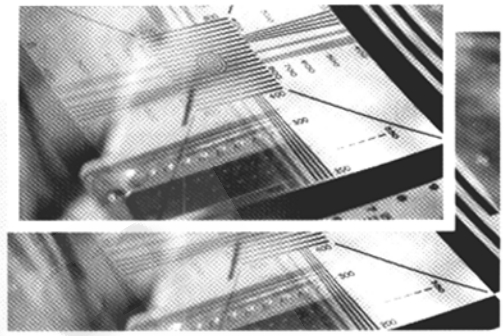
利用COM技术全面控制IE



刘 洋 (北京华北计算技术研究所 100083)

摘要：随着Internet的发展，针对IE的操作日益增多。如何开发能与IE交互的程序越来越受到关心，在一些知名的技术站点上经常有人询问类似的问题。本文利用COM技术充分展示了控制IE的各种方法。

关键词：COM/COM+ 接口 类厂 可连接对象理论 统一数据传输理论

1 引言

随着Windows 2000的出现，COM+技术对Windows软件技术正起着革命性的作用。对于Windows程序开发人员来说，了解COM+的核心结构是必备的，这不仅仅表现在WindowsDNA提供的基于COM+的三层框架中和所谓的中间件技术中。其实在我们的实际编程工作中也经常要利用COM+技术解决一些非常有实际意义的小问题。

2 组件对象模型基本编程架构

COM/COM+当前实现两个领域内的功能：创建软件组件的基本编程架构，以及高级软件组件的运行环境所关联的整套集成组件服务。这类似于Windows编程和集成化开发环境的关系。在这里本文只谈前者。

简单地说，COM是一种协议，它建立了一个软件模块同另一个软件模块的连接，然后再将其描述出来。当连接建立起来后，两个模块就可以通过"接口"的机制来进行通信。COM标准包括规范和实现两部分。规范部分定义了组件和组件之间通信的机制，只要按照该规范，任何语言都能用。COM可以利用C++来实现，也可以利用VB、Delphi、C等等语言来实现。对象和接口是认识和使用COM的最基础的概念。可以这么认为，COM对象通过接口来显示其功能。接口是COM对象和外界之间的绑定约定。不管是利用COM对象还是开发与COM相关的东西，接口是交流的gateway。对象可以支持多个接口，因此对组件对象的升级可通过增加接口的办法。从而，老用户可以在不更新代码的情况下继续使用老接口。按照COM的规范，如果一个对象没有至少实现一个最小程度为IUnknown的接口，那它就不是Microsoft的组件对象模型(COM/COM+)。

类厂就是COM类的工厂，类厂的主要作用就是创建对象。也就是说，类厂创建了COM对象（CreateInstance），然后客户程序才能调用COM对象中的接口指针(QueryInterface)。(在这里不区分进程内组件和进程外组件的概念)。

既然类厂创建COM对象，那么谁创建类厂呢？答案就是COM库；COM库在整个COM对象体系结构中起着重要的作用，COM库除了定义组件程序和客户程序交互的规范外（注：有关COM规范请查阅MSDN），它也提供了COM的实现部分 ---COM库，使得这些规范被真正利用起来。

图 1 COM库的作用

COM库经过初始化（CoInitialize）后，创建类厂（CoCreateInstance），调用一系列接口（QueryInterface）。这就是组件与客户通过COM库进行交互的基本过程。

其他一些与IE相关的COM理论（可连接对象理论、统一数据传输理论）将结合实际应用进行介绍。

3 实际应用

随着因特网的广泛应用，人们与浏览器的关系越来越密切。特别是随着电子商务的快速发展，开发利用浏览器或基于浏览器的应用程序的频率正在加大(注意:这里的浏览器专指IE。Netscape的开发利用采用的是DDE(动态数据交换机制)不属于COM理论的范畴)。而IE浏览器

是典型的构件技术产品，也就是说当开发人员要利用C++等语言开发利用IE和基于IE浏览器的程序时，唯一的办法就是利用组件对象模型技术来实现客户端程序或服务器端程序。

