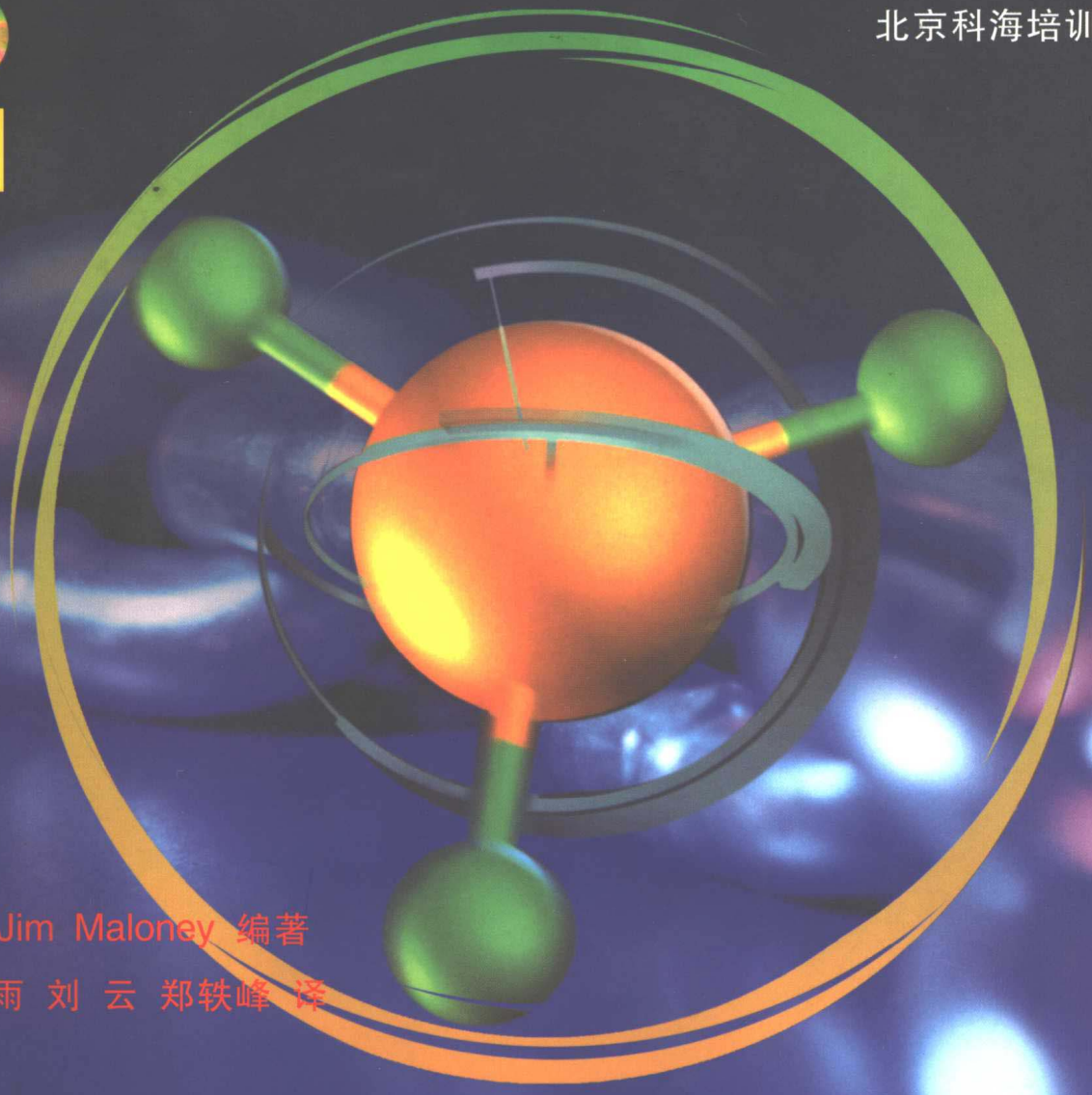
北京科海培训中心

PH PTR



[美] Jim Maloney 编著

田 雨 刘 云 郑轶峰 译

Visual C++ 6 DCOM开发指南

清华大学出版社

Visual C++ 6 DCOM开发指南

Prentice Hall PTR　　北京科海培训中心

Visual C++ 6
DCOM 开发指南

[美] Jim Maloney 编著

田 勇 刘 云 吴轶峰 译

清华大学出版社

(京)新登字 158 号

著作权合同登记号:01-2000-1865

内 容 提 要

本书是一本关于如何使用 Visual C++ 6.0 开发 DCOM 应用程序的教程。书中详细讲述了如何在对象模型中使用 COM 对象,如何使用 OLE DB 和 ADO 访问数据库,如何实现一个对象模型,如何使用微软事务处理服务器(MTS)实现数据服务层以及如何设计 ASP 等内容。

全书内容丰富实用,讲解深入浅出,并有大量例子代码,适用于对 Visual C++, COM,ATL 有一定了解,想开发分布式应用程序的读者。

书　名:Visual C++ 6 DCOM 开发指南

作　者:Jim Maloney

译　者:田雨　刘云　郑铁峰

出版者:清华大学出版社(北京清华大学校内,邮编 100084)

http://www.tup.tsinghua.edu.cn

印刷者:北京门头沟胶印厂

发　行:新华书店总店北京科技发行所

开　本:787×1092 1/16　印张:19　字数:462 千字

版　次:2000 年 9 月第 1 版　2000 年 9 月第 1 次印刷

印　数:0001～5000

盘　号:ISBN 7-900630-20-1

定　价:42.00 元

前　言

我经常说C++是程序员级的编程语言，也就是说，这门语言对那些大部分时间都在编程的专业人员而言是非常理想的。C++语言内容丰富、简洁，真正面向对象，使用它可以创建高效率的、第一流的程序。然而，专业人员所选择的C++语言的特点却使该语言看起来模糊，令人难以理解。精通C++语言可能要花费数年心血，此外还要求熟悉Windows和COM开发。使这一事实更加明显的是，当前软件开发所选择的模型是多层体系结构的开发模型，你可以看到有一个需要详细研究的学习曲线。

