



网络程序设计系列丛书

Visual

C# .NET

网络核心编程

周存杰 编著

Y



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





网络程序设计系列丛书

丛书特点

大量的实例和代码分析
以实用为出发点介绍网络编程，并配有大量的实例和代码分析。这些代码非常具有实用价值。

体贴读者、详细讲解网络开发难点
涉及的技术都是在网络开发过程中常用的技术，
对于网络开发中的主要难点，进行了深入的讲解说明。

详细讲解关键技术
在叙述方面，对于仅需了解的知识力求简明，而
对于关键技术则深入透彻。

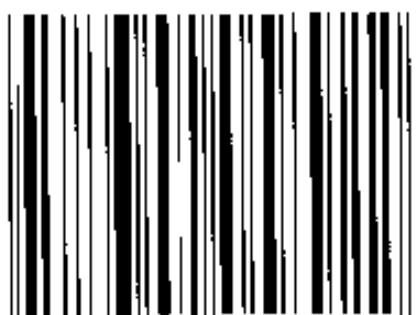
作者的体会与经验之谈
有助于读者尽快掌握网络编程。

本丛书包括

Borland C++ Builder 网络编程
Visual C++.NET 网络编程
Visual Basic.NET 网络编程
Visual C#.NET 网络核心编程
Unix 网络编程实用技术与实例分析

责任编辑 龙啟铭 封面设计 林光为

ISBN 7-302-05892-X



9 787302 058922 >

定价：35.00元

网络程序设计系列丛书

Visual C#.NET 网络核心编程

周存杰 编著

清华 大学 出 版 社

(京)新登字 158 号

内 容 简 介

本书是关于 C# 网络开发的教材，主要包括三个方面内容，首先简要介绍了有关 C# 网络开发的基础知识；接着讲解基础服务器开发、基础客户端开发、FTP 开发、SMTP 开发、POP3 开发和远程控制开发；最后是高级应用，包括 Win32 网络组件开发、Web 数据库基础、Win32 异步套接字数据库开发、XML Web Services 开发以及一个完整的分布式网络应用程序开发实例。

本书适合于 C# 开发人员进行网络开发，对 Visual C++ .NET 和 Visual Basic .NET 开发人员也很有参考意义。

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。

图书在版编目 (CIP) 数据

Visual C#.NET 网络核心编程/周存杰编著.-北京：清华大学出版社，2002.11
(网络程序设计系列丛书)

ISBN 7-302-05892-X

I. V... II. 周... III. C 语言-程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2002) 第 070810 号

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>

责任编辑：龙啟铭

印 刷 者：北京市密云胶印厂

发 行 者：新华书店总店北京发行所

开 本：787×1092 1/16 印张：24 字数：596 千字

版 次：2002 年 11 月第 1 版 2002 年 11 月第 1 次印刷

书 号：ISBN 7-302-05892-X/TP · 3498

印 数：0001~5000

定 价：35.00 元

前　　言

C#是 Microsoft 为.NET 平台量身定做的语言。随着.NET 的推广，C#越来越成为一门重要的语言。许多企业已经开始尝试使用 C#开发应用软件。在.NET 平台上，C#开发者可以方便地扩展自己的应用。C#可以将大多数组件转变为 Web 服务，并且可以被任何在 Internet 上运行的应用程序调用，C#对这一特性提供内置的支持。更重要的是，Web 服务框架可以让任何 Web 服务看起来都类似于 C#的内置对象，从而使 Internet 上的站点变成一个可以互相交换组件的地方。这一特性使程序员们欣喜不已，这不但可以避免重复劳动、提高生产效率，而且可以使整个 Internet 变成一个大的操作系统。这无疑是下一代的开发理念。

本书共三部分：第一部分只有短短的一章，简明扼要地介绍有关网络开发的一些 C#基础知识；第二部分共六章，讲解基础服务器开发、基础客户端开发、FTP 开发、SMTP 开发、POP3 开发和远程控制开发；第三部分为高级应用，共五章，内容包括：Win32 网络组件开发、Web 数据库基础、Win32 异步套接字数据库开发、XML Web Services 开发和一个完整的分布式网络应用程序开发实例。

一门好的语言，仅仅开发出来还不够，更重要的是应用它、推广它、挖掘它。本书的所有内容都是编者两年来辛苦挖掘的结果，许多内容是编者的经验总结，希望本书能为读者今后从事 C#开发提供一臂之力。

本书由周存杰负责统稿。参加编写的有：周存杰、张芯娜、王丹丹、郭东、张显明、李飞、赵明信、梁越鸿、陈琳琳、王晓莉、林志东等。在书稿交出版社前，宋元时参加本书错字的校对工作。

在本书的编写过程中，宋元时先生给予了重要的帮助，在此表示感谢。清华大学出版社的龙啟铭先生在宏观上做了重要指导，使本书章节编排更加合理，提高了本书的质量，在此表示感谢。

由于时间仓促，本书肯定还有许多不足之处，望广大读者批评指正。联系信箱：zhou868@263.net。

编　　者

目 录

第1章 C# 语法基础	1
1.1 C#的特点	1
1.2 .NET 命名空间	3
1.3 数据流	11
1.3.1 网络流	12
1.3.2 文本流	14
1.3.3 文件流	15
1.4 命令解析	16
1.4.1 普通格式命令的解析	16
1.4.2 特殊格式命令解析	17
1.5 方法参数	18
1.5.1 params 关键字	18
1.5.2 ref 关键字	19
1.5.3 out 关键字	20
1.6 常用数据类型及其传输	21
1.7 线程	22
本章小结	25
第2章 基础服务器开发	26
2.1 同步套接字服务器开发	26
2.1.1 定义主机对象	26
2.1.2 主机解析	27
2.1.3 端口绑定与监听	28
2.1.4 发送数据	29
2.1.5 接收数据	30
2.1.6 基础服务器开发实例	31
2.1.7 重要改进	33
2.2 异步套接字服务器开发	34
2.2.1 端口绑定与监听	34
2.2.2 发送数据	37
2.2.3 接收数据	38
2.2.4 异步套接字基础服务器开发实例	39
2.3 TcpListener 基础服务器开发	43
2.3.1 端口监听	43

2.3.2 发送数据与接收数据.....	44
2.3.3 基础服务器开发实例.....	44
2.3.4 重要改进	46
本章小结	47
第3章 基础客户端开发	48
3.1 同步套接字客户端开发.....	48
3.1.1 建立与服务器的连接.....	48
3.1.2 数据发送与接收	48
3.1.3 基础客户端开发实例.....	48
3.1.4 演示	48
3.2 异步套接字客户端开发.....	53
3.2.1 建立与服务器的连接.....	53
3.2.2 数据发送与接收	54
3.2.3 异步套接字操作基础客户端开发实例	54
3.2.4 演示	59
3.3 TcpClient 基础客户端开发.....	60
3.3.1 建立连接	60
3.3.2 发送数据与接收数据.....	62
3.3.3 基础客户端开发实例.....	62
3.3.4 演示	65
本章小结	66
第4章 FTP 协议开发	67
4.1 FTP 协议规范	67
4.1.1 FTP 命令格式	67
4.1.2 FTP 命令参数	68
4.1.3 FTP 命令	69
4.1.4 FTP 应答	71
4.1.5 FTP 实例	72
4.1.6 文件传输的特别要求	74
4.2 FTP 服务器开发	75
4.2.1 命令的接收与解读	76
4.2.2 响应码的发送	76
4.2.3 发送目录	77
4.2.4 发送文件	77
4.2.5 接听命令并响应	78
4.2.6 FTP 服务器开发	79
4.3 FTP 客户端开发	87
4.3.1 发送命令	87

4.3.2 接收服务器应答	87
4.3.3 检查服务器应答码	88
4.3.4 文件传输方法	88
4.3.5 下载功能	89
4.3.6 FTP 客户端开发	92
4.3.7 演示	102
本章小结	103
第 5 章 SMTP 协议开发	104
5.1 SMTP 协议简介	104
5.1.1 SMTP 命令格式	104
5.1.2 SMTP 命令参数格式	104
5.1.3 SMTP 命令	106
5.1.4 SMTP 应答码	108
5.1.5 SMTP 示例	109
5.1.6 ESMTP	109
5.2 邮件发送程序开发	112
5.2.1 身份认证	112
5.2.2 发送命令	114
5.2.3 应答码的接受	114
5.2.4 发送邮件	114
5.2.5 应答码检查	115
5.2.6 邮件发送程序开发	115
5.2.7 演示	125
5.3 SMTP 服务器开发	126
5.3.1 读取命令	126
5.3.2 发送反馈	127
5.3.3 读取邮件内容	128
5.3.4 获取邮箱字符串中的服务器名称	128
5.3.5 获取邮箱字符串中的邮箱名称	129
5.3.6 SMTP 服务器开发	129
5.3.7 演示	139
5.3.8 改进意见	141
本章小结	141
第 6 章 POP3 协议开发	142
6.1 POP3 协议简介	142
6.1.1 POP3 协议命令格式	142
6.1.2 POP3 命令参数	142
6.1.3 POP3 协议命令	143

6.1.4 POP3 简单示例	144
6.2 邮件接收程序	145
6.2.1 接收服务器应答	145
6.2.2 发送命令码	145
6.2.3 接收邮件	146
6.2.4 检查应答码	146
6.2.5 获取邮件总数	146
6.2.6 邮件接收程序开发	147
6.2.7 演示	153
6.3 POP3 服务器开发	154
6.3.1 POP3 服务器开发	154
6.3.2 演示	165
6.3.3 改进建议	166
本章小结	166
第 7 章 远程控制开发	167
7.1 服务端开发	167
7.1.1 获取客户发送的信息	168
7.1.2 获取用户命令	168
7.1.3 获取命令参数	168
7.1.4 发送反馈信息	169
7.1.5 服务器开发	169
7.2 控制端开发	179
7.3 演示	186
本章小结	186
第 8 章 网络组件开发	187
8.1 网络组件的开发基础	187
8.1.1 第一个组件的开发	188
8.1.2 带参数的组件开发	190
8.1.3 如何定义全局变量	192
8.1.4 TcpListener 基础服务器组件开发	194
8.1.5 使用基础服务器的组件	196
8.2 FTP 服务器组件开发	200
8.2.1 FTP 服务器组件开发	200
8.2.2 使用 FTP 服务器组件	210
8.2.3 演示	214
8.3 网络控件的开发	215
8.3.1 编辑控件开发与使用	215
8.3.2 TcpClient 客户端控件开发与使用	219

8.4 关于属性	226
8.4.1 在组件中使用属性	227
8.4.2 在控件中使用属性	232
本章小结	238
第 9 章 ADO.NET Web 应用开发	239
9.1 数据库建立	239
9.1.1 用 VS.NET 创建数据库	239
9.1.2 用代码创建数据库	241
9.2 数据库连接	242
9.2.1 与 SQL Server 数据库连接	242
9.2.2 与非 SQL Server 数据库连接	247
9.3 数据浏览	251
9.3.1 自定义页面表格	252
9.3.2 用 DataGrid 控件浏览 SQL Server 数据库数据	258
9.3.3 用 DataGrid 控件浏览非 SQL Server 数据库数据	258
9.4 数据查询、插入、删除和更新	259
9.4.1 数据查询	259
9.4.2 数据插入	266
9.4.3 数据删除	267
9.4.4 数据更新	268
本章小结	269
第 10 章 数据库的异步套接字网络应用	270
10.1 异步套接字的数据库服务器开发	270
10.1.1 命令识别	270
10.1.2 检查命令是否发送完毕	271
10.1.3 接收并执行命令	271
10.1.4 服务器开发	279
10.2 客户端开发	292
10.2.1 检查数据是否接收完毕	292
10.2.2 发送命令	292
10.2.3 接收数据	294
10.2.4 客户端开发	295
10.3 演示	302
第 11 章 XML Web services 开发	304
11.1 Web 服务开发基础	304
11.1.1 关于特性	304
11.1.2 第一个 Web 服务开发	306

11.1.3 Web 服务的使用	311
11.1.4 将 Web 服务修改成组件	312
11.2 Web 服务高级开发	316
11.2.1 数据库服务开发	316
11.2.2 如何将 Win32 组件转换为 Web 服务	320
11.2.3 将 Web 应用程序转换为 Web 服务	325
11.3 XML Web 服务使用实例	329
本章小结	342
 第 12 章 分布式商贸财务系统开发实例	 343
12.1 解决方案简介	343
12.1.1 程序的主要功能	343
12.1.2 基础数据库	344
12.2 XML ASP.NET Services 开发	345
12.2.1 特定时间段内全部商品流水账服务	345
12.2.2 特定时间段内特定商品流水账服务	345
12.2.3 特定时间段内所有商品的经营盈亏服务	346
12.2.4 特定时间段内特定商品的经营盈亏服务	346
12.2.5 进货数据编辑服务	347
12.2.6 售货数据编辑服务	348
12.3.1 进货部门客户端开发	355
12.3 客户端开发	355
12.3.2 售货部门客户端开发	360
12.3.3 财务部门客户端开发	364
12.3.4 管理（经理）部门客户端开发	369
12.3.5 演示	369
12.3.6 改进意见	372
本章小结	373

第1章 C# 语法基础

本书是专门针对 C# 网络高级开发而编写的教材，内容涉及常用网络协议和网络数据库的应用。由于本书篇幅的限制，在后面的章节里，往往是跳过语法知识而直接进入高级开发过程，这无疑增加了阅读本书的难度，会使许多不太熟悉网络的读者感到学习困难。为了帮助读者迅速掌握本书内容，本章首先介绍一下与网络有关的 C# 基础知识，特别是与网络有关的 C# 语法。如果读者已经对网络知识有了相当的了解，并且熟悉掌握了 C# 的语法知识，则可跳过本章，直接阅读第 2 章。

1.1 C# 的特点

C# 是 Microsoft 为.NET 平台量身定做的语言，它具有面向对象、面向 Web、语法简洁、功能强大、效率高、安全性强等特点。

1. 面向对象

C# 是面向对象的语言。在 C# 中，任何对象都会自动成为 COM 对象，开发者不再需要显式实现 IUnknown 和其他一些 COM 接口，同时可以方便而自然地使用现存的 COM 对象，而无需关心这些 COM 对象是否使用 C# 开发。对于使用 C# 的开发人员来讲，C# 允许开发人员调用 OS 所提供的 API，在经过标记的代码区域内使用指针并手工管理内存分配，这样，以前的 C/C++ 代码依然可以重用，C/C++ 开发人员可以更快地熟悉和转向 C#，而无需放弃在以前开发中形成的开发习惯。由于 C# 支持 COM，支持 API 调用，为开发人员提供了强大的开发控制功能。为了更好地实现公司的各种商业计划，在软件系统中，在商业流程和软件实现之间必须有紧密的联系。但大多数的开发语言都不能轻易地将各种应用逻辑与代码相联系。例如，开发人员会使用各种注释来标明各种类所代表的抽象商业对象。C# 允许对任何对象使用预定义数据或是经过扩展的元数据，在系统结构中可以使用区域属性，并且将这些属性添加到类、接口或者其他元素上。开发者可以独立地测试各种元素上的属性。这使得收集区域中的对象属性或编写自动工具以保证区域中的类、接口是否被正确定义等工作变得十分简单。

2. 面向 Web

Microsoft 推出.NET 的主要目的是构建下一代 Internet，使 Internet 成为一个可以互相交换组件的地方。新的开发模式意味着需要更好地利用现有的各种 Web 标准，例如 XML、SOAP 等。C# 之前的开发工具都是在 Internet 出现之前或是未得到充分应用之前出现的，所以都不能很好地适应目前 Web 技术的开发需要，而 C# 开发者可以方便地在 Microsoft 网络平台上扩展自己的应用。C# 可以将大多数组件转变为 Web 服务，并且可以被在 Internet 上

运行的各种平台的任何应用调用，C#对这一特性提供内置的支持。更重要的是，Web 服务框架使任何 Web 服务看起来都类似于 C#的内置对象，这样，开发人员在开发过程中可以继续使用他们已经具备的面向对象的开发方法和技巧。此外 C#还具有许多其他特性，使之成为最出色的 Internet 开发工具。例如，XML 目前已经成为网络中数据结构传送的标准，为了提高效率，C#允许直接将 XML 数据映射成为结构，以便更有效地处理各种数据。XML、SOAP 等是全球共有的协议和标准，不属于哪一家公司所有，对 XML、SOAP 的支持，意味着所有支持 XML 和 SOAP 的平台都可以访问 C#开发的 Web 资源。

