

第 47 章 事件的应用

在 Excel VBA 中，事件是指对象可以辨认的动作。用户可以指定 VBA 代码来对这些动作做出响应。Excel 可以监视多种不同类型的事件，Excel 中的工作表、工作簿、应用程序、图表工作表、查询表和控件等不同对象都有不同的事件，而且每个对象都有多种相关的事件，本章将主要介绍工作表和工作簿的常用事件。

本章学习要点：

要点 1： 工作表的 Change 事件。

要点 2： 工作簿的 Open 事件。

要点 3： 如何禁止事件激活。

47.1 事件过程

事件过程作为一种特殊的 Sub 过程，在事件被触发时执行，如果事件过程包含参数，系统会为相关参数赋值。事件过程必须写入相应的模块中才能发挥作用，工作簿事件过程须写入 Thisworkbook 模块中，工作表事件过程则须写入相应的工作表模块中；且只有过程所在工作表的行为可以触发该事件。

47.2 工作表事件

工作表事件发生在特定的 Worksheet 对象中。Worksheet 对象也是 Excel 最常用的对象之一，因此实际应用中经常会用到 Worksheet 对象事件。

47.2.1 Change 事件

工作表中的单元格被用户手工修改或被 VBA 代码修改时，将触发工作表的 Change 事件。值得注意的是，虽然事件的名称是 Change，但是并非工作表中单元格的任何变化都能触发该事件。

Change 事件的参数 Target 是 Range 变量，代表工作表中发生变化的区域，它可以是一个单元格也可以是多个单元格组成的区域。在实际应用中，用户通常希望只有工作表中的某些特定单元格区域发生变化时，才激活 change 事件，这就需要在 Change 事件中对 Target 参数进行判断。

示例 47.1 自动记录数据录入日期

在工作表 ChangDemo 的代码窗口中写入如下代码：

```
Private Sub Worksheet_Change(ByVal Target As Range)
    With Target
        ' 判断是否选中了单个单元格
        If .Count = 1 Then
            ' 判断单元格是否在第一列
            If .Column = 1 Then
```

```
        ' 禁止事件激活  
        Application.EnableEvents = False  
        ' 在相应行的第 2 列输入当前日期  
        Target.Offset(0, 1) = Date  
        ' 恢复事件激活  
        Application.EnableEvents = True  
    End If  
End If  
End With  
End Sub
```

返回 Excel 界面，在工作表 ChangDemo 中的 A 列中输入备忘内容，Change 事件将自动在 B 列的相应行写入当前日期，其结果如图 47-1 所示。修改工作表中其他列的单元格（如 C 列），工作表的 Change 事件同样会被触发，但是因为不满足代码中的判断条件，所以不会执行写入日期的代码。

| | A | B | C | D |
|---|--------------|-----------------|-------|---|
| 1 | 备忘 | 日期 | | |
| 2 | Excel 实战技巧精粹 | 2007 年 5 月 1 日 | | |
| 3 | Excel Home | 2007 年 5 月 2 日 | | |
| 4 | Taller | 2007 年 5 月 12 日 | | |
| 5 | | | I | |
| 6 | | | Love | |
| 7 | | | Excel | |
| 8 | | | | |

图 47-1 利用 Change 事件自动记录日期

如何禁止事件的激活

深入了解

上述代码使用 Application.EnableEvents=False 来防止事件被意外多次激活。Application 对象的 EnabledEvents 属性可以设置是否允许对象的事件被激活。上述代码中如果没有禁止事件激活的代码，在写入当前日期的代码执行后，工作表的 Change 事件被再次激活，事件代码被再次执行。某些情况下，这种事件的意外激活会重复多次发生，甚至造成死循环导致事件代码重复调用，无法结束运行。因此在可能意外触发事件的时候，需要设置 Application.EnableEvents=False 禁止事件激活。但这个设置并不能限制控件的事件被激活。

EnableEvents 属性的值不会随着事件过程的执行结束而自动恢复为 True，也就是说需要在代码运行结束之前进行恢复。如果代码被异常终止，而 EnableEvents 属性的值仍然为 False，则相关的事件都无法激活。恢复办法是在 VBE 的立即窗口中执行 Application.EnableEvents=True。

