



Microsoft 编程系列

Microsoft® Press



内附光盘

基于服务器的 应用程序技术内幕

Inside **Server-Based** **Applications**

[美] Douglas J. Reilly 著
潇湘工作室 译

在 Microsoft Windows 2000
上采用 WinSock、ASP、
ISAPI、SQL Server 7 开发服
务器端应用程序的原始资料



清华大学出版社
<http://www.tup.tsinghua.edu.cn>

Microsoft 编程系列

基于服务器的应用程序 技术内幕

[美] Douglas J. Reilly 著

潇湘工作室 译

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书分三个部分。第 I 部分说明了基于服务器开发的基础知识;第 II 部分专门介绍了在 Windows 2000 平台上进行各种服务器端应用程序的编写,列举了大量短应用程序实例;第 III 部分从传统服务应用程序和 Web 服务应用程序的管理两个方面,将在前面部分所学的各种短程序结合起来,使之成为完整的程序。本书配套光盘列举了大量各章短程序和完整程序的源代码,并配带了使用这些程序的相应软件和软件环境,其中的内容在相应的章节都作了说明。

本书适合服务器管理员、计算机编程人员阅读。

Inside Server-Based Applications

Copyright (1999) by Microsoft Corporation.

Original English language edition Copyright © (2000) by Microsoft Corporation.

All rights published by arrangement with the original publisher, Microsoft Press, a division of Microsoft Corporation, Redmond, Washington, U.S.A.

北京市版权局著作权合同登记号: 图字 01-2000-2089 号

版权所有, 翻印必究。

本书封面贴有清华大学出版社激光防伪标签, 无标签者不得销售。

书 名: 基于服务器的应用程序技术内幕

作 者: [美] Douglas J. Reilly

译 者: 潇湘工作室

责任编辑: 陈 萍

出 版 者: 清华大学出版社(北京清华大学学研大厦, 邮编 100084)

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

印 刷 者: 北京清华园胶印厂

发 行 者: 新华书店总店北京发行所

开 本: 787×960 1/16 印张: 41 字数: 869 千字

版 次: 2000 年 11 月第 1 版 2000 年 11 月第 1 次印刷

书 号: ISBN 7-900625-69-0

印 数: 0001~5000

定 价: 89.00 元(含 1 光盘)

JS432/27

前 言

本书的目的是，帮助您从一个基于服务器的应用程序的初学者，变成一个基于服务器的应用程序的编程专家。我从某些基于服务器的应用程序的背景知识开始讲起，然后讲述 API 和技术细节，通过这些，您便可以为 Microsoft Windows 2000 创建有用的基于服务器的应用程序。

本书读者对象

本书适用于这样的开发人员——他们希望了解 Win32 编程的基础要素，以便可以开发服务器应用程序。为充分了利用本书，您应该熟悉 C++，特别是 Microsoft Visual C++ 编程环境。是否熟悉 Win32 API 并不重要，但熟悉它不会妨碍您理解本书的内容。

在本书的 ASP(Active Server Pages) 和 ISAPI(Internet Server API)实例中，假设您了解 IIS(Internet Information Server)。在 ASP 实例中，还假设您熟悉 VBScript 或者 Microsoft Visual Basic。然而，对它们熟悉的程度并不重要，因为例子中所用的 VBScript 仅仅是对对象的访问方法，这是任何 C++程序员都能很容易完成的工作。其中的一个例子使用在服务器上创建的 JScript，并传递给客户机。您也需要了解 JScript，以理解这个例子的细节，但是，理解创建客户机端脚本编程并将其发送给客户机的深层含义，并不需要有关 JScript 的知识。

一些例子作为简单的数据源和服务器端开发环境而使用 Microsoft SQL Server。使用 SQL Server 作为服务器端开发环境，要求您对 SQL 有一些了解。

关于实例和配套光盘

对于前面的几个实例，需要注意的是，它们是控制台模式的应用程序，控制台模式应用程序和 16 位的 MS-DOS 文本模式应用程序类似，但是，它们却有一些很不相同的地方。首先，这些实例是真正的 32 位应用程序。其次，这些应用程序可以调用动态链接库 (DLL) 和 Win32 API 的其他部分，仅仅有少数例外。最后，本书所讨论的 Windows 2000 服务应用程序是用于特殊目的的核心控制台模式应用程序。

