

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
印数: 1 - 6 000 册

2000 年 5 月第 1 版
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的欢迎。

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	1
1.1 几个基本概念	2
1.2 客户与服务器	3
1.3 认识 GUID、CLSID、IID	4
1.4 引用计数	4
1.5 虚拟方法表	5
1.6 IUnknown 接口	5
1.7 创建 In-Process COM 服务器	6
1.8 创建 Out-of-Process COM 服务器	11
第二章 对象接口	12
2.1 接口的文法	13
2.1.1 如何声明接口	13
2.1.2 祖先接口	15
2.1.3 接口标识符	15
2.1.4 前向声明	16
2.1.5 如何实现接口	16
2.1.6 映射	17
2.1.7 方法解析子句	18
2.1.8 委托一个特性来实现接口	20
2.2 分派接口	21
2.3 双重接口	23
2.4 对接口的引用	25
2.4.1 用赋值语句获得接口引用	25
2.4.2 赋值相容和类型强制转换	27
2.4.3 对派生接口的引用	28
2.4.4 接口查询	29
2.4.5 TInterfacedObject	29
2.5 使用接口的规则	30
2.6 用接口实现代码重用	34
第三章 ActiveX 框架	36
3.1 什么是 ActiveX 框架	37
3.2 TInterfacedObject	38

3.3 TComObject.....	38
3.4 TTypedComObject.....	40
3.5 TAutoObject.....	40
3.6 TAutoIntfObject.....	41
3.7 TActiveXControl.....	42
3.8 TComServerObject.....	45
3.9 TComServer.....	46
3.10 TActiveForm.....	48
3.11 TPropertyPage.....	49
3.12 TComObjectFactory.....	50
3.13 TTypedComObjectFactory.....	53
3.14 TActiveXPropertyPageFactory.....	54
3.15 TAutoObjectFactory.....	54
3.16 TActiveXControlFactory.....	55
3.17 TActiveFormFactory.....	57
第四章 类型库.....	58
4.1 概述.....	59
4.2 “Type Library” 编辑器的基本操作.....	60
4.2.1 创建一个新的类型库.....	60
4.2.2 打开一个已有的类型库.....	61
4.2.3 刷新类型库.....	61
4.2.4 保存和注册类型库.....	62
4.2.5 把类型库引入到当前项目中.....	62
4.2.6 引出类型库.....	63
4.2.7 发布类型库.....	64
4.3 “Type Library” 编辑器的窗口.....	64
4.4 类型库的一般信息.....	65
4.5 接口.....	67
4.6 在接口中加入成员.....	68
4.7 分派接口.....	70
4.8 类型库枚举.....	71
4.9 组件类.....	72
4.10 别名、记录、联合和模块.....	73
4.11 IDL 文法.....	74
4.11.1 属性表述.....	74
4.11.2 接口的文法.....	75
4.11.3 分派接口的文法.....	76
4.11.4 组件类的文法.....	76

4.11.5 枚举的文法.....	77
4.11.6 别名的文法.....	77
4.11.7 记录的文法.....	78
4.11.8 联合的文法.....	78
4.11.9 模块的文法.....	79
第五章 ActiveX 控件.....	80
5.1 创建 ActiveX 控件.....	81
5.1.1 如何转换基于 TGraphicControl 的控件	81
5.1.2 ActiveX 控件向导.....	82
5.2 向导创建了哪些文件	84
5.2.1 项目文件.....	84
5.2.2 类型库的接口源文件.....	85
5.2.3 类型库接口的实现文件.....	90
5.2.4 About 框的源文件	96
5.2.5 许可文件.....	97
5.3 编辑类型库	98
5.3.1 在接口中加入新的成员.....	98
5.3.2 加入新的特性.....	100
5.3.3 加入新的方法.....	100
5.3.4 加入新的事件.....	101
5.4 数据绑定	103
5.5 创建特性页	104
5.6 注册和安装 ActiveX 控件.....	108
5.7 使用 ActiveX 控件.....	110
5.8 ActiveForm	111
5.9 在 Web 上发布 ActiveX	128
5.9.1 Project 页.....	128
5.9.2 Packages 页	129
5.9.3 Additional Files 页	130
5.10 与 Web 浏览器交互.....	130
第六章 OLE Automation.....	142
6.1 引入 Automation 服务器的类型库	143
6.2 Olevariant 类型.....	145
6.3 用 Olevariant 操纵 Automation 对象	149
6.3.1 创建 Automation 对象的实例.....	149
6.3.2 访问 Automation 对象的特性.....	151
6.3.3 访问 Automation 对象的方法.....	151

