完美Excel

Excel资讯 技术 技巧 资源 应用 相关技术 ..... 还有Excel外的生活

Workbook对象应用大全

2009年05月18日, 8:08 下午 分享到微博： 0
(目前还没有人投票)

Workbook对象代表一个工作簿，Workbooks集合对象则代表同一Excel进程中打开的所有工作簿对象。

[应用1] 创建新工作簿（Add方法）

使用Add方法在Workbooks集合中创建新工作簿，所创建的工作簿为活动工作簿。其语法为：

Workbooks.Add(Template)

参数Template可选，决定如何创建新工作簿。如果将该参数设置为已存在的Excel模板文件名称，那么将以该文件作为模板创建工作簿。该参数可以为下列XlWBATemplate常量之一：xlWBATChart（值-4109，代表图表）、xlWBATExcel4IntlMacroSheet（值4）、xlWBATExcel4MacroSheet（值3）、xlWBATWorksheet（值-4167，代表工作表）。在创建新工作簿时，如果指定该参数，那么将创建包含指定类型工作表的工作簿；如果省略该参数，那么将创建包含一定数量空工作表的工作簿，工作表数为SheetsInNewWorkbook属性所设置的数量。

应用示例1：创建一个新工作簿

```
Sub CreateNewWorkbook1()
    MsgBox "将创建一个新工作簿."
    Workbooks.Add
End Sub
```

应用示例2：创建一个新工作簿并命名工作表且添加数据

```
Sub CreateNewWorkbook2()
    Dim wb As Workbook
    Dim ws As Worksheet
```

```
    Dim i As Long
    MsgBox "将创建一个新工作簿,并预设工作表格式."
    Set wb = Workbooks.Add
    Set ws = wb.Sheets(1)
    ws.Name = "产品汇总表"
    ws.Cells(1, 1) = "序号"
    ws.Cells(1, 2) = "产品名称"
    ws.Cells(1, 3) = "产品数量"
    For i = 2 To 10
        ws.Cells(i, 1) = i - 1
    Next i
End Sub
```

应用示例3：创建带有指定数量工作表的工作簿

```
Sub testNewWorkbook()
    MsgBox "创建一个带有10个工作表的新工作簿"
    Dim wb As Workbook
    Set wb = NewWorkbook(10)
End Sub

Function NewWorkbook(wsCount As Integer) As Workbook
    '创建带有由变量wsCount提定数量工作表的工作簿，工作表数在1至255之间
    Dim OriginalWorksheetCount As Long
    Set NewWorkbook = Nothing
    If wsCount < 1 Or wsCount > 255 Then Exit Function
    OriginalWorksheetCount = Application.SheetsInNewWorkbook
    Application.SheetsInNewWorkbook = wsCount
    Set NewWorkbook = Workbooks.Add
    Application.SheetsInNewWorkbook = OriginalWorksheetCount
End Function
```

自定义函数NewWorkbook可以创建最多带有255个工作表的工作簿。本测试示例创建一个带有10个工作表的新工作簿。

[应用2] 打开工作簿（Open方法）

Open方法用于打开一个现有的工作簿，其语法为：

```
Workbooks.Open(FileName, UpdateLinks, ReadOnly, Format, Password, WriteResPassword, IgnoreReadOnlyRecommended, Origin, Delimiter, Editable, Notify, Converter,
```

可以看到，该方法具有很多参数，但大多数参数都很少用到。在这些参数中，除参数FileName必须外，其它参数都可选。
参数FileName指定要打开的工作簿文件的名称，参数UpdateLinks指定更新工作簿中链接的方式，参数ReadOnly用来设置是否以只读方式打开工作簿。如果需要使用密码来打开工作簿，则应该将参数Password设置为该密码；如果需要使用密码打开工作簿但没有指定密码，则会弹出询问密码的对话框。参数AddToMru指定是否将工作簿添加到最近使用的文件列表中，建议将其设置为True，默认值为False。
应用示例4：以只读方式打开某工作簿

```
Sub openWorkbook2()
    Dim fname As String
    MsgBox "将D盘中的<测试.xls>工作簿以只读方式打开"
    fname = "D:\测试.xls"
    Workbooks.Open Filename:=fname, ReadOnly:=True
End Sub
```

[应用3] 访问特定的工作簿
使用Item属性返回Workbooks集合中特定的工作簿。例如：

```
Workbooks.Item(1)
```

返回Workbooks集合中的第一个工作簿。由于Item属性是缺省的属性，因此上述代码也可以简写为：

```
Workbooks(1)
```

然而，使用索引号来指定工作簿是不可靠的，最好使用工作簿的具体名称来指定特定的工作簿，例如：

```
Workbooks("MyBook.xlsx")
```

注意，当用户使用"新建"命令创建一个新工作簿（假设该工作簿系统默认名称为Book2）时，在没有保存该工作簿前，应该使用下面的代码指定该工作簿：

```
Workbooks("Book2")
```

