第 14 章　自定义 Excel 2007 的用户界面

Excel 2003 及以前版本使用菜单栏和工具栏进行操作，在 Excel 2007 中引进功能区（RibbonX）。Ribbon 将相关的命令和功能组合在一起，并划分为不同的选项卡，以及根据所执行的任务出现的选项卡。用户可以根据当前进行的操作，很容易找到当前需要使用的命令，而不必像原来一样，逐个地进行菜单查找。本章将讲解如何用 VBA 自定义 Excel 2007 的功能区。

14.1　Excel 2007 新界面介绍

Excel 2007 有新的外观，新的用户界面用简单明了的单一机制取代了 Excel 早期版本中的菜单、工具栏和大部分任务窗格。新的用户界面帮助用户在 Excel 中更高效、更容易地找到完成各种任务的合适功能、发现新功能并且效率更高。

14.1.1　功能区用户界面

Excel 2007 中最重大的改变是使用功能区代替了以前版本中的菜单栏和工具栏。功能区将相关的命令和功能组合在一起，并划分为不同的选项卡，以及根据所执行的任务出现相应的选项卡。例如，用户需要插入一个对象，则选择“插入”选项卡，在选项卡中包括可以插入的对象和相关命令；如果用户需要处理表对象，功能区将显示和表相关的选项卡。功能区比菜单和工具栏承载更加丰富的内容，如图 14-1 所示。

图 14-1　功能区

从图 14-1 可以看出，功能区包括以下几种组件。

- 功能区面板：由不同的内容组成，包括对话框、库、一些熟悉的工具栏按钮。它们都位于应用程序顶部且由类似的组件组成。
- 选项卡：位于功能区的顶部。标准的选项卡为“开始”、“插入”、“页面布局”、“公式”、“数据”、“审阅”、“视图”和“加载项”。默认的选项卡为“开始”。
- 组：位于每个选项卡内部。例如，“开始”选项卡中包括“剪贴板”、“字体”和

“对齐”等组，相关的命令组合在一起来完成各种任务。

- 命令按钮：每个组中的命令按钮执行一个命令或显示一个命令菜单，其表现形式有框、菜单或按钮，被安排在组内。

与标准的选项卡一样，可以在功能区中出现与当前任务相关的两类其他选项卡，即“上下文”选项卡和“程序”选项卡。

当用户在编辑艺术字、图表或表时，会出现上下文选项卡。例如，当用户选择图表对象时，在“图表工具”组中会出现“设计”、“布局”和“格式”3 个选项卡，这些“上下文”选项卡为操作图表提供了更多便利的命令，如图 14-2 所示。当用户没有选中这些对象时，与之相关的“上下文”选项卡也将隐藏。

图 14-2　“上下文”选项卡

在用户使用某种创作模式或视图时，程序选项卡会替换标准选项卡。例如，单击“打印预览”命令后，“打印预览”选项卡”将取代标准选项卡，如图 14-3 所示。

对于“开发工具”选项卡，用户必须在“Excel 选项”下的“常用”选项中选择“在功能区显示‘开发工具’选项卡”，则出现在功能区中。该选项卡中包含了与程序开发和 XML 功能相关的命令，如图 14-4 所示。

图 14-3　“打印预览”选项卡

图 14-4　“开发工具”选项卡

技巧：可以在任一选项卡上双击，隐藏功能区，再次双击将重新出现。在隐藏状态下，可单击某选项卡来查看功能区并选择其中的命令。

14.1.2　Office 按钮

在 Excel 2007 中，Office 按钮是位于界面左上角的一个圆形按钮。在 Office 按钮中，除之前版本的 Excel 文件菜单或标准工具栏中的常用命令，如“打开”、“保存”、“关闭”和“打印”命令以外，还包括“准备”和“发布”命令，如图 14-5 所示。

在 Office 按钮下拉菜单的右下角，找到“Excel 选项”按钮，允许自定义 Excel 中的一些功能。在这里，也可以找到之前版本的 Excel 中“工具”菜单下的“选项”命令里的大多数命令。