3. 语法简洁

C#抛弃了许多旧的技术，如 ATL、MFC、C 运行库、标准模板库（STL）等类库。.NET 框架统一了编程类库，开发起来比使用这些旧技术容易得多。同时 C#将那些把 ATL 和 COM 搞得乱糟糟的伪关键字（如 OLE_COLOR、VARIANT_BOOL、DISPID_XXXXX 等）都抛弃了，代之以简明、清晰、易于理解的关键字。例如，在 C#中有一个 is 关键字，该关键字可以判断对象的类型，使用十分方便。下面的小程序演示了如何用 is 关键字判断对象的类型：

```
private void button1_Click(object sender, System.EventArgs e)
{
    int i=100;
    object obj=i;
    if(obj is int)
    {
        textBox1.Text="对象是整型数值。";
    }
    else(textBox1.Text="对象不是整型数值。");
}
```

执行该程序，textBox1 的文本框会出现“对象是整型数值。”一行字，如图 1-1 所示。

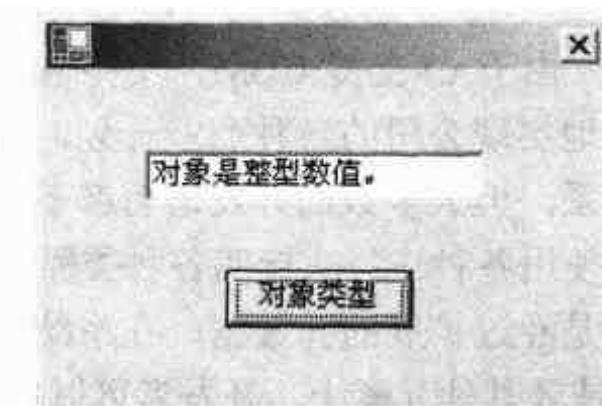


图 1-1 is 关键字

在 C#中对象的转换十分容易，任何对象都可以使用 ToString 方法转换成字符串类型；反之，使用特定对象的 Parse 方法可以把字符串类型转换成其他类型值。比如要把字符串“12345”转换成 Int32 的 12345，只要使用“Int32.Parse("12345")”即可。C#简洁的语法使得初学者非常容易入门，即使以前从来没有学过计算机语言的人也可以轻松入门。

4. 功能强大

C#可以开发任何传统风格的 Windows 程序。不仅如此，不管是控制程序、图形程序、NT 服务程序，还是普通组件，甚至是 Web 页面、Web 服务、Web 组件，除了硬件驱动程

序，都可以用 C#开发出来。而且，C#的类可以从 VB、VC++中声明的类中派生出来。只要是使用运行库的语言，都可以做到在一种语言中声明类，而在另一种语言中派生类。Visual Studio 的调试器会完全支持跨语言的程序调试，在函数堆栈调试窗口的每个条目中都会显示堆栈中的函数是什么以及它们分别用何种语言写成；此外，开发人员甚至可以跨语言地处理程序中的异常错误。

5. 高效率

在 C#推出之前的 20 年里，C 和 C++已经成为商用软件开发的主流计算机语言，但是 C 和 C++都提供一些容易使开发者产生错误的特性，可以说，C 和 C++的灵活性是以牺牲开发效率为代价。如果和其他的开发语言相比（比如说 VB），相同功能的 C/C++软件通常会需要更长的开发周期。正是由于 C/C++开发的复杂性和需要较长的开发周期，所以许多 C/C++开发人员都在寻找一种既灵活、又高效的开发语言。目前有些开发语言通过牺牲 C/C++语言的灵活性来换取开发效率，有些语言对开发人员限制过多，提供的通用命名能力很差。这些语言不能够轻易地与现存的系统相结合，并且不能与当前的 Web 开发相结合。合理的 C/C++替代语言应能对现存和潜在的平台上的高效开发提供有效和有力的支持，使 Web 开发能非常方便地与现存的应用开发结合起来，并使 C/C++开发人员在必要时能使用底层代码。为了解决这个问题，Microsoft 推出了一种命名为 C#（发音为 C Sharp）的计算机语言。C#是一种先进的、面向对象的语言，它提供大量的开发工具和服务，以帮助开发人员开发基于计算和通信的各种应用，使开发人员可以快速地建立大范围的基于 MS 网络平台的应用。C#是一种面向对象的开发语言，可以大范围地适用于高层商业应用和底层系统的开发。C#可以方便地将各种组件转变为基于 Web 的应用，并且能够通过 Internet 被各种系统或其他开发语言所开发的应用调用。从开发效率讲，C#为开发人员提供了快速的开发手段而不牺牲任何 C/C++语言的特点或优点。从继承性讲，C#在更高层次上重新实现了 C/C++，使熟悉 C/C++的开发人员可以很快转变为 C#开发人员。

6. 安全性强

即使是优秀的 C/C++开发人员都难免在编码过程犯一些常见错误，比如错误地初始化一个变量，而这类错误有可能导致各种不可以预知的错误，并且难于发现。一旦这类错误在发现之前投入生产环境，为排除这些错误将会付出昂贵的代价。而 C#的先进设计思想可以消除 C/C++开发中的许多常见错误，比如：垃圾收集机制将减轻开发人员对内存的管理负担；C#中的变量将自动根据环境被初始化；变量是类型安全的；C#在语言中内置版本控制功能以减少更新组件时产生的错误等。

综上所述，C#是一门面向对象的、功能强大、高效、安全、语法简洁的崭新的计算机语言，它必将随着.NET 的推广而被广泛应用到各种软件开发中。

1.2 .NET 命名空间

要熟悉 C#语言，必须了解.NET 命名空间，这样才能在今后的网络高级开发中得心应手地使用各种空间下的资源。下面介绍.NET 命名空间。

- Microsoft.CSharp

包含支持用 C# 语言进行编译和代码生成的类。

- Microsoft.JScript

包含支持用 JScript 语言进行编译和代码生成的 JScript 运行库和类。

- Microsoft.VisualBasic

包含 Visual Basic .NET 运行库。此运行库与 Visual Basic .NET 语言一起使用。此命名空间还包含支持用 Visual Basic .NET 语言进行编译和代码生成的类。

- Microsoft.Vsa

包含将 .NET 框架脚本引擎的脚本集成到应用程序中以及在运行时编译和执行代码的接口。

- Microsoft.Win32

提供两种类型的类：处理由操作系统引发的事件的类和对系统注册表进行操作的类。

- System

最重要的类，包含用于定义常用值和引用数据类型、事件和事件处理程序、接口、属性和处理异常的基础类和基类。

- System.CodeDom

包含可用于表示源代码文档的元素和结构的类。

- System.CodeDom.Compiler

包含源代码模型的结构，管理源代码所生成和编译的类。

- System.Collections

包含定义各种对象集合（如列表、队列、位数组、散列表和词典）的接口和类。

- System.Collections.Specialized

包含专用的强类型集合；例如，链接表词典、位向量以及只包含字符串的集合。

- System.ComponentModel

提供用于实现组件和控件的运行时和设计时行为的类。此命名空间包括用于属性和类型转换器的实现、数据源绑定和组件授权的基类和接口。

- System.ComponentModel.Design

使开发人员可以生成自定义用户界面控件，并将这些控件包括在设计时环境中以便与供应商控件一起使用。

- System.ComponentModel.Design.Serialization

提供设计器所进行的组件序列化支持。此命名空间中的类可用于提供自定义序列化程序、管理特定类型的序列化、管理设计器加载和设计器序列化，以及优化设计器重新加载。

- System.Configuration

提供以编程方式访问 .NET 框架配置设置和处理配置文件 (.config 文件) 中的错误的类和接口。

- System.Configuration.Assemblies

包含用于配置程序集的类。

- **System.Configuration.Install**

提供为组件编写自定义安装程序的类。Installer 类是 .NET 框架中所有自定义安装程序的基类。

- **System.Data**

基本上由构成 ADO.NET 结构的类组成。使用 ADO.NET 结构可以生成用于有效管理多个数据源中的数据的组件。在断开连接的方案（如 Internet）中，ADO.NET 提供可以在多层系统中请求、更新和协调数据的工具。ADO.NET 结构也可以在客户端应用程序（如 Windows 窗体）或 ASP.NET 创建的 HTML 页中实现。

- **System.Data.Common**

包含由 .NET 数据提供程序共享的类。.NET 数据提供程序描述用于在托管空间中访问数据源（如数据库）的类的集合。

- **System.Data.OleDb**

封装 OLE DB .NET 数据提供程序。.NET 数据提供程序描述用于在托管空间中访问数据源（如数据库）的类的集合。

- **System.Data.SqlClient**

封装 SQL Server .NET 数据提供程序。.NET 数据提供程序描述用于在托管空间中访问数据源（如数据库）的类的集合。

- **System.Data.SqlTypes**

提供用于 SQL Server 中本机数据类型的类。这些类提供其他数据类型更安全、更快速的替代物。使用此命名空间中的类有助于防止在可能发生精度损失的情况下出现的类型转换错误。

- **System.Diagnostics**

提供允许与系统进程、事件日志和性能计数器进行交互的类。此命名空间还提供可以调试应用程序和跟踪代码执行的类。

- **System.Diagnostics.SymbolStore**

提供允许读取和写入调试符号信息的类。面向 .NET 框架的编译器可以将调试符号信息存储到程序员的数据库 (PDB) 文件中。调试器和代码分析器工具可以在运行时读取调试符号信息。

- **System.DirectoryServices**

提供从托管代码轻松访问 Active Directory 的方法。

- **System.Drawing**

提供对 GDI+ 基本形功能的访问。System.Drawing.Drawing2D, System.Drawing.Imaging 和 System.Drawing.Text 命名空间提供了更高级的功能。

- **System.Drawing.Design**

包含扩展设计时用户界面 (UI) 逻辑和绘制的类。可以进一步扩展此设计时功能，以创建自定义工具箱项、类型特定的值编辑器（可编辑和以图形方式表示所支持的类型的值）或类型转换器（可在特定类型之间转换值）。

- **System.Drawing.Drawing2D**

提供高级的二维和向量图形功能。此命名空间包括渐变画笔、Matrix 类（用于定义几何转换）和 GraphicsPath 类。

- **System.Drawing.Imaging**

提供高级的 GDI+ 图像处理功能。

- **System.Drawing.Printing**

提供与打印相关的服务。

- **System.Drawing.Text**

提供高级的 GDI+ 版式功能。此命名空间中的类使用户可以创建和使用字体集合。

- **System.EnterpriseServices**

为企业级应用程序提供重要的基础结构。COM+ 为企业级环境中部署的组件编程模型提供服务结构。此命名空间为 .NET 框架对象提供对 COM+ 服务的访问，使 .NET 框架对象更适用于企业级应用程序。

- **System.EnterpriseServices.CompensatingResourceManager**

提供在托管代码中使用补偿资源管理器 (CRM) 的类。CRM 是由 COM+ 提供的一项服务，使用户可以在 Microsoft 分布式事务处理协调器 (DTC) 事务中包括非事务性对象。虽然 CRM 不提供完整资源管理器的功能，但它们却通过恢复日志提供事务性原子性（全有或全无行为）和持久性。

- **System.Globalization**

包含定义区域性相关信息的类，这些信息包括语言、国家/地区、正在使用的日历、日期的格式模式、货币、数字以及字符串的排序顺序。

- **System.IO**

包含允许对数据流和文件进行同步和异步读写的类型。

- **System.IO.IsolatedStorage**

包含允许创建和使用独立存储区的类型。通过使用这些存储区，可以读写信任度较低的代码无法访问的数据，防止公开可保存在文件系统其他位置的敏感信息。数据存储在独立于当前用户和代码所在的程序集的数据舱中。

- **System.Management**

提供对一组丰富的管理信息和管理事件（有关符合 Windows 管理规范 (WMI) 基础结构的系统、设备和应用程序的）的访问。

- **System.Management.Instrumentation**

提供在规范应用程序管理并通过 WMI 向潜在用户公开管理信息和事件时必需的类。这样，Microsoft Application Center 或 Microsoft Operations Manager 等用户者就可以轻松地管理您的应用程序，而管理员脚本或其他应用程序（托管应用程序和非托管应用程序）也可以监视和配置您的应用程序。

- **System.Messaging**

提供用户连接、监视和管理网络上的消息队列以及发送、接收或查看消息的类。

- **System.Net**

为当前网络采用的多种协议提供简单的编程接口。WebRequest 和WebResponse 类构成所谓的可插接式协议的基础，该协议是一种网络服务的实现，它使您可以开发使用 Internet 资源的应用程序，而不必考虑各个协议的具体细节。

- **System.Net.Sockets**

为需要严格控制网络访问的开发人员提供 Windows 套接字 (Winsock) 接口的托管实现。

- **System.Reflection**

包含提供已加载类型、方法和字段的托管视图的类和接口，并具有动态创建和调用类型的能力。

- **System.Reflection.Emit**

包含允许编译器或工具发出元数据和 Microsoft 中间语言 (MSIL) 并在磁盘上生成 PE 文件 (可选) 的类。这些类的主要客户端是脚本引擎和编译器。

- **System.Resources**

提供允许开发人员创建、存储和管理应用程序中使用的各种区域性特定资源的类和接口。

- **System.Runtime.CompilerServices**

为使用托管代码的编译器编写器提供功能，以影响在公共语言运行库运行时行为的元数据中指定的属性。此命名空间中的类只用于编译器编写器。

- **System.Runtime.InteropServices**

提供用于通过 .NET 访问 COM 对象和本机 API 的类的集合。此命名空间中的类型分为以下功能区：属性、异常、COM 类型的托管定义、包装、类型转换器和 Marshal 类。

- **System.Runtime.InteropServices.Expando**

包含 IExpando 接口，此接口允许通过添加或移除对象的成员来修改对象。

- **System.Runtime.Remoting**

提供允许开发人员创建和配置分布式应用程序的类和接口。

- **System.Runtime.Remoting.Activation**

提供支持服务器和客户端远程对象激活的类和对象。

- **System.Runtime.Remoting.Channels**

包含支持和处理信道和信道接收器的类，这些信道和信道接收器在客户端对远程对象调用方法时用作传输媒介。

- **System.Runtime.Remoting.Channels.Http**

包含使用 HTTP 协议与远程位置之间相互传输消息和对象的信道。默认情况下，HTTP 信道以 SOAP 格式对对象和方法调用进行编码以便传输，但在信道的配置属性中也可以指定其他编码和解码格式化程序接收器。

- **System.Runtime.Remoting.Channels.Tcp**

包含使用 TCP 协议与远程位置之间相互传输消息和对象的信道。默认情况下，TCP 信道以二进制格式对对象和方法调用进行编码以便传输，但在信道的配置属性中也可以指定其他编码和解码格式化程序接收器。

- **System.Runtime.Remoting.Contexts**

包含定义所有对象所驻留的上下文的对象。上下文是一个有序的属性序列，用于定义其中的对象所处的环境。上下文是在对象的激活过程中创建的，这些对象被配置为要求某些自动服务，如同步、事务、实时 (JIT) 激活、安全性等。多个对象可以存留在一个上下文内。

- **System.Runtime.Remoting.Lifetime**

包含管理远程对象生存期的类。传统上，分布式垃圾回收功能使用引用计数和 Ping 来控制对象的生存期。这种机制在每一项服务只有较少的客户端时可以正常工作，但是当每一项服务有几千个客户端时就不能正常工作了。远程处理生存期服务将每一项服务与一个租约关联，当租约到期时，就会删除该服务。生存期服务可以起到传统的分布式垃圾回收器的作用，并在每一项服务的客户端数量增加时能很好地调整。

- **System.Runtime.Remoting.Messaging**

包含用于创建和远程处理消息的类。远程处理基础结构使用消息与远程对象进行通信。消息用于传输远程方法调用、激活远程对象和交流信息。消息对象携带一组命名属性，其中包括操作标识符、代表信息和参数。