我用 Microsoft Visual Studio 6.0 和 July 1999 MSDN Library 创建了配套光盘上的所有实例。所有的实例都已编译，而且使用 Release Candidate 1(Build 2072)在 Windows 2000

上已经运行过。我用 Microsoft Visual InterDev 创建了 ASP 实例,但是,为运行 ASP 和 ISAPI 应用程序,必须要有 IIS (至少为版本 4,但版本 5 更加合适,它是和 Windows 2000 一起发行的);一些实例使用 SQL Server,这些实例只有和 Microsoft Database Engine (可以从 <http://msdn.microsoft.com/vstudio/msde/download.asp> 得到这个软件)一起,才能工作得很好;一些实例使用 TCP/IP,因此要求网络安装 TCP/IP,并用 TCP/IP 的某种命名服务(如 WINS 或 DNS)正确配置。(在配套光盘上的 Readme 文件中,也包含运行这些实例所需要的软件列表。)

概述

第 1 章和第 2 章对基于服务器的应用程序的要求进行了总体介绍。然后,在随后的各章中讨论了一些 API,它们可在 Windows 2000 下的基于服务器的应用程序完成它们必须完成的工作。第 3 章讨论了服务 API (Service API),利用它可以创建真正的基于服务器的应用程序——它甚至可以在没有任何人登录到服务器的情况下运行。在这里,我引入了一个 C++ 类,以支持基于服务器的应用程序。第 4 章介绍了假冒的 API,包括基本问题“您是谁”的安全含义。第 5 章讨论了事件记录,并将其集成到第 3 章中引入的 C++ 类中。

第 6 章使用 RAS (Remote Access Server, 远程访问服务) API 创建使用标准电话线路的虚拟网络。第 7 章讨论了 TAPI (Telephone API, 电话 API) 和 MAPI (Message API, 消息 API),并在将这些技术集成到 Windows 2000 服务应用程序中的时候,特别强调了潜在的缺陷。在第 7 章中,扩展了前面引入的 C++ 服务类,以支持集成 COM 技术,例如 MAPI,或者需要带有消息泵 (message pump) 的应用程序的功能。

第 8 章讨论了 ODBC (Open Database Connectivity, 开放式数据库连通性),并创建了一个类,以利用符合 ODBC 的数据源。虽然有很多种出色的新技术可以用于数据库访问,但是,ODBC 的优点是速度快,而且应用广泛。在第 8 章中,还介绍了几种支持 ODBC 的类。在本书的其余部分,将使用这些类。

在讨论了创建基于 Windows 2000 服务器的应用程序的基础知识,以及支持它们的 Win32 API 之后,我开始讨论几种 Web 技术,利用它们,可以为 Internet 时代创建基于服务器的应用程序。通过在第 9 章中使用 ASP,在第 10 章和第 11 章使用 ISAPI,说明了如何创建最先进的应用程序,它可以充分利用传统服务器应用程序的优点,并将它们移植到 Internet 技术中。第 12 章说明了使用命名管道和 WinSock 的客户机通信,它是利用 Win32 事件以及 I/O 完整端口而实现的。第 13 章再次研究了数据库访问,但这一次是针对 SQL Server,同时还讨论了如何完全利用 SQL Server 的服务器端应用程序支持。最后,在第 14 章和第 15 章中,我展示了一个网络监视系统的实例,它可以作为一个独立的服务应用程序使用,或者使用 ASP 和 ISAPI 与 Web 客户机协同工作。

写作方式

现在，用一点时间来说明我的写作方式：除了极少的例外，我并没有使用诸如 MFC 或者 ATL 的框架，而是使用 Win32 API。对于传统的客户机应用程序，因为这些工具最适合于支持客户机应用程序，所以，框架通常是很合适的。一般说来，这些框架对于基于服务器的应用程序并没有太多的用处。当然，也有一些例外的情况。我们可以使用 ATL 创建与 ASP 一起使用的服务器端 Microsoft ActiveX 控件。对于这个任务，没有任何理由不使用 ATL，因为 ATL 对象向导中的代码就是为服务器端使用而设计的。MFC 在 ISAPI 实例中得到了广泛使用，这里仍然没有任何理由不使用 MFC ISAPI 扩展。无需借助于其余的 MFC 类，MFC ISAPI 类就可运行。一旦了解了基础的 API 或者技术，在使用框架包装包围 API 和技术的适当时候，便可以更加有效地使用框架。我希望您能赞同我的观点。