虽然我的第一本书是关于VB编程的，但我更倾向于用C++编程。当我使用C++编程时，我感到控制更加灵活，对我的程序的高效性和更充分的利用机器的资源感到更有信心。但是设计、开发和维护一个用C++开发的多层应用程序所需要的技巧并不常见，这也是我的大多数客户坚持要求我用VB编写代码的原因所在。在微软平台上，用VB的开发似乎远远多于C++的开发，至少对企业开发而不是商业开发而言是这样的。那么，用VB开发一个大型企业应用程序是不是一个错误？开发人员是不是应该学习C++编程？如果使用C++代替VB编写企业应用程序，是否会显著提高应用程序的性能？根据我20年来作为一个独立软件顾问的经验，我知道所有以上问题的答案就是：因情况而异。

就某方面进行比较，C++程序的运行速度比VB程序快——有时甚至快得多。然而，使用C++来开发一个大型的、可扩展的多层应用程序，如果要达到或超过用VB开发的类似程序，那将是一个富有挑战性的工作。多层应用程序必定花费许多时间访问数据库，产生网络I/O，从而导致许多数据库和系统调用API。随着COM、分布式组件对象模型(DCOM)以及Microsoft Transaction Server(MTS，微软事务处理服务器)的出现，所有Microsoft操作系统的编程语言都可以共享一套用于数据库访问和分布式应用程序之间通信的对象和接口。因为VB和Visual C++可以共享这些对象和接口，所以只有从激活这些接口的方法中寻找提高性能的机会。换句话说，要获得好的性能，最主要的就是一开始进行正确的设计，选择合适的多层应用开发途径。一个设计得很糟糕的C++应用程序的性能不会比一个设计恰当的VB应用程序的性能好，无论它是不是分布式的。我是通过把多层应用程序由Visual C++转换成VB才得到这一事实的。仅仅选择C++而不是VB是不能自动地、无条件地确保应用程序的高性能的。然而，所有的事情是同样重要的——好的设计和好的途径——VB是没有办法在性能上超过C++的。

