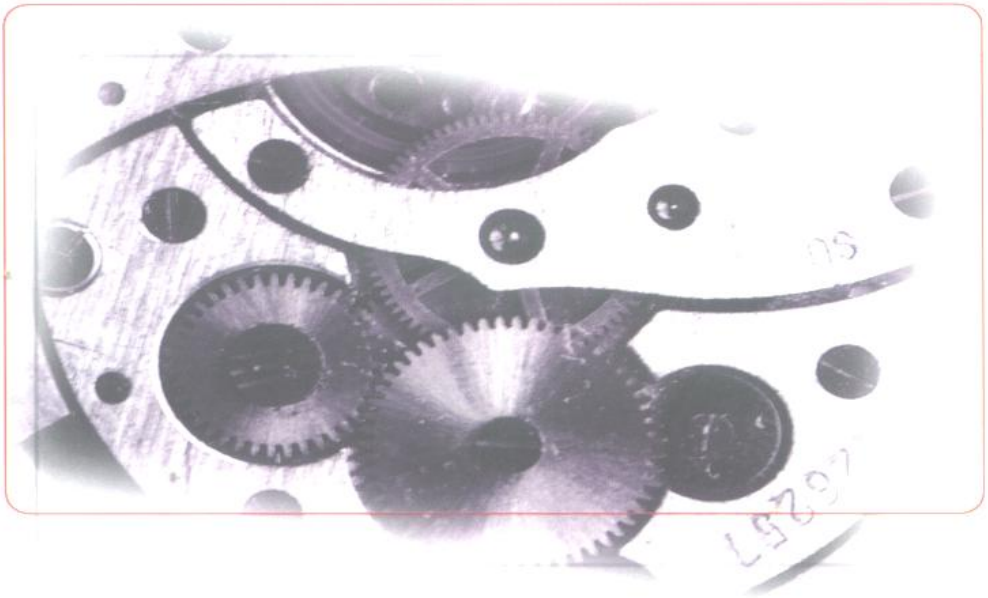
Delphi 5
高级编程丛书之四

Delphi 5 高级编程

—COM、CORBA与Internet编程

徐新华　编著



人民邮电出版社
www.pptph.com.cn

Delphi 5 高级编程丛书之四

Delphi 5 高级编程
——COM、CORBA 与 Internet 编程

徐新华　编著

人 民 邮 电 出 版 社

内 容 提 要

本书全面深入地介绍了 COM、Interface、ActiveX 框架、类型库、ActiveX 控件、OLE Automation、MTS、CORBA、ASP、WinSock、Internet 协议、Web 服务器应用程序和 MIDAS Web 应用程序。这些内容对于开发分布式应用程序和 Web 应用程序是非常重要的。

对于某些程序员来说，上述内容可能具有相当的难度。不过，只要真正领会了 Delphi 5 面向对象的编程思想，还是能够掌握这些编程技术的。

本书内容全面而又不失简洁，例子丰富。既可以作为广大读者学习 Delphi 5 的入门指导书，也可以作为程序员编程时的参考手册。

本书读者对象为计算机程序开发人员、大专院校计算机专业师生。

Delphi 5 高级编程丛书之四

Delphi 5 高级编程

——COM、CORBA 与 Internet 编程

◆ 编　　著　徐新华

责任编辑　李　际

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街 14 号

北京汉魂图文设计有限公司制作

北京朝阳展望印刷厂印刷

新华书店总店北京发行所经销

◆ 开本：787 × 1092　1/16

印张：25.25

字数：629 千字　　2000 年 5 月第 1 版

印数：1 – 6 000 册　　2000 年 5 月北京第 1 次印刷

ISBN 7-115-08496-3/TP·1607

定价：40.00 元

前　言

Delphi 5 是 Inprise 公司面向 Internet 和电子商务推出的一个战略产品。在 Internet 日益普及和众多厂商看好电子商务的今天，Delphi 5 的推出必将受到程序员的热烈欢迎。

为了帮助广大用户全面、准确地掌握 Delphi 5 的编程思想和用法，我们编写了这套《Delphi 5 高级编程丛书》。作为《Delphi 4 高级编程丛书》的升级，本套丛书在保留原丛书精髓的基础上，对丛书的结构和内容作了大刀阔斧的修改，这是考虑到很多程序员已经初步掌握了 Delphi 5 的基本用法，现在需要的是进一步提高和精通。

本套丛书分为 4 册。第一册是《Delphi 5 高级编程——IDE 与面向对象编程》，第二册是《Delphi 5 高级编程——GUI 编程》，第三册是《Delphi 5 高级编程——Database 与 MIDAS 编程》，第四册是《Delphi 5 高级编程——COM、CORBA 与 Internet 编程》。