47.2.2 SelectionChange 事件

工作表中选定区域的范围发生变化将触发工作表的 SelectionChange 事件。SelectionChange 事件的参数 Target 是 Range 变量，代表工作表中被选中的区域，相当于 Selection 属性返回的 Range 对象。

示例 47.2 高亮显示工作表中选定区域所在的行和列

在工作表 SelectionChangeDemo 中写入如下的 SelectionChange 事件代码。

```
Private Sub Worksheet_SelectionChange(ByVal Target As Range)
    With Target
        ' 清除工作表单元格的背景色
        .Parent.Cells.Interior.ColorIndex = xlNone
        ' 设置选中区域所在行的背景色
        .EntireRow.Interior.Color = vbCyan
        ' 设置选中区域所在列的背景色
        .EntireColumn.Interior.Color = vbCyan
    End With
End Sub
```

返回 Excel 界面，在工作表 SelectionChangeDemo 中选中一个单元格区域 C10:C14，显示效果如图 47-2 所示，第 10 行至第 14 行以及第 3 列单元格高亮显示。

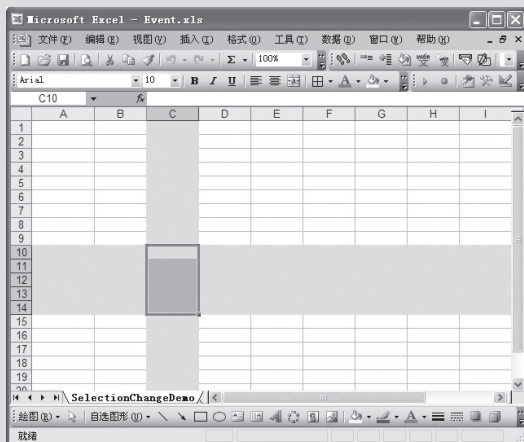


图 47-2 高亮显示选中区域所在行和列

47.3 工作簿事件

工作簿事件发生在特定的 Workbook 对象中。

47.3.1 Open 事件

Open 事件是 Workbook 对象最常用的事件之一，它发生于用户打开工作簿之时。

注意



在如下两种情况下，打开工作簿不会触发 Open 事件。

- 在按住 <Shift> 键的同时打开工作簿。
- 在打开文件时的宏安全警告提示框里，选择了“禁用宏”。


Open 事件经常被用来自动设置用户界面，这样做的好处在于，无论工作簿关闭时的状态如何，再次打开时都可以按照某个特定风格呈现在用户面前。

示例 47.3 自动设置工作簿打开时的界面风格

步骤 1 → 在 Thisworkbook 模块中写入如下的 Open 事件代码。

```
Private Sub Workbook_Open()  
    'Excel 窗口最大化  
    Application.WindowState = xlMaximized  
    With ActiveWindow  
        '工作表窗口最大化  
        .WindowState = xlMaximized  
        '禁止显示行标和列标  
        .DisplayHeadings = False  
    End With  
    '激活 Welcome 工作表  
    Sheets("Welcome").Select  
End Sub
```