6.3.4 为什么会访问失败.....	152
6.4 创建 Automation 服务器的实例	153
6.5 Automation 对象的类型库	154
6.5.1 类型库的接口描述文件.....	154
6.5.2 类型库接口的实现文件.....	159
6.5.3 编辑类型库.....	161
6.5.4 加入新的特性.....	162
6.5.5 加入新的方法.....	163
6.5.6 加入新的事件.....	164
6.6 注册和调试 Automation 服务器	165
6.7 一个典型的 Automation 客户	165
6.8 一个典型的 Automation 服务器	168

第七章 MTS.....177

7.1 MTS 组件.....	178
7.2 管理资源	179
7.3 基于角色的安全检查	181
7.4 资源分配器	181
7.5 基客户	183
7.6 MTS 与 COM、DCOM	183
7.7 创建 MTS 对象的一般步骤	184
7.8 向导生成了哪些文件	185
7.8.1 类型库的接口源文件.....	185
7.8.2 接口的实现单元.....	190
7.8.3 MTS 对象的类型库.....	191
7.9 把 MTS 对象安装到 MTS 包中	191
7.10 MTS Explorer	192
7.11 TMtsAutoObject.....	193
7.12 建立事务对象	195

第八章 CORBA.....197

8.1 CORBA 应用程序的体系结构	198
8.2 Stub、Skeleton 和 Smart Agent	199
8.3 激活 CORBA 服务器	199
8.4 创建 CORBA 服务器的一般步骤	200
8.5 定义对象接口	201
8.6 自动生成的代码	203
8.7 在接口库中注册接口	210
8.8 CORBA 客户程序	211

8.8.1 使用 Stub	211
8.8.2 使用 DII	212
8.9 自定义 CORBA 应用程序的行为	213
8.9.1 在客户程序中显示 CORBA 对象的名称	213
8.9.2 表露或隐藏 CORBA 对象	214
8.9.3 传递客户信息给服务器	214
8.10 分发 CORBA 应用程序	215
8.11 配置 Smart Agent	215
8.11.1 启动 Smart Agent	215
8.11.2 配置 ORB 域	216
8.11.3 连接不同局域网上的 Smart Agent	216
第九章 Active Server Page	218
9.1 创建 ASP 对象	219
9.2 ASP 对象的类型库	220
9.2.1 类型库的接口描述文件	220
9.2.2 ASP 对象的接口实现单元	226
9.2.3 ASP 文档	227
9.2.4 编辑 ASP 对象的类型库	227
9.2.5 加入新的特性	228
9.2.6 加入新的方法	229
9.3 在脚本中创建 ASP 对象的实例	230
9.4 注册和调试含有 ASP 对象的 Automation 服务器	231
第十章 WinSock	232
10.1 关于 Socket 的概述	233
10.2 建立服务器端 Socket	233
10.3 建立客户端 Socket	234
10.4 如何在网络上传输数据	235
10.5 在客户端使用多线程技术	236
10.6 在服务器端使用多线程技术	237
10.7 TCustomWinSocket	238
10.8 TClientWinSocket	244
10.9 TServerWinSocket	245
10.10 TServerClientWinSocket	249
10.11 TWinSocketStream	250
10.12 一个网上交谈(Chat)程序	252
第十一章 Internet 协议	256

