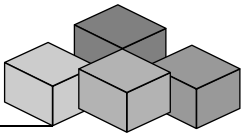
第 3 章 处理压缩文件



在日常办公中，经常用到文件的压缩与解压缩，这些操作也可以用 VBA 实现自动化。另外，Office 2007 以上版本创建的文档其实也是压缩包文件，只不过扩展名看起来是 Office 的扩展名，因此，学习压缩和解压缩的知识，有助于理解 Office 开发中自定义 Office 界面方面的知识。

文件的压缩和解压缩操作有以下两个主要方式。

- Shell 函数调用电脑默认的压缩工具。
- 使用 Shell32 对象操作 .zip 压缩文件。

第一种方式通用性比较强，几乎可以操作任意扩展名的压缩包，缺点是计算机必须安装了压缩软件。

针对第二种方式，不需要安装压缩软件，用代码即可实现压缩和解压缩，但只限于扩展名为 .zip 的压缩包。

本章包括用 Shell 函数调用 WinRAR 压缩软件以及使用 Shell32 对象操作 .zip 压缩文件两大部分内容。

本章用到的外部引用和重要对象如下。

- Microsoft Shell Controls And Automation
  - Shell32.Shell

3.1 Shell 调用 WinRAR

WinRAR 是一个文件压缩管理共享软件，由 Eugene Roshal（所以 RAR 的全名是 Roshal ARchive）开发。首个公开版本 RAR 1.3 发布于 1993 年。

WinRAR 可以把文件（夹）压缩为 .rar 或 .zip 格式，如图 3-1 所示。

WinRAR 可以解压的格式有：.CAB、.ARJ、.LZH、.TAR、.GZ、.ACE、.UUE、.BZ2、.JAR、.ISO、.Z、.7Z、.RAR5。

启动 WinRAR 软件，单击 WinRAR 软件的菜单【选项 / 设置】，弹出“设置”对话框，切换到“集成”选项卡，可以设置 WinRAR 能够解压的文件格式，如图 3-2 所示。

图 3-1 “压缩文件名和参数”对话框

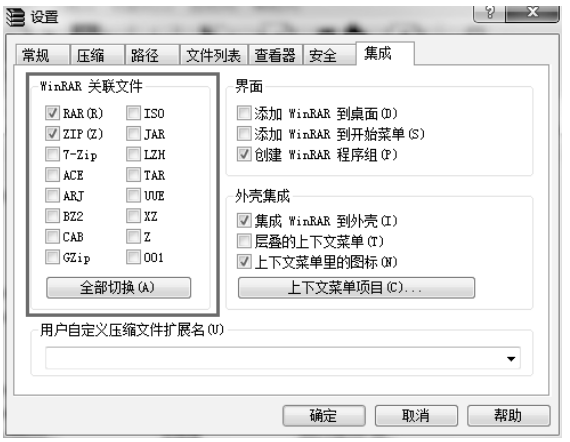


图 3-2 勾选关联的扩展名

3.1.1 获取 WinRAR 可执行文件路径

WinRAR 的执行文件一般情况下位于 C:\Program Files\WinRAR\WinRAR.exe，如果个别计算机把这个软件安装到其他位置，使用前面讲过的 WshShell 对象的 RegRead 方法读取注册表可以获取其路径。

下面的 GetSetupPath 函数用来获取指定程序名的安装路径。

```
Public Function GetSetupPath(AppName As String)
    Dim WSH As Object
    Set WSH = CreateObject("WScript.Shell")
    GetSetupPath = WSH.RegRead("HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\
CurrentVersion\App Paths\" & AppName & "\Path")
    Set WSH = Nothing
End Function
```

运行下面的过程，可以获取 WinRAR 软件的安装路径以及 PowerPoint 的安装路径。

```
Sub Test()
    Debug.Print GetSetupPath("WinRAR.exe")
    Debug.Print GetSetupPath("Powerpnt.exe")
End Sub
```

上述程序的运行结果如图 3-3 所示。

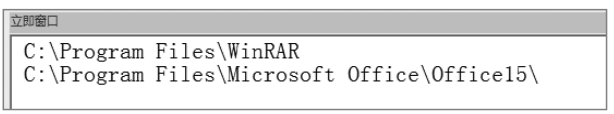


图 3-3 从注册表中获取应用程序的所在路径

3.1.2　命令和开关

获取到 WinRAR.exe 的所在路径，就可以使用 Shell 函数调用这个可执行文件完成压缩和解压缩操作。

调用格式如下。

```
Shell "WinRAR.exe 的路径 命令 开关 压缩包路径 文件路径",vbNormalFocus（或者 vbHide）
```

下面通过一个实例来介绍一下各参数的构造方法。

```
Shell "C:\Program Files\WinRAR\WinRAR.exe A C:\temp\Regdll.rar C:\temp\65.png", vbNormalFocus
```

以上语句的功能是，调用 WinRAR.exe 把 65.png 图片文件压缩到 Regdll.rar 这个压缩包中。可以看出各个参数之间用空格隔开，全部放入双引号内，形成了一个长的字符串。其中的 A 就是一个命令，表示压缩，上面这个实例没有用到开关参数。

下面分别介绍一下 WinRAR 的命令参数和开关参数。命令参数的功能是告知 WinRAR 要执行什么操作，是压缩、解压缩还是删除。开关参数是对命令参数的补充说明。

WinRAR 命令参数

单击 WinRAR 的菜单【帮助 / 帮助主题】，可以打开其帮助文件，依次展开节点“命令行模式 / 命令行”，可以看到所有命令参数的说明，如图 3-4 所示。

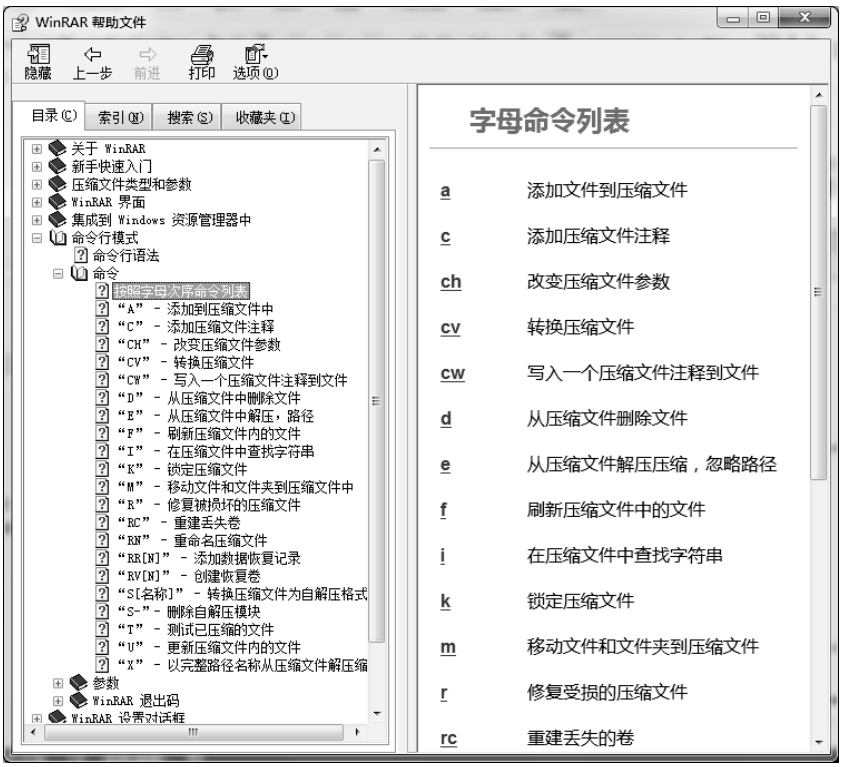


图 3-4　WinRAR 的命令参数帮助

最常用的 4 个命令参数及其功能如下。

- A：压缩，添加到压缩文件中。
- D：删除，从压缩包中删除文件。
- E：解压缩到当前目录。
- X：以完全路径解压。

可以看出，从压缩包中解压出内容，有 E 和 X 两个命令参数。其实，E 命令参数等价于 WinRAR 解压参数中的“不要提取路径”；X 命令参数等价于“提取完整路径”，使用 WinRAR 软件解压一个压缩包时，在“高级”选项卡里可以看到解压方式选项，如图 3-5 所示。