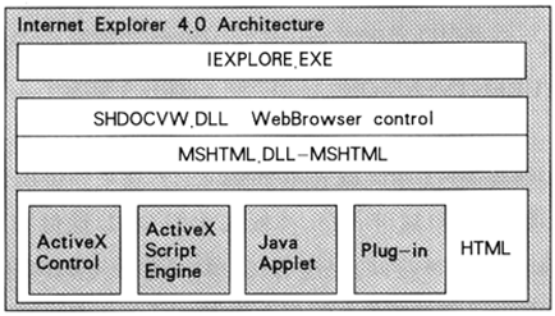


图 2 IE的组成结构

在上图的最顶层是Iexplore.exe.这个程序相当小，我们就是通过此程序来调用IE组件来执行导航(navigation)、查阅历史记录、HTML解析等等功能。Iexplore.exe直接调用Shdocvw.dll组件，shdocvw.dll调用Mshtml.dll组件和其他Activex文档。

(1) shdocvw.dll：这种组件就是我们常说的WebBrowser控件。它支持导航、查阅历史记录和就地链接功能。很多所谓的新型浏览器就是直接或间接建立在这上的。如果要用户化IE浏览器，例如改变工具栏、快捷菜单等操作也可以利用它来进行。

(2) Mshtml.dll: 在本文中，Mshtml动态连接库的作用要比WebBrowser控件的作用大。因为解析HTML等等更有实际意义的工作就是通过它来完成的。而开发人员在开发利用IE的程序时，很大程度上是与Mshtml.dll在打交道。本文所有例程均与Mshtml.dll有关。在这里，本文主要强调利用COM接口及一些COM的相关原理（可连接对象理论、统一数据传输理论）来全面控制IE浏览器。

3.1 应用示例一

在一些电子商务网站中当进行网上支付时，往往要添写你的信用卡。有一些网站支持虚拟信用卡的自动填写功能，就是在合法用户登陆到网站后，不用再手动填写每一个Form窗口，只需响应某些动作（例如，双击虚拟信用卡），就将虚拟信用卡的内容正确填写进去。这个例子很实用也很典型。当利用VC++开发类似的程序时，一般的做法如下：

(1) APPWIZARD生成普通对话框程序

(2) 在stdAfx.h文件中添加如下内容

```
#include <atlbase.h>// 为了简便安全起见，引入智能指针
#include <comdef.h>// 支持 _VARIANT_T 类型
```

(3) 始化COM库 在构造函数中调用::Coinitialize()

(4) 触发函数中添加如下内容

①利用CoCreateInstance创建实例，得到所有窗口实例，IE自然也包括在内。

②利用IshellWindows的get_counts,Item和QueryInterface得到IE实例的接口。

③得到IE对象后，既得到了Shdocvw.dll的一个接口。下面利用get_document和QueryInterface要得到文挡对象指针，既得到Mshtml.dll的接口。

④下面就可以进行我们想要进行的工作了。整个操作过程就是要求不断得到相应的接口，利用QueryInterface函数和智能指针（CCOMPtr）进行操作。

⑤利用IHTMLElementCollection接口得到指向Form实体的指针。

⑥针对相应的Form实体，判断实体的类型并进行填写。一般来说有这些实体IHTMLInputTextEleme、IHTMLSelectElement、IHTMLTextAreaElement、IHTMLInputButtonElement等 利用QueryInterface接口和Item接口得到具体某个FORM控件。

⑦下面就可以调用针对每个控件的put_values函数进行填写。需要注意的是在调用put_values接口时应该尽量利用CComBSTR产生BSTR型字符，因为这样将省去初始化等很多麻烦。

3.2 应用示例二

其实很多与IE相关的操作均可以如法炮制，例如过滤网页和自动翻译等等。但以上的做法只能单向通信，假如你想在你的intranet上开发一种应用程序来禁止你的使用者浏览集团外的URL。也就是说，如果我们要做一种工具来全面控制IE，从而能得到所有IE发生的事件的话，单纯实现客户端是不行的。原因很简单：客户程序没有实现事件接收器（Event Sink）。根据COM的理论，如果一个COM对象支持一个或多个出接口,则我们称这样的对象为可连接对象(connectable object)或源对象(source).例一中所涉及到的接口均为入接口，一旦接受到客户的请求便做出反应.同样的道理,出接口就是由客户程序实现,客户实现这些接口,并把这些接口指针告诉对象,以后对象利用出接口与客户进行通信。在客户方实现入接口的对象就是接收器(sink), 如图3。

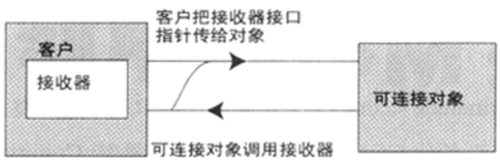


图 3　可连接对象与接收器关系

接收器对象和连接点对象之间是如何建立关系的呢?这个过程完全是由客户方控制的:

(1) 客户首先询问对象是否为可连接对象，调用对象的QueryInterface成员函数，请求IconnectionPointContainer接口，若成功则表明为可连接对象。

(2) 客户得到正确返回的IconnectionPointContainer指针后调用这个接口的成员函数FindConnectionPoint与可连接对象中的某一个连接接口建立连接，成功返回指向IconnectionPoint的指针。

(3) 利用接口IconnectionPoint的成员函数EnumConnections、Advise和Unadvise来管理可连接对象与接收器之间的连接。一般情况下，若客户没有实现接收器在这一步就会失败。

一句话，要想得到IE的事件，从而能全面控制IE就必须实现接收器。抛开MFC和ATL，单纯利用C++来接收事件要经过5个步骤:

①得到指向IconnectionPointContainer的指针。利用前面的技巧得到浏览器窗口，IwebBrowser* spBrowser;然后调用以下函数QueryInterface(IID_IConnectionPointContainer,(void**)&spContainer);

②调用FindConnectionPoint方法来找到需要的连接点，对于IE而言，要得到DwebBrowserEvent2连接点的接口。

spContainer->FindConnectionPoint(DIID_DWebBrowserEvents2, &spConnectionPoint);

③ Advise此连接点。spConnectionPoint->Advise((IDispatch *)this, &m_dwCookie);

④实现Idispatch的Invoke接口来接收任何由IE击发的事件。其中，Invoke函数的第一个参数为DISPID，它是一个长整数，它标识的是一个函数。对于IDispatch的某一个特定的实现，DISPID都是唯一的。IDispatch的每一个实现都有其自己的IID，这里dispidMemeber实际上是可以认为是和IE实例所发生的每一个事件相关的方法，比如：DISPID_BEFORENAVIGATE2，DISPID_NAVIGATECOMPLETE2等等。

⑤当要结束接收事件时，调用Unadvise。一般说来，如果在客户端没有实现接收器，以上的做法会失败的。而实现接收器的方法有简单也有复杂的方法。针对IE浏览器而言，从IE4.0版开始，引入了BHO(Browser Help Object)控件。在注册表＼＼HKEY_LOCAL_MACHINE＼＼SOFTWARE＼＼MICROSOFT＼＼WINDOWS＼＼CURENTVERSION＼＼EXPLORER下存放

当每次启动IE时，IE实例都将动态绑定BHO的实体，IE实例会通过调用CoCreateInstance()方法来创建BHO实体，假如我们利用ATL模板来生成DLL控件并实现IobjectWithSite接口，在注册表中将其注册为BHO控件，这恐怕是最简单的接收器的实现方法了。几乎不用编程就实现了接收器，上面的代码就会成功。这也是编码中最常见的技巧。但根据经验，这种做法不太有效，一般假如开发人员要对IE进行用户化时可以采用这种做法(例如每次可以在启动IE时令其最大化)。在这里，采用MFC或ATL技术，自己实现接收器的办法更好。需要说明的是在利用MFC或ATL时，一些步骤与用C++实现的5步或多或少不太一样。

首先介绍MFC方式:

在利用MFC实现COM可连接理论时，唯一烦人的就是要自己实现Sink。但由于MFC的封装性，这个问题其实也不是十分复杂。

首先，在程序中加入新的从CcmdTarget中派生类，因为CcmdTarget已经封装了Idispatch。

在CcmdTarget的派生类中，引入Dispatch Map，在派生类的实现文件中实现Dispatch Map。幸运的很，MFC的宏DECLARE_DISPATCH_MAP解决了引入的问题。在类实现文件中，引如类似代码。

```
BEGIN_DISPATCH_MAP(CIEEvents, CCmdTarget)
DISP_FUNCTION_ID(CIEEvents,......)
END_DISPATCH_MAP()
```

在构造函数中必须调用 EnableAutomation()，从而初始化Idispatch接口。

为了让可连接对象知道我们的接收器能接收的事件加入代码

```
BEGIN_INTERFACE_MAP(CIEEvents, CCmdTarget)
  INTERFACE_PART(CIE4Events, DIID_DWebBrowserEvents2, Dispatch)
END_INTERFACE_MAP()
```

