第 17 卷 第 4 期
2007 年 4 月

计 算 机 技 术 与 发 展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 17 No. 4
Apr. 2007

COM 中可连接对象的通信机制

侯 颖,孟小锁
(西安微电子研究所,陕西,西安 710075)

摘 要:COM(Component Object Model)技术是一种新的软件开发方法,它提供了跨越编程语言、操作系统甚至网络来访问组件对象的通用途径。通常,客户与组件之间的通信过程是单向的,客户创建组件对象,然后调用对象所提供的功能,这往往不能够满足实际的需要。对于一个全面的交互过程来说,组件对象也要主动与客户进行通信,可连接对象的通信机制可以很好地解决这个问题。并且对实际工作很有指导意义。

关键词:COM 技术;组件通信;可连接对象
中图分类号:TP274 **文献标识码**:A **文章编号**:1673-629X(2007)04-0222-03

The Communication Principle of Connectable Object in COM

HOU Ying, MENG Xiao-suo
(Xi'an Microelectronic Technology Research Institute, Xi'an 710075, China)

Abstract: COM (Component Object Model) technology as a new software developing method can provide general approach span programming language, operating system even network to visit component object. Usually, the communication between client and component object is one-way, the client creats the component object, then call the function supplied by component object. Such one way communication can't be able to satisfy practical need. For a mutual communication process, most time, component object should communicate with client activly, the communication principle of connectable object can resolve this problem well. This has guide meaning for practicality work.

Key words: COM technology; component communication; connectable object

在组件与客户的一般通信过程中,源对象的连接点机制和接收器的实现原理是一种理想的通信模型。在可连接对象机制中,虽然可连接对象提供了完善的双向通信机制,但客户要在运行过程中根据源对象的类型信息响应事件或请求并不容易。因此,将研究如何用 IDispatch 接口作为出接口,通过其"迟绑定[1]"特性实现运行时刻绑定事件控制函数。

1 可连接对象

一般情况下,组件通过入接口监听客户的请求,一旦接收到客户的请求便做出反应。组件对象则通过出接口与客户进行通信。

如果一个 COM 对象支持一个或多个出接口,就称这样的对象为可连接对象[2],也叫做源对象。可连接对象的出接口包含一组成员函数,每个成员函数代表了一个事件、一个通知或者一个请求,它们通过出接口的成员函数来实现。在客户方实现这些接口的对象被称为接收器,接收器也是一个 COM 对象。

1.1 可连接对象结构模型

可连接对象模型中,客户与可连接对象之间的通信是双向的,客户按照常规的途径调用对象提供的函数。对象通过它的出接口向客户发出请求。从对象一方来说,它的入接口和出接口分别承担了这两个通信过程;而从客户一方来看,前一个通信过程由客户来完成,后一个通信过程由客户方的接收器完成。

图 1 反映了可连接对象和客户、接收器之间的基本关系,客户程序把接收器的接口指针传给可连接对象,可连接对象可通过此接口指针调用接收器的成员函数。接收器有自己的引用计数,有自己的接口查询方法(QueryInterface 成员函数),但它位于客户程序内部,并不需要通过 COM 库来创建[2]。

由于接收器本身是一个 COM 对象,所以,可以通过引用计数控制自己的生存周期,这样,一个接收器可以被多个可连接对象使用,每个可连接对象都是接收器的客户。反过来,每个可连接对象也可以连接多个接收器,所以,可连接对象和接收器可以形成一对多或者多对一的关系。如图 2 所示[2]。