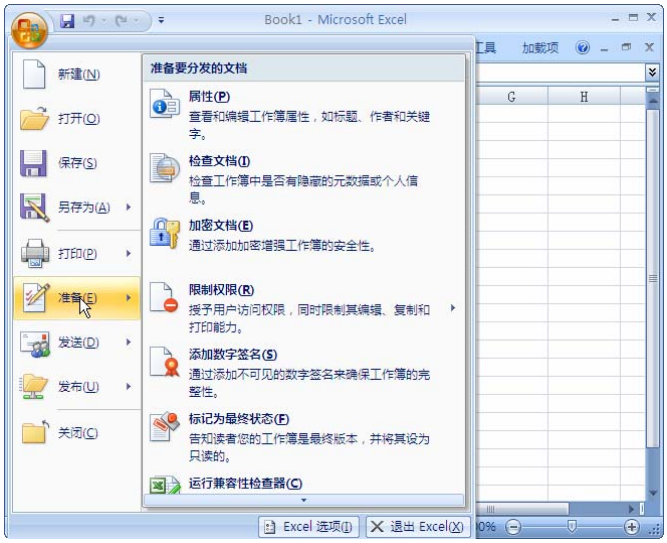

图 14-5　Office 按钮

14.1.3　向快速访问工具栏添加命令

Excel 2007 提供了一个工具栏，与显示的功能区选项卡独立，用于存放经常使用的命令。该工具栏称为快速访问工具栏，在默认状态下，位于功能区的上方、Office 按钮右侧，包含保存、撤销、恢复 3 个按钮。用户可以自定义快速访问工具栏，显示经常使用的命令，如图 14-6 所示。

图 14-6　默认状态下的快速访问工具栏

也可以将快速访问工具栏放置于功能区的下方，就像原来版本中的工具栏一样。在快速访问工具栏上右击，在弹出的下拉菜单中选择“在功能区下方显示快速访问工具栏”选项即可。

14.1.4　Excel 2007 新界面与 Ribbon

从前面的介绍中可知，Office 2007 最大的改变之一是 Ribbon3。在以前版本的 Office 中，用户可以使用 VBA 操作 CommandBars 对象模型的对象来创建菜单和工具栏。在实际开发中，经常需要成百甚至上千行 VBA 代码来维护添加、移除或重新整理菜单。

有时，最好的方式是使用表驱动的方式来创建菜单，即通过填充在工作表中的表来定义菜单和工具栏。但是仍然需要相当多的 VBA 代码，以确保仅当工作表被激活时自定义菜单是可见的，并且在关闭工作簿时必须移除定制的菜单。使用 Ribbon 设计 Excel 的用户界面的优点有以下几点。

- 用户界面在设计时就已经定义，而不是单独的编码。使用 XML 进行定义并在 XML

文件格式中存储作为一个自定义部分，而不是使用工作表中的表。

- 当打开工作簿时，Excel 自动读取 XML 部分并应用定制到 Ribbon 中。
- 如果使用一个标准工作簿，它的 Ribbon 定制仅当该工作簿激活时被应用并可见。
- 如果使用一个加载项工作簿，它的 Ribbon 定制总被应用并可用。
- 无论何时关闭工作簿，它的 Ribbon 定制都会自动删除。
- 即使定制在设计时被定义，大多数控件的属性可以使用 VBA 在运行时修改，如可见、标签等。
- 一些控件可以是完全动态的。因此，它们的结构和属性可以使用 VBA 在运行时定义。
- 所有内置的控件都是可用的，能够被重载、执行和查询它们的图像、标题等。

14.2　自定义菜单

功能区用户界面用更简单的界面系统来替换传统的分层菜单、工具栏和任务窗格。新的用户界面改进了快捷菜单、屏幕提示、浮动工具栏以及键盘快捷方式，可以提高用户的效率。用户可以使用 XML 来操作构成功能区用户界面的组件。因为 XML 是纯文本，所以可以使用任何文本编辑器来创建自定义文件。

此外，只需简单的调整就可以重用自定义功能区用户界面文件，因为每个应用程序都使用相同的编程模型。例如，可以重复使用在 Word 2007、Excel 2007、Access 2007 或 PowerPoint 2007 中创建的自定义 UI 文件。

在以前的 Office 版本中，用户使用 CommandBars 对象模型生成修改用户界面的 VBA 代码。在 Excel 2007 中，大多数情况下无需修改即可继续使用这些旧代码。但是，对 Excel 2003 中的工具栏所做的更改，现在显示在 Excel 2007 的“加载项”选项卡中。

14.2.1　使用 VBA 代码访问 Excel 的菜单

CommandBar 对象代表容器应用程序中的一个命令栏，如主菜单、“常用”工具栏和“格式”工具栏等命令栏，由 CommandBar 对象组成 CommandBars 对象集合。使用 CommandBars(index)可返回一个 CommandBar 对象，其中 index 是命令栏的名称或索引号。CommandBar 对象拥有众多的属性和方法，其常用的属性和方法如表 14-1 所示。

表 14-1　CommandBar对象的常用属性和方法

序号	名称	类型	说明
1	Delete	方法	从集合中删除 CommandBar 对象
2	FindControl	方法	获取一个符合指定条件的 CommandBarControl 对象
3	Reset	方法	将内置命令栏重置为其默认配置
4	ShowPopup	方法	将指定的命令栏作为快捷菜单，在指定坐标或当前光标位置显示