此时，如果使用下面的代码指定该工作簿：

```
Workbooks("Book2.xlsx")
```

将会产生运行时错误：下标越界。

[应用4] 激活工作簿（Activate方法）

使用Activate方法激活指定的工作簿，例如：

```
Workbooks("MyWorkbook").Activate
```

[应用5] 获得当前打开的工作簿数（Count属性）

使用Workbooks集合对象的Count属性来获得当前打开的工作簿数，例如：

```
Workbooks.Count
```

[应用6] 判断工作簿是否是只读的（ReadOnly属性）

如果工作簿以只读方式打开，那么ReadOnly属性的值为True。

[应用7] 获得工作簿的路径和名称（Name属性、FullName属性、Path属性、CodeName属性）

使用Workbook对象的Name属性可以返回工作簿的名称。例如，下面的函数可以返回当前工作簿的名称：

```
Function MyName() As String
    MyName = ThisWorkbook.Name
End Function
```

使用Workbook对象的FullName属性可以返回工作簿的路径和名称。例如，下面的函数可以返回当前工作簿的路径和名称：

```
Function MyName() As String
    MyName = ThisWorkbook.Name
End Function
```

使用Workbook对象的Path属性可以返回工作簿文件的路径。使用Workbook对象的CodeName属性返回工作簿对象的代码名。

上述属性均为只读属性。

应用示例5：一些工作簿通用属性示例

```
Sub testGeneralWorkbookInfo()
    MsgBox "本工作簿的名称为" & ActiveWorkbook.Name
    MsgBox "本工作簿带完整路径的名称为" & ActiveWorkbook.FullName
    MsgBox "本工作簿对象的代码名为" & ActiveWorkbook.CodeName
    MsgBox "本工作簿的路径为" & ActiveWorkbook.Path
    If ActiveWorkbook.ReadOnly Then
        MsgBox "本工作簿已经是以只读方式打开"
```

```
    Else
        MsgBox "本工作簿可读写."
    End If
    If ActiveWorkbook.Saved Then
        MsgBox "本工作簿已保存."
    Else
        MsgBox "本工作簿需要保存."
    End If
End Sub
```

[应用8] 保存工作簿（Save方法）

使用Save方法保存对工作簿所作的所有更改，其语法为：

```
Workbook.Save
```

应用示例6：保存已存在的所有工作簿

```
Sub SaveAllWorkbooks()
    Dim wbk As Workbook
    For Each wbk In Workbooks
        If wbk.Path <> "" Then wbk.Save
    Next wbk
End Sub
```

如果某工作簿的Path属性值为空，则表明该工作簿为新建工作簿，还没有保存。而本过程仅保存所有已存在的（即已经保存过的）工作簿。

[应用9] 保存工作簿（SaveAs方法）

使用SaveAs方法在指定的文件中保存对工作簿所做的更改，其语法为：

```
Workbook.SaveAs(FileName, FileFormat, Password, WriteResPassword, ReadOnlyRecommended, CreateBackup, AccessMode, ConflictResolution, AddToMru, TextCodepage, Te
```

所有参数均为可选参数。其中参数FileName指定要保存文件的文件名，可以包含完整的路径，如果不指定路径，Excel将文件保存到当前文件夹中。参数FileFormat指定保存文件时使用的文件格式。如果文件夹中存在相同名称的工作簿，则提示是否替换原工作簿。
参数Password用于指定文件的保护密码，是一个区分大小写的字符串（最长不超过 15 个字符）。参数WriteResPassword指定文件的写保护密码，如果文件保存时带有密码，但打开文件时没有输入密码，则该文件以只读方式打开。
将参数ReadOnlyRecommended设置为True，则在打开文件时显示一条消息，提示该文件以只读方式打开。将参数CreateBackup设

置为True，以创建一个备份文件。
参数AccessMode和参数ConflictResolution用来解决访问和冲突问题。
将参数AddToMru设置为True，以添加工作簿到最近使用的文件列表中。默认值为False。
应用示例7：创建新工作簿并保存

```
Sub AddSaveAsNewWorkbook()
    Dim Wk As Workbook
    Set Wk = Workbooks.Add
    Application.DisplayAlerts = False
    Wk.SaveAs Filename:="D:\SalesData.xlsx"
End Sub
```

这里使用了Add方法和SaveAs方法，添加一个新工作簿并将该工作簿以文件名SalesData.xlsx保存在D盘中。其中，语句Application.DisplayAlerts = False表示禁止弹出警告对话框。
应用示例8：另存已有的工作簿

```
Sub SaveWorkbook2()
    Dim oldName As String, newName As String
    Dim folderName As String, fname As String
    oldName = ActiveWorkbook.Name
    newName = "new" & oldName
    MsgBox "将<" & oldName & ">以<" & newName & ">的名称保存"
    folderName = Application.DefaultFilePath
    fname = folderName & "\" & newName
    ActiveWorkbook.SaveAs fname
End Sub
```

上述代码将当前工作簿以一个新名(即new加原名)保存在默认文件夹中。
应用示例9：备份工作簿

```
Sub CreateBak1()
    MsgBox "保存工作簿并建立备份工作簿"
    ActiveWorkbook.SaveAs CreateBackup:=True
End Sub
```

上述代码在当前文件夹中建立工作簿的备份。

```
Sub CreateBak2()
```

```
    MsgBox "保存工作簿时,若已建立了备份,则将出现包含True的信息框,否则出现False."
    MsgBox ActiveWorkbook.CreateBackup
End Sub
```

[应用10] 保存工作簿副本（SaveCopyAs方法）

使用SaveCopyAs方法保存指定工作簿的一份副本，但不会修改已经打开的工作簿，其语法为：

```
Workbook.SaveCopyAs(Filename)
```