本书是此套丛书的第四册，全面深入地介绍了 COM、Interface、ActiveX 框架、类型库、ActiveX 控件、OLE Automation、MTS、CORBA、ASP、WinSock、Internet 协议、Web 服务器应用程序和 MIDAS Web 应用程序。

由于我们的水平有限，再加上时间很紧，尽管我们作了严格的审核和测试，书中还是难免会有一些错误，敬请广大读者不吝赐教，我们谨在此表示感谢。

为了帮助广大程序员更好地掌握这个优秀的开发工具，北京东大阿尔发软件技术有限公司愿意为购买此书的读者提供技术咨询。

北京东大阿尔发软件技术有限公司
地址：北京市上地信息产业基地上地村路 1 号(100085)
电话：(010)62987260　传真：(010)62985141
网址：http://www.allfa.com.cn　邮件：books@allfa.com.cn

徐新华
2000 年 2 月

目 录

第一章 COM

本章有以下内容：

- 几个基本概念；
- 客户与服务器；
- 认识 GUID、CLSID、IID；
- 引用计数；
- 虚拟方法表；
- IUnknown 接口；
- 创建 In-Process COM 服务器；
- 创建 Out-of-Process COM 服务器。

组件对象模型(Component Object Model，以下简称 COM)是组件对象之间相互接口的规范，凡是遵循 COM 接口规范的对象彼此之间能相互通信和交互，即使这些对象是由不同的厂商、用不同的语言、在不同的 Windows 版本甚至不同的机器上编写和建立的。

Delphi 支持 COM 接口规范，Object Pascal 语言增加了对象接口的文法。用 Delphi 创建的 COM 对象还可以工作在 MTS(Microsoft Transaction Server)环境中。本章将详细介绍组件对象模型的体系结构，并用 Delphi 5 实际创建一个 COM 服务器。

1.1 几个基本概念

软件重用是业界追求的目标，人们一直希望能够像搭积木一样随意“装配”应用程序，组件对象就充当了积木的角色。所谓组件对象，实际上就是预定义好的、能完成一定功能的服务或接口。问题是，这些组件对象如何与应用程序、如何与其他组件对象共存并相互通信和交互？这就需要制定一个规范，让这些组件对象按统一的标准方式工作。

COM 是个二进制规范，它与源代码无关。这样，即使 COM 对象由不同的编程语言创建，运行在不同的进程空间和不同的操作系统平台，这些对象也能相互通信。COM 既是规范，也是实现，它以 COM 库(OLE32.dll 和 OLEAut32.dll)的形式提供了访问 COM 对象核心功能的标准接口以及一组 API 函数，这些 API 函数用于创建和管理 COM 对象。

COM 本质上仍然是客户/服务器模式。客户(通常是应用程序)请求创建 COM 对象并通过 COM 对象的接口操纵 COM 对象。服务器根据客户的请求创建并管理 COM 对象。客户和服务器这两种角色并不是绝对的。

组件对象与一般意义上的对象既相似也有区别。一般意义上的对象是一种把数据和操纵数据的方法封装在一起的数据类型的实例，而组件对象则使用接口(Interface)而不是方法来描述自己并提供服务。所谓接口，其精确定义是“基于对象的一组语义上相关的功能”，实际上是一个纯虚类，真正实现接口的是接口对象(Interface Object)。一个 COM 对象可以只有一个接口，例如 Windows 95/98 外壳扩展；也可以有许多接口，例如 ActiveX 控件一般就有多个接口，客户可以从很多方面来操纵 ActiveX 控件。

接口是客户与服务器通信的唯一途径。如果一个组件对象有多个接口，则通过一个接口不能直接访问其他接口。但是，COM 允许客户调用 COM 库中的 QueryInterface()去查询组件对象所支持的其他接口。从这个意义上讲，组件对象有点像接口对象的经纪人。

在调用 QueryInterface()后，如果组件对象正好支持要查询的接口，则 QueryInterface()将返回该接口的指针。如果组件对象不支持该接口，则 QueryInterface()将返回一个出错信息。所以，QueryInterface()是很有用的，它可以动态了解组件对象所支持的接口。

接口是面向对象编程思想的一种体现，它隐藏了 COM 对象实现服务的细节。COM 对象可以完全独立于访问它的客户，只要接口本身保持不变即可。如果需要更新接口，则可以重新定义一个新的接口，对于使用老接口的客户来说，代码得到了最大程度的保护。