图 3-5　命令参数相应的含义

简言之，E 就是忽略压缩包中的路径，释放所有文件到目标文件夹，而 X 则按照压缩包原有的路径结构释放到目标文件夹。

WinRAR 开关参数

在 WinRAR 软件的帮助文件中，依次展开节点“命令行模式 / 参数”，可以看到所有开关参数的说明，如图 3-6 所示。

下面是比较常用的开关参数。

- -ep：忽略路径。
- -ep1：忽略基准路径，但保持现有文件层次结构。
- -p 或 -hp：压缩时加密码。
- -df：压缩后删除原文件。
- -dr：压缩后删除原文件到回收站。

大致了解命令参数和开关参数后，下面通过具体实例加深学习。

图 3-6　WinRAR 开关参数帮助

3.1.3　压缩

假设文件夹“东北三省”中的内容如图 3-7 所示。

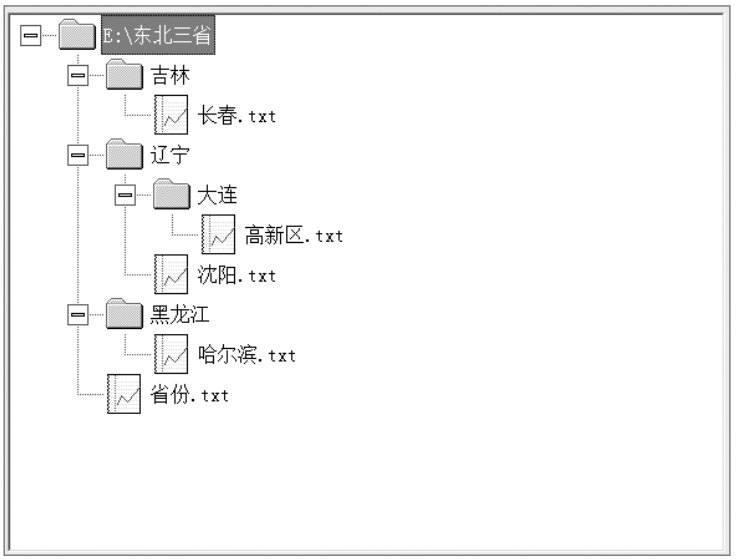


图 3-7　文件夹内容的示意图

如果采用命令 A-ep 则是忽略所有路径，也就是忽略文件夹及其子文件夹，把“东北三省”下面管辖的所有文件（含递归）压缩进去。完整代码如下。

```
    Public Function GetSetupPath(AppName As String)
        Dim WS As New IWshRuntimeLibrary.WshShell   ' 前期绑定
        Set WS = CreateObject("WScript.Shell")
        GetSetupPath = WS.RegRead("HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\
CurrentVersion\App Paths\" & AppName & "\Path")
        Set WS = Nothing
    End Function
    Private Function AddQuote(path As String) As String
        AddQuote = Chr(34) & path & Chr(34) & " "
    End Function

    Sub 文件夹及其内容添加到压缩包 ()
        Const WorkDir As String = "E:\"      ' 默认目录
        Dim WinrarExe As String              'WinRAR 可执行文件路径
        Dim Command As String                ' 命令、开关参数
        Dim RarFile As String                ' 压缩包路径
        Dim Contents As String               ' 文件、文件夹的路径
        WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
        Command = " A -ep "
        RarFile = WorkDir & "package1.rar"
        Contents = WorkDir & " 东北三省 "      ' 把 " 东北三省 " 文件夹及其内容添加到压缩包
        Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & AddQuote (Contents),
vbNormalFocus
    End Sub
```

代码分析：函数 GetSetupPath 用来获取 WinRAR 软件的安装目录，函数 AddQuote 用来处理 Shell 命令路径中的空格。要注意 Command 中要保留必要的空格。

以上代码段中，最重要的一句就是最后 Shell 函数的应用。

运行上面的“文件夹及其内容添加到压缩包”过程，会在 E: 盘下生成 package1.rar 压缩包，手工打开后，如图 3-8 所示。

图 3-8　自动执行压缩

可以看到，A-ep 命令把文件夹中所有的“文件”掏出来，放入压缩包，而不管这些文件原先在何处。

现在只需把上述代码中的 Command 换成 "A-ep1"，删除原先的压缩包，再运行一次程序，效果如图 3-9 所示。

图 3-9　连文件夹一起压缩

可以看出，该命令保留了原先文件结构。因此，可以简单地理解为 -ep 参数只压缩文件，-ep1 带文件夹压缩。

在实际应用中，根据需要选择开关参数即可。

3.1.4　解压缩

解压缩是将压缩包中的内容释放到磁盘下的操作。解压缩的命令有 E 和 X。

下面的过程把压缩包 package1.rar 中所有的文件解压到 Destination 文件夹下。

```
Sub 解压 ()
    Const WorkDir As String = "E:\"      '默认目录
    Dim WinrarExe As String              'WinRAR 可执行文件路径
    Dim Command As String                '命令、开关参数
    Dim RarFile As String                '压缩包路径
    Dim Contents As String               '文件、文件夹的路径
    WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
    Command = " E "
    RarFile = WorkDir & "package1.rar"
    Contents = WorkDir & "Destination"  '释放到 Destination 路径下
    Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & AddQuote (Contents),
vbNormalFocus
End Sub
```

运行上述程序，把压缩包中的文件直接释放到目标文件夹，如图 3-10 所示。

如果把 Command 改为 Command = " X "，再次运行上述程序，压缩包中的文件夹和文件一律解压到目标文件夹中，如图 3-11 所示。

上面的实例把压缩包中所有内容解压到目标文件夹，使用以下代码可以解压压缩包中指定路径的文件，而不是解压全部文件。

```
Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & " 东北三省 \ 辽宁 \*.* "
& AddQuote(Contents), vbNormalFocus
```