收稿日期:2006-06-04

作者简介:侯 颖(1979-),女,陕西西安人,硕士研究生,研究方向为实时嵌入式软件;孟小锁,研究员,硕士生导师,研究方向为实时嵌入式软件。

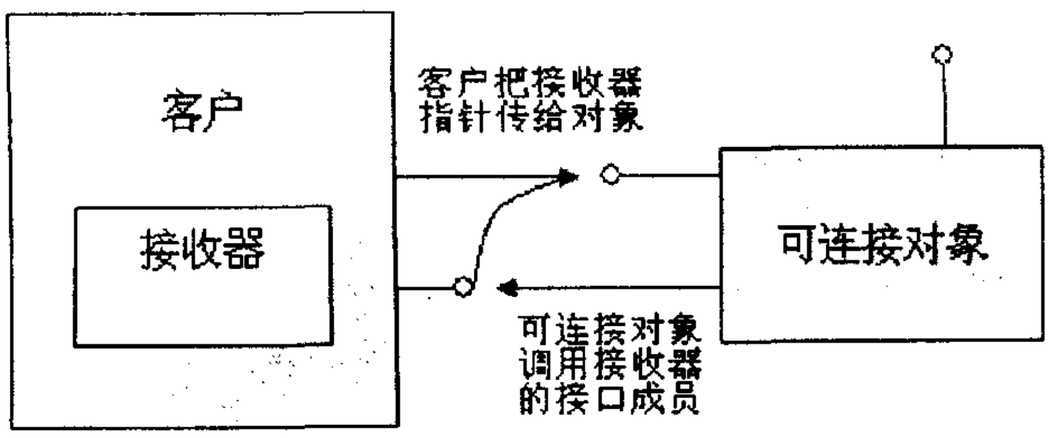

图 1　简单的客户和可连接对象模型

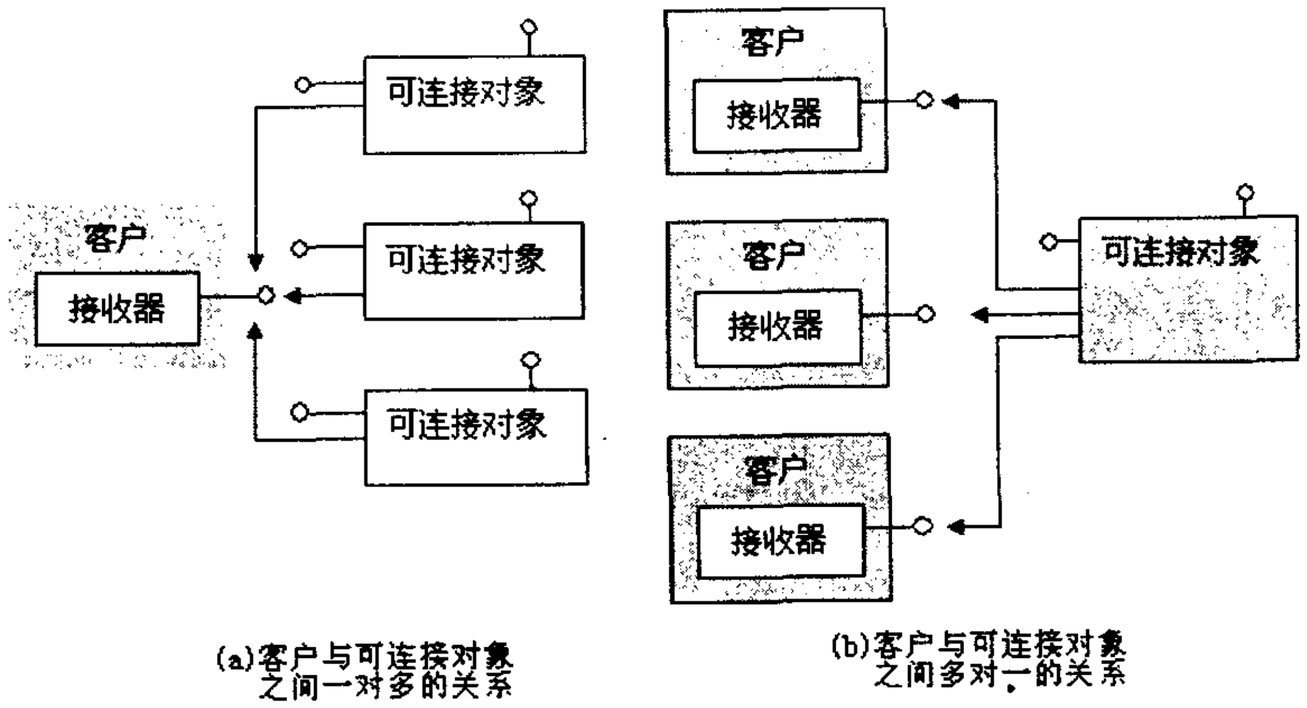

图 2　客户与可连接对象之间的两种结构图

1.2　可连接对象的基本结构

可连接对象支持一个或多个出接口,它通过接口 IConnectionPointContainer 管理所有的出接口。对应于每个出接口,可连接对象又管理了一个称为连接点的对象,并在其中实现了 IconnectionPoint 接口,客户通过连接点对象建立接收器与可连接对象的连接。连接点对象包含在可连接对象的内部,它既可以访问可连接对象的内部信息,也可以访问客户方的接收器。连接对象的基本结构如图 3 所示[2]:在 IConnectionPointContainer 的成员函数中的枚举器用于提供此对象所支持的所有出接口。