续表

序号	名称	类型	说明
5	AdaptiveMenu	属性	获取一个 Boolean 类型的值，该值指定命令栏是否应包含自适应菜单。可读写
6	BuiltIn	属性	如果指定的命令栏是容器应用程序的内置命令栏，则获取 True。如果是自定义命令栏，则返回 False。只读
7	Context	属性	获取或设置一个可确定命令栏保存位置的字符串。该字符串由应用程序定义和解释。可读写
8	Controls	属性	获取一个代表命令栏上的所有控件的 CommandBarControls 对象。只读
9	Creator	属性	获取一个 32 位整数，指示创建 CommandBar 对象时所使用的应用程序。只读
10	Enabled	属性	获取或设置用于指定是否启用了指定 CommandBar 的 Boolean 值。可读写
11	Height	属性	获取或设置 CommandBar 的高度。可读写
12	Index	属性	获取一个 Long 类型的值，该值代表集合中 CommandBar 对象的索引号。只读
13	Left	属性	设置或获取从对象左边缘算起 CommandBar 相对于屏幕的水平距离（以像素为单位）。可读写
14	Name	属性	获取内置的 CommandBar 对象的名称。只读
15	NameLocal	属性	获取以容器应用程序的语言版本显示的内置命令栏名称，或者返回或设置自定义命令栏的名称。可读写
16	Position	属性	获取或设置一个代表命令栏位置的 MsoBarPosition 常量。可读写
17	Protection	属性	获取或设置一个 MsoBarProtection 常量，代表防止用户对命令栏进行自定义的方式。可读写
18	RowIndex	属性	获取或设置一个命令栏相对于同一停靠区域中其他命令栏的停靠顺序。该属性值可以是大于零的整数，也可以是 msoBarRowFirst 或 msoBarRowLast 之一。可读写
19	Top	属性	设置或获取指定的命令栏顶边到屏幕顶边的距离。对于停靠命令栏，此属性返回或设置从命令栏到停靠区域顶部的距离。可读写
20	Type	属性	获取命令栏的类型。只读
21	Visible	属性	获取或设置命令栏的 Visible 属性。如果命令栏可见，则为 True。可读写
22	Width	属性	获取或设置指定命令栏的宽度（以像素为单位）。可读写

下例遍历命令栏集合以查找名为“Forms”的命令栏。如果找到，则显示该命令栏并保护其停靠状态。在本例中，变量 cb 代表一个 CommandBar 对象，代码如下。

```
foundFlag = False
For Each cb In ActiveSheet.CommandBars
    If cb.Name = "Forms" Then
        cb.Protection = msoBarNoChangeDock
        cb.Visible = True
        foundFlag = True
    End If
Next cb
If Not foundFlag Then
    MsgBox "命令栏集合中没有包含名为‘Forms’的命令栏"
End If
```

说明：这段代码无论在 Excel 2003，还是 Excel 2007 中都能很好的执行。但在 Excel 2007 的环境中，程序执行后，系统界面没有任何反应，并不会显示相应的命令栏。

14.2.2　创建自定义菜单

在 Excel 2003 及以前的版本中，工具栏及菜单等项目都是属于 Application 对象下的 CommandBars 集合对象；而在应用程序中的命令栏 CommandBar 对象则包含了菜单栏（Menu）、工具栏（ToolBar）及快捷菜单（PopUpMenu）等元素。它们之间的对象层次关系如图 14-7 所示。

下面将介绍这些对象以及如何利用这些对象创建自定义菜单。

1．CommandBarControl 对象

CommandBarControl 对象表示具体的命令按钮或菜单项，该对象是 CommandBarControls 集合中的成员。每个 CommandBar 对象包含一个 CommandBarControls 集合对象。CommandBarControl 对象拥有众多的属性和方法，CommandBarControl 对象的常用属性和方法如表 14-2 所示。

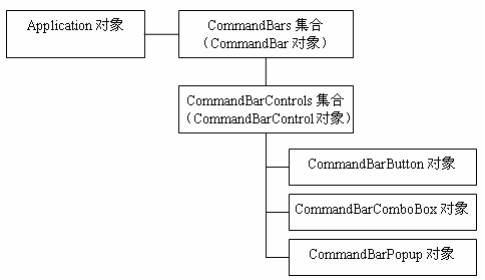

图 14-7　CommandBars 对象层次关系图

表 14-2　CommandBarControl对象的常用属性和方法

序号	名称	类型	说明
1	Copy	方法	将一个命令栏控件复制到已有的命令栏中
2	Delete	方法	将 CommandBarControl 对象从其集合中删除
3	Execute	方法	运行分配给指定 CommandBarControl 控件的过程或内置命令
4	Move	方法	将指定的 CommandBarControl 移动到已有的命令栏
5	Reset	方法	将内置命令栏重置为其默认配置，或将内置 CommandBarControl 重置为其初始功能和图符
6	SetFocus	方法	将键盘的焦点移到指定 CommandBarControl。如果该控件被禁用或不可见，那么此方法将失败
7	BeginGroup	属性	如果指定的命令栏控件显示在命令栏上控件组的最前面，则获取 True。可读写
8	BuiltIn	属性	如果指定的命令栏控件是容器应用程序的内置控件，则获取 True。如果是自定义控件或是已设置了 OnAction 属性的内置控件，则返回 False。只读
9	Caption	属性	获取或设置命令栏控件的标题文字。可读写
10	Enabled	属性	获取或设置指定是否启用 CommandBarControl 的 Boolean 值。可读写
11	Height	属性	获取或设置 CommandBarControl 控件的高度。可读写

