROLLLOCK}
Tab	{TAB}
F1至F15	{F1}至{F15}

图 10.1

除了上述按键外，还可以指定 Alt 键、Ctrl 键和 Shift 键与它们组合，这 3 个键的代表符号分别为：

- Alt 键：百分号（%）
- Ctrl 键：脱字号（^）
- Shift 键：加号（+）

下面的代码当按下 Alt+F10 组合键时执行指定的过程：

```
Application.OnKey "%{F10}", "过程名"
```

下面的代码在按下 Ctrl+Shift+Y 组合键时执行指定的过程：

```
Application.OnKey "^+Y", "过程名"
```

要使用%、^、+作为指定过程的快捷键，就要将这些字符放置在花括号中，例如下面的代码：

```
Application.OnKey "^{+}", "过程名"
```

按下 Ctrl++组合键执行指定的过程。

示例 1：禁用已有的快捷键

使用 OnKey 方法能够禁用已有的快捷键。通过赋给一个空过程，下面的代码禁止用户使用 Ctrl+c 组合键进行复制操作。

```
Sub DisableCopyShortCut()
    Application.OnKey "^c", ""
End Sub
```

要恢复 Ctrl+c 组合键的复制功能，则运行下面的代码。

```
Sub ClearCopyShortCut()
    Application.OnKey "^c"
End Sub
```

示例 2：自定义快捷键运行特定的操作

下面是 Excel 2007 VBA Programmer's Reference 中的一个示例。当运行 AssignDown 过程后，我们在 Excel 中按向下箭头键时，活动单元格将从当前单元格向下移动 10 行，而不是通常的 1 行。

```
Sub AssignDown()
    Application.OnKey "{Down}", "DownTen"
End Sub


Sub DownTen()
    ActiveCell.Offset(10, 0).Select
End Sub
```

运行下面的程序恢复向下箭头键的正常功能。

```
Sub ClearDown()
    Application.OnKey "{Down}"
End Sub
```


本章内容 2018 年 9 月 3 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（111）：向应用程序发送键盘命令——SendKeys 方法

11.向应用程序发送键盘命令——SendKeys 方法

SendKeys 方法允许发送按键到当前应用程序窗口，其语法为：

```
Application.SendKeys(Keys,Wait)
```

说明：

- 参数 Keys，必需，想要发送给应用程序的键或键组合。可以指定 Alt 键（%）、Ctrl 键（^）和 Shift 键（+）与下图 1 所列按键进行组合。
- 参数 Wait，可选，设置为 True，将控制权返回给宏之前 Excel 等待键被处理；设置为 False（或忽略）继续运行宏而不等待键被处理。

下图 11.1 列出了 SendKeys 方法中可以使用的按键代码：

键	代码
Backspace	{BACKSPACE}或{BS}
Break	{BREAK}
Caps Lock	{CAPSLOCK}
Clear	{CLEAR}
Delete	{DELETE}或{DEL}
向下箭头	{DOWN}
向左箭头	{LEFT}
向右箭头	{RIGHT}
向上箭头	{UP}
End	{END}
回车键	~ 或 {ENTER}
Escape	{ESCAPE} 或 {ESC}
Help	{HELP}
Home	{HOME}
Insert	{INSERT}
NumLock	{NUMLOCK}
Page Down	{PGDN}
Page Up	{PGUP}
Return	{RETURN}
Scroll Lock	{SCROLLLOCK}
Tab	{TAB}
F1至F15	{F1}至{F15}

图 11.1

下面的代码退出 Excel 应用程序：

```
Application.SendKeys ("%fx")
```

注意，在 Excel 应用程序界面窗口执行上述代码才能正确实现退出 Excel 的效果。

示例 1：打开计算器并设置计算模式

下面的过程开启 Windows 计算器并显示为“科学型”模式。

```
Sub UseCalc()
    Shell "calc.exe", vbNormalFocus
    Application.SendKeys "%vs"
End Sub
```

示例 2：清除立即窗口中的内容

下面的程序清除 VBE 编辑器中立即窗口里的内容。

```
Sub ClearImmediateWindow()
    Application.VBE.Windows.Item("立即窗口").SetFocus
    Application.SendKeys "^a"
    Application.SendKeys "{Del}"
End Sub
```