参数Filename用来指定副本的文件名。

应用示例10：使用与活动工作簿相同的名称但后缀名为.bak来备份工作簿

```
Sub SaveWorkbookBackup()
    Dim awb As Workbook, BackupFileName As String, i As Integer, OK As Boolean
    If TypeName(ActiveWorkbook) = "Nothing" Then Exit Sub
    Set awb = ActiveWorkbook
    If awb.Path = "" Then
        Application.Dialogs(xlDialogSaveAs).Show
    Else
        BackupFileName = awb.FullName
        i = 0
        While InStr(i + 1, BackupFileName, ".") > 0
            i = InStr(i + 1, BackupFileName, ".")
        Wend
        If i > 0 Then BackupFileName = Left(BackupFileName, i - 1)
        BackupFileName = BackupFileName & ".bak"
        OK = False
        On Error GoTo NotAbleToSave
        With awb
            Application.StatusBar = "正在保存工作簿..."
            .Save
            Application.StatusBar = "正在备份工作簿..."
            .SaveCopyAs BackupFileName
            OK = True
        End With
    End If
NotAbleToSave:
    Set awb = Nothing
    Application.StatusBar = False
    If Not OK Then
```

```
        MsgBox "备份工作簿未保存!", vbExclamation, ThisWorkbook.Name
    End If
End Sub
```

在当前工作簿中运行本示例代码后，将以与工作簿相同的名称但后缀名为.bak备份工作簿，且该备份与当前工作簿在同一文件夹中。

应用示例11：保存当前工作簿的副本到其它位置来备份工作簿

```
Sub SaveWorkbookBackupToFloppyD()
    Dim awb As Workbook, BackupFileName As String, i As Integer, OK As Boolean
    If TypeName(ActiveWorkbook) = "Nothing" Then Exit Sub
    Set awb = ActiveWorkbook
    If awb.Path = "" Then
        Application.Dialogs(xlDialogSaveAs).Show
    Else
        BackupFileName = awb.Name
        OK = False
        On Error GoTo NotAbleToSave
        If Dir("D:\" & BackupFileName) <> "" Then
            Kill "D:\" & BackupFileName
        End If
        With awb
            Application.StatusBar = "正在保存工作簿..."
            .Save
            Application.StatusBar = "正在备份工作簿..."
            .SaveCopyAs "D:\" & BackupFileName
            OK = True
        End With
    End If
NotAbleToSave:
    Set awb = Nothing
    Application.StatusBar = False
    If Not OK Then
        MsgBox "备份工作簿未保存!", vbExclamation, ThisWorkbook.Name
    End If
End Sub
```

上述程序将当前工作簿进行复制并以与当前工作簿相同的名称保存在D盘中。其中，使用了Kill方法来删除已存在的工作簿。

[应用11] 判断工作簿是否发生变化（**Saved**属性）

如果工作簿自上次保存以来没有发生任何变化，那么该工作簿的Saved属性值为True。由于该属性值是可读写的，因此我们能将该属性的值设置为True，即使该工作簿自上次保存之后发生过变化。这样，我们能设置该属性的值为True，关闭被修改过的工作簿而不提示保存

当前已发生的变化，即让Excel误认为已经保存了所作的变化。

[应用12] 关闭工作簿（Close方法）

使用Workbooks对象的Close方法关闭所有工作簿，其语法为：

```
Workbooks.Close
```

使用Workbook对象的Close方法关闭指定的工作簿，其语法为：

```
Workbook.Close(SaveChanges, Filename, RouteWorkbook)
```

参数均为可选参数。其中，参数SaveChanges用于在关闭工作簿前保存工作簿所发生的变化。特别地，如果工作簿中没有变化，则忽略该参数；如果工作簿中有变化但工作簿显示在其他打开的窗口中，则忽略该参数；如果工作簿中有改动且工作簿未显示在任何其他打开的窗口中，则由该参数指定是否应保存更改。如果将该参数设置为True，则保存对工作簿所做的更改；如果工作簿尚未命名，则使用参数FileName指定的名称保存。如果忽略参数Filename，则要求用户提供文件名。如果将该参数设置为False，则不会保存工作簿中的变化。如果忽略该参数，那么Excel将显示一个对话框询问是否保存工作簿中的变化。

参数RouteWorkbook指出工作簿传送的问题。如果工作簿不需要传送给下一个收件人（没有传送名单或已经传送），则忽略该参数。否则，Excel将根据该参数的值传送工作簿。如果将该参数设置为True，则将工作簿传送给下一个收件人。如果设置为False，则不发送工作簿。如果忽略，则要求用户确认是否发送工作簿。

注意，Close方法检查工作簿的Saved属性，以决定是否提示用户保存工作簿所发生的变化。如果将Saved属性的值设置为True，那么Close方法将不会警告而直接关闭工作簿，并不会保存工作簿中所发生的任何变化。

应用示例12：保存并关闭所有工作簿

```
Sub SaveAndCloseAllWorkbooks()
    Dim wbk As Workbook
    For Each wbk In Workbooks
        If wbk.Name <> ThisWorkbook.Name Then
            wbk.Close SaveChanges:=True
        End If
    Next wbk
    ThisWorkbook.Close SaveChanges:=True
End Sub
```