续表

序号	名称	类型	说明
12	HelpContextId	属性	获取或设置附加于 CommandBarControl 的“帮助”主题的帮助上下文 ID 号。可读写
13	HelpFile	属性	获取或设置附加于 CommandBarControl 的“帮助”主题的文件名。可读写
14	Id	属性	获取内置的 CommandBarControl 的 ID。只读
15	Index	属性	获取一个 Long 类型的值，该值代表集合中 CommandBarControl 对象的索引号。只读
16	Left	属性	获取指定的 CommandBarControl 相对于屏幕左边缘的水平位置（以像素为单位）。返回距离停靠区域左侧的距离。只读
17	OLEUsage	属性	获取或设置特定的 OLE 客户端和 OLE 服务器角色，在合并两个 MicrosoftOffice 应用程序时，会用到这些角色中的 CommandBarControl。可读写
18	OnAction	属性	获取或设置一个 VB 过程的名称，该过程在用户单击或更改 CommandBarControl 的值时运行。可读写
19	Parameter	属性	获取或设置一个应用程序可用于执行 CommandBarControl 中命令的字符串。可读写
20	Tag	属性	获取或设置有关 CommandBarControl 的信息，例如，可作为过程参数的数据或用于识别该控件的信息。可读写

本小节将用到 Caption 和 OnAction 两个属性，下面详细介绍这两个属性的含义。

- Caption 属性：设置指定命令栏控件的题注文字。
- OnAction 属性：返回或设置 VBA 的宏或子过程名，该宏在用户单击或更改某命令栏控件的值时运行。

2．CommandBarControls 集合

CommandBarControls 集合对象有 CommandBarControl 对象组成，通过 Add 方法可新建一个 CommandBarControl 对象，并将其添加到指定命令栏上的控件集合中，其语法格式如下。

```
表达式.Add(Type, Id, Parameter, Before, Temporary)
```

该方法共有 5 个参数，全部为可选参数，其含义如下。

- Type：要添加到指定命令栏中的控件类型。可以为以下 msoControl 常量之一，即 msoControlButton（按钮）、msoControlEdit（文本框）、msoControlDropdown（下拉列表框）、msoControlComboBox（下拉组合框）或 msoControlPopup（快捷菜单）。
- Id：指定内置控件的整数。如果该参数为 1，或者忽略该参数，将在命令栏中添加一个空的指定类型的自定义控件。
- Parameter：设置参数信息，对于内置控件，容器应用程序使用该参数运行命令。对于自定义控件，可以使用该参数向 VB 过程传递信息，也可以用它存储控件的信息。
- Before：设置位置信息，一个指示新控件在命令栏上位置的数值。新控件将插入到位于此位置的控件之前。如果忽略该参数，控件将添加到指定命令栏的末端。
- Temporary：若设置为 True 将使新控件为临时控件。临时控件在关闭容器应用程序时自动删除。默认值为 False。

3．CommandBars 集合的 Add 方法

使用 CommandBars 对象的 Add 方法可新建一个命令栏并添加到命令栏集合，其语法格式如下。

```
表达式.Add(Name, Position, MenuBar, Temporary)
```

该方法共有 4 个参数，全部为可选参数，其含义如下。

- Name：新命令栏的名称。如果省略此参数，则为命令栏指定默认名称（如 Custom1）。
- Position：新命令栏的位置或类型。可以是 MsoBar 常量之一，即 mosBarLeft（位于应用程序的左侧）、mosBarTop（位于应用程序的上方）、mosBarRight（位于应用程序的右侧）、mosBarBottom（位于应用程序的下方）、mosBarFloating（表示命令栏是可移动的）、mosBarPopup（表示命令栏是一个快捷菜单）。
- MenuBar：若设置为 True，则将以新命令栏替换活动菜单栏。默认值为 False。
- Temporary：若设置为 True，则将使新命令栏变成临时命令栏。临时命令栏将在容器应用程序关闭时删除。默认值为 False。