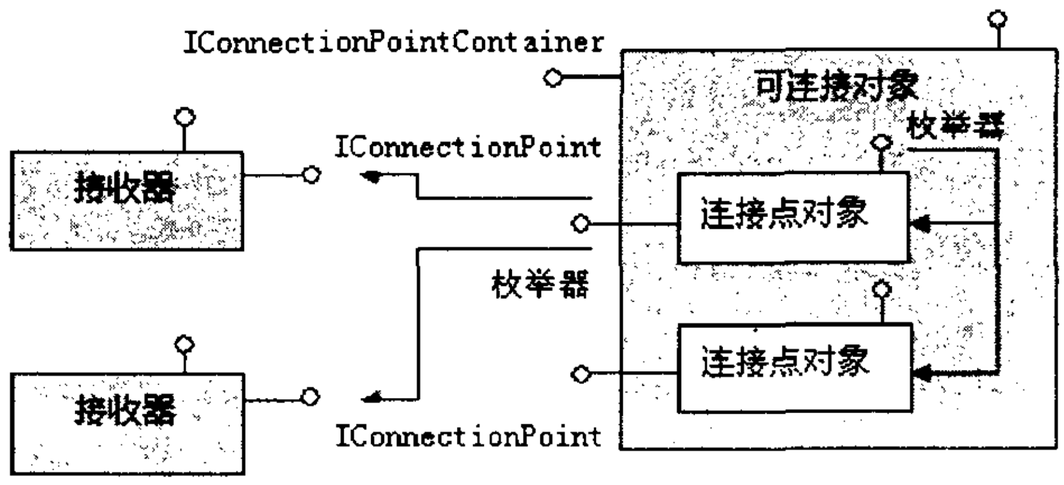

图 3　可连接对象的基本结构

1.3　客户的基本结构

客户方除了实现接收器对象之外,还要建立接收器与连接点对象之间的连接关系。客户中的接收器可以支持多个与可连接对象之间的连接,这样便于把其它对象发过来的事件或请求集中处理。

接收器对象与连接点之间的连接是由客户控制的,客户首先调用对象的 QueryInterface 成员函数,请求 IConnectionPointContainer 接口,如果成功,返回 IConnectionPointContainer 接口指针,然后,获取相应出接口的连接点对象,再调用连接点对象的 Iconnection-Point 接口成员函数建立连接。一旦连接建立,可连接对象激发的事件或发出的请求,接收器中的成员函数就会被调用。当客户取消连接时,同样调用 IconnectionPoint 接口的成员函数即可。可连接对象通过 IConnectionPointContainer 接口提供自己的出接口信息,只要 COM 对象支持出接口,客户就可以通过该接口的成员函数访问连接点。