代码首先将焦点转到立即窗口，然后发送 Ctrl+a 组合键选择该窗口中的所有文本，再发送 Del 键删除所选文本。

注意，要使该程序正常运行，必须在“信任中心——宏设置”中选取“信任对 VBA 工程对象模型的访问”。

本章内容 2018 年 9 月 6 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（112）：在代码中使用工作表函数——WeeksheetFunction 属性

12.在代码中使用工作表函数——WeeksheetFunction 属性

在 VBA 代码中，使用 WorksheetFunction 属性，将允许调用工作表函数，其语法为：

```
Application.WorksheetFunction
```

例如，下面的代码：

```
Application.WorksheetFunction.Min(Range("A1:A6"))
```

返回单元格区域 A1:A6 中的最小值。其中，Min 函数是工作表函数。

在 VBA 中提供有自己的一些函数，通常情况下，如果存在着与 Excel 函数相同用途的 VBA 函数，那么在 VBA 中就不能够直接使用这个 Excel 函数。如果一定要使用这样的 Excel 函数，则可以使用完美 Excel 微信公众号中讲解过的 Evaluate 方法。此外，还可以使用 Range 对象的 FormulaR1C1 属性使用工作表函数设置单元格中的公式，详细用法参见本系列前面的相关文章。

示例 1：替换单元格中指定的字符

下面的过程替换当前工作表列 A 中指定的字符。

```
Sub ReplaceSpecialChar()

    Dim rng As Range

    For Each rng In
ActiveSheet.Columns("A:A").SpecialCells(xlCellTypeConstan
ts, xlTextValues).Cells
```


```
        With rng
            .Value =
Application.WorksheetFunction.Substitute(.Value, "Co", "
Co. ")
            .Value =
Application.WorksheetFunction.Substitute(.Value, "Ltd", "
Ltd ")
            .Value =
Application.WorksheetFunction.Substitute(.Value, "Inc",
"Inc. ")
            .Value =
Application.WorksheetFunction.Substitute(.Value, ",", "
")
        End With
    Next rng
End Sub
```

本示例代码可以扩展为你所需要替换的字符。

示例 2：查找数据列中的最大值

下面的代码在列 A 的最后一个单元格之后输入该列的最大值并加上 1。

```
Sub GetMaxValue()
    Range("A1").End(xlDown).Offset(1, 0).Select
    ActiveCell =
WorksheetFunction.Max(ActiveCell.EntireColumn) + 1
End Sub
```

示例 3：显示所选单元格区域的汇总信息

下面的代码使用工作表函数 Count 函数、Sum 函数、Average 函数显示所选单元格区域的单元格数量、数值之和、数值的平均值等信息。

```
Sub SummaryInfo()
    Dim rngSelect As Range
    Dim lngCount As Long, dblSum As Double, dblAvg As
```

```
Double


    Set rngSelect = Application.InputBox(Prompt:="选择要汇总
的单元格区域", _
                Default:=Selection.Address, _
                Type:=8)
    rngSelect.Select

    lngCount = WorksheetFunction.Count(rngSelect)
    dblSum = WorksheetFunction.Sum(rngSelect)
    dblAvg = WorksheetFunction.Average(rngSelect)

    MsgBox "所选单元格区域" & vbCrLf & _
          "数量: " & lngCount & vbCrLf & _
          "和: " & FormatNumber(dblSum, 2) & vbCrLf & _
          "平均值:" & FormatNumber(dblAvg, 2)
End Sub
```

示例 4：清理数据

下面的代码使用工作表函数 Trim 清理数据两侧的空格并使用 Proper 函数使数据大小写合适。

```
Sub CleanUpData()
    Dim rngCell As Range
    For Each rngCell In ActiveCell.CurrentRegion
       rngCell = WorksheetFunction.Trim(rngCell)
       rngCell = WorksheetFunction.Proper(rngCell)
    Next rngCell
End Sub
```


本章内容 2018 年 9 月 11 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（113）：设置或获取 Excel 窗口大小的属性

13.设置或获取 Excel 窗口大小的属性

使用 Application 对象的 WindowState 属性，可以设置或者返回 Excel 窗口的状态，其语法为：

```
Application.WindowState
```