应用示例13：不保存而关闭工作簿

```
Sub CloseWorkbook1()
    MsgBox "不保存所作的改变而关闭本工作簿"
    ActiveWorkbook.Close False
    '或ActiveWorkbook.Close SaveChanges:=False
```

```
    '或ActiveWorkbook.Saved=True
End Sub<pre>
<span style="color: #0000ff;">应用示例14：保存而关闭工作簿</span>
<pre lang="vb">Sub CloseWorkbook2()
    MsgBox "保存所作的改变并关闭本工作簿"
    ActiveWorkbook.Close True
End Sub
```

应用示例15：关闭工作簿并将其彻底删除

```
Sub KillMe()
    With ThisWorkbook
        .Saved = True
        .ChangeFileAccess Mode:=xlReadOnly
        Kill .FullName
        .Close False
    End With
End Sub
```

[应用13] 打印预览工作簿（PrintPreview方法）

使用PrintPreview方法按工作簿打印后的外观效果显示工作簿的预览，其语法为：

```
Workbook.PrintPreview(EnableChanges)
```

参数EnableChanges指定用户是否可更改边距和打印预览中可用的其他页面设置选项。

[应用14] 打印工作簿（PrintOut方法）

使用PrintOut方法打印完整的工作簿（当然，该方法也应用于其它一些对象，例如Range、Worksheet、Chart），其语法为：

```
Workbook.PrintOut(From, To, Copies, Preview, ActivePrinter, PrintToFile, Collate, PrToFileName, IgnorePrintAreas)
```

所有参数均为可选参数。参数From指定需要打印第一页的页码，参数To指定要打印的最后一页的页码，如果忽略这些参数，将打印整个对象。
参数Copies指定要打印副本的数量，默认值为1。
如果参数Preview设置为True，那么将弹出打印预览而不是立即打印。默认值为False。
使用参数ActivePrinter设置活动打印机的名称。
如果将参数PrintToFile设置为True，那么将工作簿打印到文件。此时，如果没有指定参数PrToFileName的值，那么Excel将提示用户输入要使用的输出文件的文件名。使用参数PrToFileName指定要打印到的文件名。

将参数Collate设置为True，以逐份打印副本。
将参数IgnorePrintAreas设置为真，则忽略打印区域而打印整个对象。

[应用15] 保护工作簿（Protect方法）

使用Protect方法保护工作簿，使其不能够被修改，其语法为：

```
Workbook.Protect(Password, Structure, Windows)
```

所有参数均为可选参数。其中，参数Password用来指定一个密码，所设置的密码区分大小写。如果省略该参数，不用密码就可以取消对工作簿的保护。否则，必须指定密码才能取消对工作簿的保护。
将参数Structure的值设置为True，以保护工作簿的结构，即工作簿中工作表的相关位置。此时不能对工作簿中的工作表进行插入、复制、删除等操作。默认值为False。
将参数Windows的值设置为False，以保护工作簿窗口。此时，该工作簿右上角的最小化、最大化和关闭按钮消失。默认值为False。

应用示例16：保护工作簿示例代码

```
Sub ProtectWorkbook()
    MsgBox "保护工作簿结构,密码为123"
    ActiveWorkbook.Protect Password:="123", Structure:=True
    MsgBox "保护工作簿窗口,密码为123"
    ActiveWorkbook.Protect Password:="123", Windows:=True
    MsgBox "保护工作簿结构和窗口,密码为123"
    ActiveWorkbook.Protect Password:="123", Structure:=True, Windows:=True
End Sub
```

[应用16] 解除工作簿保护（Unprotect方法）

使用Unprotect方法取消工作簿保护，其语法为：

```
Workbook.Unprotect(Password)
```

参数Password为一个字符串，指定用于解除工作表或工作簿保护的密码，区分大小写。如果工作簿不设密码保护，则省略该参数。如果对工作簿省略该参数，而该工作簿又设有密码保护，则该方法将失效。

应用示例17：解除工作簿保护

```
Sub UnprotectWorkbook()
    MsgBox "取消工作簿保护"
    ActiveWorkbook.Unprotect "123"
End Sub
```

[应用17] 判断工作簿是否有密码保护（HasPassword属性）

如果指定工作簿有密码保护，则HasPassword属性值为 True。

应用示例18：检查工作簿是否有密码保护

```
Sub IsPassword()
    If ActiveWorkbook.HasPassword = True Then
        MsgBox "本工作簿有密码保护,请在管理员处获取密码."
    Else
        MsgBox "本工作簿无密码保护,您可以自由编辑."
    End If
End Sub
```

[应用18] ThisWorkbook对象和ActiveWorkbook对象

有时，在代码中经常会碰到ThisWorkbook对象和ActiveWorkbook对象，虽然在某些情况下其所代表的工作簿相同，但是在某些情况下还是有较大的差别，特别是制作加载项时。

ThisWorkbook对象代表的是代码所在的工作簿，而ActiveWorkbook对象代表的是活动工作簿。

[应用19] 工作簿的属性（BuiltinDocumentProperties属性）

“文件—属性”或者“Office按钮—准备—属性”将显示一个对话框，包含了有关当前工作簿的信息，可以从VBA访问工作簿的属性。

应用示例19：显示已经保存的当前工作簿的日期和时间

```
Sub LastSaved()
    Dim SaveTime As String
    On Error Resume Next
    SaveTime = ActiveWorkbook.BuiltinDocumentProperties("Last Save Time").Value
    If SaveTime = "" Then
        MsgBox ActiveWorkbook.Name & "还没有被保存."
    Else
        MsgBox "保存于:" & SaveTime, , ActiveWorkbook.Name
    End If
End Sub
```