1.2 客户与服务器

COM 本质上仍然是客户/服务器模式。COM 服务器实际上是组件对象的容器，而组件对象向 COM 客户提供服务。COM 客户通常是 EXE，也可能是 DLL，甚至就是 Windows 自己。COM 客户独立于 COM 服务器，因为 COM 客户并不知道 COM 服务器在哪里，甚至有没有这样的 COM 服务器都不知道。

当一个客户请求某个 COM 对象的服务时，客户需要传递一个类标识符(CLSID)，请求 Windows 去查找组件对象。如果 Windows 找到一个组件对象，就把接口的指针传递给客户。Windows 将从注册表中查找 COM 服务器的位置并定位一个合适的 COM 对象。

根据 COM 服务器与 COM 客户是否运行在同一个进程地址空间，COM 服务器分为 3 种，分别是 In-Process 服务器、Out-of-Process 服务器和 Remote 服务器。

In-Process 服务器通常是 DLL，它可以输出 COM 对象，并映射到客户的进程地址空间中运行。例如，一个嵌入在 Web 网页中的 ActiveX 控件，与 Internet Explorer 在同一个进程地址空间中运行。对于 In-Process 服务器来说，客户可以直接调用 COM 对象的接口。

Out-of-Process 或 Local 服务器通常是 EXE，它与 COM 客户虽在同一个机器但在不同的进程地址空间运行。例如一个嵌入到 Word 文档中的 Excel 电子表格就是 Local 服务器。

Remote 服务器可以是 EXE，也可以是 DLL，它与 COM 客户运行在不同的机器上。例如，用 Delphi 5 编写的应用服务器与“瘦”客户程序通常不在同一个机器上运行。

图 1.1 是 Out-of-Process 服务器和 Remote 服务器的示意。

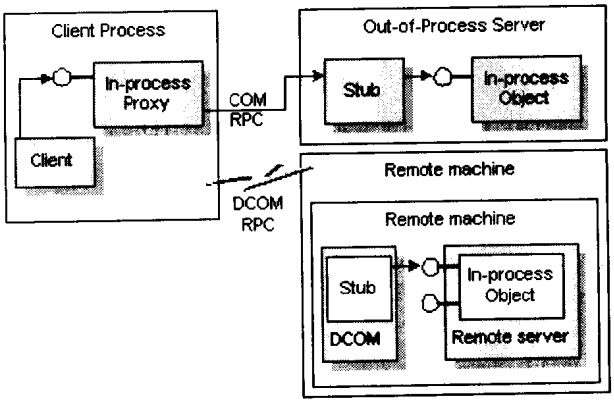


图 1.1 跨进程或机器边界的调用

Remote 服务器跨越了机器边界，如果仍然用 COM 来描述这种模型显然是不够的。Microsoft 扩展了 COM 模型，推出了称为 Distributed COM(简称 DCOM)的模型。DCOM 最初是在 Visual Basic 4.0 的专业版中被引入的，目前只有 Windows 98、Windows NT 4.0 真正实现了 DCOM，而 Windows 95 需要安装一个特殊的软件才能支持 DCOM。

为什么能够跨进程甚至机器边界访问组件对象呢？这就是 COM 模型中称为 Marshaling 的策略。Marshaling 实际上是一种打包技术，它先把要跨边界传递的函数调用信息进行必要的转换并打包，然后传递给另一个进程或者另一个机器，到达目的地后再解包并调用当地的组件对象。不过，这种跨进程边界或跨机器边界的传递是相当费劲的。如果 COM 服务器是

DLL 的话，就用不着 Marshaling，因为 COM 客户和 COM 服务器在同一个进程地址空间中运行。这种情况下，COM 客户只要搜索虚拟方法表就可以找到要调用的方法。

基于标准的远程过程调用(RPC)，IDispatch 接口提供了通用的 Marshaling 策略。COM 对象也可以自定义 Marshaling 策略。不过，自定义的 Marshaling 策略不具有通用性。

1997 年，Microsoft 又宣布了雄心勃勃的 COM+计划。COM+计划打算把类型库整合到 C++的编译器中，并且增加一个垃圾回收运行期环境，这样就不再需要 IUnknown 接口的 AddRef()和 Release()。后来，Microsoft 又宣布 COM+将把 Microsoft Transaction Server(MTS)整合到 COM 库中。不过，到目前为止，关于 COM+还没有一个具体的说明。

1.3 认识 GUID、CLSID、IID

在一个复杂的系统中，可能充斥着大量的组件对象，每个组件对象可能又有大量的接口。为了保证这些接口彼此不会冲突，Microsoft 规定用 GUID 来标识组件对象和接口。