例如，下面的代码：

```
Application.WindowState = xlMinimized
```

最小化 Excel 工作簿。

代码：

```
Application.WindowState = xlMaximized
```

将 Excel 工作簿最大化。

代码：

```
Application.WindowState = xlNormal
```

设置 Excel 工作簿为标准大小。

使用 Application 对象的 UsableHeight 属性和 UsableWidth 属性，返回工作簿窗口可以在 Excel 应用程序窗口区中占据空间的最大高度和宽度，以磅为单位。其语法为：

```
Application.UsableHeight

Application.UsableWidth
```


例如，下面的程序代码：

```
With ActiveWindow
    .WindowState = xlNormal
    .Top = 1
    .Left = 1
    .Height = Application.UsableHeight
    .Width = Application.UsableWidth
End With
```

将扩展当前窗口至最大可用的尺寸。

Application 对象的 Width 属性返回或设置一个 Double 类型的值，代表从 Excel 应用程序窗口左边至其右边的距离，以磅为单位。如果窗口被最小化，那么该属性只读，且将返回窗口图标的宽度。

Application 对象的 Height 属性返回或设置一个 Double 类型的值，代表 Excel 应用程序窗口的高度，以磅为单位。如果窗口被最小化，那么该属性只读，且将返回窗口图标的高度。如果窗口被最大化，那么不能设置该属性。使用 WindowState 属性来确定窗口的状态。

示例：获取窗口状态信息

下面的代码演示获取窗口状态、工作簿可用宽高以及应用程序宽高。

```
Sub GetWindowInfo()
    Dim lngState As Long
    Dim strInfo As String
    Dim lngResponse As Long

    '确定窗口的状态
    lngState = Application.WindowState
    Select Case lngState
        Case xlMaximized
            strInfo = "窗口最大化." & vbCrLf
```


```
        Case xlMinimized
            strInfo = "窗口最小化." & vbCrLf
        Case xlNormal
            strInfo = "窗口正常." & vbCrLf
    End Select

    strInfo = strInfo & "可用的高度 = " & _
            Application.UsableHeight & vbCrLf
    strInfo = strInfo & "可用的宽度 = " & _
            Application.UsableWidth & vbCrLf
    strInfo = strInfo & "高度 = " & _
            Application.Height & vbCrLf
    strInfo = strInfo & "宽度 = " & _
            Application.Width & vbCrLf & vbCrLf
    strInfo = strInfo & "你想要将窗口最小化吗?"

    lngResponse = MsgBox(strInfo, vbYesNo, "窗口信息")

    '如果单击"是"则最小化窗口
    If lngResponse = vbYes Then
        Application.WindowState = xlMinimized
    End If
End Sub
```

运行上述代码后的效果如图 13.1 所示。

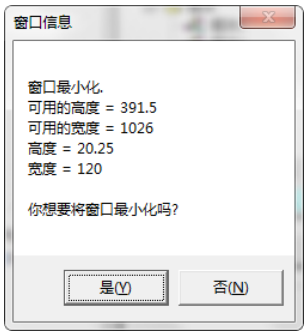


图 13.1

本章内容 2018 年 9 月 18 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（114）：在 Excel 中使用 Application 事件

14.在 Excel 中使用 Application 事件

在本系列技术电子书中，我们曾经介绍过 Workbook 对象事件、Worksheet 对象事件，他们都可以直接在相应的对象模块中直接获取并编写代码，这是因为在 Excel 中，已经为 Workbook 对象或 Worksheet 对象提供了类模块（例如，ThisWorkbook、Sheet1、Sheet2，等等），其中包含事件供我们使用。

Excel 并没有为 Application 对象提供类模块，因此，其本身的应用程序层级的事件——Application 事件，需要创建自己的类模块。（有关类的知识，在完美 Excel 微信公众号中我们会详细讲解）

使用类模块，可以创建自己的对象，可以扩展现有对象的功能。本章主要讲解如何使用类模块来钩挂 Application 对象的事件。Application 对象事件将会影响 Excel 会话中所有打开的工作簿。