根据前面介绍的内容，本小节将介绍创建自定义菜单。本例中，首先获取当前活动菜单栏，接着在活动菜单栏中添加一个下拉菜单，最后为下拉菜单添加菜单项，并设置各菜单项的名称、调用的子过程等参数，代码如下。

```
Sub AddItems()
Dim CommandBar1 As CommandBar                                    '声明变量，保存自定义菜单
    Dim myFirstMenu As Object
    Dim myFirstMenuitem As Object
    Set CommandBar1 = ActiveSheet.CommandBars.ActiveMenuBar      '获取活动菜单栏
    Set myFirstMenu = CommandBar1.Controls.Add Type = msoControlPopup,_
        Temporary =True                                          '在活动菜单后面添加一个下拉菜单
    myFirstMenu.Caption = "我的菜单"
    Set myFirstMenuitem   = myFirstMenu.Controls.Add(Type:=msoControlButton, ID:=1)
    myFirstMenuitem.Caption = "菜单项 1"
    myFirstMenuitem.Style = msoButtonCaption                     '设置为菜单项
    myFirstMenuitem.OnAction = "method1"                         '设置调用的子过程
    Set myFirstMenuitem   = myFirstMenu.Controls.Add(Type:=msoControlButton, ID:=2)
    myFirstMenuitem.Caption = "菜单项 2"
    myFirstMenuitem.Style = msoButtonCaption
    myFirstMenuitem.OnAction = " method 2"
    Set myFirstMenuitem   = myFirstMenu.Controls.Add(Type:=msoControlButton, ID:=3)
    myFirstMenuitem.Caption = "菜单项 3"
    myFirstMenuitem.Style = msoButtonCaption
    myFirstMenuitem.OnAction = " method 3"
End Sub
```

下面给出了每个菜单项的子过程，代码如下。

```
Sub method1()
    MsgBox "您选择的是“菜单项 1”。", , "创建自定义菜单"
End Sub
Sub method2()
    MsgBox "您选择的是“菜单项 2”。", , "创建自定义菜单"
End Sub
Sub method3()
    MsgBox "您选择的是“菜单项 3”。", , "创建自定义菜单"
```

```
End Sub
```

在 Excel 2003 中，程序运行后，可以看到菜单栏中多出了“我的菜单”选项，如图 14-8 所示。

图 14-8　Excel 2003 中的自定义菜单

在 Excel 2007 中运行代码后，自定义菜单将添加到功能区的“加载项”选项卡中，如图 14-9 所示。

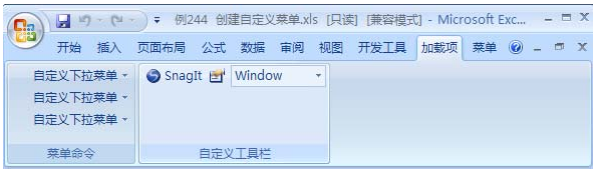

图 14-9　Excel 2007 中的自定义菜单

14.2.3　创建快捷菜单

Excel 的快捷菜单有很多，在不同的对象上右击，将弹出不同的快捷菜单项。本小节将介绍如何利用 VBA 代码控制快捷菜单。

1．在快捷菜单中添加菜单项

在快捷菜单中添加菜单项如同往常规的菜单中添加菜单项一样。下例演示了如何在 Cell 快捷菜单中添加菜单项，代码如下。

```
Dim NewItem As CommandBarControl
Set NewItem = CommandBars("Cell").Controls.Add
With NewItem
    .Caption = "新快捷菜单项"
    .OnAction ="MethodForCell"
    .BeginGroup =true
End With
```

当在单元格上右击时，将出现这个快捷菜单，将把新的菜单项添加到这个快捷菜单的末尾，上面用一个分隔栏分隔，如图 14-10 所示。

2．禁用快捷菜单项

在设计 Excel 的应用程序时，可能希望在应用程序运行期间禁用某个特定快捷菜单上的一个或者多个菜单项。当禁用菜单中的某一项时，它就会变成浅灰色，单击它不会有任何反应。下例运行后，将禁用 Excel 的行和列的快捷菜单中的“隐藏”菜单项，代码如下。

```
CommandBars("Column").Controls("Hide").Enabled = False
```

```
CommandBars("Row").Controls("Hide").Enabled = False
```

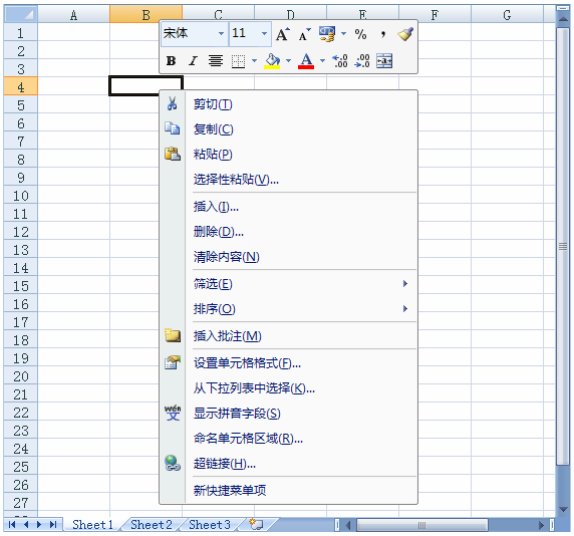

图 14-10　往快捷菜单中添加菜单项

3．禁用快捷菜单

还可以禁用整个快捷菜单。例如，如果不希望用户通过右击单元格访问到通常可以访问的命令，可以使用来禁用 Cell 快捷菜单，代码如下。

```
CommandBars("Cell").Enabled = False
```