我很愿意编写这本书，因为它的内容是我几乎每天都用于创建基于服务器的应用程序的技术。我希望您将在这里找到创建可靠而稳健的应用程序所需要的基础知识，在 21 世纪不断增长的基于服务器的计算环境中，人们将需要这些应用程序。

本书作者 Douglas J. Reilly 简介

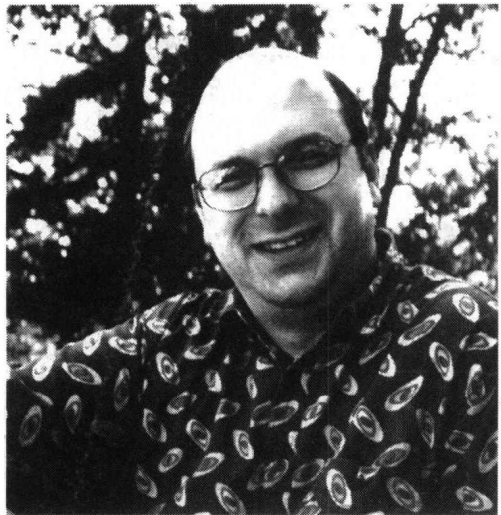
自从他说服他的太太花了很大一笔钱买了一台 Atari 800 之后, Douglas J. Reilly 就已经喜欢上了玩计算机。在买 Atari 计算机之前的许多年, 他靠修理复印机和早期的个人计算机为生。之后不久, 对他来说, 软件比硬件更有意思。

Doug 现在拥有 Access Microsystems 公司, 这是一家小咨询公司, 使用 Microsoft Visual C++, Borland Delphi、Microsoft Access 以及本书中所涉及的许多 API 和技术, 开发软件。他已经创建了许多应用程序, 包括电子测试求职人员, 跟踪零售商的存货清单, 以及修复损坏的数据库。目前, 他正在为 St. Barnabas Health Care System 公司开发各种卫生保健应用程序; 不久前, 他曾为 Home Care Information Systems 公司开发各种卫生保健应用程序。这两家公司都在新泽西州。除了开发软件外, Doug 已经为《Dr. Dobb's Journal》和《Software Development》撰写文章, 他还是《Pervasive Software Developer's Journal》的专栏作家。

Doug 与他的太太 Jean 以及两个孩子 Tim 和 Erin 一起生活。在不编程的时候, 他欣赏音乐, 读书, 骑自行车, 尽管今年的写作任务和其他问题已经妨碍了他参加重大的自行车比赛。可能明年他还参加自行车比赛。

通过下列电子邮件地址, 可以和 Doug 联系:

doug@accessmicrosystems.com



目 录

第 I 部分 服务器基础

第 1 章 基于服务器的开发	3
1.1 什么是服务器	3
1.2 基于服务器的开发需求增长	5
1.2.1 NetWare	5
1.2.2 胖客户机模型的缺点	6
1.2.3 Internet: TCO 问题的解决方案	7
1.2.4 内部网: 带宽的天堂	8
1.2.5 Windows 2000 硬件支持	8
1.3 基于服务器的应用程序开发的不同之处	9
1.3.1 服务器是无人照管的	9
1.3.2 资源泄漏是隐藏的错误	10
1.3.3 多任务的必要性	11
1.4 基于服务器的应用程序	13
1.4.1 传统的基于服务器的系统	13
1.4.2 基于 Web 的服务器	14
1.5 基于服务器编程的 WIN32 支持	15
1.5.1 控制台 API	17
1.5.2 服务 API	17
1.5.3 假冒 API	18
1.5.4 事件记录 API	18
1.5.5 RAS、MAPI 和 TAPI	19
1.5.6 数据库连通性	19
1.5.7 ASP	19
1.5.8 ISAPI	20

1.5.9 通信 API	21
1.6 SQL Server 简介	22
1.7 结论	23
第 2 章 基于服务器开发的基础	24
2.1 多任务处理技术基础	24
2.1.1 协作式多任务处理	24
2.1.2 抢占式多任务技术	26
2.2 多线程	26
2.3 同步对象	28
2.3.1 临界区	30
2.3.2 事件	36
2.3.3 互斥体	37
2.3.4 信号量	38
2.3.5 可等待的定时器	40
2.3.6 Wait 函数和线程同步	41
2.4 创建新线程	42
2.4.1 线程创建函数	43
2.4.2 多线程实例程序	44