IConnectionPointContainer 接口定义如下:

```
class IConnectionPointContainer:public IUnknown
{
    virtual HRESULT EnumConnectionPoints(IEnum-
Connectionpoints* *)=0;
    virtual HRESULT FindConnectionPoint(const IID
*,IConnectionPoint* *)=0;
};
```

该接口只有两个成员函数,EnumConnectionPoints 成员函数返回连接点枚举器用于访问 COM 对象的所有连接点。FindConnectionPoint 成员函数根据给定的接口 IID 返回出接口的连接点。若不支持该接口 IID,返回 CONNECT_E_NOCONNECTION。

每个连接点对象都对应一个出接口,一般情况下只实现 IconnectionPoint 接口。

该接口的定义[3]如下:

```
class IConnectionPoint:public IUnknown
{
    virtual HRESULT GetConnectionInterface(IID* pIID)=0;
    virtual HRESULT GetConnectionPointContainer(IConnection-
PointContainer* * ppCPC)=0;
    virtual HRESULT Advise(IUnknown * pUnk,DWORD *
pdwCookie)=0;
    virtual HRESULT Unadvise(DWORD dwCookie)=0;
    virtual HRESULT EnumConnections(IEnumConnections* *
ppEnum)=0;
};
```

该接口有 5 个成员函数:GetConnectionInterface 成员函数返回对应出接口的 IID; GetConnection-PointContainer 成员函数返回可连接对象的 IConnectionPointContainer 接口指针;成员函数 EnumConnections,Advise,Unadvise 用于管理可连接对象与接收器之间的连接。

2　客户 - 可连接对象 - 接收器的协作

如前所述,接收器[2]是客户方的一个内部对象,它

没有 CLSID 和类厂。它只实现一个接口,较为简单,它的基本框架的定义如下:

```
class CSomrEventSet:public ISomeEventSet
{
  private:
    ULONG m_cRef;
  ……
  public:
    DWORD m_dwCookie;
  public:
    CSomeEventSet();
    CSomeEventSet();
  //IUnknown members
    STDMETHODIMP QueryInterface(REFIID,PPVOID);
    STDMETHODIMP_(DWORD)AddRef(void);
    STDMETHODIMP_(DWORD)Release(void);
    //ISomeEventSet member
    STDMETHODIMP SomeEventFunction(……);
    ……
};
```

其中 IUnknown 的成员函数的实现方法与一般 COM 对象的实现方法相同,而其它的接口成员函数才是真正的事件控制函数,是接收器的主体。

客户方的接收器实现出接口,如果客户已经明确知道了它所希望连接的出接口,以及出接口的 IID 和成员函数的定义,那么客户可以很方便地在源代码中创建连接器对象。但在实际中,因为客户和组件通常是独立开发的,客户在编译时刻不一定知道对象支持什么样的出接口,即使客户知道了出接口的所有类型信息,包括成员函数、函数的参数个数和参数类型等,那么要在程序运行过程中根据可连接对象的类型信息响应事件或请求实现动态接收器对象并不容易。为此将在后面研究以 IDispatch 接口作为出接口,利用 IDispatch 接口中方法的分发功能实现事件控制函数。

3 IDispatch 接口实现出接口

前面对一般的可连接对象机制,包括源对象的连接点机制和接收器的实现原理进行了研究。但是用一般的自定义接口作为出接口,则客户和源对象在编译时刻就要确定出接口的定义,客户要在运行时刻获取出接口信息,并动态创建接收器对象实现起来很不容易。但是可以用 IDispatch 接口作为出接口[4],通过其"迟绑定"特性实现运行时刻绑定事件控制函数。

接口 IDispatch 的定义如下:

```
class IDispatch:public IUnknown
{
  public:
    virtual HRESULT GetTypeInfoCount(UNIT * pctinfo)=0;
    virtual HRESULT GetTypeInfo(UNIT iTInfo,LCID lcid,ITypeInfo * * ppTInfo)=0;
    virtual[5] HRESULT GetIDsOfNames(REFIID riid,LPOLESTR * rgszName,UNIT cNames,LCID lcid,DISPID * rgDispId)=0;
    virtual HRESULT Invoke(DISPID dispIdMember,REFIID riid,LCID lcid,WORD wFlage,DISPPARAMS * pDispParams,VARIANT * pVarResult,EXCEPINFO * pExcepInfo,UINT * puArgErr)=0;
};
```