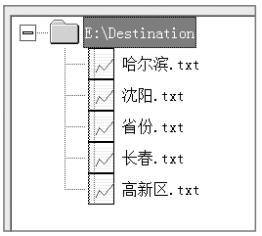


图 3-10　自动解压

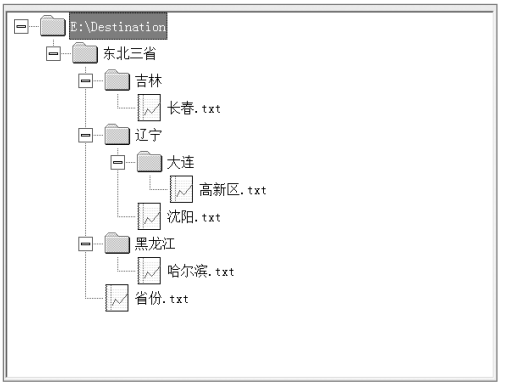


图 3-11　连文件夹一起解压

3.1.5　删除

WinRAR 使用命令 D 删除压缩包中的文件或路径。

下面的实例把 package1.rar 压缩包中的“吉林”文件夹删除。

```
Sub 删除压缩包中内容()
    Const WorkDir As String = "E:\"                    '默认目录
    Dim WinrarExe As String                            'WinRAR 可执行文件路径
    Dim Command As String                              '命令、开关参数
    Dim RarFile As String                              '压缩包路径
    Dim Contents As String                             '压缩包中待删除的文件、文件夹的路径
    WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
    Command = " D "
    RarFile = WorkDir & "package1.rar"
    Contents = "东北三省\吉林"   '把"东北三省\吉林"压缩包中的路径删除
    Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & AddQuote (Contents),
vbNormalFocus
End Sub
```

运行上述程序，删除压缩包中的“吉林”文件夹，如图 3-12 所示。

图 3-12　删除压缩包中指定的文件夹

3.1.6　使用通配符

无论是压缩、解压缩，还是删除命令，路径设置中均可使用通配符。* 表示 0 个以上任

意字符，? 表示 1 个任意字符。

下面的程序把文件夹下所有 4 位扩展名的 Word 文档添加到 Word.rar 压缩包中。

```
Sub 指定类型的文件添加到压缩包 ()
    Const WorkDir As String = "C:\temp\"          ' 默认目录
    Dim WinrarExe As String                       'WinRAR 可执行文件路径
    Dim Command As String                         ' 命令、开关参数
    Dim RarFile As String                         ' 压缩包路径
    Dim Contents As String                        ' 文件、文件夹的路径
    WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
    Command = " A -ep "
    RarFile = WorkDir & "Word.rar"
    Contents = WorkDir & "*.doc?" ' 把四位扩展名的 Word 文档 (docx、docm) 添加到压缩包。
    Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & AddQuote (Contents),
vbNormalFocus
End Sub
```

运行上述程序，将文件夹中所有的 Word 文档添加到压缩包，如图 3-13 所示。

图 3-13　批量压缩特定类型的文件

下面的代码从 Word.rar 压缩包中删除启用宏的 Word 文档（扩展名为 .docm）。

```
Sub 删除压缩包指定类型的文件 ()
    Const WorkDir As String = "C:\temp\"      ' 默认目录
    Dim WinrarExe As String                   'WinRAR 可执行文件路径
    Dim Command As String                     ' 命令、开关参数
    Dim RarFile As String                     ' 压缩包路径
    Dim Contents As String                    ' 文件、文件夹的路径
    WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
    Command = " D "
    RarFile = WorkDir & "Word.rar"
    Contents = "*.docm"                       ' 删除扩展名为 docm 的文件
    Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & AddQuote(Contents),
vbNormalFocus
End Sub
```

需要注意的是，由于是从压缩包里面删除内容，所以代码中 Content 的赋值不需要 WorkDir。

运行上述程序，删除压缩包中扩展名为 docm 的文件，如图 3-14 所示。

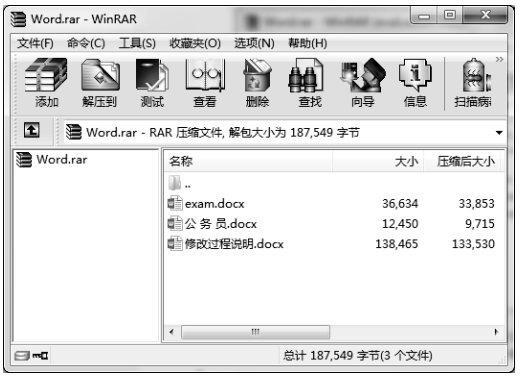


图 3-14　从压缩包中删除指定扩展名的文件

3.1.7　处理压缩包的密码

使用 -p 或 -hp 开关参数，可以压缩为加密的文件，也可以从添加了密码的压缩包中解压文件。-p 后面带上密码，表示普通加密，手工双击压缩包，会看到压缩包中的文件列表，每个文件后面有 *。

下面的程序把文件夹中所有扩展名为 .pdf 的文件添加到压缩包，并且设置解压密码为 ryueifu。

```
Sub 加密的压缩包()
    Const WorkDir As String = "C:\temp\"          '默认目录
    Dim WinrarExe As String                       'WinRAR 可执行文件路径
    Dim Command As String                         '命令、开关参数
    Dim RarFile As String                         '压缩包路径
    Dim Contents As String                        '文件、文件夹的路径
    WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
    Command = " A -ep -pryueifu "
    RarFile = WorkDir & "Lock.rar"
    Contents = WorkDir & "*.pdf"
    Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & AddQuote(Contents),
vbNormalFocus
End Sub
```

运行上述程序，然后打开压缩包，会看到处于加密状态，如图 3-15 所示。

图 3-15　自动压缩并设置解压密码

如果使用开关参数 -hp，表示高度加密，这种方式生成的压缩包，连其中的文件列表也看不到，如图 3-16 所示。

图 3-16　使用 -hp 参数高度加密

对于设置了密码的压缩包，可以在解压命令中把密码传递进去。

下面的程序把刚刚加密生成的 Lock.rar 解压到 test 文件夹中。

```
Sub 解压带密码的压缩包()
    Const WorkDir As String = "C:\temp\"
                                                    '默认目录
    Dim WinrarExe As String                         'WinRAR 可执行文件路径
    Dim Command As String                           '命令、开关参数
    Dim RarFile As String                           '压缩包路径
    Dim Contents As String                          '文件、文件夹的路径
    WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
    Command = " X -pryueifu "
    RarFile = WorkDir & "Lock.rar"
    Contents = WorkDir & "test"                     '解压到 test 文件夹
    Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & AddQuote (Contents),
vbNormalFocus
End Sub
```

3.1.8　使用 WinRAR 修改 Office 文档

Office 2007 以上版本的 Office 文档（.docx、.xlsx、.pptx 等格式）其实是一种压缩包格式，使用 WinRAR 可以直接打开。下面介绍一下用 WinRAR 查看和修改 Excel 文件的方法。