第 II 部分 Windows 2000 作为服务器平台

第 3 章 Service API	49
3.1 什么是 Service API	49
3.2 谁运行服务	50
3.3 Service API 实例	51
3.3.1 安装服务	60
3.3.2 使用 CreateService 安装服务	61
3.3.3 使用 DeleteService 卸载服务	64
3.3.4 启动服务	65
3.3.5 ServiceMain 内部	65
3.4 Windows 2000 服务的 C++ 框架	68
3.4.1 为 Windows 2000 服务编写 C++ 类	69
3.4.2 实现 C++ 服务类	72

3.4.3 增强功能.....	95
3.5 结论.....	97
第 4 章 假冒 API	98
4.1 Windows 2000 安全概述.....	99
4.2 假冒的基础.....	100
4.3 假冒的类.....	101
4.3.1 基于连接的假冒.....	101
4.3.2 基于客户机信息的假冒.....	102
4.3.3 保护服务器特定的资源.....	106
4.3.4 联合使用假冒和自定义安全.....	108
4.4 结论.....	114
第 5 章 事件记录	115
5.1 事件记录 API.....	118
5.1.1 在事件日志中写入事件.....	118
5.1.2 在 CPPService 中添加事件记录.....	125
5.1.3 从事件日志中读取事件.....	153
5.2 结论.....	161
第 6 章 Remote Access Service API	162
6.1 RAS API 概述.....	163
6.2 使用 RasDial 的实例.....	163
6.2.1 控制多个线程.....	171
6.2.2 RasDialFunc2 回调函数.....	173
6.3 RAS 的其他用途.....	174
6.4 结论.....	175
第 7 章 TAPI 和 MAPI	177
7.1 TAPI.....	177
7.1.1 TAPI 版本.....	177
7.1.2 TAPI 设备.....	178
7.1.3 TAPI 错误.....	178
7.1.4 得到 TAPI 的反馈.....	178
7.1.5 TAPI 实例.....	179
7.1.6 TAPI 的其他作用.....	198

7.2 MAPI.....	198
7.2.1 MAPI 的版本.....	199
7.2.2 选择正确的客户机接口.....	199
7.2.3 简单 MAPI.....	200
7.2.4 完整 MAPI.....	219
7.2.5 在服务中综合使用 MAPI 和 TAPI.....	242
7.3 结论.....	297
第 8 章 ODBC API	298
8.1 ODBC 的功能	298
8.2 ODBC 的替代品	299
8.3 ODBC 组件	301
8.3.1 应用程序.....	302
8.3.2 Driver Manager	302
8.3.3 驱动程序.....	303
8.3.4 数据源.....	303
8.4 ODBC 类	303
8.4.1 实例: 使用 ODBC 类.....	304
8.4.2 ODBCDatabase 类.....	309
8.4.3 CODBCxCursor 类.....	328
8.5 ODBC 陷阱和缺陷	329
第 9 章 Active Server Pages.....	330
9.1 什么是 ASP	331
9.1.1 ASP 与 HTML 页面的相似性	331
9.1.2 ASP 与 HTML 页面的不同	332
9.2 ASP 的优势	337
9.3 ASP 的弱点	338
9.4 ASP 应用程序中的可用对象	339
9.4.1 Application 对象.....	339
9.4.2 Session 对象.....	342
9.4.3 Server 对象	343
9.4.4 Request 对象.....	345
9.4.5 Response 对象.....	349
9.4.6 ObjectContext 对象	351

9.5 ASP 实例	352
9.5.1 简单的电话目录	352
9.5.1 更加复杂的 ASP 实例: Troubleshooter	359
9.6 结论	386
第 10 章 Internet Server API	387
10.1 DLL 回顾	388
10.2 ISAPI 概述	390
10.2.1 ISAPI 比 CGI 的优越之处	391
10.2.2 ISAPI 比 ASP 的优越之处	392
10.3 简单的 ISAPI 扩展	392
10.4 ISAPI 扩展的其他用途	424
10.5 结论	432
第 11 章 ISAPI 过滤器	433
11.1 ISAPI 过滤器概述	433
11.2 ISAPI 过滤器实例	439
11.3 MFC ISAPI 过滤器	447
11.4 结论	449
第 12 章 客户机/服务器通信 API	450
12.1 客户机与服务器的通信方式	450
12.1.1 TCP/IP(WinSock)	450
12.1.2 命名管道	452
12.1.3 如何选择 WinSock 和命名管道	452
12.2 分派通信请求	453
12.2.1 管理客户机的状态	453
12.2.2 管理故障	454
12.2.3 分派请求	455
12.2.4 WinSock 和命名管道的比较	495
12.2.5 事件与 I/O 完成端口与 QueueUserWorkItem 之间的比较	495
12.3 通用的客户机通信类	496
12.4 结论	503
第 13 章 集成 SQL Server	504
13.1 Transact-SQL	505