说明：这段程序执行之后，在单元格上右击就不会有任何反应了。

14.3　自定义功能区

功能区作为 Office 2007 新引进的用户界面，也可由用户进行定制。不过，定制功能区并不需要通过编写 VBA 代码来实现，而是用 XML 编写相应的代码来完成。如果有必要的话，再通过 VBA 代码编写完成功能区控件的相应功能。

14.3.1　使用记事本定制功能区选项卡

功能区作为 Office 2007 新引进的用户界面，也可由用户进行定制。定制功能区不是通过 VBA 编写代码，而是用 XML 编写相应的代码来完成，也称这种代码为 RibbonX 代码。有必要的话，再通过 VBA 代码编写完成功能区控件的相应代码。

RibbonX 代码是 XML 代码，遵守 Microsoft 提供的 XML 架构。所谓 XML 是描述数据的一种方式，一个 XML 文件也可以包含数据。要定制 Office Ribbon UI，用 XML 描述每件事情。所有关于尺寸、位置、可见性、标签、ID 等，习惯于在代码中指定这些操作的开发者应在 RibbonX XML 格式中指定。用户必须学习这些，以便于可以定制新的 Ribbon UI。

本小节以在功能区中添加一个新的选项卡，并在该选项卡中放置两个内置的组为例，说明如何使用记事本定制功能区选项卡。

1．创建自定义 UI 文件

可以使用普通的文本编辑工具，创建一个 UI 文件，其步骤如下。

（1）创建一个名为 customUI 的文件夹。

（2）在该文件夹中，创建一个名为 customUI.xml 的文本文件。

（3）打开该文本文件，并输入如下程序代码。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customUI">
    <ribbon startFromScratch="false">
        <tabs>
            <tab id="rxtabCustom"
                 label="my Tab"
                 insertBeforeMso="TabHome">
                <group idMso="GroupFont">
                </group>
                <group idMso="GroupZoom">
                </group>
            </tab>
        </tabs>
    </ribbon>
</customUI>
```

（4）保存该文件。

2．添加 XML 到文件中

Office 2003 及以前的版本使用二进制文件存储文档，而 Office 2007 中采用了 OpenXML 文件格式作为新的标准，新的文件实际上被包含在被压缩的 XML 格式中。Excel 2007 工作簿的默认格式是基于 Open XML 文件格式（.xlsx），其他基于 OpenXML 格式是基于 XML 且启用宏工作簿格式（.xlsm）、Excel 模板格式、启用宏模板和启用宏加载项。

将 Office 2007 文件的扩展名改为.zip 后，可以直接用 WinRAR 打开，可看到该压缩包中有 3 个文件夹和一个文件，如图 14-11 所示。

图 14-11　用 WinRAR 查看 Office 2007 文件

Office 2007 新文件格式是 Office 2007 应用程序使用.zip 压缩技术来存储文档。在打开文件时，这种格式可以自动解压缩，而在保存文件时，这种格式又可以重新自动压缩，这些解压和压缩的操作都是在后台进行的，对用户是透明的。所有的 OpenXML 文件都是压缩容器，

按照 RibbonX 代码读取的顺序放置在压缩容器内。

（1）选择 Excel 2007 文件，例如，选择文件“花名册.xlsx”，右击该文件，在弹出的快捷菜单中选择“重命名”命令，保存完整的文件名，但是在其后面添加.zip 扩展名。现在该文件改成了 zip 压缩文件，代替刚才标准的 Excel 文档文件。

（2）双击该压缩文件，将其打开。

（3）拖动上一步编辑的 customUI 文件夹，并将其放置在压缩文件夹中。

（4）拖动压缩文件夹中的_rels 文件夹到桌面，编辑.rels 文件链接 UI 更改，指定该文件和 customUI 文件夹之间的联系。

（5）打开_rels 文件夹并使用记事本编辑其.rels 文件。在该文件末尾元素之前，即在</Relationships>之前插入，插入如下程序代码。

