VB5.0 中函数 CreateObject 和 GetObject 的使用

曹进克　（郑州解放军电子技术学院研究所　450004）

随着面向对象编程(OOP)技术的日益普及，VB5.0 的功能也日益被人们所认同，特别是基于部件的应用程序开发，VB5.0 又是首选的开发工具。CreateObject 和 GetObject 是部件调用过程中经常用到的两个函数，确切地理解它们的使用方法，则是用 VB5.0 进行基于部件的应用程序开发的基础。

1. CreateObject 函数

CreateObject 函数的功能：创建并返回对一个 ActiveX 对象的引用。

语法：CreateObject(class)。

class 参数：该参数的形式为 appname.objecttype，其中 appname 是必需的，指提供对象的应用程序的名称(字符串类型)，objecttype 也是必需的，指要创建的对象的类或类型。

每一个支持 OLE 自动化的应用程序都至少提供一种类型的对象。例如 Word 字处理程序提供 Application 对象、Document 对象和 Toolbar 对象。为了创建一个 ActiveX 对象(ActiveX 是 Microsoft OLE 技术的扩展)，需要把 CreateObject 函数返回的对象赋给一个对象变量。像下面的代码表示的那样：

```
Dim ExcelSheet As Object ' 声明一个对象变量
Set ExcelSheet = CreateObject("Excel.WorkSheet")
```

这段代码将启动 Excel 应用程序来创建这个对象。一旦一个对象被创建，就可以通过定义的对象变量在程序代码中应用它。在下面的代码中，我们利用这个对象变量 ExcelSheet 及 Excel 的其他对象来访问这个新对象的属性和方法。

```
ExcelSheet.Application.Visible = True '通过 Application 对象使 Excel 可见
ExcelSheet.Cells(1,1).Value = "This is column A, row 1" '给这个表单的第一个单元赋值
ExcelSheet.SaveAs "C:\tmp.doc" '保存这个表单
ExcelSheet.Application.Quit '关闭 Excel 应用程序
Set ExcelSheet = Nothing '释放对象变量
```

利用 As Object 形式声明的对象变量可以引用任意类型的对象，这时只有当程序代码运行时对象变量才能与指定的对象绑定，称之为后绑定(Late binding)；如果在声明对象变量时就指明了对象的类型，那么在程序编译的时候对象变量就与指定的对象绑定，称之为前绑定(Early binding)。下面代码为前绑定例子：

```
Dim xlApp As Excel.Application
Dim xlBook As Excel.Workbook
Dim xlSheet As Excel.Worksheet
Set xlApp = CreateObject("Excel.Application")
Set xlBook = xlApp.Workbooks.Add
Set xlSheet = xlBook.Worksheets(1)
```

利用前绑定变量，执行的效率较后绑定要高些。

当对象没有当前例程运行时，可使用 CreateObject 函数来创建一个对象例程。如果有一个例程正在运行，使用 CreateObject 函数将有一个新的例程启动，创建一个指定类型的对象。为了使用当前例程，或者启动应用程序使其装载一个文件，可使用 CreateObject 函数。

如果一个对象将自己注册为单例程对象(a single-instance object)，那么不管执行多少次 CreateObject 函数，都只能创建对象的一个例程。

2. GetObject 函数

GetObject 函数的功能：对来自一个文件的 ActiveX 对象的引用。

语法：GetObject([pathname][, class])。其中 pathname 参数为可选项，指包含指定对象的文件的有效路径。如果该参数省略，必须指定 class 参数；class 参数也为可选项，是一字符串，表示指定对象的类，其语法格式为 appname.objecttype(意义同上)。

使用 GetObject 函数可以访问来自一个文件的 ActiveX 对象，并把该对象赋给一个对象变量。例如：

```
Dim CADObject As Object
Set CADObject = GetObject("c:\cad\schema.cad")
```