关于本书

这是一本适合于进行分布式应用程序开发的书籍。虽然严格地说，这是一本中间层次的书，但我希望读者能成为一个有经验的C++和Windows开发人员。最好读者具有一些ATL(Active Template Library，活动模板库)的经验，或者至少读过一本介绍ATL的书。也许你看见过或试过一些简单的类似于hello，world的ATL的演示，知道一些关于ATL和

COM 的基本概念，现在面临的是应用这些概念来开发一个分布式应用程序。如果这些描述正是你所面对的情况的话，你会发现这本书非常有用，这主要是因为只要明白一些简单的 ATL 概念就可以使用 Visual C++开发高效率的多层应用程序。这也是本书的主旋律——通过最可能快的方式使你用 Visual C++进行分布式应用程序的开发。

可能这本书的最不平常之处就在于我是从第 1 章开发多层分布式应用程序开始的。我在课堂上使用该方法非常奏效，我希望在这本书中也能达到同样的效果。因为以后章节所使用的许多示例都依赖于安装的例子程序，所以我建议您首先阅读第 1 章，并且执行一下安装过程。

在第 2 章，我们讨论 Visual C++应用程序中 COM 对象的用途，使用了本书提供的 Video Store 对象作为例子，即使你对在 C++应用程序中使用 COM 对象非常熟悉，你也应该阅读这一章。这一章对应用程序而言，也许看起来没有什么很大的用处，但是它提供了一些对象设计和文档的很好的例子。

第 3 章比较深层次地概述了 OLE 数据库(OLE DB)，即数据库访问的面向 COM 的接口。本章将会澄清你对 OLE DB 和 ActiveX Data Objects(ADO)之间差异的模糊认识，并将提供一些很难找到的关于使用 OLE DB 接口访问数据库的完整应用程序的例子。虽然大多数多层应用程序应使用 ADO 而不是低级 OLE DB 接口来完成数据库的访问，但你还是应该仔细阅读这一章，因为它提供了 OLE DB 的坚实基础。

作为分布式应用程序中间层开发的最基本的介绍，第 4 章是应读的。这一章介绍了 ADO，并提供了几个关于数据库访问的例子，包括许多从本书附带 CD-ROM 上提供的完整多层样例应用程序中提取的例子。

在第 5 章中，我解释了如何设计和实现一个对象模型，一个客户端组件，它是一个围绕在 MTS 下开发服务器端无状态对象中的面向对象的封装。对象模型中的对象，也被称为客户端状态保持层，是商业对象的简洁抽象。一个对象模型包括客户端逻辑，但不包括用户界面——使用对象模型使得开发和发布客户端程序变得十分容易，尤其是 Win32 下基于对话框的应用程序。

第 6 章介绍了有关 DCOM 的最基本的知识。阅读本章会使你得到 DCOM 的坚实基础，也会欣赏到 MTS 所提供的一些特征。一个认真的分布式应用程序的开发者将需要阅读并理解这一章。

根据我的经验，我可以肯定你们大多数人买这本书的目的是在于第 7 章和第 8 章中所要涉及的问题——使用 MTS。许多年来，我经常乐于观察学生们回答一个问题时的反应，这个问题就是："你们中有多少人对学习 MTS 有兴趣？"即使那些在公共场合不愿举手的人也会经常作出一副精神饱满的样子。这就毫无疑问地表明你也会需要阅读这两章。

全书最后着重讲述在第 9 章"使用 Active Server Pages 创建 Internet 界面"。这一章有些超出了开始时所涉及的内容，我认为本章是要读的，因为现今的 Web 应用程序大多是用 ASP 页来实现的。