成员函数 GetTypeInfoCount 和 GetTypeInfo 用于获取类型信息;成员函数 GetIDsOfNa－Mes 用于返回成员函数的分发 ID(DISPID,dispatch identifier),IDispatch 用分发 ID 管理属性和方法;成员函数 Invoke 是个关键函数,客户必须通过 Invoke 函数访问属性或方法,也可以认为 Invoke 函数是自动化对象的命令翻译器,Invoke 函数根据第一个参数 dispIdMember 给出的分发 ID 执行有关的属性访问函数或者方法函数;wFlage 参数指示是访问属性还是调用方法;pDispParams 参数类型为 DISPPARAMS,它包括方法和属性调用的参数数组、参数的分发 ID 数组、数组中参数的个数等信息;参数 pVarResult 保存返回值信息。

因此,可以看到用 IDispatch 接口作为出接口可以很好地解决接收器的动态创建过程。IDispatch 接口把所有的调用都通过 Invoke 成员函数来实现,并提供了管理属性和方法的分发 ID 机制,以及一套描述参数和返回值的方法。图 4 为用 IDispatch 接口作为出接口的模型图。

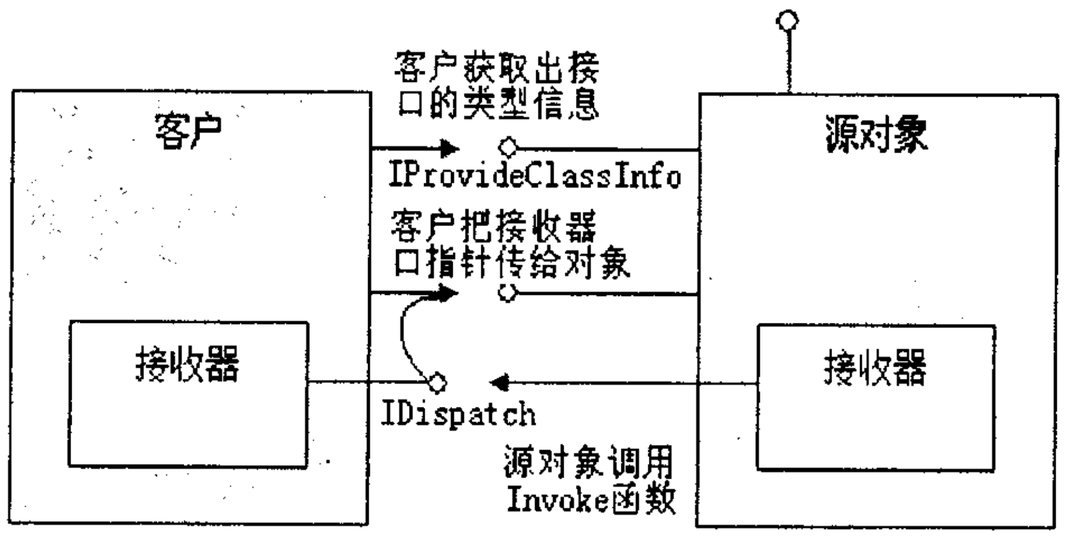

图 4 用 IDispatch 作为出接口的模型图

下面来描述用 IDispatch 接口作为出接口的过程。

首先,从 IDispatch 派生新的接口作为出接口,把属性和方法加到派生接口中,并为每一个属性和方法赋予分发 ID,源对象通过某种途径把出接口的类型信息暴露出来。比如通过类型库文件或者 IProvideClassInfo 接口。当源对象激发事件或请求时,把调用参数和返回信息设置正确,再调用 IDispatch::Invoke 成员函数即可。