- **System.Runtime.Remoting.Metadata**

包含可用于为对象和字段自定义 SOAP 的生成和处理的类和属性。此命名空间中的类可用于指示 SOAPAction、类型输出、XML 元素名和 XML 命名空间 URI 方法。

- **System.Runtime.Remoting.Metadata.W3cXsd2001**

包含由 WWW 联合会（W3C）在 2001 年定义的 XML 架构定义（XSD）。W3C 中的“XML Schema Part2: Data types”（XML 架构第二部分：数据类型）规范确定了各种数据类型的格式和行为。此命名空间包含符合 W3C 规范的数据类型的包装类。所有日期和时间类型都符合 ISO 标准规范。

- **System.Runtime.Remoting.MetadataServices**

包含利用 Soapsuds.exe 命令行工具和用户代码在元数据和远程处理基础结构的 XML 架构之间相互转换的类。

- **System.Runtime.Remoting.Proxies**

包含控制和提供代理功能的类。代理是作为远程对象映像的本地对象。代理使客户端可以跨远程处理边界访问对象。

- **System.Runtime.Remoting.Services**

包含为 .NET 框架提供功能的服务类。

- **System.Runtime.Serialization**

包含可用于序列化和反序列化对象的类。序列化是将对象或对象图转换为线性的字节序列以存储或传输到其他位置的过程。反序列化是接受存储的信息并用这些信息重新创建对象的过程。

- **System.Runtime.Serialization.Formatters**

提供由序列化格式化程序使用的通用枚举、接口和类。

- **System.Runtime.Serialization.Formatters.Binary**

包含可以用二进制格式序列化和反序列化对象的 BinaryFormatter 类。

- **System.Runtime.Serialization.Formatters.Soap**

包含可以用 SOAP 格式序列化和反序列化对象的 SoapFormatter 类。

- **System.Security**

提供公共语言运行库安全系统的基础结构，包括权限的基类。

- **System.Security.Cryptography**

提供加密服务，包括数据的安全编码和解码，以及其他许多操作，如散列处理、随机数生成和消息身份验证。

- **System.Security.Cryptography.X509Certificates**

包含 Authenticode X.509 v.3 证书的公共语言运行库实现。此证书用惟一明确标识证书持有者的私钥签名。

- **System.Security.Cryptography.Xml**

包含在 .NET 框架安全系统中供独占使用的 XML 模型。此 XML 模型不用于常规用途。此模型允许对 XML 对象进行数字签名。

- **System.Security.Permissions**

定义根据策略控制操作和资源访问的类。

- **System.Security.Policy**

包含代码组、成员条件和证据。这三种类型的类用于创建由公共语言运行库安全策略系统应用的规则。证据类是安全策略的输入，成员条件是开关；二者共同创建策略语句并确定授予的权限集。策略级别和代码组是策略层次的结构。代码组是规则的封装并且在策略级别中分层排列。

- **System.Security.Principal**

定义表示运行代码的安全上下文的用户对象。

- **System.ServiceProcess**

提供使您可以实现、安装和控制 Windows 服务应用程序的类。服务是不需要用户界面、长期运行的可执行文件。实现服务的过程包括：从 ServiceBase 类继承，定义在传入启动、停止、暂停和继续命令时处理的特定行为，以及定义当系统关闭时所采取的自定义行为和操作。

- **System.Text**

包含表示 ASCII、Unicode、UTF-7 和 UTF-8 字符编码的类，用于在字符块和字节块之间相互转换的抽象基类，以及不需要创建字符串的中间实例就可以操作和格式化字符串对象的帮助器类。

- **System.Text.RegularExpressions**

提供对 .NET 框架正则表达式引擎的访问的类。此命名空间提供可在 Microsoft .NET 框架上运行的任何平台或语言中使用的正则表达式功能。

- **System.Threading**

提供支持多线程编程的类和接口。此命名空间包括管理线程组的 ThreadPool 类、允许在指定的一段时间后调用委托的 Timer 类，以及用于同步互相排斥的线程的 Mutex 类。此命名空间还提供用于线程安排、等待通知和死锁解析的类。

- **System.Timers**

提供允许以指定的间隔引发事件的 Timer 组件。

- **System.Web**

提供支持浏览器/服务器通信的类和接口。此命名空间包括提供有关当前 HTTP 请求的大量信息的 HttpResponse 类、管理 HTTP 到客户端的输出的 HttpResponse 类，以及提

供对服务器端实用工具和进程访问的 `HTTPServerUtility` 对象。`System.Web` 还包括用于 `Cookie` 操作、文件传输、异常信息和输出缓存控制的类。

- `System.Web.Caching`

提供用于在服务器上缓存常用资源的类。这些资源包括 ASP.NET 页、Web 服务和用户控件。另外，缓存词典可供您存储常用资源，如散列表和其他数据结构。

- `System.Web.Configuration`

包含用于设置 ASP.NET 配置的类。

- `System.Web.Hosting`

提供从 Microsoft Internet Information Server (IIS) 外部的托管应用程序承载 ASP.NET 应用程序的功能。

- `System.Web.Mail`

包含使用 CDOSYS 消息组件构造和发送消息的类。邮件消息通过 Microsoft Windows 2000 中内置的 SMTP 邮件服务或任意的 SMTP 服务器发送。此命名空间中的类可从 ASP.NET 或任何托管应用程序中使用。

- `System.Web.Security`

包含用于在 Web 服务器应用程序中实现 ASP.NET 安全的类。

- `System.Web.Services`

包含可以生成和使用 Web 服务的类。Web 服务是驻留在 Web 服务器上并通过标准 Internet 协议公开的可编程实体 (XML, SOAP)。

- `System.Web.Services.Configuration`

由一些类组成，这些类配置用 ASP.NET 创建的 XML Web services 的运行方式。

- `System.Web.Services.Description`

由一些类组成，这些类使您能够使用 Web 服务描述语言 (WSDL) 公开描述 XML Web services。此命名空间中的每个类对应于 WSDL 规范中的一个特定元素，并且类层次结构对应于有效的 WSDL 文档的 XML 结构。

- `System.Web.Services.Discovery`

由一些类组成，这些类允许 XML Web services 客户端通过称为“XML Web services 发现”的进程来定位 Web 服务器上可用的 XML Web services。

- `System.Web.Services.Protocols`

由一些类组成，这些类定义在通信期间通过网络在 XML Web services 客户端和用 ASP.NET 创建的 XML Web services 之间传输数据的协议。

- `System.Web.SessionState`

提供支持在服务器上存储特定于 Web 应用程序中的单个客户端的数据的类和接口。会话状态数据用于向客户端提供与应用程序的持久连接的外观。状态信息可以存储在本地进程内存中，或者，对于网络场配置来说，可以使用 ASP.NET 状态服务或 SQL Server 数据库将状态信息存储在进程外。

- `System.Web.UI`

提供创建以 Web 页上用户界面形式出现在 Web 应用程序中的控件和页的类和接口。此命名空间包括 `Control` 类，该类为所有控件（不论是 HTML 控件、Web 控件还是用户控件）提

供一组通用功能。它还包括 Page 控件，每当对 Web 应用程序中的页发出请求时，都会自动生成此控件。另外还提供了一些类，这些类提供 Web 窗体服务器控件数据绑定功能、保存给定控件或页的视图状态的能力，以及对可编程控件和文本控件都适用的分析功能。

- **System.Web.UI.Design**

包含用于扩展 Web 用户界面设计时的支持的类。

- **System.Web.UI.Design.WebControls**

包含用于扩展 Web 服务器控件设计时所支持的类。

- **System.Web.UI.HtmlControls**

提供可以在 Web 页上创建 HTML 服务器控件的类。HTML 服务器控件在服务器上运行，并直接映射到所有浏览器支持的标准 HTML 标记。这使您能够以编程方式控制 Web 页上的 HTML 元素。

- **System.Web.UI.WebControls**

包含可以在 Web 页上创建 Web 服务器控件的类。Web 控件在服务器上运行，并包括窗体控件（如按钮和文本框）以及特殊用途的控件（如日历）。这使您能够以编程方式控制 Web 页上的这些元素。Web 控件比 HTML 控件更抽象。它们的对象模型不一定反映 HTML 语法。

- **System.Windows.Forms**

包含用于创建基于 Windows 的应用程序的类，这些应用程序可以充分利用 Microsoft Windows 操作系统中的丰富用户界面功能。

- **System.Windows.Forms.Design**

包含可用于扩展 Windows 窗体设计时支持的类。

- **System.Xml**

提供基于标准的 XML 处理支持。

- **System.Xml.Schema**

提供基于标准的 XML 架构 (XSD) 支持。

- **System.Xml.Serialization**

包含用于将对象序列化为 XML 格式的文档或流的类。

- **System.Xml.XPath**

包含 XPath 分析器和计算引擎。它支持 W3C XML 路径语言 (XPath) 1.0 版建议 (www.w3.org/TR/xpath)。

- **System.Xml.Xsl**

提供可扩展样式表转换 (XSLT) 支持。它支持 W3C XSL 转换 (XSLT) 1.0 版建议 (www.w3.org/TR/xslt)。

1.3 数 据 流

在网络编程中，常用的有网络流、文本流和文件流。网络流用于传输数据，文本流用于读写文本，文件流用于读写文件。下面简单介绍这几种流。

1.3.1 网络流

在.NET 平台上，`NetworkStream` 类提供用于网络访问的基础数据流，`NetworkStream` 实现了通过网络套接字发送和接收数据的标准 .NET 框架流机制。`NetworkStream` 支持对网络数据流的同步和异步访问。`NetworkStream` 不支持对网络数据流的随机访问。`CanSeek` 属性总为 `false`；读取 `Position` 属性或调用 `Seek` 方法将引发 `NotSupportedException`（系统不支持异常）。

`NetworkStream` 列的构造方法有以下几种。

原型一：

```
public NetworkStream(  
    Socket socket  
) ;
```

参数是套接字对象。下面例子演示了该方法的用法。

```
NetworkStream newStream=new NetworkStream(mySock);
```

其中“mySock”是套接字对象。

原型二：

```
public NetworkStream(  
    Socket socket,  
    bool ownsSocket  
) ;
```

第一个参数是套接字对象，第二个对象用于声明该网络流实例是否拥有该套接字对象，也就是创建对指定套接字具有读/写访问权限的 `NetworkStream` 对象。如果 `ownsSocket` 为 `true`，则 `NetworkStream` 拥有基础套接字，因而调用 `Close` 方法也将关闭该基础套接字。下面例子演示了该方法的用法。

```
NetworkStream newStream=new NetworkStream(mySock,true);
```

原型三：

```
public NetworkStream(  
    Socket socket,  
    FileAccess access  
) ;
```

该方法用于创建不拥有套接字但可控制 `NetworkStream` 对象访问权限的 `NetworkStream` 对象。第一个参数是套接字对象，第二个参数是读写权限，共有三种：`Read`、`Write`、`ReadWrite`。下面例子演示了该方法的用法。

```
NetworkStream newStream=new  
    NetworkStream(msocck,System.IO.FileAccess.Read);
```

原型四：

```
public NetworkStream(
    Socket socket,
    FileAccess access,
    bool ownsSocket
);
```

该方法用于创建可拥有套接字对象的附加访问权限的网络流对象。如果 `ownsSocket` 为 `true`，则网络流对象拥有该套接字对象，在使用 `Close` 方法时，也关闭该套接字。`access` 用于指定网络流的访问权限。下面例子演示了该方法的用法：

```
NetworkStream newStream=new NetworkStream
(mesock, System.IO.FileAccess.Read, true);
```

`NetworkStream` 类的主要属性如表 1-1 所示。

表 1-1 `NetworkStream` 类的主要属性

属性	用途
<code>CanRead</code>	已重写。获取一个值，该值指示当前流是否支持读取
<code>CanSeek</code>	已重写。获取一个值，该值指示流是否支持查找。该属性总是返回 <code>false</code>
<code>CanWrite</code>	已重写。获取一个值，该值指示当前流是否支持写入
<code>DataAvailable</code>	获取一个值，该值指示是否可以在流上读取数据
<code>Length</code>	已重写。流上可用数据的长度。此属性始终引发 <code>NotSupportedException</code>
<code>Position</code>	已重写。获取或设置流中的当前位置。此属性始终引发 <code>NotSupportedException</code> 。

`NetworkStream` 类最常用的方法是 `Read`、`Write` 和 `Flush` 方法。`Read` 方法用于读取数据；`Write` 方法用于向流写数据；`Flush` 用于刷新流，由于数据有时会存放在缓冲区中，而没有真正实现指定的操作，这时需要刷新流。下面简单介绍这几个方法。

● Read 方法

该方法定义如下：

```
public override int Read(
    [in] byte[] buffer,
    [int] offset,
    [int] size
);
```

该方法第一个参数是字节数组，用于暂存数据（缓存），第二个参数是开始读取的位置，第三个参数是要读取的字节总数。返回值是 `int` 值，表示已经读取的字节数。下面代码演示了该方法的用法。

```
NetworkStream newStream=new NetworkStream(mesock);
byte[] myByte=new Byte[1024];
newStream.Read(myByte, 0, myByte.Length);
newStream.Flush();
```

在上述代码中时用了 `Flush` 方法，该方法定义非常简单，如下所示：

```
public override void Flush();
```

既没有参数也没有返回值。

- Write 方法

该方法定义如下：

```
public override void Write(
    byte[] buffer,
    int offset,
    int size
);
```

该方法第一个参数是字节数组，用于暂存数据（缓存），第二个参数是开始写的位置，第三个参数是要写的字节总数。该方法没有返回值。下面代码演示了该方法的用法：

```
NetworkStream newStream=new NetworkStream(mesock);
byte[] myByte=new Byte[1024];
newStream.Write(myByte,0,myByte.Length);
newStream.Flush();
```

1.3.2 文本流

文本流用于文本的读写。要谈文本，首先要了解的是文本编码。在网络编程中，常用的是 ASCII、UTF8 和 BigEndianUnicode 码。其中 ASCII 码最为常用，几乎所有协议都支持 ASCII 码，SMTP 和 POP3 协议使用 UTF8 编码，但也支持 ASCII 码，BigEndianUnicode 码是双字节码，一般协议不用该种编码作为命令码，因为双字节码在解码时较为繁琐，容易出错。UTF8 和 BigEndianUnicode 编码支持汉字传输，其中 UTF8 码使用起来较为方便，可以像使用 ASCII 码一样使用 UTF8 码。下面演示向文件中和向网络中写文本流：

- 向文件写文本流

下列代码演示新建或打开（如果不存在，就新建一个文件；如果已经存在，则打开）一个文件，然后向文件写文本数据，并且新数据代替旧数据。

```
StreamWriter sw=null;
sw=new StreamWriter("e:\\temp\\aa.txt",false,System.Text.Encoding.UTF8);
sw.WriteLine("aaaaaaaaaaaaaaaaaaaa");
sw.Close();
```

下列代码演示新建或打开（如果不存在，就新建一个文件；如果已经存在，则打开）一个文件，然后向文件写文本数据，并且新数据放在旧数据后面。

```
StreamWriter sw=null;
sw=new StreamWriter("e:\\temp\\aa.txt",true,System.Text.Encoding.UTF8);
sw.WriteLine("aaaaaaaaaaaaaaaaaaaa");
sw.Close();
```

- 向网络写入文本

下列代码演示向网络流写文本。

```
string str="aaaaaaaaaaaaaaaaaaaaaaa";
```

```
byte[] myByte=System.Text.Encoding.UTF8.GetBytes(str);
NetworkStream netStream=new NetworkStream(mysock);
netStream.Write(myByte,0,myByte.Length);
```

- 从文件读取文本

下列代码演示如何从文件读取文本：

```
StreamReader sr=null;
sr=new StreamReader("e:\\temp\\aa.txt",System.Text.Encoding.UTF8);
string myStr=sr.ReadToEnd();
sr.Close();
```

- 从网络读取文本

下列代码演示如何从网络读取文本：

```
string str="aaaaaaaaaaaaaaaaaaaaaaaa";
byte[] myByte=System.Text.Encoding.UTF8.GetBytes(str);
NetworkStream netStream=new NetworkStream(mysock);
netStream.Read(myByte,0,myByte.Length);
```