实例文件“example01.xlsx”有 3 个工作表，表名从左到右依次分别为 Jan、Feb、Mar，如图 3-17 所示。

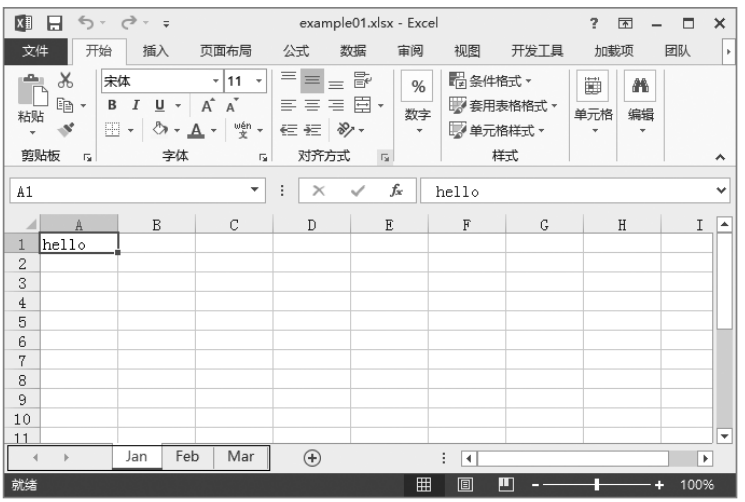


图 3-17　Excel 工作簿文件

在 Excel 中关闭该文件，然后打开 WinRAR 软件，按下快捷键【 Ctrl+O 】，浏览到 example01.xlsx，如图 3-18 所示。

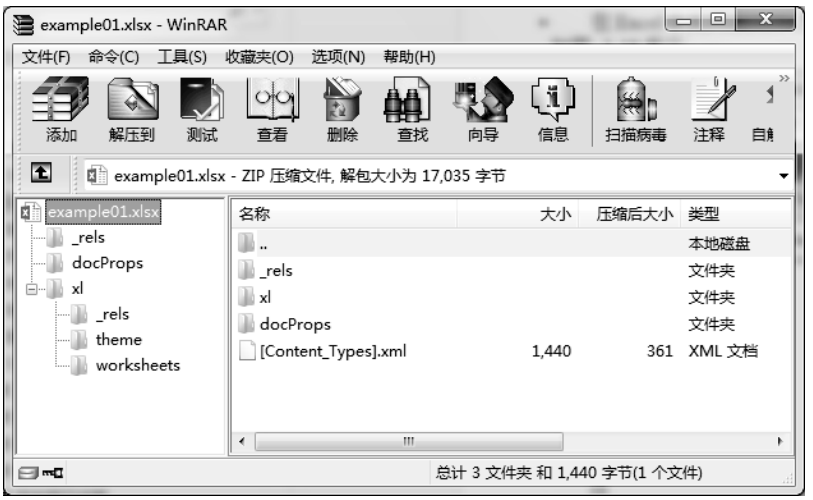


图 3-18　使用 WinRAR 打开 Excel 文件

在 WinRAR 中看到 Excel 文件由 3 个文件夹和一个文件构成，继续展开名为 xl 的文件夹，可以看到和工作表信息有关的内容，如图 3-19 所示。

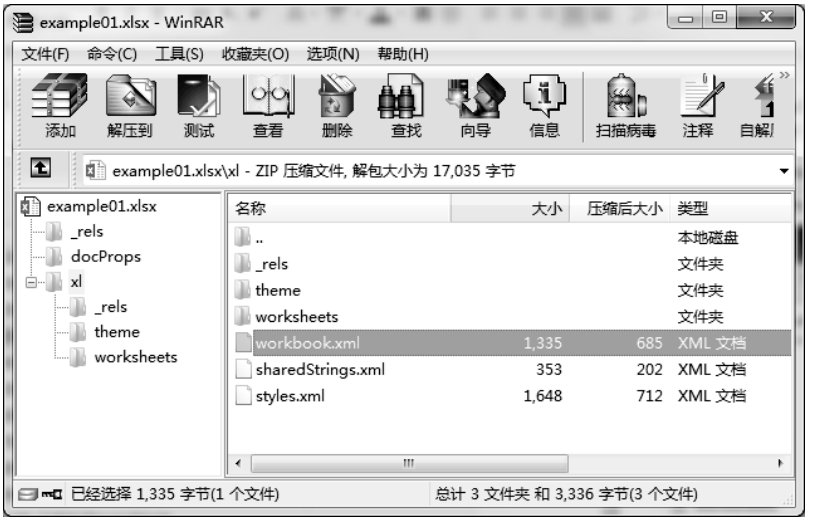


图 3-19　查看 Excel文件的内容

手工把 example01.xlsx 解压到磁盘下，生成了一些扩展名为 .xml 的文件。双击 Workbook.xml，会在 IE 浏览器中打开该文件，如图 3-20 所示。

其中 <Sheets> 这个节点中存储的就是各个工作表的名称和顺序。

xml 文件可以用记事本程序编辑，因此接下来用记事本程序打开 Workbook.xml 文件，把 Jan 这个工作表移动到最后，并且把 Feb 这个表的名称修改为“二月”。修改好后，在 IE 中再次预览，如图 3-21 所示。

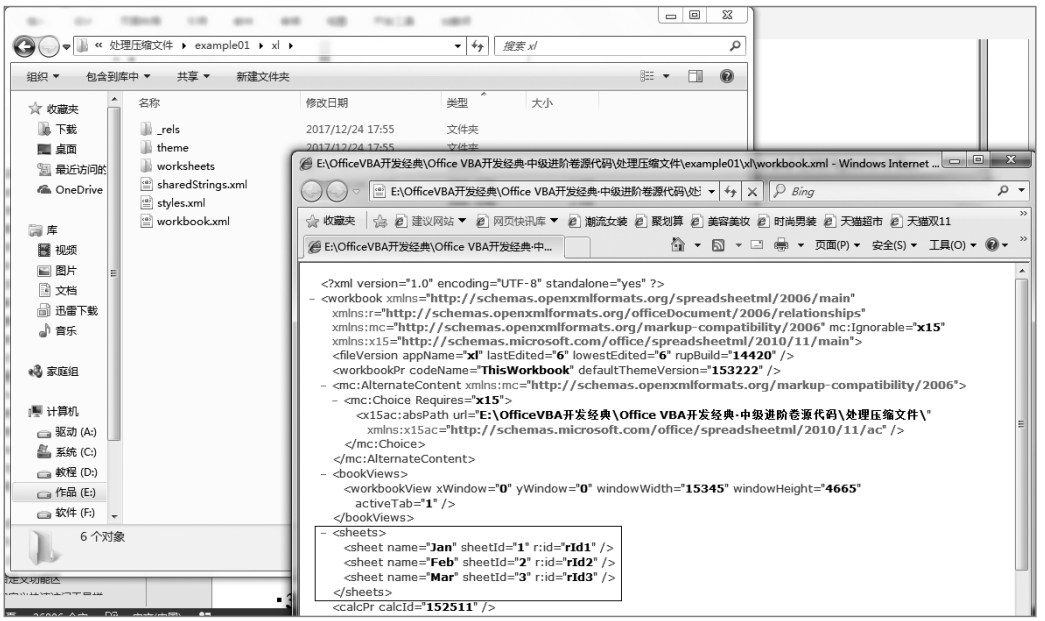


图 3-20　查看 Excel 文件内部的部署清单

```
<sheets>
  <sheet name="二月" sheetId="2" r:id="rId2" />
  <sheet name="Mar" sheetId="3" r:id="rId3" />
  <sheet name="Jan" sheetId="1" r:id="rId1" />
</sheets>
```

图 3-21　调整工作表的 XML 代码

然后把修改了的 Workbook.xml 文件压入 example01.xlsx 工作簿中，在 Excel 中再次打开，看到工作表的名称和次序发生了变化，如图 3-22 所示。