光盘说明

简介

随书附上的 CD-ROM 包含了本书中所有例子程序的完整源代码和一个 README.TXT 文件，该文件包含了 CD-ROM 有关内容的最新信息。所有例子都以 Microsoft Visual Studio 6.0 或 Microsoft Visual Basic 6.0 项目文件的形式出现。

系统描述

安装和运行例子程序需要 Microsoft Visual Studio 6.0 专业版(Service Pack 3 以上)，Microsoft Windows NT 4.0 Server 或者 Workstation (Service Pack 3 以上)，Windows NT 4.0 Option Pack，包括 MTS 和 IIS，以及 Microsoft SQL Server 6.5 或 7.0。许多 C++服务器演示程序都有一个相应的用 Visual Basic 6.0 编写的用户测试界面，如果要运行这些程序的话，还需要 Visual Basic 6.0。

如何使用该 CD-ROM

你如果愿意的话，可以把 CD-ROM 中的 VCENT 文件夹拷贝到硬盘上。这样，源文件、可执行文件和相应的动态链接库都被拷贝到了硬盘上。你也可以单独拷贝你感兴趣的例子程序。尽管并不必要，但建议你在硬盘上保留与 CD-ROM 上同样的相对目录结构。许多例子都可以直接运行。但是有些例子，特别是 n 层版本的 Mom-n-Pop Video Store 例子需要进行安装。你可以在文本文件中找到相应的安装说明。

操作系统

大多数演示程序都可以在 Microsoft Windows 95/98 下运行。但是，n 层结构的例子需要 Windows NT 4.0 Server 或者 Workstation，MTS，IIS 以及 SQL Server。

其他信息

作者会在他的主页 http://www.jmalco.com 上发布错误纠正和提问问题。任何错误都将发布在 http://www.jmalco.com/vcerrata.htm 上。你可以在 http://www.jmalco.com/vcfaq.htm 上找到常见问题列表。报告错误或者提问问题请发 email 给 vcbook@jmalco.com。另外，一定要阅读本书中与例子有关的部分以及 CD-ROM 上的 README.TXT 文件。

技术支持

Prentice Hall 不为 CD-ROM 上的软件提供技术支持。但是，如果 CD 本身有问题，发 e-mail 给 disc_exchange@prenhall.com，说明你的问题，就可以获得一份新的拷贝。

目　录

第 1 章　开发多层客户/服务器应用程序

我喜欢的 Seinfeld 片断是一个倒述的片断。如果你是一个 Seinfeld 迷，就会知道我说的是哪一个片断。它不是真正的倒述——不是把录像带倒放，使人们说话和走路完全反过来。而是该片断首先上演了一小段不同的事件，然后上演了这一事件的前一事件，诸如此类。该片断的最有趣的事情是它如何结尾的，让你知道一点信息来吸引你的注意力，你就会急切地想知道事情是如何开始的。如果该剧是按照传统的从开始到结束的平铺直述的手法，我会怀疑它是否能抓住你的注意力。

本书首先概述了什么是多层应用程序以及如何开发它。自然，在应用开发工作中，编写多层客户/服务器应用程序是最后一个步骤。为什么要这样做呢？原因就在于开发多层应用程序的方法对大多数开发人员来说是陌生的，因此我要使用一个完整的 3 层应用程序来倒述开发的过程。这将有助于你看到使用这些方法开发程序所带来的好处和所面临的挑战。像我喜欢的 Seinfeld 片断一样，我希望能够通过结果来引起你的兴趣，这样你就会急于去读它的开始部分。

在开始之前我要提醒一下读者：要想获得成功，你必须认真地执行书中所提供的指令。因为这些步骤将修改你的注册表，也会创建新的 Microsoft SQL Server 数据库，强烈建议你在一台不重要的具有开发环境的机器上来执行这些操作，在执行这些步骤前，你应该备份你的系统。