1.3.3 文件流

文件流主要用于读写非文本的文件。文件类型可以是各种各样的，无论是文本文件，还是图片文件、压缩文件、可执行文件等等，都可以用文本流实现读写功能。下例演示如何用文件流读写数据。

- 将网络流数据写入文件

下列代码演示如何从网络流中读取数据并将内容写入文件，其中 path 代表文件路径，stream 代表已经连接并且存在数据传输的网络流。

```
FileStream filestream=new FileStream(path,
 FileMode.OpenOrCreate, FileAccess.Write);
int readNumber=0;
byte[] bye=new byte[8];
while((readNumber=stream.Read(bye,0,8))>0)
{
    filestream.Write(bye,0,readNumber);
    filestream.Flush();
}
filestream.Close();
```

- 读文件并写入网络流

下列代码演示如何用文件流从文件读取数据，并将数据发送到网络流，其中 path 代表文件路径，stream 代表已经连接并且存在数据传输的网络流。

```
filestream=new FileStream(path,FileMode.Open,FileAccess.Read);
int number;
//定义缓冲区
```

```

byte[] bb=new byte[8];
//循环读文件
NetworkStream stream=new NetworkStream(newClient);
while((number=filestream.Read(bb,0,8))!=0)
{
    //向客户端发送流
    stream.Write(bb,0,8);
    //刷新流
    stream.Flush();
    bb=new byte[8];
}
filestream.Close();

```

1.4 命令解析

在网络协议中，协议的命令通常用 ASCII 码表示，但在实际应用中，单纯使用 ASCII 码很不方便，还可能使用其他编码，比如 UTF8 码、BigEndianUnicode 码等。下面简单介绍一下如何获取命令及其参数。

1.4.1 普通格式命令的解析

在所有网络协议中，最常用的命令是，命令与参数之间一般以空格（<SP>）间隔，参数与参数之间用空格间隔，命令以回车加换行符结束（<CRLF>）。下面例子说明如何解析参数以空格间隔且命令以<CRLF>结束的命令。以其他符号间隔的命令可参照下列代码解析即可。

比如有一条命令：

RENA<SP>E:\AA.TXT<SP>C:\BB.TXT<CRLF>

假定上述代码是一条要求将 E:\AA.TXT 重命名为 C:\BB.TXT 的命令。但发送方一般不可能只发送上述代码，在上述代码的后面可能还跟着许多空字符（“\0\0\0.....”），如果将这些无用的字符也接受过来，程序（服务器或客户端）就无法识别，也就不能完成指定操作。这时就要把无用的字符过滤掉。

下面代码可以做到只接受有用命令，而且能过滤掉无用字符：

```

byte[] myByte=new Byte[1024];
//下面的“mySock”假定是建立连接的套接字对象
NetworkStream netStream=new NetworkStream(mysock);
netStream.Read(myByte,0,myByte.Length);
string str=System.Text.Encoding.ASCII.GetString(myByte);
int x=str.IndexOf("\r\n");
string allCommand=str.Substring(0,x);
char[] a=new char[]{' '};
string[] commStr=allCommand.Split(a);

```

```
string command=commStr[0];
string parameter1=commStr[1];
string parameter2=xommStr[2];
```

在上述代码中，使用了 `String` 类的 `IndexOf` 方法，该方法可在特定字符串（A）中查找指定的字符串（B），并返回指定字符串（B）所在的第一个位置。

上述代码还使用了 `String` 类的 `Split` 方法，该方法可以将字符串用特定的字符数组分割成若干段。

上述代码执行的结果如下：

- 字符串 `command` 的值为 RENA;
- 字符串 `parameter1` 的值为 E:\AA.TXT;
- 字符串 `parameter2` 的值为 C:\BB.TXT。

有关 `UTF8` 码和 `BigEndianUnicode` 码命令的解析，可参照 `ASCII` 码命令进行。

1.4.2 特殊格式命令解析

有一些程序，由于特殊需求，不能以`<CRLF>`结束，而以`<EOF>`结束，主命令码与参数之间、参数与参数之间用`<CRLF>`间隔。这是一种不太常用的命令，但在特殊领域有所应用，有时使用起来还很方便。只是解析命令时稍微繁琐一点。

对这种命令的解析可使用如下两种方法进行。一种方法是：首先在字符串中寻找“EOF”，以确定总的有效字符串长度，然后分别查找“`\r\n`”的位置，最后分段将数据赋值给字符串对象。另外一种方法是：首先构造一个 `RichTextBox` 控件对象，将有效字符串的值赋给该控件，然后再在该控件中按行读取数据。这两种方法都不失为一种可单独使用的办法。

下面有一个字符串：USER<CRLF>AAA<CRLF>PASS<CRLF>BBBEOF；

```
string readMessage="USER"+"\r\n"+ "AAA" + "\r\n" + "PASS" + "\r\n" + "BBB" + "EOF";
int x=readMessage.IndexOf("EOF");
string ss=readMessage.Substring(0,x);
int y=ss.IndexOf("\r\n");
string command=ss.Substring(0,y);//结果
string aaa=ss.Substring(y+2,x-y-2);
y=aaa.IndexOf("\r\n");
string command1=aaa.Substring(0,y);//结果
aaa=aaa.Substring(y+2,aaa.Length-y-2);
y=aaa.IndexOf("\r\n");
string command2=aaa.Substring(0,y);
y=aaa.IndexOf("\r\n");
aaa=aaa.Substring(y+2,aaa.Length-y-2);
string command3=aaa;
```

执行结果是：`command` 为“UAER”，`command1` 为“AAA”，`command2` 为“PASS”，`command3` 为“BBB”。

下面用第二种方法完成解析命令的操作。

```

string readMessage="USER"+"\r\n"+ "AAA" + "\r\n" + "PASS" + "\r\n" + "BBB" + "EOF";
RichTextBox rich=new RichTextBox();
rich.Text=readMessage;
int i=rich.Lines.Length;
string comm=rich.Lines[0];
string comm1=rich.Lines[1];
string comm2=rich.Lines[2];
string comm3=rich.Lines[3].Substring(0,rich.Lines[3].Length-3);

```

其中 comm 为 “UAER” , comm1 为 “AAA” , comm2 为 “PASS” , comm3 为 “BBB” 。

1.5 方法参数

在使用方法的关键字时，如果为没有 `ref` 或 `out` 的方法声明一个参数，则此参数可以具有关联的值。可以在方法中更改该值，但当控制传递回调用过程时，不会保留更改的值。通过使用方法参数关键字，可以更改这种行为。

1.5.1 params 关键字

`params` 关键字是可以指定在参数数目可变处采用参数的方法参数。在方法声明中的 `params` 关键字之后不允许任何其他参数，并且在方法声明中只允许一个 `params` 关键字。

下列代码演示如何使用该关键字声明和使用数目可变的参数。

```

private void UseParams(params string[] list)
{
    for ( int i = 0 ; i < list.Length ; i++ )
    {
        richTextBox1.AppendText(list[i]+"\r\n");
    }
}

private void UseParams1(params object[] list)
{
    for ( int i = 0 ; i < list.Length ; i++ )
        richTextBox1.AppendText((object)list[i].ToString()+"\r\n");
}

private void button1_Click(object sender, System.EventArgs e)
{UseParams("how","are","you");
 UseParams1(6,'c',"good study!");}
}

```

图 1-2 是演示结果。

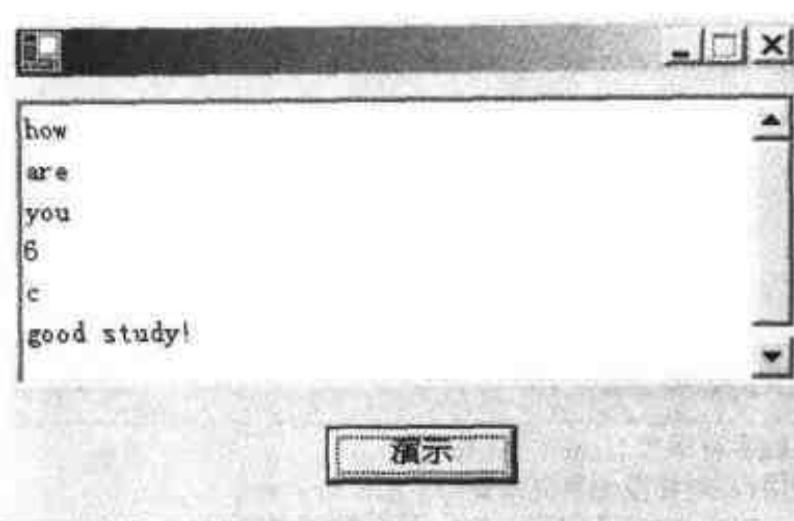


图 1-2 params 关键字

params 关键字是非常有用的，在许多程序中，事先并不知道用户要输入多少参数以及参数的类型，这时即可使用该关键字。下面方法可以接受任何类型的参数：

```
private void UseParams1(params object[] list)
{
    ...
}
```

1.5.2 ref 关键字

方法参数中的 ref 关键字可以传递参数值，即在方法内部对参数的任何改变会影响到外部变量，也就是说，外部变量的值也跟着变。若要使用 ref 参数，必须将参数作为 ref 参数显式传递到方法，而且，传递到 ref 参数的参数必须最先初始化。如果两个方法的声明只在 ref 的使用方面不同，则会发生重载。需要注意的是，属性不是变量，不能作为 ref 参数传递。下列代码演示使用 ref 关键字与不使用 ref 关键字的区别：

```
private void testRef(ref string str)
{
    str="how are you";
}

private void testNoRef( string str)
{
    str="how do you do";
}

private void button1_Click(object sender, System.EventArgs e)
{
    string myStr="good study";
    richTextBox1.AppendText("原始字符串为: "+myStr+"\r\n");
    testRef(ref myStr);
    richTextBox1.AppendText("使用 ref 时将原始字符串变为: "+myStr+"\r\n");
    testNoRef(myStr);
```

```

richTextBox1.AppendText("testNoRef 方法不使用 ref 时，原始字符串仍为：
"+myStr+"\r\n");
}

```

图 1-3 为执行结果。

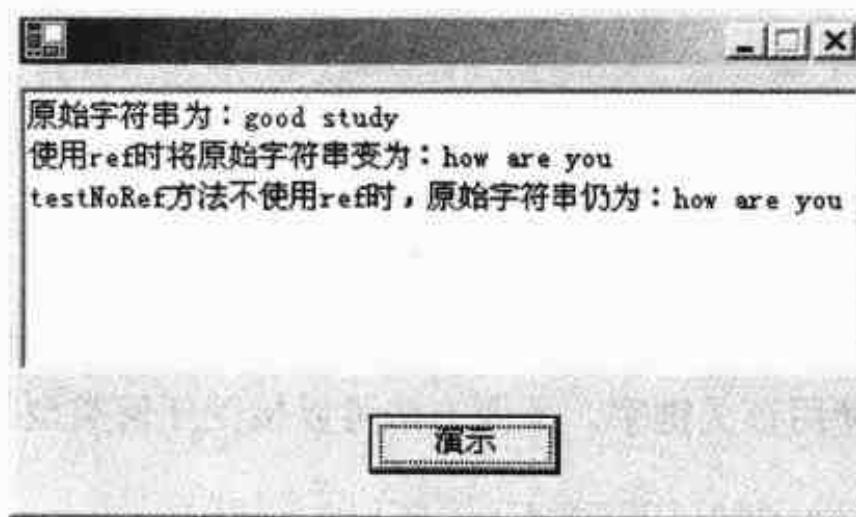


图 1-3 ref 关键字

上述结果表明，使用 `ref` 时，将方法内部对参数的改变传递给了外部变量；而不使用 `ref` 关键字时，方法内部对参数的改变不影响外部变量的值。

1.5.3 out 关键字

`out` 关键字的主要用途是返回多个返回值。与 `ref` 关键字相同之处为：在方法内部对参数的任何改变都会反映到外部变量上。不同的是，使用 `out` 关键字时，变量不需要初始化。若要使用 `out` 参数，必须显式声明使用 `out` 关键字。如果两个方法的声明仅在 `out` 的使用方面不同，则会发生重载。需要注意的是，与 `ref` 关键字一样，属性不是变量，不能作为 `out` 参数传递。

下列代码演示 `out` 关键字的用法：

```

private string TestOut(out char i)
{
    i = 'a';
    return "good study!";
}

private void button1_Click(object sender, System.EventArgs e)
{
    char i;//无需初始化
    string getReturn=TestOut(out i);
    richTextBox1.AppendText("方法执行的返回值是：" +getReturn+"\r\n");
    richTextBox1.AppendText("返回 out 参数的值为：" +i.ToString()+"\r\n");
}

```

图 1-4 为执行结果。

从执行结果可以发现，使用 `out` 关键字可以根据需要返回多个值，使用非常方便。

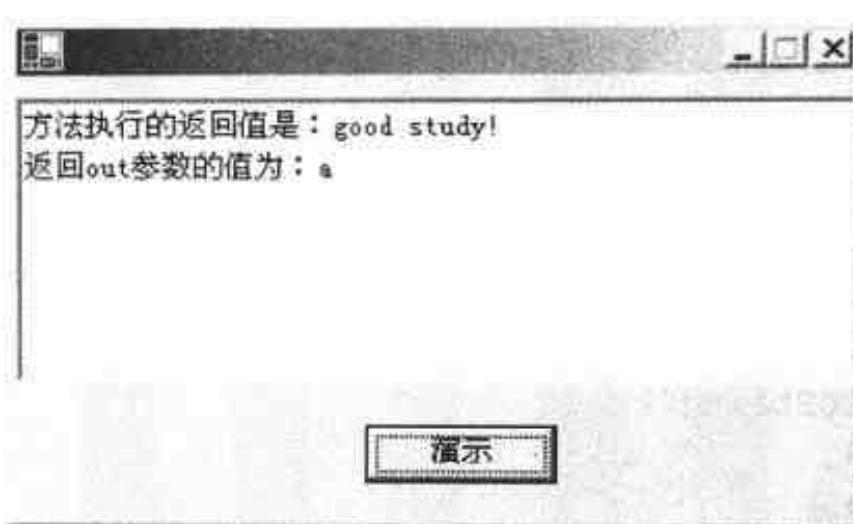


图 1-4 out 关键字

1.6 常用数据类型及其传输

在网络上，文本数据是非常容易传输的，这在 1.3 节的代码中已经演示过。但在网络编程中，还经常用到其他类型的数据，比如 Int 值、Float 值、Double 值、DateTime 值、IPAddress 值等。这些类型值在传输之前，一般可用 ToString 方法转换成字符串文本，然后进行传输。另一终端接收到该文本后，用相应的 Parse 方法将字符串转换成相应的类型值。下列代码演示上述用法。

1. 时间类型值

```
DateTime aa=new DateTime(2002,1,1,12,21,12);
//将时间类型值转换为字符串
string timescr=aa.ToString();
//将字符串转换为时间，年月日之间用“-”隔开，日期与钟点之间用空格隔开，
//时、分秒之间用“：“隔开
string date="2002-5-1 12:22:22";
DateTime bb=DateTime.Parse(date);
```

2. 整型值

● 长整型

```
long x=12345;
//将长整型转化为字符串
string intStr=x.ToString();
string ss="12345";
//将字符串转化为长整型
long y=Int64.Parse(ss);
```

● 整型

```
int x=12345;
//将整型转化为字符串
string intStr=x.ToString();
string ss="12345";
```

```
//将字符串转化为整型
int y=Int32.Parse(ss);
```

● 短整型

```
Int16 x=12345;
//将短整型转化为字符串
string intStr=x.ToString();
string ss="12345";
//将字符串转化为短整型
Int16 y=Int16.Parse(ss);
```

3. float 和 double 值

需要注意的是，double 类型值不能隐式地转换为 float 值，所以在为 float 类型值赋值时必须显式地转换。下列代码演示 float 值的转换。

```
float aa=(float)123.12;
string floatStr=aa.ToString();
float bb=float.Parse("123.3");
```

下列代码演示 double 值的转换：

```
double aa=123.12;
string floatStr=aa.ToString();
double bb=float.Parse("123.3");
```