这样我们就可以在例如OnBeforeNavigate2的函数中

接收这类消息了。例如可以在这种事件发生后将其记录到文件中。

虽然接收器构造完成了，但必须在客户程序中创建后它才能发挥作用。

首先要在构造函数中加入创建对象的函数，并在析构函数中将其delete掉。这样，一个完整的接收器就建立好了。下面就可以在对话框程序中响应代码

· 利用 CoCreateInstance 创建 IE 的实例

· 利用 AfxConnectionAdvise 建立接受器。然后，你就可以按照自己的意愿接收 IE 的事件了。需要注意的是一定要在程序结束时调用 AfxUnConnectionAdvise 函数。

对于ATL方式，其实和MFC方式没有什么区别。如果一定要指出不同的话，那就是对同一种功能的实现方式不太一样，对于ATL程序，是利用实现 Idispatch::Invoke 函数来实现的。这相对于MFC程序而言，要显得简洁一些。在 Invoke 函数内部，接收事件。关于IE能击发什么事件请参考 Internet Sdk 的技术支持资料。

3.3 应用示例三

拖拉的概念相信大家一定很熟悉了，使用过网络蚂蚁的人一定记得拖拉窗口，这几乎是COM理论最大众化的应用了。拖拉其实就是COM统一数据传输理论的简单应用。下面结合实现类似于网络蚂蚁的拖拉窗口来阐述如何在IE中应用统一数据传输理论。

数据交换是操作系统中不同应用之间通信的重要手段，所以建立一种有效的数据交换机制对于像 Windows 这样的桌面操作系统特别有意义。早期的DDE机制曾支持了很长一个阶段，就是现在DDE也非常常用，例如大名鼎鼎的Netscape依然利用DDE机制来解决IE利用COM机制解决的问题。不过，DDE毕竟过时了，而COM提供的统一数据传输机制既提高了效率也节约了资源。

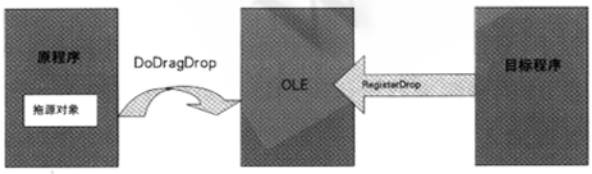


图 4 拖拉应用原理

在不同应用之间进行数据传输操作包括两方面内容，首先是数据格式的统一，其次是传输协议的建立。数据对象的概念使统一数据传输机制不仅可以用于应用之间传输数据，也可以成为组件程序之间的信息交换标准。拖放过程传输的是一个数据对象，所以在拖放前原程序必须创建一个数据对象，并且选中它作为传输的对象，经过拖放操作，此对象将被移动或复制到目标程序中。

在原程序方，为了支持“拖”的特性，它不仅要提供一个数据对象，还要实现一个“拖源”对象，“拖源”对象必须实现接口 IDropSource。

在目标程序方，为了接收数据对象即支持“放”特性，它必须提供一个“放目标”对象。放目标对象实现了 IDropTarget，并且，目标程序还必须把“放目标”与一个窗口联系在一起，以后OLE将根据窗口句柄找到“放目标”对象。因此，应用程序为了支持“放”的特性，它必须调用OLE提供的API RegisterDragDrop。把“放目标”与窗口联系起来。

相对于IE而言，在IE内部已经封装了IDropTarget和IdropSource接口，所以在客户端只要实现了IDropTarget接口就能够接收IE中的超级连接。

下面就一个示例来演示拖拉的实现过程。关键之处在于在从 CdropTarget 派生类中实现以下接口函数。

OnDragEnter OnDragOver OnDragLeave OnDrop

其中只有OnDrop是我们自己调用的，其他三个函数均为OLE调用而准备，实现方法也是十分的标准。所以主要介绍 OnDrop 函数。

(1) 判断数据对象是否为期望的类型 pDataObject->IsDataAvailable;

(2) 得到传进来的数据 GetData;

(3) 利用 GlobalLock 函数申请一块内存得到数据的首地址。

(4) 保存数据。

(5) 释放全局内存空间。

根据COM拖拉理论，在创建函数中必须调用登记函数，从而将小窗口定义为OLE接受窗口。

BOOL success = m_OleTarget.Register(this);

这样，一个标准的IE浏览器作为数据源，客户程序作为拖放目标的程序就基本实现了。

（注：本文所有例子均有源码，需要者请与本人联系：lyang3@sohu.com）■

参考文献

1 http://msdn.microsoft.com

2 《COM原理及应用》潘爱民