步骤 2 → 返回 Excel 界面，选中 Sheet1 工作表。

步骤 3 → 单击 Excel 窗口右上角的向下还原按钮，取消窗体最大化。

步骤 4 → 单击“文件” “保存”。

步骤 5 → 单击“文件” “退出”关闭工作簿。

步骤 6 → 单击“文件” “打开”，再次打开刚才保存的工作簿。

步骤 7 → 单击安全警告提示框的“启用宏”按钮。

工作簿打开后，Excel 窗口是最大化的，Welcome 工作表成为活动工作表，而不是关闭工作簿时的 Sheet1 工作表，如图 47-3 所示。

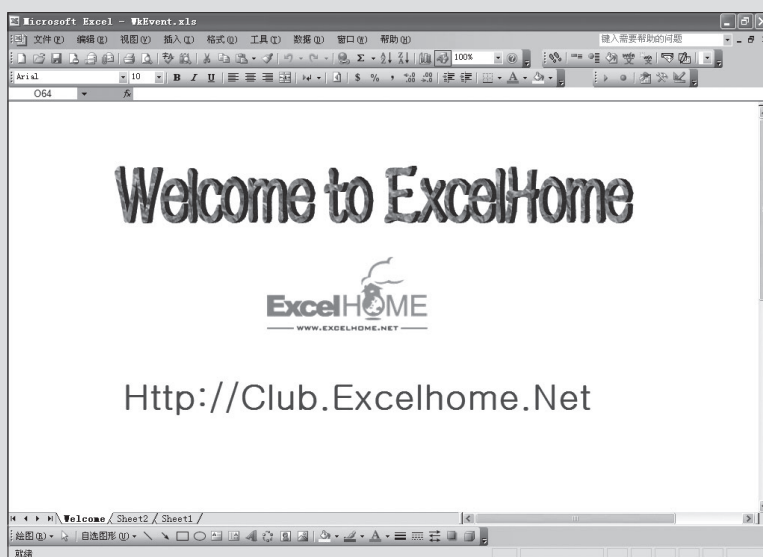


图 47-3 工作簿 Open 事件运行结果

47.3.2 BeforeClose 事件

工作簿被关闭之前 BeforeClose 事件被激活。BeforeClose 事件经常和 Open 事件配合使用，在 Open 事件中修改的 Excel 设置和用户界面，可以在 BeforeClose 事件中进行恢复。

示例 47.4 关闭工作簿时自动恢复 Excel 默认界面风格

在 Thisworkbook 模块中写入如下的代码：

```
Private Sub Workbook_Open()  
    With Application  
        ' 隐藏公式编辑栏  
        .DisplayFormulaBar = False  
        ' 设置鼠标指针为沙漏型  
        .Cursor = xlWait  
    End With  
End Sub  
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    With Application  
        ' 显示公式编辑栏  
        .DisplayFormulaBar = True  
        ' 恢复系统默认鼠标指针  
        .Cursor = xlDefault  
    End With  
End Sub
```

保存并关闭工作簿，然后再次打开工作簿，Excel 的窗口如图 47-4 所示，公式编辑栏已经隐藏且鼠标指针改为沙漏形。而在 BeforeClose 事件中，对相应的设置进行了恢复，所以工作簿关闭后，Excel 将恢复默认的系统设置。



图 47-4 隐藏公式编辑栏并设置沙漏形鼠标指针

47.3.3 全部工作表使用相同的事件代码

工作簿事件有几个名称是以“Sheet”开头的，这些事件的一个共同特点是，工作簿内的任意工作表的行为都将触发事件代码的执行。

如果希望所有的工作表都相应相同的工作表事件代码，有两种实现方法：

- 在每个工作表代码模块中写入相同的事件代码。
- 使用相应的工作簿事件代码。

毫无疑问，第二种方法是最简洁的实现方法。

示例 47.5 高亮显示任意工作表中选定区域所在的行和列

与示例 47.2 相对应, 如果希望在工作簿中的任意工作表都拥有这种高亮显示的效果, 可以在 Thisworkbook 模块中写入如下事件代码:

```
Private Sub Workbook_SheetSelectionChange(ByVal Sh As Object, ByVal Target As Range)
    With Target
        ' 清除工作表单元格的背景色
        .Parent.Cells.Interior.ColorIndex = xlNone
        ' 设置选中区域所在行的背景色
        .EntireRow.Interior.Color = vbCyan
        ' 设置选中区域所在列的背景色
        .EntireColumn.Interior.Color = vbCyan
    End With
End Sub
```