4. IPAddress 值

下列代码演示 IPAddress 类型值的转换。

```
IPAddress aa=new IPAddress(12345);
string IPString=aa.ToString();
IPAddress bb=IPAddress.Parse("123.2.2.1");
```

1.7 线 程

操作系统使用进程将正在执行的不同应用程序分开。线程是操作系统分配处理器时间的基本单元，并且该进程中可以有多个线程同时执行代码。每个线程都维护异常处理程序、调度优先级和一组系统用于在调度该线程前保存线程上下文的结构。线程上下文包括使线程在线程的宿主进程地址空间中无缝地继续执行所需的所有信息，包括线程的 CPU 寄存器组和堆栈。要提高对用户的响应速度并且处理所需数据以便几乎同时完成工作，使用多个线程是一种功能强大的技术。多个线程可以通过利用用户事件之间很小的时间段在后台处理数据来达到这种效果。例如，当另一个线程正在计算程序的数据时，用户可以同时输入数据。使用线程的缺点是：它占用系统资源，而且在处理线程时有可能产生许多错误。下面简单介绍一下线程的常用操作。欲进一步了解线程，读者可参考有关 C# 语法。

1. 线程创建

线程创建可用 System.Threading 命名空间下的 Thread 类的构造方法来完成。该构造方法的原型如下：

```
public Thread(  
    ThreadStart start  
)
```

参数为 ThreadStart 的类型值。下列代码演示如何创建线程。

```
Thread thread=new Thread(new ThreadStart(accp));  
Private void accp() {  
    ...  
    ...  
}
```

2. 线程启动

线程启动可用 Thread 类的 Start 方法来完成。该方法既无参数也无返回值。其定义如下：

```
public void Start();
```

下列代码演示线程的启动：

```
Thread thread=new Thread(new ThreadStart(accp));  
thread.Start();  
  
Private void accp() {  
    ...  
    ...  
}
```

3. 线程暂停与重新启动

在启动线程后，可以调用 Thread.Sleep 使当前线程立即阻塞一段时间（参数单位为毫秒），调用 Thread.Sleep(Timeout.Infinite) 将使线程休眠，直到它被调用 Thread.Interrupt 的另一个线程中断或被 Thread.Abort 中止为止。需要注意的是，一个线程不能对另一个线程调用 Sleep。还可以通过调用 Thread.Suspend 来暂停一个线程。当线程对其自身调用 Thread.Suspend 时，该调用将阻塞，直到该线程被另一个线程继续为止。当一个线程对另一个线程调用 Thread.Suspend 时，该调用就成为使另一个线程暂停的非阻塞调用。调用 Thread.Resume 将使另一个线程跳出挂起状态并使该线程继续执行，而与调用 Thread.Suspend 的次数无关。例如，如果连续调用 Thread.Suspend 三次，然后调用 Thread.Resume，则该线程将在对 Resume 的调用后面立即继续执。

下列代码演示线程休眠：

```
Thread.Sleep(10000);
```

下列代码演示线程挂起：

```
Thread thread=new Thread(new ThreadStart(accp));
```

```
thread.Start();
...
...
thread.Suspend();
```

下列代码演示线程的重新开始：

```
Thread thread=new Thread(new ThreadStart(accp));
thread.Start();
...
...
thread.Suspend();
...
...
thread.Resume();
```

还可以使用许多其他方法来阻塞线程。例如，可以通过调用 `Thread.Join` 使一个线程等待另一个线程停止。因为本书不是写语法的，其他方法这里就不一一介绍了。

`Thread` 类的 `Join` 方法有以下几种原型。

原型一：

```
public void Join();
```

原型二：

```
public bool Join(
    int millisecondsTimeout
)
```

参数是毫秒。

原型三：

```
public bool Join(
    TimeSpan timeout
)
```

参数是“`TimeSpan`”（时间间隔）类型值。

下列代码演示该方法的用法：

```
Thread thread=new Thread(new ThreadStart(accp));
thread.Start();
...
...
thread.Join(10000);
```

4. 线程销毁

线程要占用大量的系统资源，完成特定使命后，必须适时销毁。销毁线程可用 `Thread` 类的 `Abort` 方法或 `Interrupt` 方法。

在调用 `Abort` 方法时，在指定线程上引发 `ThreadAbortException`，以开始终止此线程的过程。`ThreadAbortException` 是一个可以由应用程序代码捕获的特殊异常，但除非调用

ResetAbort，否则会在 catch 块的结尾再次引发它。ResetAbort 可以取消 Abort 的请求，并阻止 ThreadAbortException 终止此线程。但是，线程不一定会立即中止，或者根本不中止。如果线程在作为中止过程的一部分被调用的 finally 块中做非常大量的计算，从而无限期延迟中止操作，则会发生这种情况。若要确保线程已经中止，请在调用 Abort 之后对线程调用 Join 方法。如果对尚未启动的线程调用 Abort，则当调用 Start 时该线程将中止。如果对已挂起的线程调用 Abort，则该线程将继续，然后中止。如果对被阻塞或正在休眠的线程调用 Abort，则该线程被中断，然后中止。

Thread 类的 Abort 方法有以下几种原型。

原型一：

```
public void Abort();
```

原型二：

```
public void Abort(  
    object stateInfo  
)
```

下列代码演示该方法的用法：

```
Thread thread=new Thread(new ThreadStart(accp));  
thread.Start();  
...  
...  
thread.Abort();  
thread.Join(10000);
```

Thread 类的 Interrupt 方法用于中断处于 Wait、Sleep、Join 线程状态的线程。如果此线程当前未阻塞，在等待、休眠或连接状态中，则下次开始阻塞时它将被中断。该方法定义原型如下：

```
public void Interrupt()
```

该方法既无参数也无返回值。下列代码演示该方法的用法：

```
Thread thread=new Thread(new ThreadStart(accp));  
thread.Start();  
...  
...  
thread.Join(10000);  
thread.Interrupt();
```

本章小结

本章简单介绍了 C#的基础知识以及网络编程中的常用语法。掌握这些知识，对今后学习 C#网络高级开发有一定的帮助。

第 2 章 基础服务器开发

本章使用.NET 平台的同步套接字技术、异步套接字技术以及 TcpListener 类开发三个基础服务器。在这些服务器的基础上，稍加修改就可以成为适合商业需要的应用服务器。

2.1 同步套接字服务器开发

本节首先介绍一下同步套接字服务器的主机解析、主机绑定、端口监听、数据接收、数据发送的基础知识，然后利用 System.Net 名字空间的 Sockets 类开发一个套接字同步服务器实例。

2.1.1 定义主机对象

定义主机对象可用 IPEndPoint 类的 IPEndPoint 构造方法。IPEndPoint 类位于 System.Net 命名空间下，该类的构造方法可以构造一个新的主机对象。该方法有两种原型。

原型一：

```
public IPEndPoint(  
    IPAddress address,  
    int port  
)
```

该方法有两个参数，第一个参数是 IP 地址，第二个参数是端口号。下列代码演示该方法的用法：

```
IPAddress myIP=IPAddress.Parse("127.0.0.1");  
IPEndPoint MyServer=new IPEndPoint(myIP,myIP);
```

上述代码使用了 IPAddress 类的 Parse 方法。该方法可以把字符串转换为指定的格式。这里是把字符串转换为 IPAddress 格式。IPAddress 类位于命名空间 System.Net 下，该类可以定义一个 IP 地址格式的对象。

原型二：

```
public IPEndPoint(  
    long address,  
    int port  
)
```

该方法有两个参数，第一个参数是整型数值 (Int64)，第二个参数是端口号 (Int32)。下列代码演示该方法的用法

```
IPEndPoint aa=new IPPEndPoint(12345,90);
```

2.1.2 主机解析

在服务器绑定与监听端口之前，首先要利用 DNS 服务器解析主机。解析主机可以用 Dns 类的 Resolve 方法，该方法原型如下：

```
public static IPHostEntry Resolve(  
    string hostName  
)
```

该方法只有一个参数，即待解析主机的主机名称，但输入 IP 格式的字符串时也不会出错。该方法返回值是一个 IPHostEntry 类型值，这是一个为 Internet 主机地址信息提供容器的类。通俗地说，该类包含了诸多的主机信息（包括 IP 地址列表、主机名称等）。下列代码演示 Dns 类 Resolve 方法的使用方法：

```
IPHostEntry myHost=new IPHostEntry();  
myHost=Dns.Resolve("aaa.bbb.com");  
  
for(int i=0;i<myHost.AddressList.Length;i++)  
{  
    richTextBox1.AppendText(myHost.AddressList[i].ToString()+"\r\n");  
}
```

下列代码用 Resolve 方法获取主机的信息：

```
IPHostEntry myHost=new IPHostEntry();  
myHost=Dns.Resolve(textBox1.Text);  
for(int i=0;i<myHost.AddressList.Length;i++)  
{  
    richTextBox1.AppendText(myHost.HostName.ToString()+"\r\n");  
}
```

Dns 类的 GetHostName 可以获取本地主机的名称，使用这个方法可以代替手动输入主机名称，十分方便。

```
public static string GetHostName()
```

下列代码可以获取本地主机名称，并把主机名称赋给 textBox1 的 Text 属性。

```
textBox1.Text=Dns.GetHostName();
```

另外 Dns 类的 GetHostByName 方法可以通过主机名称获取主机信息，GetHostByAddress 方法可以通过 IP 地址获取主机信息。GetHostByName 方法的原型如下：

```
public static IPHostEntry GetHostByName(  
    string hostName  
)
```

该方法只有一个参数，即主机名称。下列代码演示该方法的用法：

```

IPHostEntry myHost=new IPHostEntry();
try
{
    myHost=Dns.GetHostByName(textBox1.Text);
}
catch(Exception ee){MessageBox.Show(ee.Message);}
for(int i=0;i<myHost.AddressList.Length;i++)
{
    textBox2.AppendText(myHost.AddressList[i].ToString()+"\r\n");
}

```

GetHostByAddress方法的原型有两个。

原型一：

```

public static IPHostEntry GetHostByAddress(
    IPAddress address
)

```

该方法只有一个参数，即 IP 地址，返回值是 IPHostEntry 类型值。下列代码演示该方法的用法：

```

IPAddress myIP=IPAddress.Parse(textBox1.Text);
IPHostEntry myHost=new IPHostEntry();
myHost=Dns.GetHostByAddress(myIP);
textBox2.Text=myHost.HostName.ToString();

```

原型二：

```

public static IPHostEntry GetHostByAddress(
    string address
)

```

该方法只有一个参数，即 IP 地址格式的字符串，如“127.0.0.1”。下列代码演示该方法的用法：

```

IPHostEntry myHost=new IPHostEntry();
myHost=Dns.GetHostByAddress("127.0.0.1");
textBox2.Text=myHost.HostName.ToString();

```

另外，Dns 类还有许多其他方法可以获取特定主机的信息，这里就不一一介绍了。

2.1.3 端口绑定与监听

同步套接字服务器主机的绑定与端口的监听要使用 Socket 类的 Bind 方法、Listen 和 Accept 方法。Bind 方法用于绑定主机，Listen 方法用于监听端口，Accept 方法用于接收客户端的连接请求。Bind 方法的原型为：

```

public void Bind(
    EndPoint localEP
)

```

)

该方法只有一个参数，即主机对象，比如 IPEndPoint。

Listen 方法原型为：

```
public void Listen(
    int backlog
)
```

该方法的参数为整型数值，意义为挂起对列的最大数。

Accept 方法原型为：

```
public Socket Accept()
```

该方法没有参数，返回值是套接字对象。

下列代码演示如何开始监听过程。

```
IPAddress myIP=IPAddress.Parse("127.0.0.1");
IPEndPoint MyServer=new IPPEndPoint(myIP,2020);
Socket sock =new Socket(AddressFamily.InterNetwork,
                        SocketType.Stream,ProtocolType.Tcp);
sock.Bind(MyServer);
sock.Listen(50);
Socket bbb=sock.Accept();
```

2.1.4 发送数据

发送数据可以用 **Socket** 类的 **Send** 方法或 **NetworkStream** 类的 **Write** 方法。

Send 方法有以下几种原型。

原型一：

```
public int Send( byte[] buffer)
```

该方法只有一个参数，就是字节数组。

原型二：

```
public int Send(byte[], SocketFlags);
```

该方法有两个参数，第一个参数为字节数组，第二个参数为 **SocketFlags** 值。**SocketFlags** 成员如表 2-1 所示。

表 2-1 **SocketFlags** 成员名及其说明

成员名称	说 明
DontRoute	不使用路由表发送
MaxIOVectorLength	为发送和接收数据的 WSABUF 结构数量提供标准值
None	不对此调用使用标志
OutOfBand	处理带外数据
Partial	消息的部分发送或接收
Peek	查看传入的消息

原型三：

```
public int Send(byte[], int, SocketFlags);
```

该方法第一个和第三个参数和原型二相同，第二个参数为要发送的字节数。

原型四：

```
public int Send(byte[], int, int, SocketFlags);
```

该方法第一个、第三个、第四个参数与原型三相同，第二个参数是 byte[] 中开始发送的位置。

下列代码演示如何使用该方法发送数据：

```
...
Socket bbb=sock.Accept();
Byte[] bytee=new Byte[64];
string send="aaaaaaaaaaaaaa";
bytee=System.Text.Encoding.BigEndianUnicode.GetBytes(send.ToCharArray());
bbb.Send(bytee,bytee.Length,0);
```

NetworkStream 类的 Write 方法也能发送数据。该方法的原型为：

```
public override void Write(
    byte[] buffer,
    int offset,
    int size
)
```

该方法第一个参数为字节数组，第二个参数为开始发送字节的位置，第三个参数是发送的数据的总字节数。

下列代码演示了该方法的用法：

```
...
Socket bbb=sock.Accept();
NetworkStream stre=new NetworkStream(bbb);
Byte[] ccc=new Byte[512];
string sendMessage="aaaaaaaaaaaaaa";
ccc=System.Text.Encoding.BigEndianUnicode.GetBytes(sendMessage);
stre.Write(ccc,0,ccc.Length);
```

2.1.5 接收数据

接收数据可以用 Socket 类的 Receive 方法或 NetworkStream 类的 Read 方法。Socket 类的 Receive 方法有以下几种形式，每个参数的含义和 Socket 类的 Send 方法相同。

```
public int Receive(byte[])
public int Receive(byte[], SocketFlags)
public int Receive(byte[], int, SocketFlags)
public int Receive(byte[], int, int, SocketFlags)
```

下列代码演示用该方法的接收数据的方法：

```
...
Socket bbb=sock.Accept();
...
Byte[] ccc=new Byte[512];
bbb.Receive(ccc,ccc.Length,0);
string rece=System.Text.Encoding.BigEndianUnicode.GetString(ccc);
richTextBox1.AppendText(rece+"\r\n");
```

NetworkStream 类的 Read 方法可以接收数据，该方法的原型为：

```
public override int Read(
    in byte[] buffer,
    int offset,
    int size
)
```

该方法三个参数的含义与 NetworkStream 类的 Write 方法相同。下列代码演示该方法的使用方法：

```
...
bbb=sock.Accept();
...
NetworkStream stre=new NetworkStream(bbb);
Byte[] ccc=new Byte[512];
stre.Read(ccc,0,ccc.Length);
string readMessage=System.Text.Encoding.BigEndianUnicode.GetString(ccc);
```

2.1.6 基础服务器开发实例

下面开发一个同步套接字基础服务器。该程序使用 Socket 类的 Bind 方法绑定本地主机，使用 Listen 方法监听端口，使用 Accept 接收连接请求。使用 NetworkStream 类的 Read 方法读取数据，使用 Write 方法发送数据。为了避免程序等待，使用 Thread 类的 Start 方法实现线程同步。另外，为了能够收发中文，文本编码使用 BigEndianUnicode 码。下面是该程序的界面（图 2-1）。