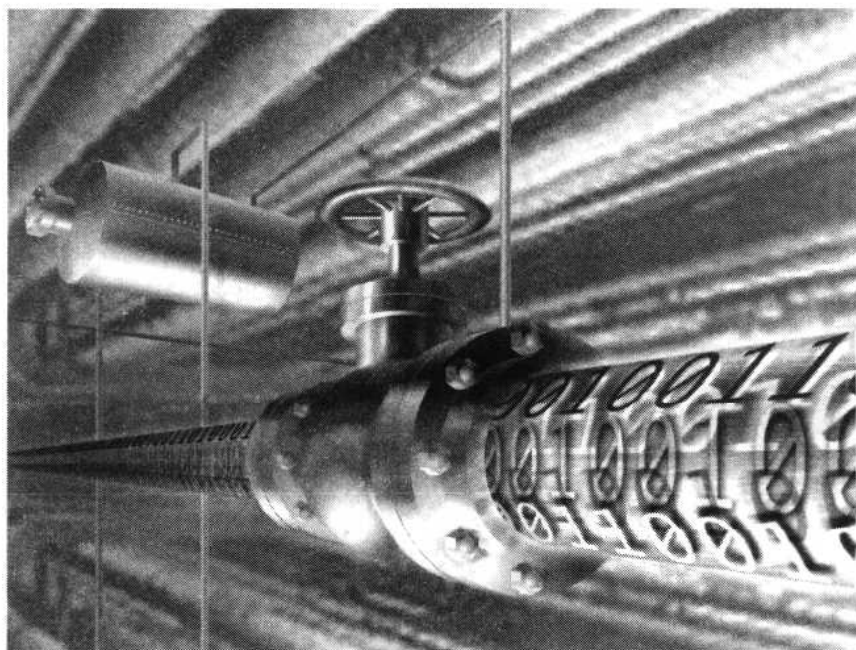
13.1.1 触发器.....	505
13.1.2 存储过程.....	509
13.1.3 扩展存储过程.....	511
13.2 结论.....	528

第 III 部分 建立 Windows 2000 服务器应用程序

第 14 章 传统的服务应用程序.....	533
14.1 网络监视器.....	533
14.2 什么是网络监视器.....	533
14.1.2 网络监视器的结构.....	537
14.1.3 CMonitorService 类.....	538
14.2 监视任务函数.....	570
14.3 结论.....	590
第 15 章 管理 Web 上的服务应用程序.....	591
15.1 使用 ASP 监视 MONITORSERVICE.....	591
15.1.1 同警报进行交互.....	595
15.1.2 同任务进行交互.....	604
15.2 使用 ISAPI 监视 MONITORSERVICE.....	620
15.2.1 CMonitorService 修改.....	620
15.2.2 ISAPIMonitor ISAPI 扩展.....	626
15.3 结论.....	632
附录 A 在 IIS 中设立虚拟目录.....	634
附录 B 服务失败的反应.....	640
参考文献.....	644

第 I 部分

服务器基础



第 1 章 基于服务器的开发

在电影“成功秘诀(The Secret of My Success)”中，有一个叫 Michael J.Fox 的人物，他刚刚从堪萨斯州到达纽约，从电话亭中打电话回家，向他的母亲报平安。他告诉母亲，这里和堪萨斯州一样。突然响起了枪声，这让他蹲下了几秒钟，然后他继续说，“是的，和堪萨斯州类似，但是更加强烈”。基于服务器的开发让我想起了上面的这段话——它和常规编程类似，但是却更加强大。

1.1 什么是服务器

在进一步讨论问题之前，首先必须明确一些定义。准确地定义什么是服务器就好像定义什么是美一样——当您看见它的时候，您就认识了它。大多数程序员在看见服务器程序的时候，知道什么是服务器程序，但是，要提出一个定义并不容易。大多数服务器系统存在于局域网(LAN)或者广域网(WAN)上，而且和相同网络上的客户机进行通信，但事情并不总是这样的。甚至所使用的操作系统也不再是服务器程序的标志：Microsoft Windows 2000 是第一个客户机操作系统和服务器操作系统一致的系统(暂时忽略 Windows 2000 Professional 版和 Windows 2000 Server 版之间的差别——稍后，我们将更加详细地介绍这些)。