与示例 47.2 相比, 这种方法不必在每个工作表代码模块中写入相同的事件代码, 而且当工作簿中新增工作表时, 也无需为新建工作表添加 Change 事件代码。

47.4 非对象事件

Excel 提供了两种不与对象关联的特殊事件, 利用 Application 对象的相应方法可以设置这些特殊事件。

47.4.1 OnTime 事件

OnTime 事件指定一个过程在将来的特定时间运行, 此处的特定事件既可以是具体指定的某个时间点, 也可以是指定的一段时间之后。

示例 47.6 文件保存提醒

步骤 1 → 在工作簿中插入标准模块, 并在其中写入如下代码。

```
' 定义全局变量
Public iTime As Date
Sub SaveReminder()
    ' 判断当前工作簿是否被修改
    If ThisWorkbook.Saved = False Then
        ' 显示消息框
        If MsgBox(" 为了防止数据丢失请保存文件 " & _
            vbCrLf & " 单击 < 是 > 进行保存 ", vbYesNo, "OnTimeDemo") = vbYes Then
            ' 如果用户选择 < 是 >, 则保存当前工作簿
            ThisWorkbook.Save
        End If
    End If
    ' 记录下次运行的时间点
    iTime = Now + TimeValue("0:0:10")
```

```
' 设置 10 秒后再次运行 SaveReminder 过程  
Application.OnTime iTime, "SaveReminder"  
End Sub
```

步骤 2 → 在 Thisworkbook 中写入如下代码。

```
Private Sub Workbook_Open()  
    ' 记录下次运行的时间点  
    iTime = Now + TimeValue("0:0:10")  
    ' 设置 10 秒后再次运行 SaveReminder 过程  
    Application.OnTime iTime, "SaveReminder"  
End Sub  
  
Private Sub Workbook_BeforeClose(Cancel As Boolean)  
    ' 取消设置的自动运行  
    Application.OnTime iTime, "SaveReminder", Schedule:=False  
End Sub
```

步骤 3 → 保存并关闭工作簿。

重新打开工作簿，在 10 秒钟之后将看到如图 47-5 所示的工作簿保存提醒消息框。

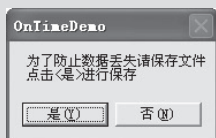


图 47-5 工作簿保存提醒消息框

Now 函数返回当前计算机系统设置的日期和时间；TimeValue("0:0:10") 函数返回一个 Date 类型数据，相当于 10 秒；SaveReminder 是标准模块中在指定时间执行的过程的名称。为了演示方便，示例中设定了较短的代码执行时间间隔。

首先在工作簿打开时会触发工作簿的 Open 事件，其中的 OnTime 设置 10 秒后运行 SaveReminder 过程。

在 SaveReminder 过程中，判断工作簿的 Saved 属性的值，如果为 False 说明工作簿已经被修改，进而提示用户进行保存，如果用户单击“是”按钮，就保存当前工作簿。在过程的最后，设置 10 秒后再次执行 SaveReminder 过程代码。

在工作簿的 BeforeClose 事件中，利用 Onkey 方法的 Schedule 参数清除已经设置的定时运行过程，如果省略此代码，即使工作簿已经关闭，到达指定时间时，Excel 将再次打开工作簿运行 SaveReminder 过程代码。

47.4.2 OnKey

使用 OnKey 方法可以设置按下特定的键或组合键时运行指定的过程代码。Excel 会一直监视着用户的任何键盘操作，因此理论上可设置任何一个键或组合键来运行指定的过程代码。

注意

在工作表中输入公式或在对话框中时，OnKey 设置的组合键无效。

示例 47.7 为 Excel 设置自定义快捷键

步骤 1 → 在工作簿中插入标准模块，在模块中写入如下代码。

```
Sub OnKeyDemo()  
    Application.OnKey "^a", "CtrlA"  
End Sub  
Sub CtrlA()  
    MsgBox "您按下了 Ctrl+A 组合键"  
End Sub
```

步骤 2 → 返回 Excel 界面，按 <Ctrl+A> 组合键，将出现如图 47-6 所示的消息框。



图 47-6 自定义按 <Ctrl+A> 组合键的消息框

OnKey 方法的参数“ ^a ”中的“ ^ ”代表 <Ctrl> 键，关于其他功能键的表示方法请参考 VBA 帮助。

默认情况下，Excel 中 <Ctrl+A> 组合键为选中工作表中的全部单元格。运行 OnKeyDemo 过程之后，按 <Ctrl +A> 组合键将执行 CtrlA 过程代码显示消息框。这也就是说，OnKey 方法设置的组合键与系统默认的组合键相比有更高的优先级。

使用如下的代码可以恢复 <Ctrl+A> 组合键的默认设置功能。

```
Application.OnKey "^a"
```