当以上代码被执行时，路径指定的应用程序启动，

© 中国科学院软件研究所 http://www.c-s-a.org.cn

应用程序中的对象激活。

如果 pathname 参数为零长度字符串(""),GetObject 函数返回指定类型对象的一个新的例程;如果省略 pathname 参数,GetObject 函数返回一个指定类型当前活动的对象。如果指定类型的对象不存在,将出现错误。

当对象有一个当前例程,或者你想用一个已装入内存的文件创建对象时,可使用 GetObject 函数。如果没有当前例程,或者你不想用一个装入的文件来启动对象,这时可使用 CreateObject 函数。

对于单例程对象(a single - instance object),当 GetObject 函数的参数为零长度字符串("")时,GetObject 函数总是返回同样的例程,如果省略 pathname 参数,则会引起错误。不能用 GetObject 函数来对由 VB 生成的类进行引用。

下例中的程序代码用 GetObject 函数来得到对 Excel 的一个表单的引用。

```
' 声明有关 API 函数
Declare Function FindWindow Lib "user32" Alias "FindWindowA" (ByVal lpClassName as String, ByVal lpWindowName As Long) As Long
Declare Function SendMessage Lib "user32" Alias "SendMessageA" (ByVal hWnd as Long, ByVal wMsg as Long ByVal wParam as Long ByVal lParam As Long) As Long
Sub GetExcel()
  Dim MyXL As Object ' 声明对象变量
  Dim ExcelWasNotRunning As Boolean ' 声明一逻辑变量
  On Error Resume Next ' 错误处理语句
Set MyXL = Getobject(, "Excel.Application") ' 第一个参数省略
If Err.Number < > 0 Then ExcelWasNotRunning = True
  Err.Clear ' 清除 Err 对象
  DetectExcel ' 子程序调用
  Set MyXL = Getobject("c:\vb5\MYTEST.XLS") '给对象变量赋值
  MyXL.Application.Visible = True
  MyXL.Parent.Windows(1).Visible = True
  ' 这里可添加其他语句对该文件进行处理
  ' ...
  If ExcelWasNotRunning = True Then
    MyXL.Application.Quit
  End IF
  Set MyXL = Nothing ' 释放对象变量
End Sub
Sub DetectExcel()
  ' 该子程序检测是否有一 Excel 例程在运行,如果有就注册它
  Const WM-USER = 1024
  Dim hWnd As Long
  hWnd = FindWindow("XLMAIN", 0)
  If hWnd = 0 Then ' 没有 Excel 运行
    Exit Sub
  Else
    SendMessage hWnd, WM-USER + 18, 0, 0 ' 若 Excel 正在运行,就将其放进运行的对象表中
  End If
End Sub
```

解释:DetectExcel 子程序是用于寻找 Excel 的,若 Excel 正在运行,就将其放进运行的对象表中。代码段中,第一次调用 GetObject 的语句行在 Excel 没有运行时会产生一个错误,这个错误使逻辑变量 ExcelWasNotRunning 设置为 True。第二次调用 GetObject 的语句行明确一个需要打开的文件。若 Excel 没有运行,第二次调用 GetObject 的语句行将启动 Excel,并返回文件 MYTEST.XLS 所代表的 WorkSheet 的引用。这个文件必须是存在的,否则 VB 将产生错误。接下来的代码使 Excel 和包括指定工作单的窗口都可见。最后若程序执行前没有 Excel 在运行,这时在代码执行完前关闭 Excel,否则不关闭它,仅释放对象变量。

3. 讨论

CreateObject 和 GetObject 在客户程序访问服务部件时非常有用,不管服务部件是放在本地机器上还是放在网络上。要注意这两个函数使用上的区别,特别是处理支持 OLE 自动化的文件对象时,用 GetObject 较合适,而处理其他支持 OLE 自动化的对象时,则常用 CreateObject。

(注:文中程序代码均在 VB5.0 中调试通过。)

(来稿时间:1998 年 6 月)

© 中国科学院软件研究所 http://www.c-s-a.org.cn