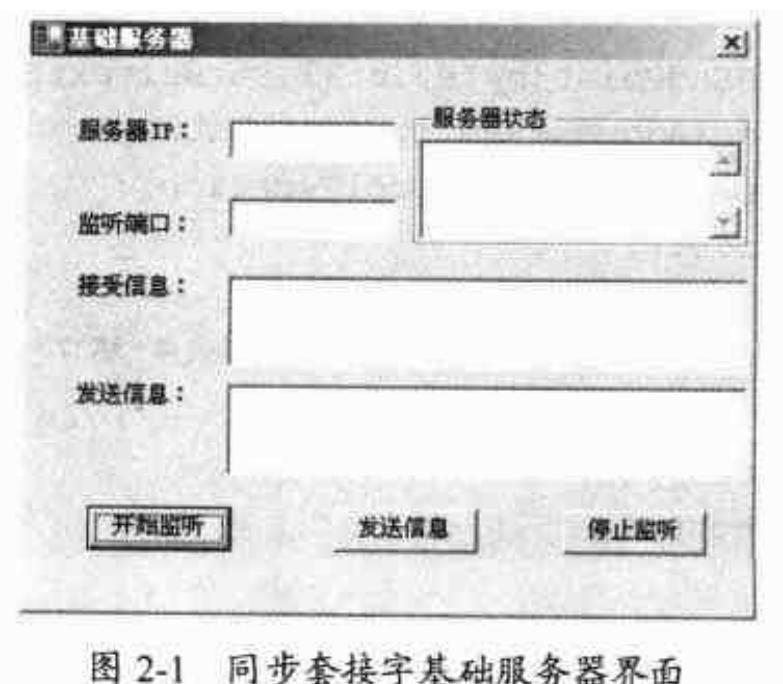


图 2-1 同步套接字基础服务器界面

在上述界面中，“服务器 IP”文本框是 textBox1，“监听端口”文本框是 textBox2，服务器状态文本框是 textBox3，“接收信息”文本框是 richTextBox1，“发送信息”文本框是 richTextBox2。下面开始编码。

(1) 添加引用。

```
using System.Net;
using System.Net.Sockets;
using System.Threading;
```

(2) 添加私有成员。

```
private IPAddress myIP=IPAddress.Parse("127.0.0.1");
private IPEndPoint MyServer;
private Socket sock;
private bool check=true;
private Socket accSock;
```

(3) “开始监听”按钮的 Click 事件的代码。

```
private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        myIP = IPAddress.Parse(textBox1.Text);
    }
    catch{MessageBox.Show("您输入的 IP 地址格式不正确, 请重新输入!");}
    try
    [
        Thread thread=new Thread(new ThreadStart(accp));
        thread.Start();
    ]
    catch(Exception ee){textBox3.AppendText(ee.Message);}
}
```

(4) 线程同步方法 accp 的代码。

```
private void accp()
{
    MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
    sock =new Socket(AddressFamily.InterNetwork,
                      SocketType.Stream,ProtocolType.Tcp);
    sock.Bind(MyServer);
    sock.Listen(50);
    textBox3.AppendText("主机"+textBox1.Text+"端口 "+textBox2.Text
                        +"开始监听.....\r\n");
    accSock=sock.Accept();
    if(accSock.Connected)
    {
        textBox3.AppendText("与客户建立连接。");
        while(check)
```

```
{  
    Byte[] Rec=new Byte[64];  
    NetworkStream netStream=new NetworkStream(accSock);  
    netStream.Read(Rec, 0, Rec.Length);  
    string RecMessage=  
        System.Text.Encoding.BigEndianUnicode.GetString(Rec);  
    richTextBox1.AppendText(RecMessage + "\r\n");  
}  
}  
}  
}
```

(5) “发送信息”按钮 Click 事件代码。

```
private void button2_Click(object sender, System.EventArgs e)  
{  
    try  
    {  
        Byte[] sendByte=new Byte[64];  
        string send=richTextBox2.Text + "\r\n";  
        NetworkStream netStream=new NetworkStream(accSock);  
        sendByte=System.Text.Encoding.BigEndianUnicode.GetBytes  
            (send.ToCharArray());  
        netStream.Write(sendByte, 0, sendByte.Length);  
    }  
    catch(MessageBox.Show("连接尚未建立！无法发送！"));  
}
```

(6) “停止监听”按钮的 Click 事件代码。

```
private void button3_Click(object sender, System.EventArgs e)  
{  
    try  
    {  
        sck.Close();  
        textBox3.AppendText("主机" + textBox1.Text + "端口" + textBox2.Text  
            + "监听停止！\r\n");  
    }  
    catch(MessageBox.Show("监听尚未开始，关闭无效！"));  
}
```

到此为止，本程序开发完毕，这是一个基础服务器，在此基础上加以改造，可以开发实用的网络服务器。

2.1.7 重要改进

不知读者发现了没有，上述服务器有一个漏洞，客户端一旦断开连接，就再也连接不上了，其实这是程序设计的问题，稍加改造就可以解决这一问题。问题出在代码

“accSock=sock.Accept();”位于循环语句之外，只要把它放到循环语句之内，一切就解决了。看下面代码：

```
private void accp()
{
    MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
    sock =new Socket(AddressFamily.InterNetwork,
                      SocketType.Stream,ProtocolType.Tcp);
    sock.Bind(MyServer);
    sock.Listen(50);
    textBox3.AppendText("主机"+textBox1.Text+"端口 "+textBox2.Text
        +"开始监听.....\r\n");
    while(true)
    {
        accSock=sock.Accept();
        if(accSock.Connected)
        {
            textBox3.AppendText("与客户建立连接。");
            Thread thread=new Thread(new ThreadStart(round));
            thread.Start();
        }
    }
}
private void round()
{
    while(true)
    {
        Byte[] Rec=new Byte[64];
        NetworkStream netStream=new NetworkStream(accSock);
        netStream.Read(Rec,0,Rec.Length);
        string RecMessage=
            System.Text.Encoding.BigEndianUnicode.GetString(Rec);
        richTextBox1.AppendText(RecMessage+"\r\n");
    }
}
```

2.2 异步套接字服务器开发

本节首先介绍一下异步套接字服务器的主机绑定、端口监听、数据接收、数据发送的基础知识，然后利用 System.Net 名字空间下 Sockets 类开发一个套接字异步服务器实例。

2.2.1 端口绑定与监听

异步套接字服务器主机的绑定与端口的监听仍然要使用 Socket 类的 Bind 方法与 Listen 方法，这与同步套接字没有区别，这里不再赘述。需要注意的是，异步套接字服务器使用 BeginAccept 方法异步接收客户端的连接请求。BeginAccept 的原型为：

```
public IAsyncResult BeginAccept(
    AsyncCallback callback,
    object state
)
```

该方法有两个参数，第一个参数为异步回调方法，第二个参数为自定义对象。返回值是IAsyncResult类型值。下面演示一下该方法的用法。该方法的使用需要有一个自定义类，用于获取操作状态。这里首先添加一个自定义类。打开类视图，在项目名称上单击鼠标右键，在弹出的快捷菜单上单击菜单“添加”|“添加类”，打开“C#类向导”对话框。在类名的文本框里输入适当的类名称（本例为 StateObject），设置好其他属性，然后单击完成，系统自动添加如下代码：

```
using System;
namespace mySocket
{
    /// <summary>
    ///
    /// </summary>
    public class StateObject
    {
        public StateObject()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }
    }
}
```

将上述代码修改为：

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Text;

namespace mySocket
{
    /// <summary>
    ///
    /// </summary>
    public class StateObject
    {
        public Socket workSocket = null;
        public const int BufferSize = 1024;
        public byte[] buffer = new byte[BufferSize];
```

```

public StringBuilder sb = new StringBuilder();
public StateObject()
{
    //
    // TODO: 在此处添加构造函数逻辑
    //
}
}

```

自定义类完成后，就可以编写代码进行监听了。下列代码实现主机的绑定和端口的监听。

首先自定义几个私有对象：

```

private IPAddress myIP=IPAddress.Parse("127.0.0.1");
private IPEndPoint MyServer;
private Socket mySocket;
private Socket handler;
private static ManualResetEvent myReset = new ManualResetEvent(false);

```

然后开始绑定主机和监听端口：

```

try
{
    IPHostEntry myHost=new IPHostEntry();
    myHost=Dns.GetHostByName("//主机名称，如 aaa.bbb.com");
    string IPstring=myHost.AddressList[0].ToString();
    myIP = IPAddress.Parse(IPstring);
}
catch	MessageBox.Show("您输入的 IP 地址格式不正确，请重新输入！");
try
{
    MyServer=new IPEndPoint(myIP,Int32.Parse("//端口字符串，如 1234"));
    mySocket =new Socket(AddressFamily.InterNetwork,
        SocketType.Stream,ProtocolType.Tcp);
    mySocket.Bind(MyServer);
    mySocket.Listen(50);
    Thread thread=new Thread(new ThreadStart(target));
    thread.Start();
}
catch(Exception ee){ }

```

下面是线程 target 方法的代码：

```

private void target()
{
    while(true)
    {

```

```
//设为非终止  
myReset.Reset();  
mySocket.BeginAccept( new AsyncCallback(AcceptCallback),  
    mySocket );  
//阻塞当前线程，直到收到请求信号  
myReset.WaitOne();  
}  
}
```

下面是异步回调方法 AcceptCallback 的代码：

```
private void AcceptCallback(IAsyncResult ar)  
{  
    // 将事件设为终止  
    myReset.Set();  
    Socket listener = (Socket) ar.AsyncState;  
    handler = listener.EndAccept(ar);  
    // 获取状态  
    StateObject state = new StateObject();  
    state.workSocket = handler;  
    //下面代码发送数据并开始接收数据  
    try  
    {  
        byte[] byteData = System.Text.Encoding.BigEndianUnicode.GetBytes  
            ("已经准备好，请通话！" + "\n\r");  
        // Begin sending the data to the remote device.  
        handler.BeginSend(byteData, 0, byteData.Length, 0,  
            new AsyncCallback(SendCallback), handler);  
    }  
    catch(Exception ee){MessageBox.Show(ee.Message);}  
    Thread thread=new Thread(new ThreadStart(begReceive));  
    thread.Start();  
}
```

2.2.2 发送数据

发送数据可以用 Socket 类的 BeginSend 方法。BeginSend 方法原型如下：

```
public IAsyncResult BeginSend(  
    byte[] buffer,  
    int offset,  
    int size,  
    SocketFlags socketFlags,  
    AsyncCallback callback,  
    object state  
)
```

该方法共有六个参数，第一个参数是字节数组，第二个参数是开始发送的位置，第三个参数是发送的字节数，第四个参数是SocketFlags 值的按位组合，第五个参数是异步回调，第六个参数是自定义对象。下列代码演示该方法的用法：

```

...
try
{
    byte[] byteData = System.Text.Encoding.BigEndianUnicode.GetBytes
        ("已经准备好，请通话！" + "\n\r");
    // Begin sending the data to the remote device.
    handler.BeginSend(byteData, 0, byteData.Length, 0,
        new AsyncCallback(SendCallback), handler);
}
catch(Exception ee){MessageBox.Show(ee.Message);}
Thread thread=new Thread(new ThreadStart(begReceive));
thread.Start();
}

```

下面是异步回调方法 SendCallback 的代码：

```

private void SendCallback(IAsyncResult ar)
{
    try
    {
        handler = (Socket) ar.AsyncState;
        int bytesSent = handler.EndSend(ar);
    }
    catch(Exception eee)
    {
        MessageBox.Show(eee.Message);
    }
}

```

2.2.3 接收数据

接收数据可以用 Socket 类的 BeginReceive 方法，该方法的原型如下：

```

public IAsyncResult BeginReceive(
    byte[] buffer,
    int offset,
    int size,
    SocketFlags socketFlags,
    AsyncCallback callback,
    object state
)

```

该方法共有 6 个参数。第一个参数是字节数组，第二个参数是开始发送的位置，第三个参数是发送的字节数，第四个参数是 SocketFlags 值的按位组合，第五个参数是异步回调，第六个参数是自定义对象。下列代码演示该方法的用法：

```
...
Thread thread=new Thread(new ThreadStart(begReceive));
thread.Start();
```

下面是线程 begReceive 方法的代码：

```
private void begReceive()
{
    StateObject state = new StateObject();
    state.workSocket = handler;
    handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
        new AsyncCallback(ReadCallback), state);
}
```

下面是异步回调方法 ReadCallback 的代码：

```
private void ReadCallback(IAsyncResult ar)
{
    StateObject state = (StateObject) ar.AsyncState;
    Socket tt = state.workSocket;
    // 结束读取并获取读取字节数
    int bytesRead = handler.EndReceive(ar);
    state.sb.Append(System.Text.Encoding.BigEndianUnicode.GetString(
        state.buffer, 0, bytesRead));
    string content = state.sb.ToString();
    // 清除 state.sb 内容，准备重新赋值
    state.sb.Remove(0, content.Length);
    richTextBox1.AppendText(content + "\r\n");
    // 重新开始读取数据
    tt.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
        new AsyncCallback(ReadCallback), state);
}
```

2.2.4 异步套接字基础服务器开发实例

下面开发一个异步套接字基础服务器。该程序使用 Socket 类的 Bind 方法绑定本地主机，使用 Listen 方法监听端口，使用 BeginAccept 接收连接请求，使用 BeginSend 方法发送数据，使用 BeginReceive 方法接收数据。为了避免程序等待，使用 Thread 类的 Start 方法实现线程同步。另外，为了能够收发中文，文本编码使用 BigEndianUnicode 吗。下面是该程序的界面（图 2-2）。

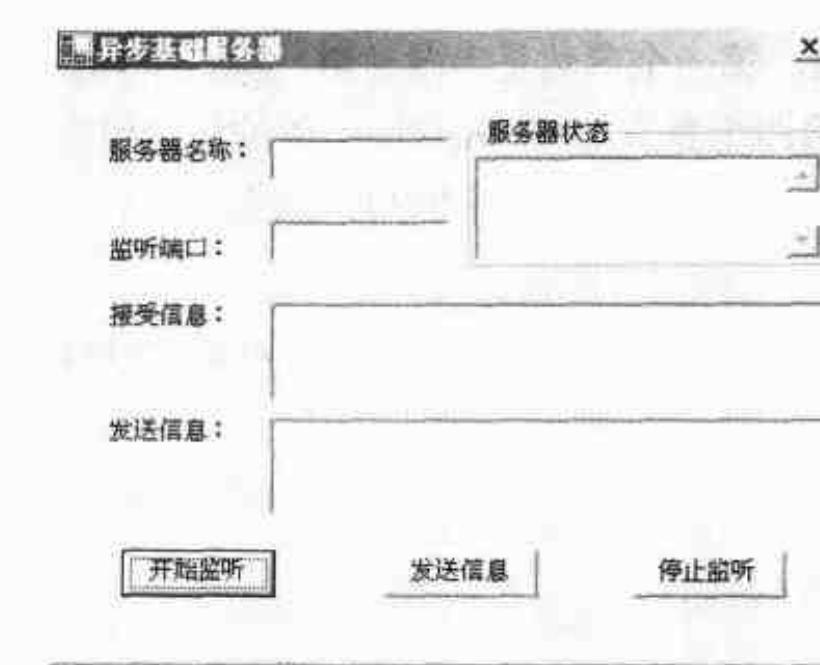


图 2-2 异步套接字基础服务器界面

在上述界面中，“服务器名称”文本框是 `textBox1`，“监听端口”文本框是 `textBox2`，服务器状态文本框是 `textBox3`，“接收信息”文本框是 `richTextBox1`，“发送信息”文本框是 `richTextBox2`。下面开始编码。

(1) 添加引用。

```
using System.Net;
using System.Net.Sockets;
using System.Threading;
```

(2) 添加私有成员。

```
private IPAddress myIP=IPAddress.Parse("127.0.0.1");
private IPEndPoint MyServer;
private Socket mySocket;
private Socket handler;
private static ManualResetEvent myReset = new ManualResetEvent(false);
```

(3) “开始监听”按钮的 Click 事件代码。

```
private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        IPHostEntry myHost=new IPHostEntry();
        myHost=Dns.GetHostByName(textBox1.Text);
        string IPstring=myHost.AddressList[0].ToString();
        myIP =IPAddress.Parse(IPstring);
    }
    catch{MessageBox.Show("您输入的 IP 地址格式不正确，请重新输入！");}
    try
    {
        MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
        mySocket =new Socket(AddressFamily.InterNetwork,
            SocketType.Stream,ProtocolType.Tcp);
        mySocket.Bind(MyServer);
        mySocket.Listen(50);
```