1.1　应用程序体系结构

目前业界认为应用程序的开发有 3 种不同的体系结构。然而，业界并没有就每种体系结构的具体组成达成一致的意见。在 Microsoft 环境下，通常认为有 3 种类型的体系结构：桌面、2 层和 3 层体系结构。

1.1.1　桌面应用程序体系结构概述

桌面体系结构(desktop architecture)适用于传统的应用程序，它们是完全自包含的，因此也是单一的。在桌面体系结构中，所有层都驻留在客户机上，包括数据库本身，如图 1.1 所示。当数据库是远程数据库时，应用程序可能就不再是桌面体系结构而是 2 层体系结构了。

许多采用桌面体系结构的应用程序在传统的各层应用程序之间没有明显的界线。这些层包括用户界面、商业逻辑以及数据库访问逻辑。由于各层之间没有显著的区别，所以把桌面应用程序转换成任何类型的分布式应用程序体系结构都很困难。

1.1.2　2 层应用程序体系结构概述

一个访问远程数据库的应用程序是 2 层体系结构的。数据库可以被任何客户/服务器数据库 API 访问，比如 ActiveX Data Objects(ADO)，Remote Data Objects(RDO)和 Open

Database Connectivity (ODBC)API 等等(参见图 1.2)。

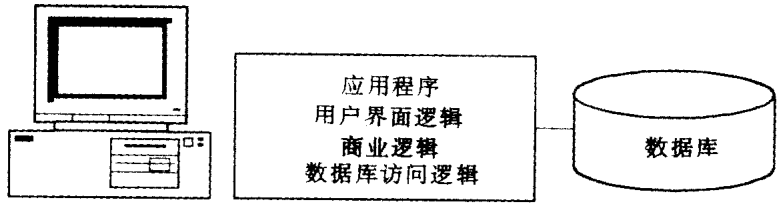


图 1.1 桌面应用程序体系结构是单一的。一个应用程序包括用户界面、商业逻辑以及数据库访问逻辑。数据库是本地的

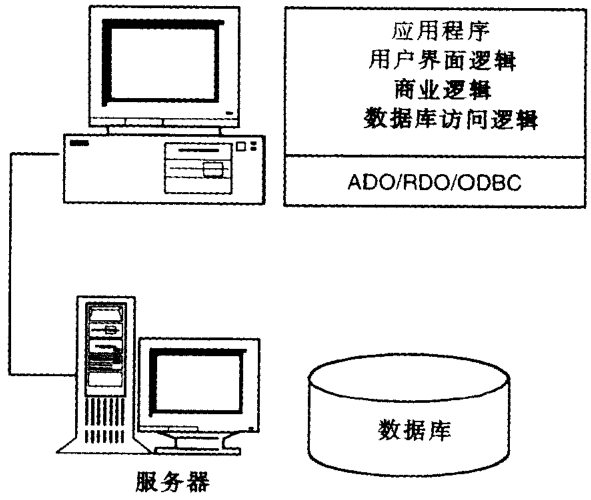


图 1.2 在 2 层体系结构中,数据库是远程的,但是商业逻辑和数据库访问逻辑仍然驻留在客户端,用户界面代码也是如此

把桌面应用程序转换成 2 层体系结构是容易的。事实上,使用上面所说的数据库访问 API(ADO,RDO,ODBC),转换工作就简单变成修改数据源的属性,使得应用程序访问远程数据库而不是本地数据库(这个事实容易造成对分布式应用分类的混乱,也就是当一个 2 层体系结构的应用程序的数据库在本地时,它可不可以被认为是桌面体系结构应用呢?)。换句话说,n 层(n-tier)这个术语是应用于逻辑模型还是物理模型呢?专家们不同意这个观点。然而随着分布式结构的日趋普通,越来越多的开发人员开始认为 n 层是应用于逻辑模型的,这也是本书的定义。例如,3 层应用程序的所有组件,包括数据库,都可以布置在一台机器上。实现这样一个开发计划并不会使你的应用程序变成一个桌面体系结构,因为它可以在分布式环境中展开。