如果没有保存过工作簿，那么对Last Save Time属性的访问将产生错误，使用On Error语句忽略这个错误。

应用示例20：列出当前工作簿的内置属性

```
Sub listWorkbookProperties()
    On Error Resume Next
    '在名为"工作簿属性"的工作表中添加信息,若该工作表不存在,则新建一个工作表
    Worksheets("工作簿属性").Activate
    If Err.Number <> 0 Then
```

```
        Worksheets.Add after:=Worksheets(Worksheets.Count)
        ActiveSheet.Name = "工作簿属性"
    Else
        ActiveSheet.Clear
    End If
    On Error GoTo 0
    ListProperties
End Sub

Sub ListProperties()
    Dim i As Long
    Cells(1, 1) = "名称"
    Cells(1, 2) = "类型"
    Cells(1, 3) = "值"
    Range("A1:C1").Font.Bold = True
    With ActiveWorkbook
        For i = 1 To .BuiltinDocumentProperties.Count
            With .BuiltinDocumentProperties(i)
                Cells(i + 1, 1) = .Name
                Select Case .Type
                    Case msoPropertyTypeBoolean
                        Cells(i + 1, 2) = "Boolean"
                    Case msoPropertyTypeDate
                        Cells(i + 1, 2) = "Date"
                    Case msoPropertyTypeFloat
                        Cells(i + 1, 2) = "Float"
                    Case msoPropertyTypeNumber
                        Cells(i + 1, 2) = "Number"
                    Case msoPropertyTypeString
                        Cells(i + 1, 2) = "string"
                End Select
                On Error Resume Next
                Cells(i + 1, 3) = .Value
                On Error GoTo 0
            End With
        Next i
    End With
    Range("A:C").Columns.AutoFit
End Sub
```

[应用20] 重命名工作簿（Name方法）

Name方法用来重命名一个文件、目录或文件夹，其语法为：

```
Name oldpathname As newpathname
```

应用示例21：重命名未打开的工作簿

```
Sub rename()
    Name "<工作簿路径>\<旧名称>.xlsx" As "<工作簿路径>\<新名称>.xlsx"
End Sub
```

代码中<>的内容为需要重命名的工作簿所在路径及新旧名称。该方法只是对未打开的文件进行重命名，如果该文件已经打开，使用该方法会提示错误。

[应用21] 获取或设置工作簿密码（Password属性）

使用Password属性返回或设置在打开指定工作簿时必须提供的密码。

应用示例22：设置工作簿密码

```
Sub UsePassword()
    Dim wb As Workbook
    Set wb = Application.ActiveWorkbook
    wb.Password = InputBox("请输入密码:")
    wb.Close
End Sub
```

代码运行后，提示设置密码，然后关闭工作簿；再次打开工作簿时，要求输入密码。

[应用22] 工作簿中形状的显示方式（DisplayDrawingObjects属性）

使用DisplayDrawingObjects属性返回或设置工作簿中形状的显示方式，可以是下列常量之一：xlDisplayShapes（显示所有形状）、xlPlaceholders（仅显示占位符）、xlHide（隐藏所有形状）。

应用示例23：控制工作簿中图形显示方式

```
Sub testDraw()
    MsgBox "隐藏当前工作簿中的所有图形"
    ActiveWorkbook.DisplayDrawingObjects = xlHide
    MsgBox "仅显示当前工作簿中所有图形的占位符"
    ActiveWorkbook.DisplayDrawingObjects = xlPlaceholders
    MsgBox "显示当前工作簿中的所有图形"
    ActiveWorkbook.DisplayDrawingObjects = xlDisplayShapes
End Sub
```

[应用23] 工作簿文件格式（FileFormat属性）

使用FileFormat属性返回工作簿文件格式或类型。

[应用24] 决定工作簿计算使用的数值（PrecisionAsDisplayed属性）

在工作簿进行计算时，如果将PrecisionAsDisplayed属性设置为True，则仅使用工作表中所显示的数值进行计算，而不是单元格中实际存储的值。该属性的默认值为False，表明工作簿计算基于单元格中实际存储的值。

应用示例24：设置数字精度

```
Sub SetPrecision()
    Dim pValue
    MsgBox "在当前单元格中输入1/3，并将结果算至小数点后两位"
    ActiveCell.Value = 1 / 3
    ActiveCell.NumberFormatLocal = "0.00"
    pValue = ActiveCell.Value * 3
    MsgBox "当前单元格中的数字乘以3等于：" & pValue
    MsgBox "然后,将数值分类设置为[数值],即单元格中显示的精度"
    ActiveWorkbook.PrecisionAsDisplayed = True
    pValue = ActiveCell.Value * 3
    MsgBox "此时,当前单元格中的数字乘以3等于：" & pValue & "而不是1"
    ActiveWorkbook.PrecisionAsDisplayed = False
End Sub
```

上述代码在计算前将PrecisionAsDisplayed属性的值设置为True，则表明采用单元格中所显示的数值进行计算。

[应用25] 删除自定义数字格式（DeleteNumberFormat方法）

使用DeleteNumberFormat方法从工作簿中删除一个自定义数字格式，其语法为：

```
Workbook.DeleteNumberFormat(NumberFormat)
```