如何在 Excel 中使用 Application 事件

首先，创建工作簿，在 VBE 中插入一个标准模块和一个类模块。在标准模块中，输入下面的代码声明一个公共变量：

```
Public gxlApp As 类 1
```

如下图 14.1 所示。

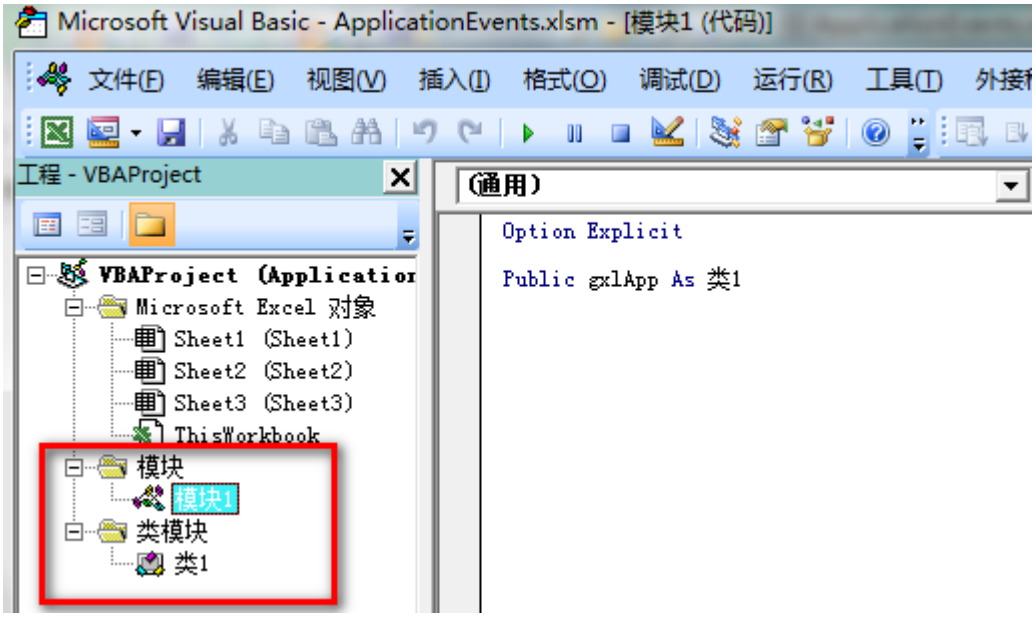


图 14.1

“类 1”是我们类模块的名称。在类模块中，使用下面的代码声明另一个变量：

```
Public WithEvents xlApp As Application
```

关键字 WithEvents 告诉 VBA，我们想要访问 Application 对象的事件。此时，通过展开下拉列表，你将可以在 VBE 编辑器代码窗口顶部看到可用的事件列表，如图 14.2 所示。

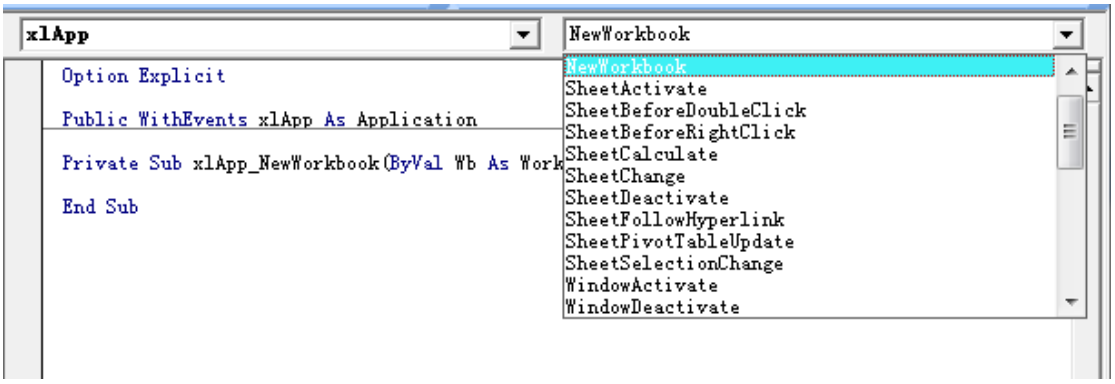


图 14.2

下面，创建一个事件过程。

在类模块中，使用代码窗口右上角的下拉列表插入 WorkbookNewSheet 事件。在事件过程中添加下面的代码，使得无论何时在任意工作簿中插入新工作表时，定置该工作表的页眉。

```
Private Sub xlApp_WorkbookNewSheet(ByValWb As Workbook, ByValSh As Object)
    If TypeName(Sh) = "Worksheet" Then
        Sh.PageSetup.CenterHeader = "完美 Excel"
    End If
End Sub
```