1.1.3 3 层应用程序体系结构概述

在用户界面和数据库之间增加一个附加层就使得应用程序成为 3 层体系结构(见图 1.3)。这个中间层通常被认为是关于商业规则或商业逻辑的,它一般和数据库一起,或驻留在服务器上。

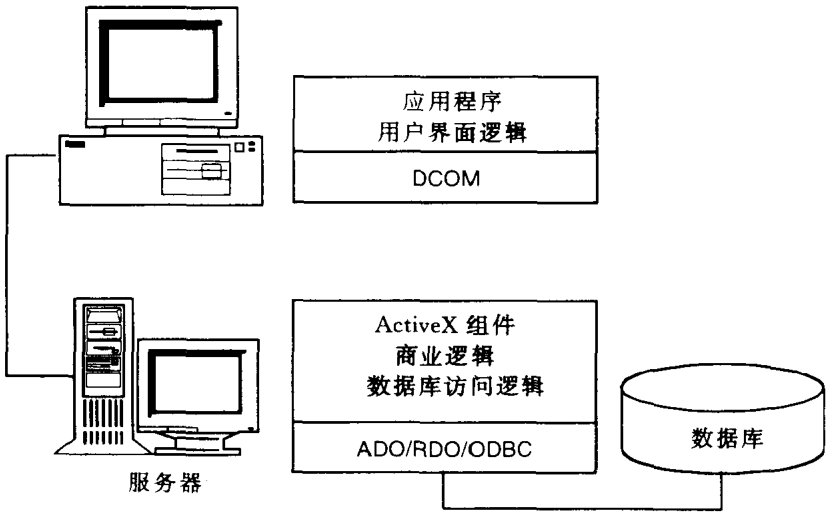


图 1.3　3 层体系结构中，商业逻辑和数据库访问逻辑与用户界面分离，驻留在服务器上

Visual C++ MFC/AppWizard 开发环境使得基于对话框的应用程序很容易开发，在这种应用程序中，用户与控件交互时相应的事件处理函数会被自动调用。因此，将这些已有的应用程序改造成 3 层体系结构会比较困难。

许多专家所提议的 3 层体系结构建议把商业逻辑和数据库访问逻辑放在一个独立的组件中，并把它放置在服务器上，用户界面从远程访问它。尽管任何类型的远程通信协议都可用于用户界面和服务器之间的通信，包括 TCP/IP 套接字，命名管道(named pipes)，网络 DDE 等等，但是最好的途径是使用分布式组件对象模型(Distributed Component Object Model，DCOM)。更进一步说，最好通过 Microsoft Transaction Server(MTS)使用 DCOM，MTS 提供了一个 DCOM 实用工具的高级环境，可以分布和监视服务器端的对象，还提供了一个可以管理对象的运行时环境，组合数据库连接，安全管理和协调分布式事务。

很明显，在 3 层体系结构中，在用户界面逻辑和商业逻辑之间引入了一个附加的潜在的瓶颈——网络。实际上，是一个瓶颈代替了另外一个瓶颈。3 层模型获得了处理数据的良好性能。这就是说，通过把商业逻辑移到数据库服务器上，客户机和服务器只要交换很少的数据。但是，用户界面和商业逻辑间的通信变成远程了。根据用户界面逻辑和商业逻辑之间所需的通信量不同，从本地处理数据所获得的性能也许会被网络流量的增加所抵消。

解决这个问题的办法就是把中间层的对象设计成无状态的。与传统的对象不同，无状态的对象没有固定的属性。典型地，这些对象只有方法(函数)，在一个单一函数调用中必须声明足够的参数才能完成一个会话。在这种情况下，状态由客户机维持，一直到用户界面收集到足够的信息，此时激活中间层某个对象的方法。虽然使用这种技术可以得到容易升级的程序，但是很多程序员并不喜欢使用很多参数来激活一个方法，尤其是那些好的面向对象的设计技术。这种进退两难的困境以及解决方法将在以后章节中作更深一步的讨论。