GUID 是 Globally Unique Identifier 的缩写，意为全局唯一标识符。GUID 可以标识组件对象的类，这时候 GUID 也称为 CLSID(Class Identifier 的缩写)。GUID 也可以标识组件对象的接口，这时候 GUID 也称为 IID(Interface Identifier 的缩写)。

GUID 是一个 128 位整数(16 字节)，绝对能保证每一个组件对象具有唯一的标识符。

当用向导创建一个组件对象或者接口时，向导会自动生成组件对象或接口的 GUID。实际上，GUID 是通过一个叫 CoCreateGUID()的 API 来产生的。CoCreateGUID()会综合考虑当前的日期、时间、CPU 时钟序列、网卡编号以及 Bill Gates 的银行帐号上的余额来生成一个 GUID。如果您的机器中安装了网卡，所生成的 GUID 肯定是唯一的，因为每块网卡的编号都是唯一的。如果机器中没有安装网卡，CoCreateGUID()会考虑其他硬件信息。

在 Delphi 5 的代码编辑器中，按 Ctrl+Shift+G 键可以生成一个新的 GUID。

1.4 引用计数

引用计数是一种机制，使组件对象具有一定的“智能性”。它的工作原理是这样的：当接口对象第一次创建时，引用计数的初始值为 1。当有一个客户请求获得接口对象的指针时，就调用 AddRef()使该计数加 1。当一个客户不再需要组件对象的服务时，它应当调用 Release()。注意，Release()并不真正释放接口对象，因为可能还有其他客户正在使用接口。Release()只是使引用计数减 1。只有当引用计数正好减为零时，接口对象才被删除。

下面举例说明引用计数的作用。假设客户 A 向服务器请求 IMalloc 接口，服务器收到请求后，首先看该接口对象是否存在。如果没有，就创建一个接口对象，并调用 AddRef()使引用计数变为 1，同时把该接口对象的指针传递给客户 A。假设这时候客户 B 也加入进来，并且也是请求 IMalloc 接口。由于此时 IMalloc 接口对象已存在，所以服务器只是简单地返回一个指针，并且调用 AddRef()使引用计数变为 2。当客户 A 不再需要 IMalloc 接口

时，它就调用 Release()试图释放这个接口。显然，这时候不能删除 IMalloc 接口对象，因为客户 B 还在用着呢。可见，引用计数这种机制使服务器知道如何管理自己的接口。

引用计数这种机制也带来一个问题，就是调用 AddRef()和 Release()不能出现混乱。一旦出现混乱，可能导致接口对象永远不被删除或者过早地被删除。

1.5 虚拟方法表

COM 是个二进制规范，任何开发环境只要遵守这个规范都可以生产出 COM 对象。COM 采用一种称为虚拟方法表(*virtual method table*)的文法来解决方法调用。不过，COM 接口与 Object Pascal 的类还是有一些区别的。COM 接口中凡是要表露给客户的方法必须声明为纯虚的，客户得到的只是指向虚拟方法表的指针，具体实现接口的是接口对象。

如果建立了同一个 COM 对象的多个实例，则虚拟方法表是共享的，但每个实例的数据是私有的。图 1.2 是虚拟方法表的示意。

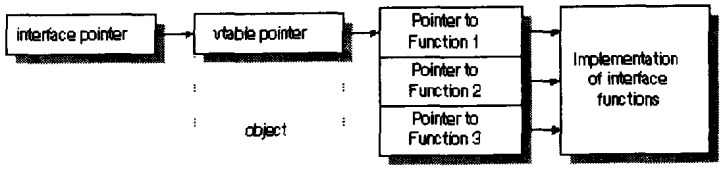


图 1.2 虚拟方法表的示意

在 Delphi 2 中，用 abstract 指示字来声明纯虚方法。例如：

```
TMyPureVirtualClass = class
public
    procedure MyMethod;virtual; abstract;
    ...
end;
```

上面这个例子中，MyMethod 被声明为抽象的，这意味着 TMyPureVirtualClass 并没有给出该方法的定义，而是由 TMyPureVirtualClass 的派生类具体实现这个方法。

而从 Delphi 3 开始，Object Pascal 的文法得到扩展，专门引入了对象接口(Object interface)的概念来描述 COM 接口。

1.6 IUnknown 接口

Addref()、Release()和 QueryInterface()是 COM 对象提供的三个核心服务，这三个方法构成了 IUnknown 接口。

在 System 单元中，IUnknown 接口是这样声明的：

```
IUnknown = interface
    ['{00000000-0000-0000-C000-000000000046}']
```