最后，在 Workbook_Open 事件中编写代码，告诉 VBA 公共变量 gxlApp 代表 Application 对象。

```
Private Sub Workbook_Open()
    Set gxlApp = New 类 1
    Set gxlApp.xlApp = Application
End Sub
```

小结

使用 Application 事件的基本步骤如下：

第 1 步：在标准模块中声明一个公共变量，代表类模块

第 2 步：创建一个类模块，使用 WithEvents 关键字声明一个公共变量，代表 Application 对象

第 3 步：在某个过程中，通常为 Workbook_Open 事件，将应用程序对象赋值给该类模块变量

本章内容 2018 年 9 月 24 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（115）：Application 对象事件概览

15.Application 对象事件概览

本章列出了 Application 对象的一系列常用事件及说明，供需要时查阅。其中，xlApp 是在类模块中声明的变量名称。

Private Sub xlApp_NewWorkbook(ByValWb As Workbook)

在创建新工作簿时发生，参数 wb 代表新工作簿。例如，下面的代码：

```
Private Sub xlApp_NewWorkbook(ByValWb As Workbook)
MsgBox "欢迎使用 excelperfect!"
End Sub
```

在创建新工作簿时给出一条提示信息，如图 15.1 所示。

图 15.1

Private Sub xlApp_SheetActivate(ByValSh As Object)

在激活工作表时发生，参数 sh 代表激活的工作表。例如，下面的代码：

```
Private Sub xlApp_SheetActivate(ByValSh As Object)
    MsgBox "你将使用工作表: " &Sh.Name
End Sub
```

在选择新工作表后会弹出一条提示信息。

Private Sub xlApp_SheetBeforeDoubleClick(ByValSh As Object, ByVal Target As Range, Cancel As Boolean)

在双击任意工作表中的单元格时发生，参数 Sh 代表工作表，参数 Target 代表工作表中双击的单元格，参数 Cancel 的值默认为 False，若设置为 True，则阻止通过双击单元格进入输入模式。

Private Sub xlApp_SheetBeforeRightClick(ByValSh As Object, ByVal Target As Range, Cancel As Boolean)

在任意工作表的单元格中点击右键时发生，参数 Sh 代表工作表，参数 Target 代表工作表中右击的单元格，参数 Cancel 的值默认为 False，若设置为 True，则阻止右键单击单元格时弹出快捷菜单。

Private Sub xlApp_SheetCalculate(ByValSh As Object)

在工作簿中的任意工作表被重新计算或者更新图表时发生，参数 Sh 代表当前工作表。

Private Sub xlApp_SheetChange(ByValSh As Object, ByVal Target As Range)

在工作簿中任意工作表的任意单元格的值发生改变时发生，参数 Sh 代表当前工作表，参数 Target 代表修改的单元格。

Private Sub xlApp_SheetDeactivate(ByValSh As Object)

在工作簿中选择其他工作表为活动工作表时发生，参数 Sh 代表之前的活动工作表。该事件发生在 **xlApp_SheetActivate** 事件之前。

Private Sub xlApp_SheetSelectionChange(ByValSh As Object, ByVal Target As Range)

在工作簿中选取任意工作表中的单元格时发生，参数 Sh 代表当前工作表，Target 代表选取的单元格区域。该事件发生在 **xlApp_SheetChange** 事件之后。

Private Sub xlApp_WorkbookBeforeClose(ByValWb As Workbook, Cancel As Boolean)

在关闭工作簿时发生，参数 Wb 代表被关闭的工作簿，参数 Cancel 默认值为 False，设置参数 Cancel 为 True 将阻止关闭工作簿。

Private Sub xlApp_WorkbookBeforePrint(ByValWb As Workbook, Cancel As Boolean)