网络附件也并不是服务器程序的要求。大多数服务器程序是为了与网络环境或者客户工作站交互而建立的，但这种交互作用不是必需的。刚才提到过，Windows 2000 以客户机/服务器的模式建立，在同一台计算机上包含了客户机和服务器。让我们看一看图 1.1，其中显示了 Windows 2000 Services 小程序。顺着服务列表向下看，我们将看见一些明显和网络相关的服务(DHCP Client 和 FTP Publishing Service)以及一些明显和网络无关的内容(Event Log)。所谓 Event Log，就是本地计算机用于记录通知消息、警告和错误的服务。的确，Event Log 所创建的日志也可以从网络上的另一台计算机上阅读，但是，此功能并不是该服务的基本部分，而且在独立的计算机上，此服务也一样可以工作得很好。(如何用这种方式构造 Microsoft Windows NT，需要另外一本完全不同的书来讲述——参见“参考文献”中的《Show-Stopper and Inside Windows NT，第 2 版》——但是，重要的是要认识到，客户机和服务器没有必要位于不同的计算机上。)

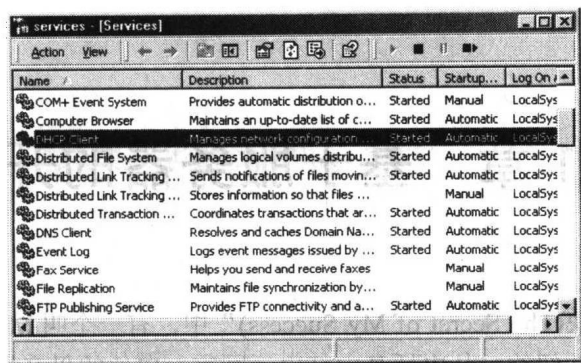


图1.1 Windows 2000 Services小程序

因为我们并不依赖这些标准来定义服务器程序——使用什么操作系统，是否在客户机和服务器之间进行通信，或者程序是否在网络上运行——所以，我们需要另外一个标准。我的标准就是：服务器程序就是这样的一种程序，其中并不存在用户界面(UI)，或者用户界面与程序的功能是独立的，或者是程序功能的附属物。

让我们考虑两个服务器程序，看看它们如何满足这个标准的。第一个实例就是联网的 MS-DOS 程序，它监视 AS/400 中档计算机上的虚拟驱动器，复制并处理 LAN 驱动器上的文件，然后等待出现下一个文件。唯一真实的用户界面就是基于文本的显示，它告诉操作者系统正在进行的工作。这个用户界面是服务器程序的主要任务的附属品——因为没有用户选项——而且消息的存在仅仅是为了让用户知道系统是否正在运行。

服务器程序的第二个实例就是 Microsoft SQL Server。如果在图 1.1 中向下滚动列表，将在服务列表中看见两个和 SQL Server 相关的条目：MSSQLServer，这是一个真正的服务器程序，它管理数据库操作；SQLExecutive，这是一个处理 SQL Server 数据库的定时操作的程序。图 1.2 显示了当开始运行 SQL Enterprise Manager 的时候所打开的程序屏幕。

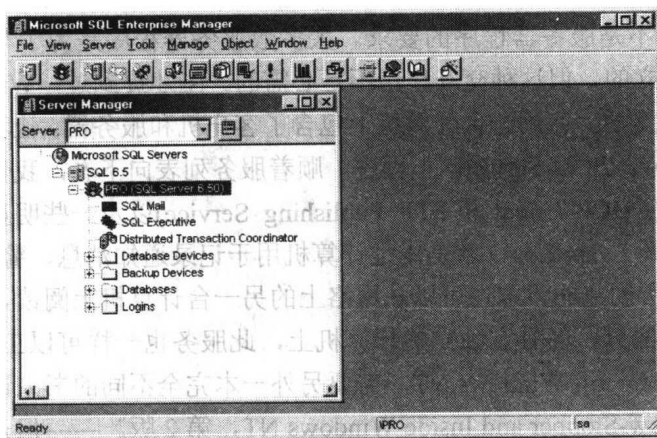


图1.2 SQL Enterprise Manager, SQL Server的用户界面