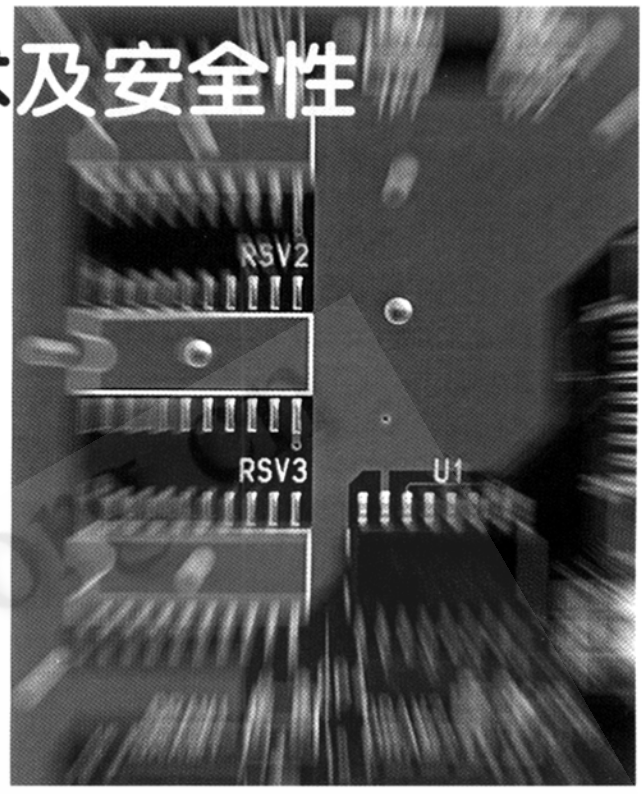
VB的多接口技术及安全性控件的制作

杨莉萍（山东财政学院计算机信息工程系 250014）



摘要：本文以具体应用--给组件增加安全性接口为出发点，简述了COM组件支持的多接口技术，组件中的类允许支持多个接口，正是对同一标准接口的支持，实现了组件的多态；本文着重介绍了多接口技术在VB开发环境下的具体实现，同时给出了一个极具实用价值的完整实例。

开发Web应用时，常需要在页面上嵌入各种控件，这样会大大改善页面操作的灵活性，但在具体开发时，往往会遇到控件安全性问题。例如：本人曾为某单位开发了一套基于Web的综合数据查询系统，其中的日报查询页面(如下图)为了方便用户选择日期，避免错误输入，嵌入了一个DateTimePicker控件。但由于该控件提供的一些方法能在Browser端实现写操作，被认为是一个不安全控件，要想在浏览器中解释执行此页面必须将浏览器的级别降至最低，如果每次访问日报查询页面都要先降浏览器级别，无疑增加了系统使用的复杂性，大大限制了系统的使用面。实际上，该页面只使用了DateTimePicker控件的Value属性及Change事件，其他那些可能对Browser端构成威胁的方法均未使用，因此可以考虑只保留我们要用的属性和事件，将那些有危险的方法屏蔽掉，生成一个新控件，再增加一个安全性接口，将其改造成一个安全性控件。下面就是改造过程涉及的主要技术。

1 页面上控件的安全运行设置

Browser正常安全级别下，要想在HTML页面上使用控件，该控件必须被设置脚本安全性和初始化安全性。所谓脚本安全性是指没有任何脚本可以使控件对用户的计算机或数据造成破坏，且不能从用户的计算机中获取未授权的信息，即不能为系统造成任何破坏；而初始化安全性是指控件在初始化时无论使用什么数据和脚本都不会执行有损于最终用户计算机的操作。借此保证了在执行该控件过程中客户机是安全的。

一个控件是否设置了脚本安全性和初始化安全性，途径之一是通过IobjectSafety接口告知客户机。

2 控件的多接口技术及其在VB开发环境下的具体实现

2.1 多接口技术及应用

所谓接口就是指组件中类的一组相关属性、方法和事件的集合，它可视为组件的开发人员和使用人员之间的协议。接口的改变可能导致使用该组件编译的应用程序的改变。

每一个公共类和接口都有一个唯一标识GUID，通常称之为类ID（CLISID）和接口ID（IID）。当编译使用了某组件的程序时，该程序所创建的所有对象的类ID和接口ID都包含在可执行程序中。程序使用类ID请求组件创