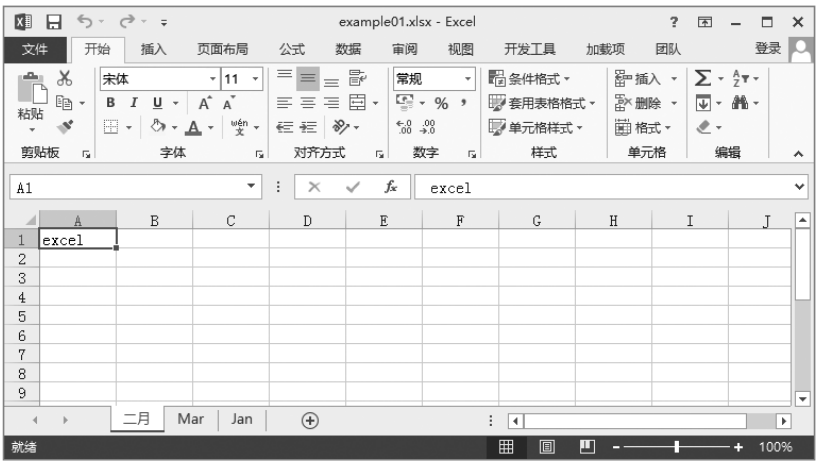


图 3-22　修改后 Excel 文件的内容

可以看到 Jan 工作表移到了最后，Feb 工作表被重命名。

Word、PowerPoint 文档也是压缩文件，可以用 WinRAR 查看和打开。以上演示的是手工使用 WinRAR 查看、修改 Office 文档，当然也可以用 Shell 调用 WinRAR 进行自动解压。

下面的过程把 example01.xlsx 文件中的 Workbook.xml 部分解压到 Destination 文件夹中。

```
Sub 解压 Office 文档 ()
    Const WorkDir As String = "E:\"                    '默认目录
    Dim WinrarExe As String                            'WinRAR 可执行文件路径
    Dim Command As String                              '命令、开关参数
    Dim RarFile As String                              '压缩包路径
    Dim Contents As String                             '文件、文件夹的路径
    WinrarExe = GetSetupPath("WinRAR.exe") & "\WinRAR.exe"
    Command = " E "
    RarFile = WorkDir & "example01.xlsx"
    Contents = WorkDir & "Destination"                 '释放到 Destination 路径下
    Shell AddQuote(WinrarExe) & Command & AddQuote(RarFile) & " xl\workbook.xml "
& AddQuote(Contents), vbNormalFocus
End Sub
```

如果是带有自定义功能区的文档，用压缩包打开后，里面有更多的内容，这些在稍后的章节讲解。

以上内容的源代码文件为“实例文档 10.xlsm”。

3.2 使用 Shell32 对象

Shell32 对象中的 Folder 对象与 FSO 文件系统对象中的 Folder 用法非常相似，都能代表计算机中的一个文件夹。但是，Shell32.Folder 还可以表示一个扩展名为 .zip 的压缩文件，而 FSO 不能操作到压缩文件的内部。

因此，本节介绍一下用 Shell32 对象处理计算机中的文件夹和文件以及 .zip 压缩包中的内容。

3.2.1 引入 Shell32 对象

前期绑定方式：在 VBA 工程中添加外部引用“ Microsoft Shell Controls And Automation”，如图 3-23 所示。

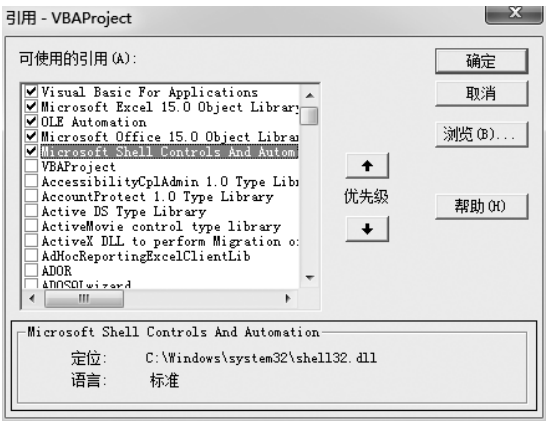


图 3-23 添加外部引用

代码中使用 Dim ShellApp As New Shell32.Shell 声明了一个新的 Shell32 的应用程序对象。后期创建对象的方法如下。

```
Set ShellApp = CreateObject("Shell.Application")
```

3.2.2　使用 namespace 返回文件夹

FSO 对象中，使用 GetFolder 返回一个 Folder 对象，而 Shell32 对象中，使用 namespace 返回一个 Folder 对象。

```
Sub Test1()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder
    With ShellApp
        Set fd = .Namespace("C:\temp")
        Debug.Print fd.Items.Item.Path  '返回文件夹的路径
        Debug.Print fd.Items.Count      '返回该文件夹中包含的内容个数（子文件夹 + 文件）
    End With
End Sub
```

以上过程中，fd 是一个文件夹对象变量，本例用它来指代 C:\temp 这个文件夹。

运行上述程序，在立即窗口打印文件夹的路径、子文件夹和文件的总数，运行结果如图 3-24 所示。

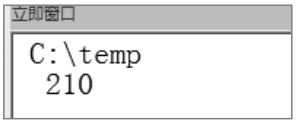


图 3-24　运行结果

上面的实例中，文件夹的规定是在 namespace 的参数中直接指定的，如果要让用户选择一个文件夹，则可以使用 BrowseForFolder 函数。

3.2.3　文件夹选择对话框

BrowseForFolder 函数的语法如下。

```
BrowseForFolder(Hwnd, Title, Options, RootFoler)
```

返回一个 Folder 对象。各参数说明如下。

- Hwnd：对话框的所属句柄，长整型。设置为 0 表示在桌面弹出对话框。
- Title：对话框显示的提示语。
- Options：对话框样式设定参数，十六进制数。
- RootFolder：文件夹选择对话框的起始路径，也可以是特殊的文件夹常量。

下面的代码在 Excel 上面弹出一个文件夹选择对话框，起始目录设置为宏所在工作簿的路径。

```
Sub Test2()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder
    With ShellApp
        Set fd = .BrowseForFolder(Hwnd:=Application.Hwnd, Title:="你必须选择一个
文件夹: ", Options:=&H0, RootFolder:=ThisWorkbook.Path)
        If fd Is Nothing = False Then
```

```
            Debug.Print fd.Items.Item.Path
        End If
    End With
End Sub
```

代码分析：BrowseForFolder 函数弹出浏览文件夹对话框，用户选择文件夹并单击“确定”按钮，fd 会返回一个 Folder 对象，如果单击“取消”按钮，那么 fd 就是 Nothing。