1.1.4　n 层应用程序体系结构概述

如果你在表示层和数据库之间插入一层，那么是不是将得到一个 4 层的客户/服务器体系结构？这个问题问得非常好，它演示了术语是怎样产生的。通常业界已经采用术语 n 层

(n-tier)来描述一种新的体系结构，在这种体系结构中，应用程序被清楚地划分成若干层，每层之间通过定义好的接口进行访问。因此，在分布式应用程序的技术描述中你不太可能遇到这个术语。

我在这里提出 4 层结构的观点是因为围绕 3 层体系结构有许多争议，主要是关于 3 层体系结构中层对象的无状态属性。从本章开始，本书始终将讨论无状态对象的概念和重要性。在这里我要特别提及的是无状态对象就是没有属性的对象。无状态对象只有方法(函数)，这些函数具有那些训练有素的面向对象编程人员感到不舒服的一大堆参数。设计无状态对象主要是用于 Internet 或者是基于 intranet 的客户应用程序，在这种应用程序中，表示层或者用户界面是在 Web 浏览器上运行的。

在本书中，我提供了一种分布式应用程序开发的方法，它考虑了 Internet/intranet 客户及 Win32 客户(参见图 1.4)。在使用这种方法时，我建议 Internet/intranet 客户表示层使用商业规则层(也称作数据服务层)的无状态对象。在 Win32 客户表示层中使用一个对象模型，它包括状态对象，其作用是在商业规则层中的无状态对象上建立一层面向对象代码。因为我在表示层和数据库层增加了一层，有人也许会认为本书提出了一种 4 层体系结构。然而，我更愿意把这种对象模型看作是 Win32 客户应用程序表示层的一种扩展。

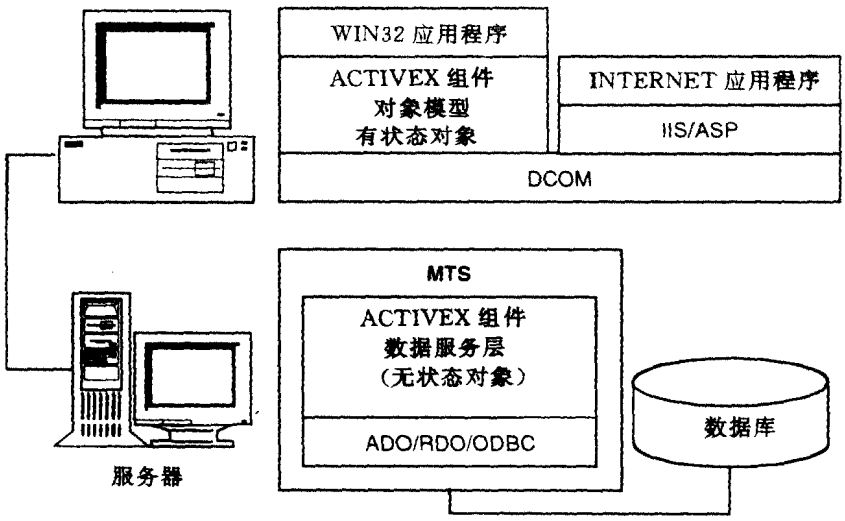


图 1.4 带有客户端对象模型的 3 层体系结构是本书所使用的模型

1.1.5 关于 2 层客户/服务器应用程序

在 2 层客户/服务器应用程序中，数据库访问从客户端进行，或直接从表示层(用户界面)直接访问，或通过对象模型访问数据库(参见图 1.5)。对象模型一般是驻留在客户端，这是因为它的对象是有状态的。一般来说，存在于表示层和数据库之间的代码被看作中间件(middleware)，虽然许多客户/服务器开发人员只把这一术语用于 3 层客户/服务器应用程序，其实这是不正确的。中间件是存在于表示层和数据库之间的任何一层定制的软件。