建对象，然后查询该对象的接口ID。如果接口ID不存在则会发生错误。

组件中的类允许支持多个接口，其中必有一个为缺省接口，其余均为附加接口。借助于多接口可实现组件多态和版本兼容。

多态意味着许多类可以提供同样的属性和方法，调用者在调用这些属性和方法前，不必知道某个对象属于什么类。也就是说如果两个类实现了相同的辅助接口，就这个接口而言这两个
类就是多态的。

如果创建标准的接口并在多个类中实现，这些类由一个或多个组件提供，就可以在应用程序或整个系统中获得多态性的优点。

此外，多接口还为版本兼容提供了一种方式。组件的设计必须遵守下面的AxctivX规则：一旦接口投入使用，就不能再改动。需要时再通过实现附加的接口来扩展其功能。由于组件的新版本在添加新接口或增强接口时，可以继续提供原有的接口，这样就简化了保持兼容的问题。既保持与使用较早版本组件编译生成的旧应用程序的兼容，又能对组件进行升级和扩展。也可以说多个接口还提供了增量式或渐进式开发的方法，在发生改动时不必重新编译系统中的所有部件。

2.2 多接口技术在VB开发环境下的实现

在VB中创建一个类，系统会自动为其设置一个缺省接口。若不作说明，为该类定义的属性和方法均属于默认接口。缺省接口的名称是类名前加下划线，但通常使用类名引用缺省接口。

借助于Implementing语句可类增设附加接口。接口就象一个契约，如果一个类支持它，就必须实现该接口的所有属性和方法。增设了附加接口，就要为接口中的所有属性和方法编写实现代码。

接口源可以是一个事先在某类型库中定义的标准接口，也可以是一个抽象类。一个抽象类，只包含属性和方法，不包含实现的任何代码。它不是用来创建对象的，其用途是为添加其他类中的接口提供摸板。

3 改造DateTimePicker控件添加IobjectSafety接口的具体实现

3.1 具体步骤

(1) 在VB开发环境下，新建一个AxtiveX Control工程。

(2) 在UserControl1的窗体界面上，嵌入DateTimePicker控件，调整DateTimePicker位置及尺寸，满意为止，然后调整UserControl1界面与DateTimePicker大小一致。

(3) 新控件的缺省接口只设一个Change事件和一个Value属性，其中Change事件要在代码窗口中用Public Event Change()语句进行声明，它由DateTimePicker控件的Change事件引发；而Value属性是控件默认属性，不必声明，其值DateTimePicker控件的Value值。

至此，一个替代DateTimePicker的新控件创建完毕，下面的工作就是为其附加一个IobjectSafety接口。

(4) 在工程中，添加对IObjectSafety Interface类型库的引用，，该库可在VB实例集的Samples目录(其中提供了一个 IObjectSafety 的示例应用程序)中找到，名为Iobjsafe.tlb。然后在UserControl1的代码窗口中，增加下列语句：Implements IobjectSafety，附加IobjectSafety接口。

(5) 为方便IobjectSafety接口方法的代码编写，在工程中添加一个标准模块Module1，在其中声明一些常量、另外编写一个自定义函数GetIIDFromPTR，功能是由指定对象的接口ID获取它的标识符串。

(6) 在UserControl1的代码窗口，为IobjectSafety接口规定的两个方法编写实现代码。

i IobjectSafety接口中的两个方法：

方法IObjectSafety_GetInterfaceSafetyOptions(ByVal riid As Long, pdwSupportedOptions As Long, pdwEnabledOptions As Long)用于获取对象支持的安全选项以及对象正在使用的安全选项，若成功则返回S_OK，若对象识别给定的接口则返回E_NOINTERFACE，其中参数riid传递对象的接口ID，参数pdwSupportedOptions是输出参数，传递riid接口支持的安全选项位，参数pdwEnabledOptions也是输出参数，传递riid接口正在使用的安全选项位。