在打印工作簿时发生，参数 Wb 代表被打印的工作簿，参数 Cancel 默认值为 False，设置参数 Cancel 为 True 将阻止打印工作簿。

Private Sub xlApp_WorkbookBeforeSave(ByValWb As Workbook, ByValSaveAsUI As Boolean, Cancel As Boolean)

在保存工作簿时发生，参数 Wb 代表被保存的工作簿。参数 SaveAsUI 设置为 True，打开“另存为”对话框；参数 Cancel 设置为 True，将阻止保存工作簿。

Private Sub xlApp_WorkbookNewSheet(ByValWb As Workbook, ByValSh As Object)

在工作簿中添加新工作表时发生，参数 Wb 代表工作簿，参数 Sh 代表新的工作表。

Private Sub xlApp_WorkbookOpen(ByValWb As Workbook)

在打开工作簿时发生，参数 Wb 代表被打开的工作簿。

本章内容 2018 年 9 月 28 日首发于
[完美 Excel]微信公众号 *excelperfect*
原标题为
Excel VBA 解读（116）：Application 对象事件示例

16.Application 对象事件示例

在运行本章的 Application 事件代码之前，请先按照 14.在 Excel 中使用 Application 事件的三步骤设置基本代码：

第 1 步：在标准模块中声明一个公共变量，代表类模块

第 2 步：创建一个类模块，使用 WithEvents 关键字声明一个公共变量，代表 Application 对象

第 3 步：在某个过程中，通常为 Workbook_Open 事件，将应用程序对象赋值给该类模块变量

下面列举几个 Application 对象事件示例。

示例 1：总是将 Excel 中新建工作表放到最后

在 Excel 工作簿中插入新工作表时，Excel 会将新工作表放置在最左侧或者当前工作表的左侧，下面的 WorkbookNewSheet 事件代码总是将 Excel 新建工作表放置在最后位置。

```
Private Sub xlApp_WorkbookNewSheet(ByVal Wb As Workbook,
ByVal Sh As Object)
    If Sh.Index < Wb.Sheets.Count - 1 Then
        Sh.Move After:=Wb.Sheets(Wb.Sheets.Count)
    End If
End Sub
```

示例 2：在打印的工作表页脚放置指定的文本

下面的代码在工作簿的工作表打印时在页脚放置文本“完美 Excel”：

```
Private Sub xlApp_WorkbookBeforePrint(ByVal Wb As Workbook,
Cancel As Boolean)

    Wb.ActiveSheet.PageSetup.LeftFooter = "完美 Excel"

End Sub
```

示例 3：强迫用户另存为启用宏的工作簿

有时候，我们发送给用户一个含有宏的工作簿，当用户对工作簿进行修改后，如果使用另存为命令保存工作簿，很可能保存为后缀为 xlsx 的工作簿，这样会移除工作簿中的宏，导致实现不了自定义的功能。下面的代码强迫用户在另存工作簿时必须将工作簿保存为 xlsm 扩展名的工作簿。

```
Private Sub xlApp_WorkbookBeforeSave(ByVal Wb As Workbook,
ByVal SaveAsUI As Boolean, Cancel As Boolean)
    Dim strFileName As String


    '检查是否调用"另存为"命令
    If SaveAsUI = True Then
        Cancel = True


        '调用自定义的对话框
        strFileName    =    Application.GetSaveAsFilename(,
"Excel 启用宏的工作簿(*.xlsm),*.xlsm", , "另存为 XLSM 文件")
        If strFileName = "False" Then
            MsgBox "操作失败", vbOKOnly
            Cancel = True
            Exit Sub
        End If
```

```
        '保存文件
        Application.EnableEvents = False
        Wb.SaveAs Filename:=strFileName, _
            FileFormat:=xlOpenXMLWorkbookMacroEnabled
        Application.EnableEvents = True
    End If
End Sub
```

代码着先检查参数 SaveAsUI 的值是否为 True，如果为 False，意味着用户单击了保存按钮；如果为 True，代表用户是否调用了另存为命令。如果用户调用了另存为命令，则使用 GetSaveAsFilename 方法调用对话框，限制对话框仅使用 xlsm 文件。

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 Application 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。