```
    textBox3.AppendText("主机"+textBox1.Text+"端口"+textBox2.Text  
        +"开始监听.....\r\n");  
    Thread thread=new Thread(new ThreadStart(target));  
    thread.Start();  
}  
catch(Exception ee){textBox3.AppendText(ee.Message+"\r\n");}  
}
```

线程同步方法 target 的代码:

```
private void target()  
{  
    while(true)  
    {  
        //设为非终止  
        myReset.Reset();  
        mySocket.BeginAccept( new AsyncCallback(AcceptCallback),  
            mySocket );  
        //阻塞当前线程，直到收到请求信号  
        myReset.WaitOne();  
    }  
}
```

异步回调方法 AcceptCallback 代码:

```
private void AcceptCallback(IAsyncResult ar)  
{  
    // 将事件设为终止  
    myReset.Set();  
    Socket listener = (Socket) ar.AsyncState;  
    handler = listener.EndAccept(ar);  
    // 获取状态  
    StateObject state = new StateObject();  
    state.workSocket = handler;  
    textBox3.AppendText("与客户建立连接。 \r\n");  
    try  
    {  
        byte[] byteData = System.Text.Encoding.BigEndianUnicode.GetBytes  
            ("已经准备好，请通话！ "+ "\n\r");  
        // 开始发送数据  
        handler.BeginSend(byteData, 0, byteData.Length, 0,  
            new AsyncCallback(SendCallback), handler);  
    }  
    catch(Exception ee){MessageBox.Show(ee.Message);}  
    Thread thread=new Thread(new ThreadStart(begReceive));  
    thread.Start();  
}
```

异步回调方法 **SendCallback** 如下：

```
private void SendCallback(IAsyncResult ar)
{
    try
    {
        handler = (Socket) ar.AsyncState;
        int bytesSent = handler.EndSend(ar);
    }
    catch (Exception eee)
    {
        MessageBox.Show(eee.Message);
    }
}
```

线程同步方法 **begReceive** 代码如下：

```
private void begReceive()
{
    StateObject state = new StateObject();
    state.workSocket = handler;
    handler.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
        new AsyncCallback(ReadCallback), state);
}
```

异步回调方法 **ReadCallback** 的代码如下：

```
private void ReadCallback(IAsyncResult ar)
{
    StateObject state = (StateObject) ar.AsyncState;
    Socket tt = state.workSocket;
    // 结束读取并获取读取字节数
    int bytesRead = handler.EndReceive(ar);
    state.sb.Append(System.Text.Encoding.BigEndianUnicode.GetString(
        state.buffer, 0, bytesRead));
    string content = state.sb.ToString();
    state.sb.Remove(0, content.Length);
    richTextBox1.AppendText(content + "\r\n");
    // 重新开始读取数据
    tt.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
        new AsyncCallback(ReadCallback), state);
}
```

(4) “发送信息”按钮的 Click 事件代码：

```
private void button2_Click(object sender, System.EventArgs e)
{
    try
```

```
        {
            byte[] byteData = System.Text.Encoding.BigEndianUnicode.GetBytes
                (richTextBox2.Text + "\n\r");
            //开始发送数据
            handler.BeginSend(byteData, 0, byteData.Length, 0,
                new AsyncCallback(SendCallback), handler);
        }
        catch (Exception ee) { MessageBox.Show(ee.Message); }
    }
```

(5) “停止监听”按钮的 Click 事件代码:

```
private void button2_Click(object sender, System.EventArgs e)
{
    try
    {
        mySocket.Close();
        textBox3.AppendText("主机" + textBox1.Text + "端口 " + textBox2.Text
            + "监听停止! \r\n");
    }
    catch { MessageBox.Show("监听尚未开始, 关闭无效!"); }
}
```

到此为止, 本程序开发完毕, 这是一个异步套接字基础服务器, 在此基础上加以改造, 可以开发实用的网络服务器。

2.3 TcpListener 基础服务器开发

本节首先介绍一下服务器端口监听、数据接收、数据发送的基础知识, 然后利用 System.Net 名字空间下 TcpListener 类开发一个服务器实例。

2.3.1 端口监听

TcpListener 服务器端口的监听要使用 TcpListener 类的 Start 方法和 AcceptSocket 方法。Start 方法用于开始监听, AcceptSocket 方法用于接收来自客户端的连接请求。

Start 方法原型为:

```
public void Start()
```

该方法没有参数, 也没有返回值。

AcceptSocket 的原型为:

```
public Socket AcceptSocket()
```

该方法没有参数, 返回值是套接字对象。

下列代码演示了上述两个方法的用法：

```
try
{
    listener=new TcpListener(1234); //参数是端口号
    listener.Start();
    Thread thread=new Thread(new ThreadStart(recieve));
    thread.Start();
}
catch(Exception ee){MessageBox.Show(ee.Message);}
}
```

下列是线程同步方法 receive 的代码：

```
private void recieve()
{
    mySock=listener.AcceptSocket();
    if(mySock.Connected)
    {
        while(!control)
        {
            netStream=new NetworkStream(mySock);
            byte[] messageByte=new Byte[64];
            netStream.Read(messageByte,0,messageByte.Length);
            string readMessage=System.Text.Encoding.BigEndianUnicode
                .GetString(messageByte);
            richTextBox1.AppendText(readMessage);
        }
    } //对应于 if(mySock.Connected) 的 “{”
}
```

2.3.2 发送数据与接收数据

发送数据与接收数据的方法与本章前两节介绍的方法相同，具体可参考本章 2.1.4、2.1.5、2.2.2 和 2.2.3 节，这里不再赘述。

2.3.3 基础服务器开发实例

下面开发一个 TcpListener 基础服务器，该程序使用 TcpListener 类的 Start 方法监听端口，使用 AcceptSocket 方法接收连接请求。使用 NetworkStream 类的 Read 方法读取数据，使用 Write 方法发送数据。为了避免程序等待，使用 Thread 类的 Start 方法实现线程同步。另外，为了能够收发中文，文本编码使用 BigEndianUnicode 码。下面是该程序的界面（图 2-3）。

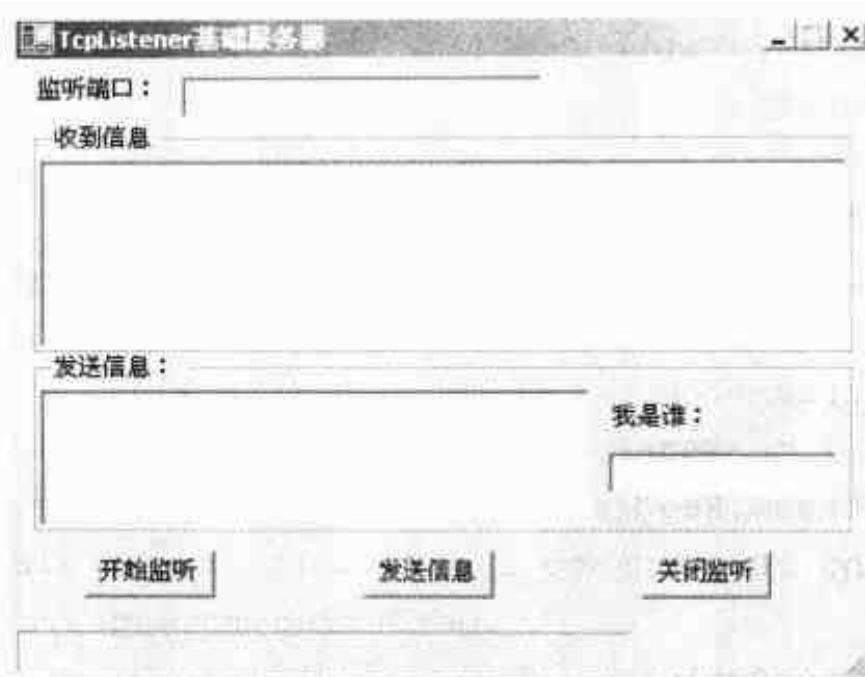


图 2-3 TcpListener 基础服务器界面

在上述界面中，“监听端口”文本框是 textBox1，“我是谁”文本框是 textBox2，“收到信息”文本框是 richTextBox1，“发送信息”文本框是 richTextBox2。另外，在窗体的下面有一个状态条 statusBar1，该控件的 Panels 属性为 statusBarPanel1。下面开始编码。

(1) 添加引用

```
using System.Net;
using System.Net.Sockets;
using System.Threading;
```

(2) 添加私有成员

```
private Socket mySock;
private TcpListener listener;
private NetworkStream netStream;
private bool control=false;
```

(3) “开始监听”按钮的 Click 事件的代码:

```
private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        listener=new TcpListener(Int32.Parse(textBox1.Text));
        listener.Start();
        statusBarPanel1.Text="现处于待机状态.....";
        Thread thread=new Thread(new ThreadStart(recieve));
        thread.Start();
    }
    catch(Exception ee){MessageBox.Show(ee.Message);}
}
```

(4) 线程同步方法 receive 的代码:

```
private void recieve()
{
```

```

mySock=listener.AcceptSocket();
if(mySock.Connected)
{
    statusBarPanel1.Text="与客户建立连接";
    while(!control)
    {
        netStream=new NetworkStream(mySock);
        byte[] messageByte=new Byte[64];
        netStream.Read(messageByte,0,messageByte.Length);
        string readMessage=System.Text.Encoding.BigEndianUnicode
            .GetString(messageByte);
        richTextBox1.AppendText(readMessage);
    }
}
//对应于 if(mySock.Connected) 的 “{”
}

```

(5) “发送信息”按钮 Click 事件的代码:

```

private void button2_Click(object sender, System.EventArgs e)
{
    netStream=new NetworkStream(mySock);
    string send(textBox2.Text+">>>"+richTextBox2.Text+"\r\n");
    byte[] messageByte=System.Text.Encoding.BigEndianUnicode
        .GetBytes(send.ToCharArray());
    netStream.Write(messageByte,0,messageByte.Length);
    netStream.Flush();
}

```

(6) “关闭监听”按钮的 Click 事件的代码:

```

private void button3_Click(object sender, System.EventArgs e)
{
    try{
        mySock.Close();
        listener.Stop();
    }
    catch{}
}

```

到此为止，本程序开发完毕，这是一个基础服务器，在此基础上加以改造，可以开发实用的网络服务器。

2.3.4 重要改进

同 2.1.7 的改进意见一样，为了可以使客户端断开连接后，重新连接上，将代码修改为如下所示：

```
private void receive()
{
    while(true)
    {
        mySock=listener.AcceptSocket();
        if(mySock.Connected)
        {
            statusBarPanel1.Text="与客户建立连接";
            Thread thread=new Thread(new ThreadStart(round));
            thread.Start();
        }
    }
}

private void round()
{
    while(true)
    {
        netStream=new NetworkStream(mySock);
        byte[] messageByte=new Byte[64];
        netStream.Read(messageByte,0,messageByte.Length);
        string readMessage=
            System.Text.Encoding.BigEndianUnicode.GetString(messageByte);
        richTextBox1.AppendText(readMessage);
    }
}
```

本章小结

本章详细介绍了同步套接字、异步套接字以及用 `TcpListener` 类开发基础服务器的方法。了解了这些方法后，可根据具体的应用层协议开发实用的服务器程序。

第3章 基础客户端开发

本章使用.NET 平台的同步套接字技术、异步套接字技术以及 TcpClient 类开发三个基础客户端，在这些客户端的基础上，稍加改造就可以成为适合商业需要的客户端程序。

3.1 同步套接字客户端开发

本节首先介绍一下同步套接字客户端与服务器的基础知识，然后利用 System.Net 名字空间下 Sockets 类开发一个套接字同步客户端实例。

3.1.1 建立与服务器的连接

建立与服务器的连接要使用 Socket 类的 Connect 方法。该方法定义原型如下：

```
public void Connect(  
    EndPoint remoteEP  
)
```

该方法只有一个参数，就是EndPoint 对象，如 IPEndPoint。IPEndPoint 在本书 1.1 节已经介绍过，这里不再赘述。下列代码演示如何连接远程服务器。

```
IPAddress myIP=IPAddress.Parse("111.111.111.111");  
IPEndPoint MyServer=new IPEndPoint(myIP,1234); // "1234" 是端口号  
sock =new Socket(AddressFamily.InterNetwork,SocketType.Stream,  
    ProtocolType.Tcp);  
sock.Connect(MyServer);
```

3.1.2 数据发送与接收

发送数据与接收数据的方法与第 2.1 节介绍的方法相同，具体可参考 2.1.4 和 2.1.5 节。

3.1.3 基础客户端开发实例

下面开发一个同步套接字基础客户端。该程序使用 Socket 类的 Connect 方法连接远程服务器，使用 NetworkStream 类的 Read 方法读取数据，使用 Write 方法发送数据。为了避免程序等待，使用 Thread 类的 Start 方法实现线程同步。另外，为了能够收发中文，文本编码使用 BigEndianUnicode 码。下面是该程序的界面（图 3-1）：

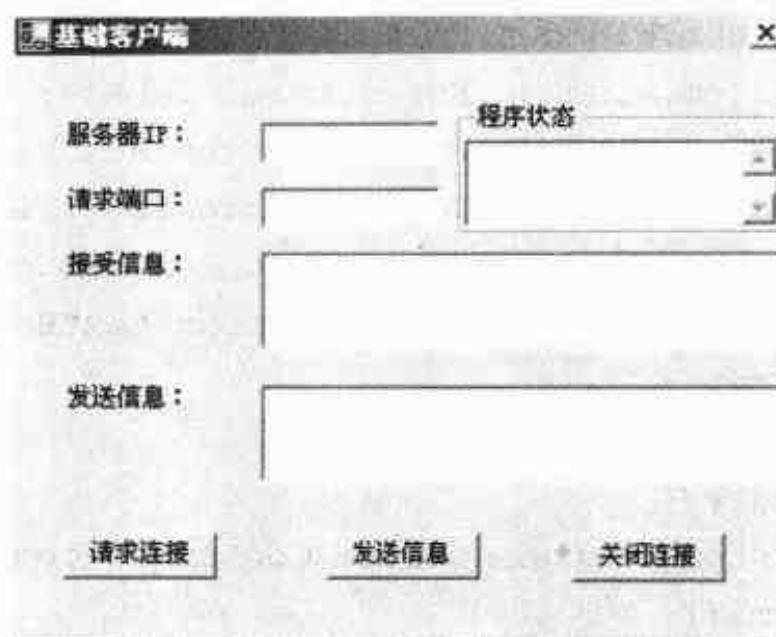


图 3-1 同步套接字基础客户端界面

在上述界面中，“服务器 IP”文本框是 textBox1，“请求端口”文本框是 textBox2，“程序状态”文本框是 textBox3，“接收信息”文本框是 richTextBox1，“发送信息”文本框是 richTextBox2。“请求连接”按钮是 button1，“发送信息”按钮是 button2，“关闭连接”按钮是 button3。下面是该程序的代码。

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace Sock_Conn
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Button button2;
        private IPAddress myIP=IPAddress.Parse("127.0.0.1");
        private IPEndPoint MyServer;
        private Socket connectSock;
        private bool check=true;
        private System.Windows.Forms.Button button3;
```

```
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.RichTextBox richTextBox1;
private System.Windows.Forms.RichTextBox richTextBox2;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.TextBox textBox3;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;
public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();
    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}
//下面是系统自动产生的代码，已删除
#region Windows Form Designer generated code
...
...
#endregion
//上面是系统自动产生的代码，已删除
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
```

```
        Application.Run(new Form1());
    }

    private void button1_Click(object sender, System.EventArgs e)
    {
        try
        {
            myIP = IPAddress.Parse(textBox1.Text);
        }
        catch { MessageBox.Show("您输入的 IP 地址格式不正确, 请重新输入!"); }

        try
        {
            MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
            connectSock =new Socket(AddressFamily.InterNetwork,
                SocketType.Stream,ProtocolType.Tcp);
            connectSock.Connect(MyServer);

            textBox3.AppendText("与主机"+textBox1.Text
                +"端口"+textBox2.Text+"连接成功! \r\n");
            Thread thread=new Thread(new ThreadStart(receive));
            thread.Start();
        }
        catch(Exception ee){MessageBox.Show(ee.Message);}
    }

    private void button3_Click(object sender, System.EventArgs e)
    {
        try
        {
            connectSock.Close();
            textBox3.AppendText("与主机"+textBox1.Text
                +"端口"+textBox2.Text+"断开连接! \r\n");
        }
        catch{MessageBox.Show("连接尚未建立, 断开无效!");}
    }

    private void button2_Click(object sender, System.EventArgs e)
    {
        try
        {
            Byte[] sendByte=new Byte[64];
            string send=richTextBox2.Text+"\r\n";
            NetworkStream netStream=new NetworkStream(connectSock);
            sendByte=System.Text.Encoding.BigEndianUnicode.GetBytes
                (send.ToCharArray());
            netStream.Write(sendByte,0,sendByte.Length);
            netStream.Flush();
        }
    }
}
```

```

        }
        catch{MessageBox.Show("连接尚未建立！无法发送！");}
    }
    private void receive()
    {
        while(check)
        {
            Byte[] Rec=new Byte[64];
            NetworkStream netStream=new NetworkStream(connectSock);
            netStream.Read(Rec,0,Rec.Length);
            string RecMessage=
                System.Text.Encoding.BigEndianUnicode.GetString(Rec);
            richTextBox1.AppendText(RecMessage+"\r\n");
        }
    }
}