```
<Relationship Id="customUIRelID"
    Type="http://schemas.microsoft.com/office/2006/relationships/ui/extensibility"
    Target="customUI/customUI.xml"/>
```

注意：在输入语句时，应非常小心地使用标点符号、空格和大小写。

（6）保存该.rels 文件，关闭记事本。

（7）在压缩文件夹中，删除原来的_rels 文件夹。

（8）将编辑过的_rels 文件夹拖回到压缩文件夹中。

（9）关闭压缩文件夹。

（10）在压缩文件中右击，在弹出的快捷菜单中选择“重命名”命令，去掉名称后面的.zip 扩展名，恢复为“花名册.xlsx”文件名。

（11）打开“花名册.xlsx”文件。

此时，如果出现错误消息，重新检查.rels 文件和 customUI.xml 文件，可能由于代码输入错误而导致错误。操作完成后，在“开始”选项卡左侧会出现一个名为“My Tab”的新选项卡，包含内置的“字体”选项组和“显示比例”选项组，即一个内置组的副本，如图 14-12 所示。

图 14-12　添加一个新的选项卡

需要引起读者注意的是，选项卡的名称并不支持中文。笔者将“My Tab”换成中文，试验了多次都没有成功。

14.3.2　使用 Microsoft Office 2007 Custom UI Editor 修改 UI

Microsoft Office 2007 Custom UI Editor 是一个用来编辑 OpenXML 文件的小型工具，方便且实用。下载地址为：http://openxmldeveloper.org/articles/customuieditor.aspx。

1．为 Windows XP 用户安装 Microsoft .NET Framework 2.0

首先，确保系统中已安装了 Microsoft .NET Framework 2.0。可以通过选择“控制面板”中的“添加或删除程序”命令，在“添加或删除程序”对话框中查找是否有“Microsoft .NET Framework 2.0”选项。如果没有安装框架，应当先安装 Microsoft .NET Framework 2.0。

2．安装 Microsoft Office 2007 Custom UI Editor

下载后，双击图标即可安装 Microsoft Office 2007 Custom UI Editor。然后打开 Custom UI Editor，在该编辑器中打开“花名册.xlsx”文件，如图 14-13 所示。

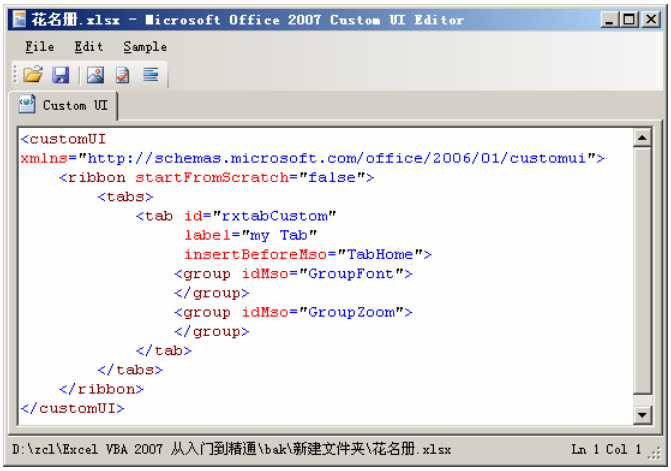

图 14-13 “Microsoft Office 2007 Custom UI Editor”编辑器

从图中可以看到，该编辑器中的文件使用了颜色来区分不同的标记文本，并使用了缩排，版面清新自然。使用这个编辑器的优势在于：颜色能帮助阅读和解释代码，容易嵌入图片、验证代码、存储代码片断，甚至生成响应的回调的代码框架（回调，即当单击自定义的 Ribbon 控件时触发的 VBA 程序）。

3．CustomUI Editor 的使用方法

虽然使用 CustomUI Editor 能够很容易地编辑 XML 代码，但在使用前还是要对其进一步了解。

（1）CustomUI Editor 不会检查 XML 标记的形式，因此应确保只使用在 XML 架构中定义的属性。然而，CustomUI Editor 能核查在引号内提供的有效的属性。

（2）在编写和调试 RibbonX 代码时，不能够同时打开要定制的应用程序文件和 CustomUI Editor。当试图在 CustomUI Editor 中保存在 Office 应用程序中打开的文件将导致错误。此外，即便关闭了正在编辑的 Office 应用程序文档然后将其在 CustomUI Editor 中保存，CustomUI Editor 仍将覆盖在应用程序中编辑文档所作的任何变化。在其他的工具中做出变化之前，关闭应用程序是更安全的。

（3）CustomUI Editor 没有查找/替换工具，因此，如果对 XML 作大量的编辑，可以先复制信息到另一个应用程序，编辑后再复制回来。

（4）当处理多行 XML 的文件时，CustomUI Editor 习惯刷新屏幕使光标总是在屏幕的最

后一行。

4．使用 CustomUI Editor 来定制功能区

这里，将使用 CustomUI Editor 来定制 Excel 的功能区，使用与前面相同的代码，以展示用 CustomUI Editor 功能区时的优势。

（1）打开 CustomUI Editor，在该编辑器中打开“花名册.xlsx”工作簿，然后输入前面小节中在记事本中输入的代码。

（2）验证代码。单击编辑器工具栏右侧第二个按钮（Validate），此时如果有错误，将会指出存在的错误，如果没有错误，则弹出如图 14-14 所示的对话框。

图 14-14　“验证代码”提示框

（3）保存并关闭 CustomUI Editor。

（4）打开“花名册.xlsx”工作簿，已在该工作簿的功能区中添加了新的选项卡，如图 14-15 所示。

图 14-15　添加选项卡

14.3.3　RibbonX 和 VBA

用户除了可以自定义选项卡外，同样可以为每个选项添加对应的回调函数。回调意味着当对 RibbonX 定义的控件进行单击、变化等操作时，就会运行与控件相对应的过程，这与前面介绍的 Application.OnKey、CommandBarButton.OnAction 等原理相同。在 RibbonX 中，主要的不同在于传递到所调用的函数的参数，因此函数必须被正确地声明以便于调用。而且，这与用户窗体控件事件过程的代码也不同。

用户可以使用调用在运行时改变控件属性。不是在 XML 中为属性定义一个特定的值，而是提供 Excel 应该调用的过程名。例如，在前面自定义的组中，可以使用特定的方法代替 label="Miscellaneous"属性，代码如下。