参数NumberFormat为要删除的数字格式。

应用示例25：删除自定义数字格式

```
Sub DeleteNumberFormat()
    MsgBox "从当前工作簿中删除000-00-0000的数字格式"
    ActiveWorkbook.DeleteNumberFormat ("000-00-0000")
End Sub
```

[应用26] 添加名称（Names属性）

Workbook对象的Names属性返回Names集合，代表指定工作簿中的所有名称。

应用示例26：在活动工作簿中添加名称

```
Sub testNames()
    MsgBox "将当前工作簿中工作表Sheet1内单元格A1命名为myName."
    ActiveWorkbook.Names.Add Name:="myName", RefersToR1C1:="=Sheet1!R1C1"
End Sub
```

上述代码将活动工作簿单元格A1命名为MyName。

[应用27] 获取工作簿用户状态信息（UserStatus属性）

UserStatus属性返回一个基为 1 的二维数组，该数组提供有关每一个以共享列表模式打开工作簿的用户的信息。数组第二维的第一个元素为用户名，第二个元素是用户打开工作簿的日期和时间，第三个元素是一个表示清单类型的数字（1表示独占，2表示共享）。

UserStatus属性不返回有关以只读方式打开指定工作簿的用户的信息。

应用示例27：列出工作簿用户状态信息

```
Sub UsePassword()
    Dim Users As Variant
    Dim Row As Long
    Users = ActiveWorkbook.UserStatus
    Row = 1
    With Workbooks.Add.Sheets(1)
        .Cells(Row, 1) = "用户名"
        .Cells(Row, 2) = "日期和时间"
        .Cells(Row, 3) = "使用方式"
        For Row = 1 To UBound(Users, 1)
            .Cells(Row + 1, 1) = Users(Row, 1)
            .Cells(Row + 1, 2) = Users(Row, 2)
            Select Case Users(Row, 3)
            Case 1
                .Cells(Row + 1, 3).Value = "个人工作簿"
            Case 2
                .Cells(Row + 1, 3).Value = "共享工作簿"
            End Select
        Next
    End With
    Range("A:C").Columns.AutoFit
End Sub
```

示例代码运行后，将创建一个新工作簿并带有用户使用当前工作簿的信息，即用户名、打开的日期和时间及工作簿使用方式。

[应用28] 操作工作簿中的样式（Styles集合和Style对象）

每个工作簿都有一个Styles集合，包含该工作簿的所有已定义样式。一个Style对象代表单元格区域的一组格式选项，可以使用Add方法创建Style对象，其语法为：

```
Styles.Add(Name, BasedOn)
```

参数Name必需，用来指定样式的名称。参数BasedOn可选，用来指定单元格，新样式即基于该单元格生成。如果省略此参数，就基于"常规"样式创建新样式。
如果指定名称的样式已经存在，该方法将基于参数BasedOn指定的单元格重新定义已存在的样式。
Style对象的属性代表了不同的格式特征，例如字体名称、字体大小、数字格式、对齐等。也有内置的样式，例如Normal、Currency和Percent，这些内置样式能在样式对话框中找到。
应用示例28：创建一个新样式并将其应用到当前工作表的单元格区域中

```
Sub test()
    Dim st As Style
    '如果该样式已存在则删除
    For Each st In ActiveWorkbook.Styles
        If st.Name = "Bordered" Then st.Delete
    Next st
    '创建新样式
    With ActiveWorkbook.Styles.Add(Name:="Bordered")
        .Borders(xlTop).LineStyle = xlDouble
        .Borders(xlBottom).LineStyle = xlDouble
        .Borders(xlLeft).LineStyle = xlDouble
        .Borders(xlRight).LineStyle = xlDouble
        .Font.Bold = True
        .Font.Name = "Arial"
        .Font.Size = 36
    End With
    '应用样式
    Application.ActiveSheet.Range("A1:B3").Style = "Bordered"
End Sub
```

[应用29] 打开文本文件（OpenText方法）

OpenText方法用于在一个新工作簿中装入文本文件，并将其转换为工作表，其语法为：

```
Workbooks.OpenText(Filename, Origin, StartRow, DataType, TextQualifier, ConsecutiveDelimiter, Tab, Semicolon, Comma, Space, Other, OtherChar, FieldInfo, TextVi
```

与Open方法一样，除参数Filename必须外，其它参数都可选。
参数Filename指定要打开的文本文件的名称。参数Origin指定文本文件的来源，可以为下列XlPlatform常量之一：xlMacintosh、xlWindows或xlMSDOS，此外它还可以是一个整数，表示所需代码页的代码页编号，例如"1256"指定源文本文件的编码是阿拉伯语