11.1 TPowersock.....	257
11.2 FTP.....	267
11.3 UDP.....	273
11.4 HTTP.....	277
11.5 SMTP	281
11.6 POP3	285
11.7 NNTP	289
第十二章 Web 服务器应用程序.....	297
12.1 WWW 是如何工作的	298
12.2 Web 服务器扩展	299
12.3 Web 服务器应用程序的逻辑结构.....	299
12.4 静态的 HTML 页面.....	300
12.5 动态的 HTML 页面.....	304
12.6 Web 模块	305
12.7 Web 调度器	307
12.8 Web 动作项	308
12.9 HTTP 请求消息.....	311
12.9.1 HTTP 请求消息是如何传递的	311
12.9.2 TWebRequest 对象.....	312
12.9.3 TISAPIRequest 对象.....	320
12.9.4 TCGIRequest 对象.....	321
12.9.5 TWinCGIRequest 对象	321
12.9.6 一个例子.....	322
12.10 HTTP 响应消息.....	323
12.10.1 建立 HTTP 响应消息.....	324
12.10.2 HTTP 响应消息是如何传递的	324
12.10.3 自己传递 HTTP 响应消息.....	324
12.10.4 TWebResponse 对象	325
12.10.5 TISAPIResponse 对象	331
12.10.6 TCGIResponse 对象	332
12.10.7 TWinCGIResponse 对象.....	332
12.11 Cookie	332
12.12 重定向到另一个 Web 站点	334
12.13 数据流	335
12.14 与客户交互	337
12.15 网页生成器	340
12.15.1 建立 HTML 模板	340
12.15.2 指定 HTML 模板	342

12.15.3 返回转换后的结果.....	343
12.15.4 OnHTMLTag 事件	343
12.15.5 TDataSetPageProducer.....	345
12.16 基于 Web 的数据库应用程序.....	345
12.16.1 用 TSession 管理数据库连接.....	345
12.16.2 建立数据集.....	345
12.16.3 把数据集转换为 HTML 页面	346
12.16.4 把 HTML 表格传给客户	346
12.17 TDataSetTableProducer 元件	347
12.18 TQueryTableProducer 元件	352
12.19 操纵 Web 服务器应用程序	353
12.19.1 TWebApplication 对象.....	354
12.19.2 TCGIApplication 对象.....	356
12.19.3 TISAPIApplication 对象	356
12.20 调试 Web 服务器应用程序.....	356
12.20.1 调试 ISAPI 或 NSAPI 类型的 Web 服务器应用程序	357
12.20.2 调试 CGI 或 Win-CGI 类型的 Web 服务器应用程序.....	358
12.21 两个典型的 Web 服务器应用程序.....	358

第十三章 MIDAS Web 应用程序366

13.1 以 ActiveX 控件或 ActiveForm 作为客户端	367
13.2 创建 MIDAS Server for InternetExpress	368
13.3 创建 MIDAS Web 应用程序	371
13.4 使用 JavaScript 库.....	373
13.5 授权启动和访问 MIDAS Server	373
13.6 使用 XML Broker	374
13.7 MIDAS 网页生成器.....	380
13.8 Web 网页编辑器	383
13.9 在运行期操纵 Web 组件.....	385
13.9.1 TWebComponentList.....	385
13.9.2 TWebForm.....	387
13.9.3 TDataForm	387
13.9.4 TQueryForm	387
13.9.5 TWebControlGroup	388
13.9.6 TLayoutGroup	388
13.9.7 TXMLDisplayGroup	389
13.9.8 TDataGrid.....	389
13.9.9 TDataNavigator	390
13.9.10 TFieldGroup	391

13.9.11 TQueryButtons	391
13.9.12 TQueryFieldGroup	391
13.10 自定义 HTML 模板	391

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。

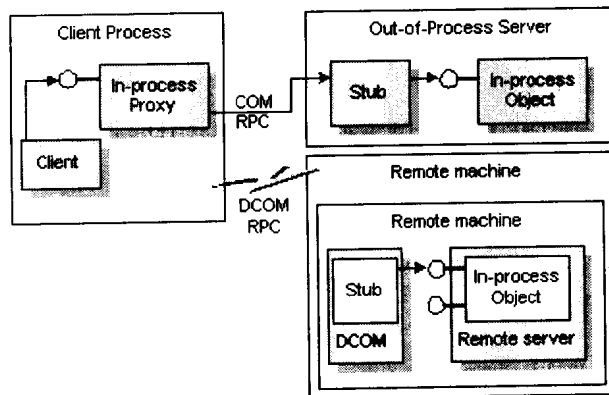


图 1.1 跨进程或机器边界的调用

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。

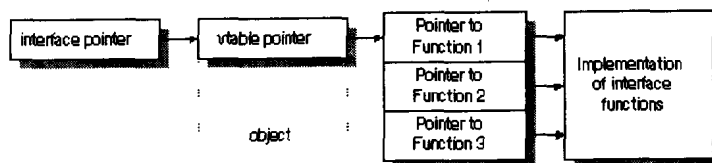


图 1.2 虚拟方法表的示意

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}']
    
```