方法IObjectSafety_SetInterfaceSafetyOptions(ByVal riid As Long, ByVal dwOptionsSetMask As Long, ByVal dwEnabledOptions As Long)用于为对象设置脚本安全性或初始化安全性，若设置成功则返回S_OK，若对象不识别给定的接口ID则返回E_NOINTERFACE，若对象不支持要改变的安全选项，则返回E_FAIL；其中参数riid传递对象的接口ID，参数dwOptionsSetMask传递要设置的安全选项，参数dwEnabledOptions传递为安全选项设置的值，其值可以取下列值：INTERFACESAFE_FOR_UNTRUSTED_CALLER (riid接口被设为脚本安全性)，INTERFACESAFE_FOR_UNTRUSTED_DATA (riid接口被设为初始化期间非受托数据是安全的)。通常，一个控件的客户程序在执行该对象的脚本或初始化该对象前首先要调用该方法获知要进行的操作是否安全，如果返回失败，客户程序会提示一个用户窗口让用户决定是否继续执行操作。

ii .IobjectSafety 接口的使用方案

使用 IobjectSafety 接口 可以将对象及其组成部分标记为以下三类：

·对未受托的 Automation 客户和脚本设置 Automation 安全性

· 对未受托数据设置初始化安全性

· 对未受托脚本设置运行安全性

可以为您的对象考虑三种方案：

· 对象的所有属性和方法总是设置脚本安全性的。在该方案中，您可以通过不在IObjectSafety_SetInterfaceSafetyOptions 接口方法中返回错误来通知客户您的对象设置了脚本安全性。这样就能成功地创建和运行您的对象了。

· 对象永不设置脚本安全性。在这种情况下，您可以通过在 IObjectSafety_SetInterfaceSafetyOptions 接口方法中返回一个错误，E_Fail 来通知客户您的对象不安全。这样就防止了客户通过脚本来访问任何方法或属性了。

· 对象上的部分属性和方法，但不是全部，未设置脚本安全性。在这种情况下，您可以通知客户整个对象都不安全，也可以通知客户您的对象是安全的，但禁用不安全的属性和方法。

在这里我们采用第一种方案。

3.2 实现代码

(1) 在标准模块 Module1 中的代码：

```
Option Explicit
' 接口标识符串常量声明
' IID_Idispatch  自动化脚本操作接口标识符串
' IID_IPersist*  初始化数据操作接口标识符串
Public Const IID_IDispatch = "{00020400-0000-0000-C000-000000000046}"
Public Const IID_IPersistStorage = "{0000010A-0000-0000-C000-0000000000046}"
Public Const IID_IPersistStream = "{00000109-0000-0000-C000-0000000000046}"
Public Const IID_IPersistPropertyBag = "{37D84F60-42CB-11CE-8135-00AA004BB851}"

' IObjectSafety 的选项位定义
Public Const INTERFACESAFE_FOR_UNTRUSTED_CALLER = &H1
Public Const INTERFACESAFE_FOR_UNTRUSTED_DATA = &H2
Public Const E_NOINTERFACE = &H80004002
' 不支持这样的接口
Public Const E_FAIL = &H80004005 ' 非指定错误
Public Const MAX_GUIDLEN = 40 '对于NT 这个必须是40
' API 声明 ...
Public Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (pDest As Any, pSource As Any, ByVal ByteLen As Long)
Public Declare Function StringFromGUID2 Lib "ole32.dll" (rguid As Any, ByVal lpstrClsId As Long, ByVal cbMax As Integer) As Long

' UDT 声明 ...
Public Type uGUID
    Data1 As Long
    Data2 As Integer
    Data3 As Integer
    Data4(7) As Byte
End Type

Public Function GetIIDFromPTR(ByVal riid As Long) As String
    Dim Rc      As Long ' 函数返回代码
    Dim rClsId   As uGUID ' 指导结构
    Dim bIID()   As Byte ' 接口标识符的字节数组
    If (riid <> 0) Then ' 对界面标识符验证点
        CopyMemory rClsId, ByVal riid, Len(rClsId) ' 复制界面指导到结构
        bIID = String$(MAX_GUIDLEN, 0) ' 预分配字节数组
        Rc = StringFromGUID2(rClsId, VarPtr(bIID(0)), MAX_GUIDLEN) ' 从指导结构获得 clsid
        Rc = InStr(1, bIID, vbNullChar) - 1 ' 寻找尾随的空字符
        GetIIDFromPTR = Left$(UCase(bIID), Rc) ' 去掉额外的空，并且为了比较转换为大写字母
    End If
End Function
```