(Windows)。如果省略该参数，则此方法将使用“文本导入向导”中“文件原始格式”选项的当前设置。
参数StartRow指定从文本文件中开始进行分析处理的文本的行号，默值为1。参数DataType指定字段中的文本格式，可以是下列XlTextParsingType常量之一：xlDelimited或xlFixedWidth，如果忽略该参数，则Excel将尝试在打开文件时确定字段格式。参数TextQualifier用来指定文本识别符，可以是下列XlTextQualifier常量之一：xlTextQualifierNone（值-4142，代表无分隔符）、xlTextQualifierDoubleQuote（值1，代表双引号）、xlTextQualifierSingleQuote（值2，代表单引号）。应将参数ConsecutiveDelimiter设置为True，这样将连续分隔符当作一个分隔符，默认值为False。
有几个参数需要参数DataType必须设置为xlDelimited，包括参数Tab、参数Semicolon、参数Comma、参数Space和参数Other。当这些参数中的任何一个设置为True时，表示Excel应该使用与文本分隔符相应的字符，这些分隔符描述如下（所有参数的缺省值均为False）：
参数Tab设置为True时，使用制表符作为分隔符；参数Semicolon设置为True时，使用分号作为分隔符；参数Comma设置为True时，使用逗号作为分隔符；参数Space设置为True时，使用空格作为分隔符；参数Other设置为True时，将使用参数OtherChar指定的字符作为分隔符。如果参数OtherChar包含多个字符时，则仅使用第一个字符而忽略其它字符。
参数FieldInfo是包含单列数据相关分列信息的数组，取决于参数DataType的值。当参数DataType的值为xlDelimited时，则参数FieldInfo为由两元素数组组成的数组，该数组的大小应该与被转换数据的列的数量相同或更小。一个二维数组的第一维是列标（起始为1），第二维是下列XlColumnDataType常量之一，用于指定列的数据类型：xlGeneralFormat（值1，常规）、xlTextFormat（值2，文本）、xlMDYFormat（值3，MDY日期格式）、xlDMYFormat（值4，DMY日期格式）、xlYMDFormat（值5，YMD日期格式）、xlMYDFormat（值6，MYD日期格式）、xlDYMFormat（值7，DYM日期格式）、xlYDMFormat（值8，YDM日期格式）、xlSkipColumn（值9，列未分列即跳过列）、xlEMDFormat（值10，EMD日期格式）。
如果提供给二维数组的列没有找到，那么该列将使用常规设置。例如，将下面的数据设置为参数FieldInfo的值，则第一列为文本，而第三列被跳过，其它列被视为常规数据：

```
Array(Array(1,2),Array(3,9))
```

参数TextVisualLayout代表文本的可视布局，参数DecimalSeparator指定小数分隔符，参数ThousandsSeparator指定千位分隔符，参数TrailingMinusNumbers用于处理末尾为减号的数字。参数Local用来指定是否分隔符、数字和数据格式应使用计算机的区域设置。
应用示例29：打开带有分隔符的文本文件
在D盘的excel文件夹中有一个名为temp1.txt的文本文件，其内容为：

```
"张三","工人","A工厂",1/2/2009
"李四","职员","B公司",3/3/2009
"王五","教师","C学校",2/2/2009
"赵六","学生","D学院",1/1/2009
```

在Excel工作簿中放置下面的代码：

```
Sub test()
    Workbooks.OpenText Filename:="D:\excel\temp1.txt", _
```

```
        Origin:=xlMSDOS, _
        StartRow:=1, _
        DataType:=xlDelimited, _
        TextQualifier:=xlTextQualifierDoubleQuote, _
        ConsecutiveDelimiter:=True, _
        Comma:=True, _
        FieldInfo:=Array(Array(1, 2), Array(2, 2), Array(3, 2), Array(4, 6))
End Sub
```

运行后，将生成如下图1所示的工作表。注意，列D中的单元格放置日期。

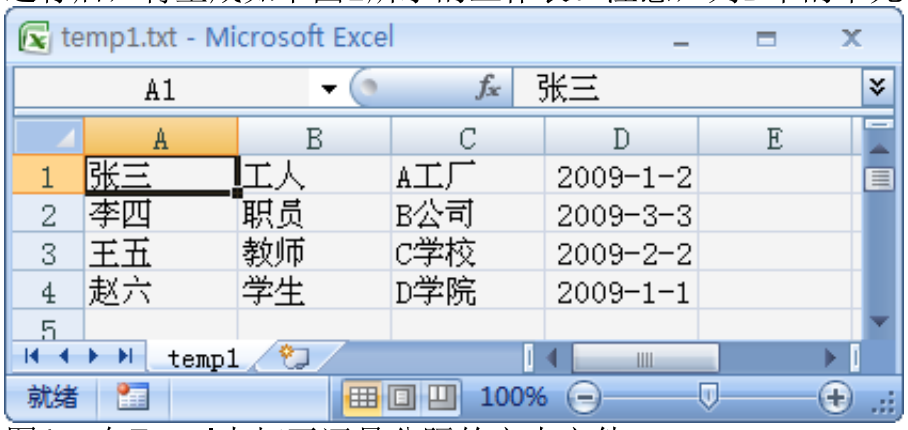

图1：在Excel中打开逗号分隔的文本文件

应用示例30：打开固定宽度的文本文件

如果参数DataType设置为xlFixedWidth，那么参数FieldInfo中每个二维数组的第一维指定字符在列中开始的位置（第一个字符的位置是0），第二维指定列的数据类型（如上面在参数介绍中所描述的）。

在D盘的excel文件夹中有一个名为temp2.txt的文本文件，其内容为：

```
0-125-689
2-523-489
3-424-664
4-125-160
```

在Excel工作簿中放置下面的代码：

```
Sub test()
    Workbooks.OpenText Filename:="D:\excel\temp2.txt", _
        Origin:=xlMSDOS, _
        StartRow:=1, _
```

```
        DataType:=xlFixedWidth, _
        FieldInfo:=Array(Array(0, 2), Array(1, 9), Array(2, 2), Array(5, 9), Array(6, 2))
End Sub
```