运行上述程序，自动弹出一个选择文件夹的对话框，如图 3-25 所示。

图 3-25　浏览文件夹对话框

3.2.4　遍历文件夹中的内容

Shell32 中的 FolderItem 对象是指包含在文件夹中的文件或子文件夹。

假设 D:\Download 下的文件内容如图 3-26 所示。

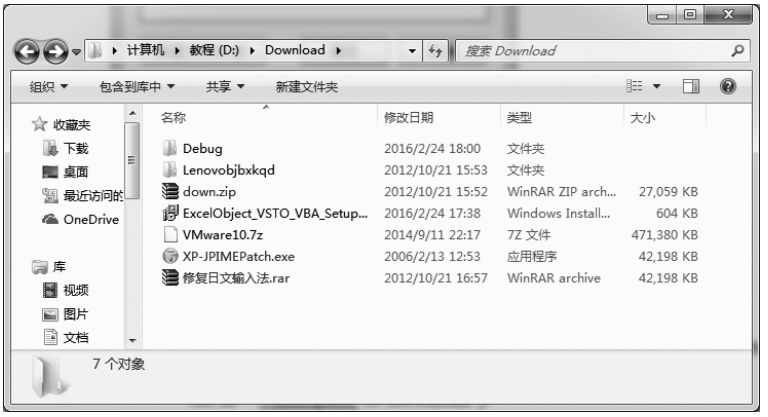


图 3-26　文件夹中的内容

文件夹中有 2 个子文件、5 个文件。

下面的代码遍历 Download 文件夹下所有项目的路径、类型。

```
Sub Test3()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, item As Shell32.
FolderItem
    With ShellApp
        Set fd = .Namespace("D:\Download")
        If fd Is Nothing = False Then
            Debug.Print fd.Items.Count '返回该文件夹中包含的内容个数（子文件夹 + 文件）
            For Each item In fd.Items
                Debug.Print item.Path, item.Type
            Next item
        End If
    End With
End Sub
```

运行上述程序，立即窗口打印出文件夹中的所有内容，如图 3-27 所示。

图 3-27　遍历文件夹中的内容

可以看出，子文件夹的 Type 是“文件夹”。

如果要遍历所有子文件夹中的所有内容，需要用下面的递归函数。

```
Sub Recursion(ByVal Parent As Shell32.Folder)
    Debug.Print Parent.Title
    For Each f In Parent.Items
        If f.Type = "文件夹" Then
            Recursion f.GetFolder
        Else
            Debug.Print f.Path
        End If
    Next f
End Sub

Sub 遍历所有内容()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, item As Shell32.
FolderItem
    With ShellApp
        Set fd = .Namespace("D:\Download")
        Recursion fd
    End With
End Sub
```

运行上面的“遍历所有内容”这个过程，立即窗口打印出 Download 文件夹下的所有内容（包含递归文件夹），如图 3-28 所示。

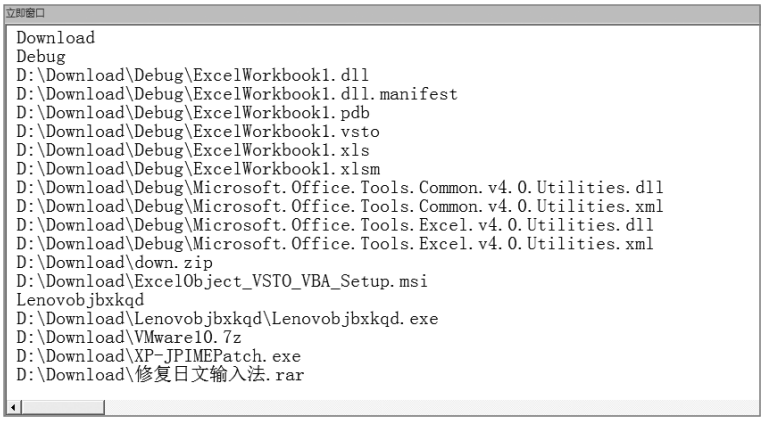


图 3-28　递归遍历文件夹中的所有内容

由于 namespace 不只限于文件夹，还能把 .zip 压缩包当成一个文件夹，因此下面讲述遍

历 .zip 压缩包中的内容。

3.2.5 遍历 .zip 压缩包中的内容

假设 D: 盘下有一个“东北三省 .zip”的压缩包，其内部文件结构如图 3-29 所示。

图 3-29 压缩包

运行下面的过程，可以把压缩包中的所有文件、路径列举出来。

```
Sub 遍历压缩包中内容()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, item As Shell32.
FolderItem
    With ShellApp
        Set fd = .Namespace("D:\东北三省.zip")
        Recursion fd
    End With
End Sub
```

其中，Recursion 是前面讲过的用于递归遍历的函数。

运行上述程序，立即窗口打印出压缩包中的所有内容，如图 3-30 所示。

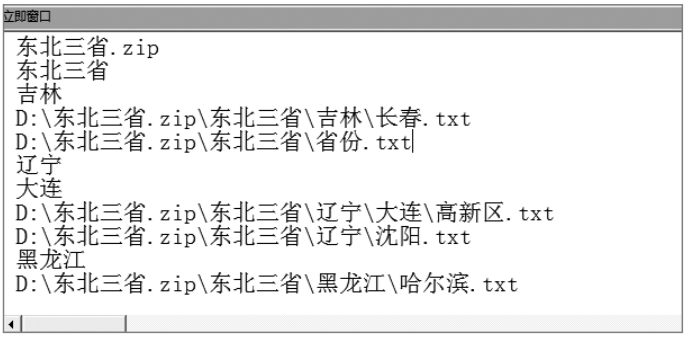


图 3-30 递归遍历压缩包中的所有内容

3.2.6 遍历 Office 文档中的内容

Office 文档的扩展名不是 .zip，理论上用 Shell32 无法遍历其内部内容，然而，可以先

把 Office 文档更改扩展名为 .zip，等遍历完后，再撤销重命名即可。

假设文件夹中有一个幻灯片文件 Presentation01.pptx。下面用 Shell32 遍历该文件的所有内部文件。

```
Sub 遍历 Office 文档 ()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, item As Shell32.
FolderItem
    Dim doc As String
    doc = "C:\temp\Presentation01.pptx"
    Name doc As doc & ".zip"
    With ShellApp
        Set fd = .Namespace(doc & ".zip")
        Recursion fd
    End With
    MsgBox "下面撤销重命名！"
    Name doc & ".zip" As doc
End Sub
```

该幻灯片的内部文件比较多，因此只打印出一部分结果，如图 3-31 所示。

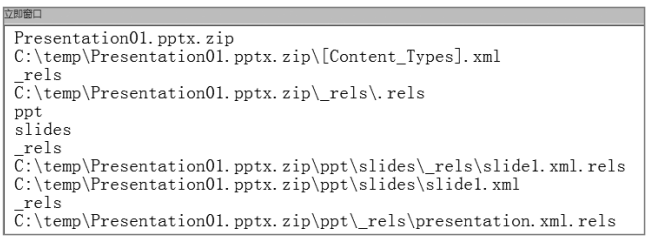


图 3-31　递归遍历 PPT 文件中的所有内容

用 WinRAR 打开该幻灯片，如图 3-32 所示。

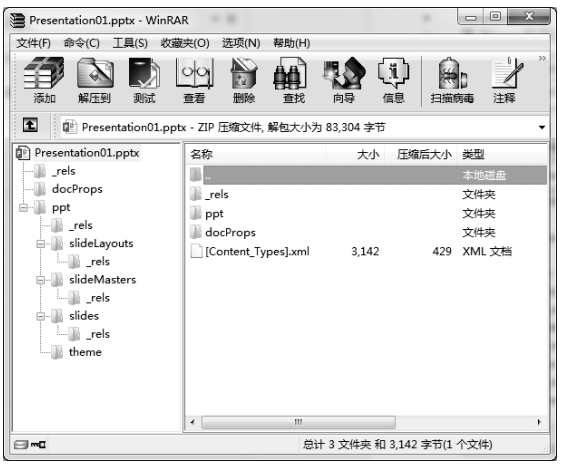


图 3-32　使用压缩包查看 PPT 文件的构造

3.2.7　CopyHere 方法

Shell32 中的 Folder 对象有 CopyHere 方法和 MoveHere 方法，作用是把其他地方的文件

（夹）复制或移动到 Folder 中。

下面的程序把 dist 路径下的所有文件、子文件夹复制到 temp 路径下。

```
Sub 复制文件到文件夹中()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, data As Shell32.Folder
    With ShellApp
        Set fd = .Namespace("C:\temp\")
        Set data = .Namespace("C:\dist\")
        fd.CopyHere data.Items
    End With
End Sub
```