(2) UserControl1 代码窗口中的代码

```
Option Explicit
Implements IObjectSafety
Public Event Change()
'---------- 下面是缺省接口的实现代码
Private Sub DTPicker1_Change()
  RaiseEvent Change
End Sub
```

```
Public Property Get value() As Date
  value = DTPicker1.value
End Property

Public Property Let value(ByVal rq As Date)
  DTPicker1.value = rq
End Property
'每次使用该控件时自动提取当前日期
Private Sub UserControl_Initialize()
  DTPicker1.value = Date
End Sub

'---------- 下面是 IobjectSafety 接口的实现代码
Private Sub IObjectSafety_GetInterfaceSafetyOptions(ByVal riid As Long, pdwSupportedOptions As Long, pdwEnabledOptions As Long)
  Dim IID As String '接口标识符串
  pdwSupportedOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER Or
  INTERFACESAFE_FOR_UNTRUSTED_DATA '设置支持的安全
  IID = GetIIDFromPTR(riid) '由接口 ID 获取接口标识符串
  Select Case IID
  Case IID_IDispatch 'Idispatch 接口
    'Idispatch 是脚本操作接口，若控件是脚本安全的，设置 pdwEnabledOptions
    '为 INTERFACESAFE_FOR_UNTRUSTED_CALLER，否则设为 0
    pdwEnabledOptions =INTERFACESAFE_FOR_UNTRUSTED_CALLER
    Exit Sub '退出成功返回
  Case IID_IPersistStorage, IID_IPersistStream, IID_IpersistPropertyBag
    'Ipersist* 是初始化操作接口，若控件是初始化安全的，设置 pdwEnabledOptions
    '为 INTERFACESAFE_FOR_UNTRUSTED_DATA，否则设为 0
  pdwEnabledOptions = INTERFACESAFE_FOR_UNTRUSTED_CALLER
    Exit Sub '退出成功返回
  Case Else
    '其他未知接口引发错误 E_NOINTERFACE
    Err.Raise E_NOINTERFACE
  End Select
  Err.Raise E_FAIL '引发错误 E_NOINTERFACE
End Sub

Private Sub IObjectSafety_SetInterfaceSafetyOptions(ByVal riid As Long, ByVal dwOptionsSetMask As Long, ByVal dwEnabledOptions As Long)
  Dim fSettings  As Long '安全设置标志
  Dim IID  As String
   fSettings = (dwEnabledOptions And dwOptionsSetMask) '获取安全设置标志
  IID = GetIIDFromPTR(riid)
  Select Case IID
  Case IID_IDispatch
    '若控件不是或不能设为脚本安全的，就返回错误 E_FAIL
    'Err.Raise E_FAIL
    '若控件是脚本安全的或通过屏蔽一些属性功能做成安全的，则成功返回
    If (fSettings = INTERFACESAFE_FOR_UNTRUSTED_CALLER) Then
      Exit Sub
    End If
  Case IID_IPersistStorage, IID_IPersistStream, IID_IPersistPropertyBag
    '若控件从来就不是初始化安全的，就返回错误 E_FAIL
    '若控件总是初始化安全的，则成功返回
    If (fSettings = INTERFACESAFE_FOR_UNTRUSTED_DATA) Then
      Exit Sub
    End If
  Case Else
    Err.Raise E_NOINTERFACE
  End Select
  Err.Raise E_FAIL
  End Sub
```

最后生成OCX控件，至此新的安全性控件制作完成。在页面上嵌入该控件，浏览器正常安全级别下可正常运行。

本例可作为一个参照，让开发者可根据需求改造控件、添加安全性接口，以满足 Web 应用的需要。但必须保证改造后的控件一定要满足安全性要求，否则如果对客户机的资源造成破坏要承担法律责任。■