运行后，将生成如下图2所示的工作表。注意看代码是如何使用数组跳过这些连字符的。

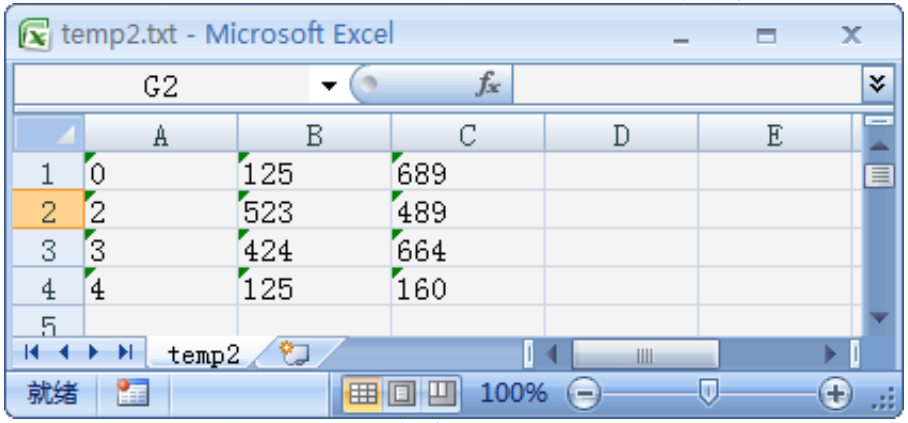

图2：在Excel中打开固定宽度的文本文件

注意到图1和图2中打开的文本文件并没有被转换为Excel 工作簿文件，因此使用下面的代码将其保存为Excel工作簿文件：

```
    Application.ActiveWorkbook.SaveAs _
            Filename:="D:\excel\temp.xlsx", FileFormat:=xlWorkbookNormal
```

[应用30] 判断工作簿是否存在

下面的示例使用自定义函数FileExists判断工作簿是否存在，若该工作簿已存在，则打开它。代码中，“C:\文件夹\子文件夹\文件.xls”代表工作簿所在的文件夹名、子文件夹名和工作簿文件名。

```
Sub testFileExists()
    MsgBox "如果文件不存在则用信息框说明,否则打开该文件."
    If Not FileExists("C:\文件夹\子文件夹\文件.xls") Then
        MsgBox "这个工作簿不存在!"
    Else
        Workbooks.Open "C:\文件夹\子文件夹\文件.xls"
    End If
End Sub

Function FileExists(FullFileName As String) As Boolean
    '如果工作簿存在,则返回True
    FileExists = Len(Dir(FullFileName)) > 0
```

```
End Function
```


相关文章

- 2009年05月23日 使用VBA合并多个Excel工作簿 (4)
  有许多实现Excel工作簿合并的方法，在《将多个工作簿中的数据合并到一个工作簿》中介绍过合并工作簿的示例。下面再列举几个示例，供有兴趣的朋友参考。例如，需要将多个Excel工作簿中的工作表合并到一个工作簿。这里假设需要合并的工作簿在"D:\示例\数据记录\"文件夹中，含有两个工作簿test1.xls、test2.xls（当然，可以不限于两个），在test1.xls工作簿中含有三张工作表，...
- 2009年05月19日 判断指定的工作簿是否已打开 (0)
  下面整理归纳了一些实用函数代码，其功能都是用来判断指定的工作簿是否打开，如果已打开则返回True，否则返回False。 实用代码1： Function IsWorkbookOpen(wbk As String) As Boolean Dim wbT As Excel.Workbook Err.Clear On Error Resume Next ...
- 2009年05月20日 从已关闭的工作簿中取值 (4)
  经常有人提出关于如何从已关闭（或未打开）的工作簿中取值的问题，我将自己收集整理的一些代码辑录于此，供参考。 示例代码1： Sub testGetValuesFromClosedWorkbook() GetValuesFromAClosedWorkbook "C:", "Book1.xls", "Sheet1", "A1:G20" End Sub Sub GetValue...
- 2009年05月9日 将多个工作簿中的数据合并到一个工作簿 (1)
  在MSDN文档资料库中，Microsoft Office Excel MVP Ron de Bruin提供了一些非常好的代码示例，用于将同一文件夹里的多个工作簿中某一工作表的数据合并到一个工作簿中。下面，来介绍这些实用代码，也通过这些代码来学习VBA。当然，您可以适当修改代码，使代码满足自己想要的功能。 查找单元格区域中的最后一个单元格、最后一行或最后一列 这是后面示例中要用到的通用代码...
- 2009年05月3日 从Excel中导出图片（一种较简单的方法） (1)
  在以前的一篇文章中，介绍了两种从Excel中导出图片的方法，即《从Excel中导出图片》。所介绍的代码都使用了Windows API，较为繁锁。前几天看到John Walkenbach先生的一个VBA技巧，很巧妙地将工作簿中的图片导出，并且代码也很简单，特在此与大家分享。 我们知道，将Excel工作簿保存为网页文件（HTML格式）时，除了主文件外，还会创建一个包含有图形、图表、样式等文件的...

标签：Workbooks集合, Workbook对象
Category: VBA | 11,949 次阅读

1条评论

1. 粉丝说道：
   2009年05月19日 4:44 下午

   谢谢 又来吸收营养啦 哈哈啊