代码分析：data.Items 表示该命名空间（路径）下的所有项目，包括文件、子文件夹。

如果要复制个别的项目，需要用 Item 属性来约定。例如把上面最后一行代码修改为如下形式。

```
fd.CopyHere data.Items.item("aaa\使用说明.txt")
```

上述代码的含义是把 C:\dist\aaa\ 使用说明 .txt 这个文件直接复制到 temp 文件夹下。其中，aaa 是 dist 路径下的一个文件夹。

3.2.8 MoveHere 方法

MoveHere 方法与 Copy 方法几乎是一样的语法，唯一不同的是，使用 MoveHere 方法是把文件或文件夹移动到 Folder 中，也就相当于文件的移动、剪切。

下面举一个把文件夹中的文件移动到压缩包中的实例。首先在桌面或者任意文件夹中新建一个“WinRAR ZIP archive”空白压缩包，并把该压缩包重命名为 blank.zip，如图 3-33 所示。

图 3-33　手工新建一个空白压缩包

路径 C:\temp\datas 下有一百多个文本文件，如图 3-34 所示。

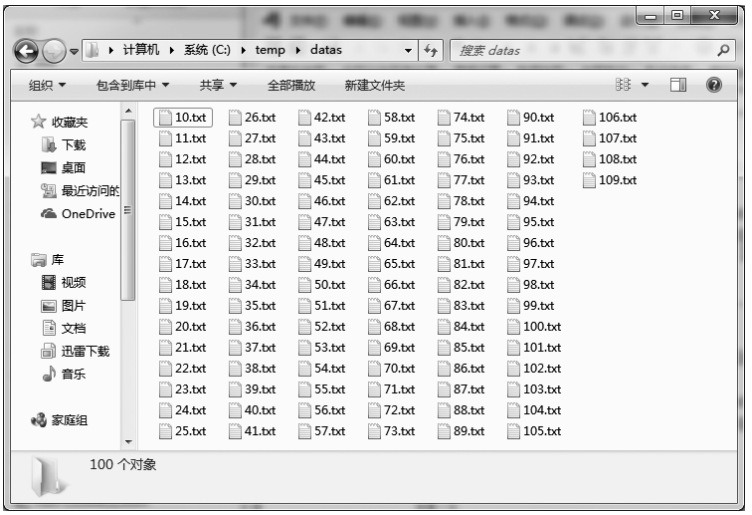


图 3-34　文件夹中包含多个文本文件

下面的程序把 datas 文件夹连同其所有子文件压缩到 blank.zip 中。

```
Sub 移动文件到压缩包中()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, data As Shell32.Folder
    With ShellApp
        Set fd = .Namespace("C:\dist\Blank.zip")
        Set data = .Namespace("C:\temp\datas")
        fd.MoveHere data
    End With
End Sub
```

执行程序后，使 WinRAR 软件查看压缩包 blank.zip 中的内容，如图 3-35 所示。

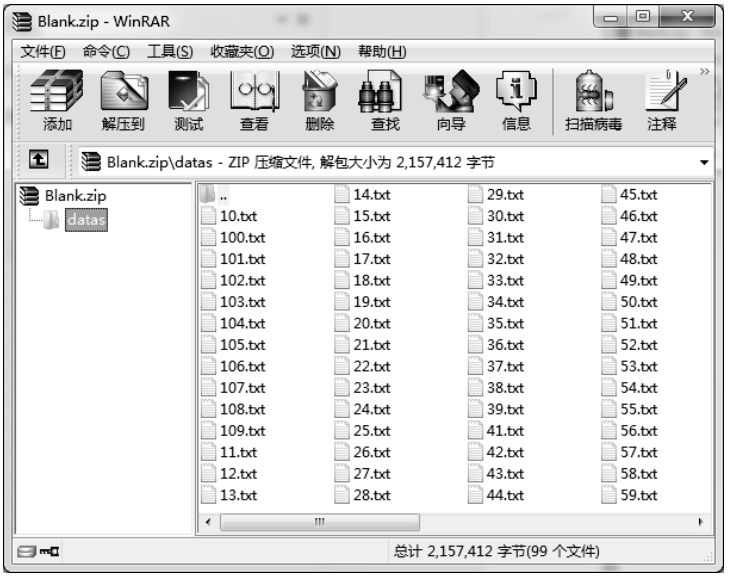


图 3-35　自动压缩文件夹

需要注意的是，如果最后一行代码修改为如下形式。

```
fd.MoveHere data.Items
```

执行的效果是，datas 下面所有的文本文件都直接压缩进去，而没有 datas 这个文件夹！

如果修改为 fd.MoveHere data.Items.Item("39.txt")，则只把一个文本文件移动到压缩包的根目录下。

反过来，CopyHere 方法、MoveHere 方法也可以从压缩包中释放内容到文件夹中。

下面的程序把上述有内容的 Blank.zip 中的 datas\40.txt 文件移动到 C:\lib 路径下。

```
Sub 释放压缩包中内容到文件夹()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, LIB As Shell32.Folder
    With ShellApp
        Set fd = .Namespace("C:\dist\Blank.zip\datas")
        Set LIB = .Namespace("C:\lib")
        LIB.MoveHere fd.Items.item("40.txt")
    End With
End Sub
```

代码分析：namespace 允许在压缩包路径后追加子路径，因此对象变量 fd 表示的是压缩包中的 datas 文件夹。

LIB.MoveHere fd.Items.item("40.txt") 把 fd 中的项目移动到 LIB 中，一定不要搞错方向。

另外，除了上述讲过的用鼠标右键新建 .zip 压缩包之外，也可以用代码自动在路径下产生一个空白的 .zip 压缩包。

```
Sub 新建空白zip压缩包()
    Open "C:\Container.zip" For Output As #1
    Print #1, Chr$(80) & Chr$(75) & Chr$(5) & Chr$(6) & String(18, 0)
    Close #1
End Sub
```

运行上述过程，会在 C: 盘下产生 Container.zip，这个压缩包里没有任何内容。

3.2.9 处理文件覆盖

使用 CopyHere 方法、MoveHere 方法进行文件复制、移动时，如果目的地已经存在名称相同的文件，执行程序过程中会弹出是否复制、替换的对话框，如图 3-36 所示。

图 3-36 存在同名文件的询问对话框

如果要默认强制替换已存在文件，屏蔽该对话框，可以在方法之后加一个 vOptions 参数，并设置为 16。例如：

```
    fd.CopyHere vItem:=data.Items.item ("aaa\使用说明.txt"), vOptions:=16
```