```
<group id="rxAuditMisc" getLabel="rxAuditMisc_getLabel">
```

说明：当选项卡第一次显示时，Excel 将调用 rxAuditMisc_getLabel VBA 过程（在一个标准模块中），该过程提供了为 Excel 使用的文本。如果想在稍后的某个时候改变文本，不能

只更新对象的属性。RibbonX 没有对象模型，取而代之的是，有一个接口告诉 Excel 先前所提供的信息不再有效，当 Excel 需要显示组或控件时，再调用过程来获取新值。

14.4 设置 Office 按钮菜单

Office 按钮菜单也是 Excel 2007 用户界面的一个组成部分，本节将介绍如何使用 XML 代码定制 Excel 按钮菜单。

14.4.1 禁用“Office 按钮”菜单

单击“Office 按钮”打开下拉菜单，每个菜单为一个<Command>对象。在标签<Command>中，通过 idMso 引用具体的菜单项，如“另存为”菜单项的 idMso 为“FileSaveAs”。下例给出了如何引用“另存为”菜单项的方法，代码如下。

```
<command idMso = "FileSaveAs" />
```

图 14-16 禁用 Office 按钮的菜单

通过 Enabled 属性可设置指定的菜单项是否可用。下例列出了禁止“Office 按钮”下拉菜单中部分菜单项的 XML 代码，代码如下。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui">
<commands>
    <command idMso="FileSaveAs" enabled = "false" />
    <command idMso="FileSaveAsMenu" enabled = "false" />
    <command idMso="FilePrint" enabled = "false" />
    <command idMso="FilePrintPreview" enabled = "false" />
</commands>
</customUI>
```

按前面介绍的方法将含有这段代码的 customUI.XML 放置到相应的 Excel 2007 文件中，即可实现禁用“Office 按钮”下拉菜单中部分菜单项的功能，如图 14-16 所示。

14.4.2 在“Office 按钮”中新建菜单

当用户使用 XML 自定义“Office 按钮”时，可能用到的 XML 标记语言有以下几个。

- 使用<officeMenu>标签表示“Office 按钮”。
- 使用<button>标签表示具体的菜单项。
- 使用<menu>标签表示弹出菜单。

下例演示了如何新建菜单，代码如下。

```
<customUI xmlns="http://schemas.microsoft.com/office/2006/01/customui" >
 <ribbon>
     <officeMenu>
       <button id="Button1" label="Custom Menu" onAction="sub1"
          imageMso="FindRelatedMenu" />
     </officeMenu>
 </ribbon>
</customUI>
```

接下来，在本例的工作簿的 VBE 中插入一个模块，在模块中编写响应各菜单项的子过程。下面的代码将响应“Custom Menu”菜单，显示一个提示信息框。

```
Sub Sub1()
     MsgBox "您好，您选择的是【Custom Menu】菜单项",,"信息提示"
End Sub
```

单击“Office 按钮”打开下拉菜单，可看到新增加的菜单项，如 14-17 所示。单击“Custom Menu”菜单，将调用相应的 VBA 子过程，弹出提示对话框，如图 14-18 所示。

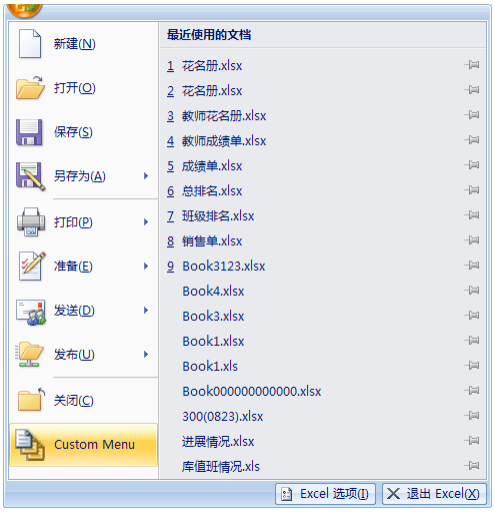

图 14-17　新增的菜单项

图 14-18　执行菜单功能

14.5　小结

本章仅仅围绕 Excel 2007 的用户界面，重点介绍了如何通过 XML 代码自定义 Excel 2007 的功能区，以及如何通过 VBA 代码自定义传统的 Excel 菜单和工具栏。通过本章的学习，希望读者通过 RibbonX，很容易地创建自定义选项卡、组、菜单、按钮、切换按钮、下拉项、库、复选框以及动态菜单以添加应用程序的功能到 Ribbon 中。