(下转第 228 页)

市提名候选人 47 名

姓名	反对	弃权	姓名	反对	弃权
王华元		X	王敏		✓
邓大荣	○		王仁元		✗
叶力行		✓	王军民	○	
丘海		✓	王进诚		✓
吕成贤	✗		王宗廉	○	
周振中		○	王春涛	X	

图 5 二值化之后的选票图

市提名候选人 47 名

姓名	反对	弃权	姓名	反对	弃权
王华元		X	王敏		✓
邓大荣	○		王仁元		✗
叶力行		✓	王军民	○	
丘海		✓	王进诚		✓
吕成贤	✗		王宗廉	○	
周振中		○	王春涛	X	

图 6 平滑后的选票图

3 小 结

预处理目的是为减少由于传输设备不精确而产生的干扰信号,是为纠正由于书写的随意性而造成的字体变形,文中针对选票选举系统中选票图像,提出了几种预处理方法,为后续的图像识别做了很好的准备。

参考文献:

[1] 刘子贵.基于 PC 机选票处理系统的设计[D].南宁:广西大学,2004.

[2] 王正群.手写体汉字识别研究[D].南京:南京理工大学,2001.

[3] 吕凤军,数字图像处理编程入门[M].北京:清华大学出版社,2001.

[4] 马 驰,张红云,苗夺谦,等.改进的多阈值动态二值化算法[J].计算机工程,2006,32(6):203-206.

[5] Kavallieratou E, Fakotakis N, Kokkinakis G. Skew angle estimation using Cohen's class distribution[J]. Pattern Recognition Letters,1999,20(12):1305-1312.

[6] 陈书海,傅录详.实用数字图像处理[M].北京:科学出版社,2005:148-155.

(上接第 224 页)

然后,客户方按照源对象提供的出接口类型信息实现接收器对象,一般情况下,IDispatch 接口的前三个函数 GetTypeInfoCount, GetTypeInfo 和 GetIDsOfNames 实际意义不大,可以不实现这三个成员函数的功能,简单地返回 E_NOTIMPL 即可。关键是 Invoke 成员函数的实现,当接收到事件或请求时,Invoke 成员函数分析参数值,如果它可以接收,则调用事件控制函数或请求的响应函数。接收器除了处理应用计数和 QueryInterface 之外,只需再实现 Invoke 函数即可。不同的开发环境和运行环境下实现 Invoke 函数的方法不同。如果在编译时刻可以决定客户应该响应哪些事件或请求,则可以在程序中建立一张表,把每个事件或请求的分发 ID 和对应的控制函数作为表项放到表中,该表称为事件映射表,然后在 Invoke 函数中进行查表操作,如果源对象的事件或请求的分发 ID 在表中能找到,则调用其相应的控制函数,否则不予处理。如果客户程序在运行过程中进行事件控制函数的绑定操作,则客户程序按一定规则找到事件控制函数,并根据源对象的类型信息找到控制函数的表项,在程序运行过程中,动态创建事件映射表,Invoke 函数用同样的查表法处理事件或请求。

4 结束语

文中通过对 COM 中组件的基本通信原理的研究与分析,研究了 COM 中可连接对象的实现及其方法。在针对通信过程中接收器的实现问题的研究中,给出了一种动态实现接收器的方法,从而为今后相关方面的研究与开发提供了方向。

参考文献:

[1] 孙延琳,詹振炎.COM 中的命名和绑定技术[M].计算机工程与设计,2002,23(9):66-68.

[2] 潘爱民.COM 原理与应用[M].北京:清华大学出版社,1999:176-193.

[3] 黄维通.Visual C++面向对象与可视化程序设计[M].北京:清华大学出版社,2000:30-32.

[4] Rogersor D.COM 技术内幕[M].杨秀章译.北京:清华大学出版社,1999:243-266.

[5] 鸿志创作组.Visual C++5.0[M].北京:科学出版社,1998:43-46.