这样，运行到这行代码时，即使存在同名文件，也强制替换。

3.2.10　处理异步问题

使用 CopyHere、MoveHere 方法移动内容时，不管移动操作是否已完成，VBA 代码都会继续向下执行。如果移动的文件容量越大，异步问题越明显。对于一些要求苛刻的程序任务，有必要让程序同步压缩进度。

为了能够让 VBA 识别到文件移动的进度，需要在 CopyHere、MoveHere 方法之后加一个 Do…Loop 循环，循环跳出的条件是目标压缩包中的文件总数达到一个指标。

假设 E:\Joker 路径下有 52 张扑克牌图片，使用下面的程序把这 52 个 .jpg 格式的图片全部移动到新建的压缩包 C:\temp\Package.zip 中。

```
Sub 处理异步问题 ()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, data As Shell32.Folder
    Open "C:\temp\Package.zip" For Output As #1
    Print #1, Chr$(80) & Chr$(75) & Chr$(5) & Chr$(6) & String(18, 0)
    Close #1
    With ShellApp
        Set fd = .Namespace("C:\temp\Package.zip")
        Set data = .Namespace("E:\Joker")
        fd.MoveHere data.Items, 16
    End With
    Do Until fd.Items.Count = 52
        Application.Wait Now() + TimeValue("00:00:01")
    Loop
    MsgBox " 移动操作已完成 !", vbInformation
End Sub
```

代码分析：首先创建一个空白 .zip 压缩包，该压缩包中项目个数为 0。其次使用对象变量 fd 来指代该压缩包，然后把 Joker 文件夹下的所有文件移动到压缩包中，移动过程中 fd.Items.Count 一定小于 52，因此根据这个特征可以让 VBA 代码阻塞在 Do 循环内。最后，压缩操作结束，跳出 Do 循环，弹出对话框，如图 3-37 所示。

图 3-37　压缩操作完成才弹出对话框

3.2.11　修改 Office 文档功能区

Office 2007 以上版本创建的文档允许自定义功能区。自定义功能区的 XML 代码存储在文档中 CustomUI 文件夹下，文件名一般为 customUI14.xml。本书源代码中的 example02.xlsx 文件用 WinRAR 打开后，可以看到自定义功能区的部分，如图 3-38 所示。

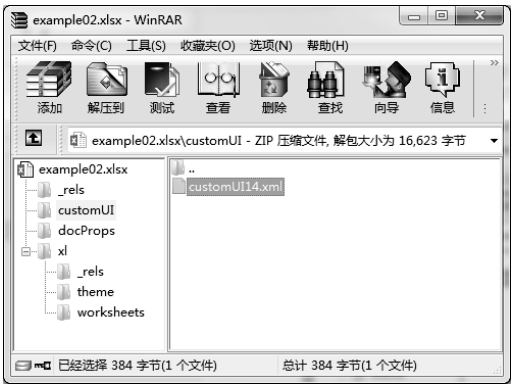


图 3-38　Excel 文件的内部构造

双击 customUI14.xml，可以看到 XML 代码，如图 3-39 所示。

```
- <customUI xmlns="http://schemas.microsoft.com/office/2009/07/customui">
  - <ribbon startFromScratch="false">
    - <tabs>
      - <tab id="exampleID1" label="RibbonXmlEditor">
        - <group id="GroupID2" label="Author:ryueifu">
            <button id="ButtonID3" label="原始按钮" imageMso="ChartTypeOtherInsertGallery" size="large" />
          </group>
        </tab>
      </tabs>
    </ribbon>
  </customUI>
```

图 3-39　customUI 代码

如果在 Excel 中打开该文件，如图 3-40 所示。

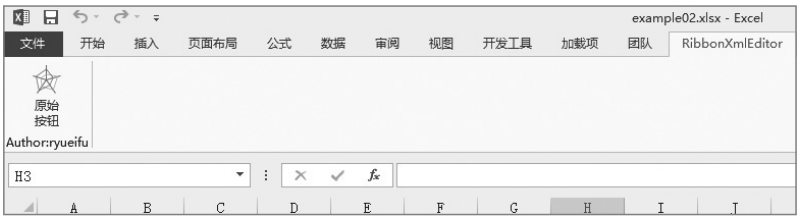


图 3-40　包含 customUI 部分的 Excel 文件

下面通过 Shell32 的 MoveHere 方法更改 XMl 代码，从而更改功能区的外观显示。

具体实现原理和步骤如下。

（1）在工作簿关闭的前提下，后面追加 .zip 扩展名，以便 Shell32 访问。

（2）使用 MoveHere 方法把 customUI14.xml 文件移动到某个文件夹中。

（3）使用 XML 外部对象自动修改文件夹中的 customUI14.xml 文件。

（4）用 MoveHere 方法把文件夹中的 customUI14.xml 文件逆向移动回到压缩包中的 customUI 文件夹下。

（5）删掉工作簿后面的 .zip，恢复为正常的 Excel 工作簿。具体代码如下。

```
Sub 修改 Office 文档功能区 ()
    Dim ShellApp As New Shell32.Shell, fd As Shell32.Folder, temp As Shell32.Folder
    Dim wbk As String
```

```
    Dim X As New DOMDocument              '引用 Microsoft XML v6.0
    Dim Ribbon As String
    wbk = "E:\处理压缩文件\example02.xlsx"
    Name wbk As wbk & ".zip"              '暂时重命名为 .zip 压缩包
    With ShellApp
        Set fd = .Namespace(wbk & ".zip\customUI") '定位到压缩包中 CustomUI 文件夹
        Set temp = .Namespace("C:\temp\")
        temp.MoveHere fd.Items.item("customUI14.xml"), 16
                                          '移动功能区代码到磁盘文件夹中
        If X.Load("C:\temp\customUI14.xml") Then       '装载 xml 文件并适当替换
            Ribbon = Replace(X.XML, "原始按钮", "被我修改")
            Ribbon = Replace(Ribbon, "ChartTypeOtherInsertGallery", "M")
            If X.LoadXML(Ribbon) Then
                X.Save "C:\temp\customUI14.xml"        '保存被修改的 xml 文件
            End If
        End If
        fd.MoveHere temp.Items.item("customUI14.xml"), 16
                                          '把被修改的 xml 文件压缩回工作簿中
        Stop
    End With
    Name wbk & ".zip" As wbk              '恢复原来的扩展名
End Sub
```

代码分析：为了确保压缩操作完成后再重命名，在重命名代码之前加一个 Stop 语句。

运行上述代码后，在 Excel 2013 中打开 example02.xlsx 文件，会看到功能区中的按钮标题和图标都发生了变化，如图 3-41 所示。

图 3-41　使用 Shell32 修改 Excel 文件的 customUI部分

以上这部分知识是自定义功能区的铺垫，同时表明可以从压缩文件的角度去研究和处理 Office 文档。

以上内容的源代码文件为“实例文档 11.xlsm”。

3.3　本章小结

Shell 调用 WinRAR 处理压缩包，Shell32 对象处理压缩包，这些操作都是异步的，也就是说，VBA 代码在不知道压缩操作是否已完成的情况下继续执行后面的代码。因此，当使用 CopyHere、MoveHere 方法之后，需要补充一些监测压缩操作进度的代码，防止后续的 VBA 代码过早执行。