```

到此为止，本程序开发完毕，这是一个基础客户端，在此基础上加以改造，可以开发实用的客户端程序。

3.1.4 演示

(1) 启动 2.1.6 节的基础服务器，输入服务器 IP 地址并设置适当的端口，然后单击“开始监听”按钮开始监听端口。

(2) 启动 3.1.3 节的实例程序，输入服务器 IP 地址和端口，然后单击“请求连接”按钮，建立与服务器的连接。

(3) 开始通话，服务器和客户端可以任意发送文本数据。

图 3-2 是服务器的运行情况。

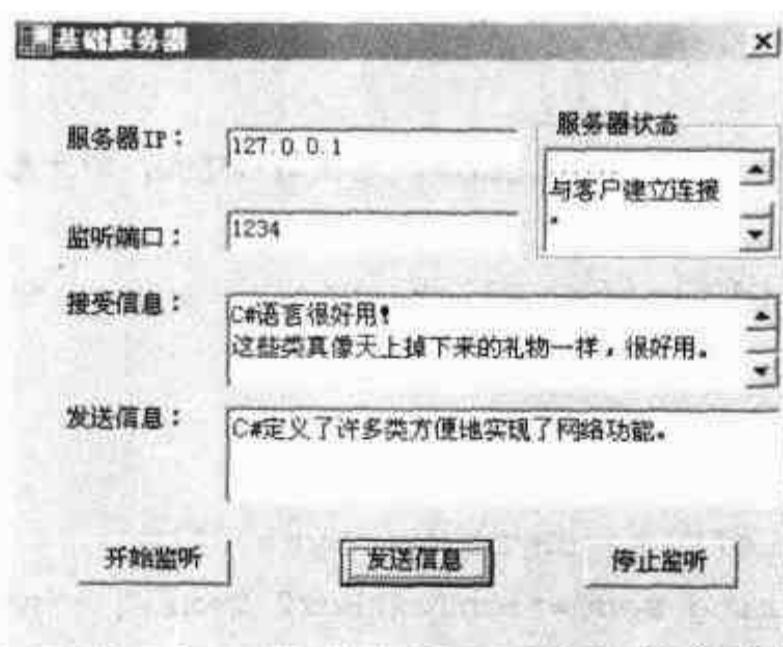


图 3-2 同步套接字服务器

图 3-3 是客户端的运行情况。

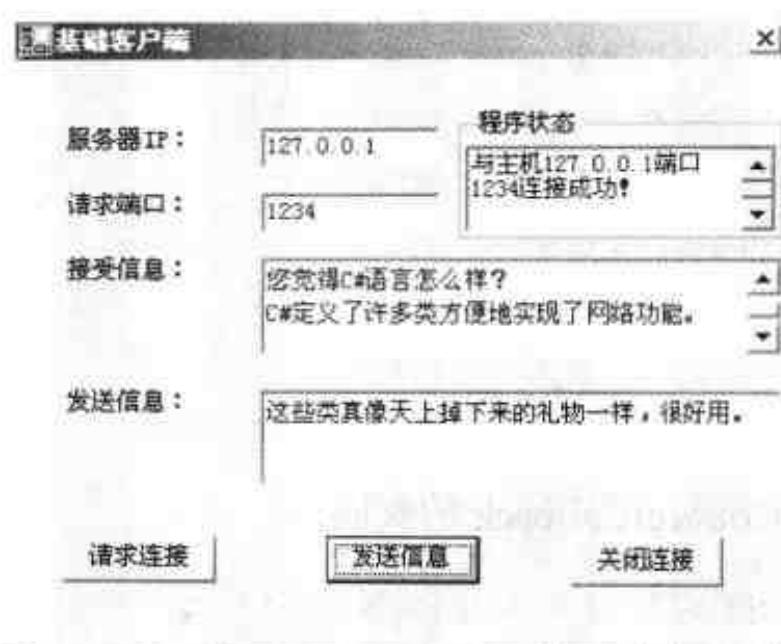


图 3-3 同步套接字客户端

3.2 异步套接字客户端开发

本节首先介绍一下异步套接字客户端与服务器的基础知识,然后利用System.Net名字空间下 Sockets 类开发一个套接字同步客户端实例。

3.2.1 建立与服务器的连接

建立与服务器的异步连接要使用 Socket 类的 BeginConnect 方法,该方法定义原型如下:

```
public IAsyncResult BeginConnect(
    EndPoint remoteEP,
    AsyncCallback callback,
    object state
)
```

该方法共三个参数,第一个参数是主机对象,如 IPEndPoint,第二个参数是异步回调方法,第三个参数是自定义对象。

下列代码演示该方法的用法。

```
try
{
    IPHostEntry myHost=new IPHostEntry();
    myHost=Dns.GetHostByName("aaa.bbb.com");
    string IPstring=myHost.AddressList[0].ToString();
    myIP =IPAddress.Parse(IPstring);
}

catch{MessageBox.Show("您输入的 IP 地址格式不正确, 请重新输入!");}
try
{
    MyServer=new IPEndPoint(myIP,1234);
```

```

mySocket = new Socket(AddressFamily.InterNetwork,
                      SocketType.Stream, ProtocolType.Tcp);
mySocket.BeginConnect( MyServer,
                      new AsyncCallback(ConnectCallback), mySocket);
connectReset.WaitOne();
}
catch(Exception ee){MessageBox.Show(ee.Message);}

```

下面是异步回调方法 ConnectCallback 的代码:

```

private void ConnectCallback(IAsyncResult ar)
{
    try
    {
        Socket client = (Socket) ar.AsyncState;
        client.EndConnect(ar);
        try
        {
            byte[] byteData =
                System.Text.Encoding.BigEndianUnicode.GetBytes
                ("准备完毕，可以通话！" + "\n\r");
            mySocket.BeginSend(byteData, 0, byteData.Length, 0,
                               new AsyncCallback(SendCallback), mySocket);
        }
        catch(Exception ee){MessageBox.Show(ee.Message);}
        Thread thread=new Thread(new ThreadStart(target));
        thread.Start();
        //设置事件终止
        connectReset.Set();
    }
    catch {}}
}

```

3.2.2 数据发送与接收

发送数据与接收数据的方法与第 2.2 节介绍的方法相同，具体可参考 2.2.2 和 2.2.3 节。

3.2.3 异步套接字操作基础客户端开发实例

下面开发一个异步套接字基础客户端。该程序使用 `Socket` 类的 `BeginConnect` 方法连接远程服务器，使用 `BeginReceive` 方法读取数据，使用 `BeginSend` 方法发送数据。为了避免程序等待，使用 `Thread` 类的 `Start` 方法实现线程同步。另外，为了能够收发中文，文本编码使用 `BigEndianUnicode` 码。下面是该程序的界面（图 3-4）：

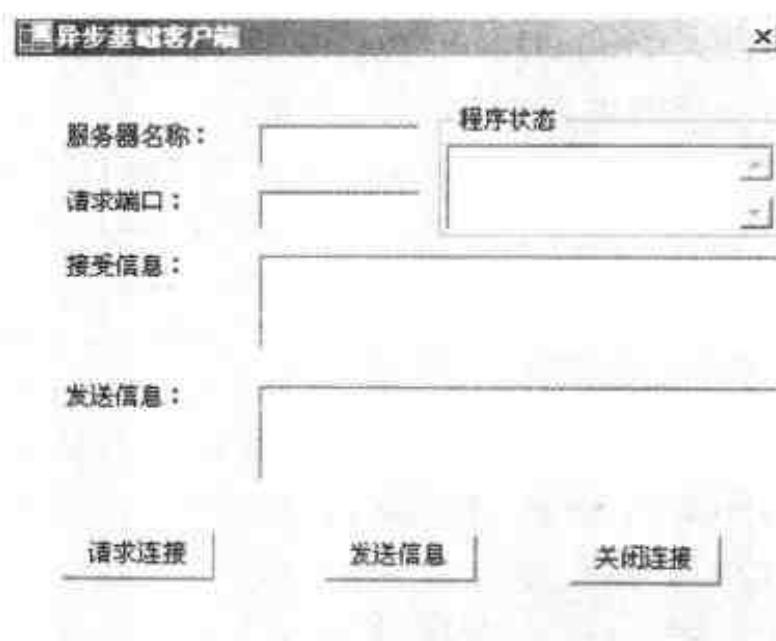


图 3-4 异步套接字客户端

在上述界面中，“服务器名称”文本框是 textBox1，“请求端口”文本框是 textBox2，“程序状态”文本框是 textBox3，“接收信息”文本框是 richTextBox1，“发送信息”文本框是 richTextBox2。“请求连接”按钮是 button1，“发送信息”按钮是 button2，“关闭连接”按钮是 button3。下面是该程序的代码。

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace Sock_Conn
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Button button2;
        private IPAddress myIP=IPAddress.Parse("127.0.0.1");
        private IPEndPoint MyServer;
        private Socket mySocket;
        private static ManualResetEvent connectReset =
            new ManualResetEvent(false);
```

```
private static ManualResetEvent sendReset =
    new ManualResetEvent(false);
private System.Windows.Forms.Button button3;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.RichTextBox richTextBox1;
private System.Windows.Forms.RichTextBox richTextBox2;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.TextBox textBox3;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;

public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();

    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
#endregion
//上面系统自动产生的代码已经删除
```

```
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        IPHostEntry myHost=new IPHostEntry();
        myHost=Dns.GetHostByName(textBox1.Text);
        string IPstring=myHost.AddressList[0].ToString();
        myIP =IPAddress.Parse(IPstring);
    }
    catch{MessageBox.Show("您输入的IP地址格式不正确，请重新输入！");}
    try
    {
        MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
        mySocket =new Socket(AddressFamily.InterNetwork,
            SocketType.Stream,ProtocolType.Tcp);
        mySocket.BeginConnect( MyServer,
            new AsyncCallback(ConnectCallback), mySocket);
        connectReset.WaitOne();
    }
    catch(Exception ee){MessageBox.Show(ee.Message);}
}
private void ConnectCallback(IAsyncResult ar)
{
    try
    {
        Socket client = (Socket) ar.AsyncState;
        client.EndConnect(ar);
        try
        {
            byte[] byteData =
                System.Text.Encoding.BigEndianUnicode.GetBytes
                ("准备完毕，可以通话！\r\n");
            mySocket.BeginSend(byteData, 0, byteData.Length, 0,
                new AsyncCallback(SendCallback), mySocket);
        }
        catch(Exception ee){MessageBox.Show(ee.Message);}
        textBox3.AppendText("与主机"+textBox1.Text)
```

```
        +"端口 "+textBox2.Text+"建立连接! \r\n");
        Thread thread=new Thread(new ThreadStart(target));
        thread.Start();
        //设置事件终止
        connectReset.Set();
    }
    catch
    {
    }

}

private void button3_Click(object sender, System.EventArgs e)
{
    try
    {
        mySocket.Close();
        textBox3.AppendText("与主机"+textBox1.Text
        +"端口 "+textBox2.Text+"断开连接! \r\n");
    }
    catch(MessageBox.Show("连接尚未建立，断开无效！"));
}

private void button2_Click(object sender, System.EventArgs e)
{
    byte[] byteData =
        System.Text.Encoding.BigEndianUnicode.GetBytes
        (richTextBox2.Text);
    mySocket.BeginSend(byteData, 0, byteData.Length, 0,
        new AsyncCallback(SendCallback), mySocket);
}

private void SendCallback(IAsyncResult ar)
{
    try
    {
        Socket client = (Socket) ar.AsyncState;
        //设为终止
        sendReset.Set();
    }
    catch (Exception e)
    {
        Console.WriteLine(e.ToString());
    }
}
```

```
private void target()
{
    try
    {
        StateObject state = new StateObject();
        state.workSocket = mySocket;
        mySocket.BeginReceive( state.buffer, 0,
            StateObject.BufferSize, 0,
            new AsyncCallback(ReceiveCallback), state );
    }
    catch( Exception ee )
    {
        MessageBox.Show(ee.Message);
    }
}

private void ReceiveCallback( IAsyncResult ar )
{
    try
    {
        StateObject state = (StateObject) ar.AsyncState;
        Socket client = state.workSocket;
        int bytesRead = client.EndReceive(ar);
        state.sb.Append(System.Text.Encoding.BigEndianUnicode
            .GetString(state.buffer, 0, bytesRead));
        string aa=state.sb.ToString();
        state.sb.Remove(0,aa.Length);
        richTextBox1.AppendText(aa+"\r\n");
        client.BeginReceive(state.buffer, 0, StateObject.BufferSize
            , 0, new AsyncCallback(ReceiveCallback), state );
    }
    catch {}
}
}
```

3.2.4 演示

- (1) 启动 2.2.4 节的程序，输入服务器名称和适当的端口，单击“开始监听”按钮，启动服务器。
- (2) 启动 3.2.3 节的程序，输入适当的服务器端口和名称，然后单击“请求连接”按钮，建立与服务器的连接。
- (3) 开始通话。图 3-5 是服务器的运行情况。

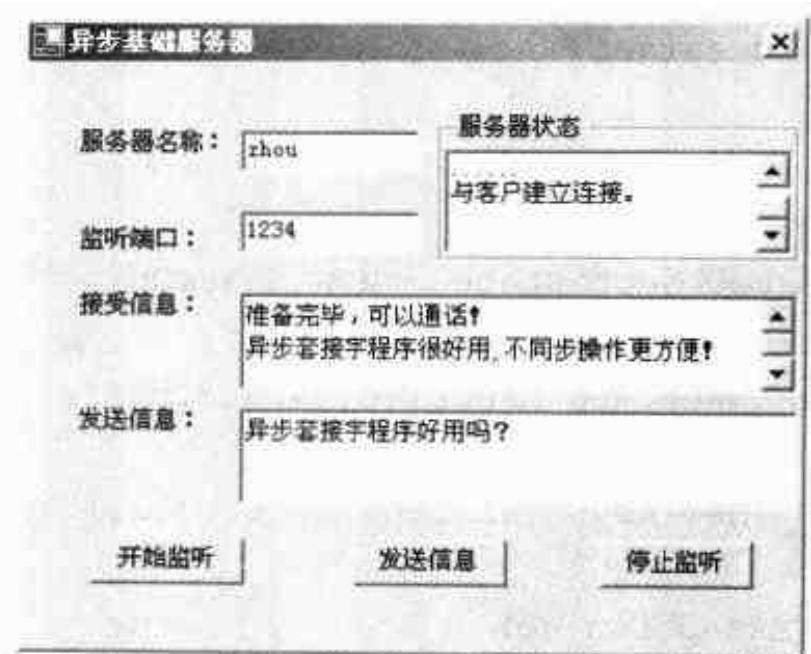


图 3-5 异步服务器已连接

图 3-6 是客户端的运行情况。

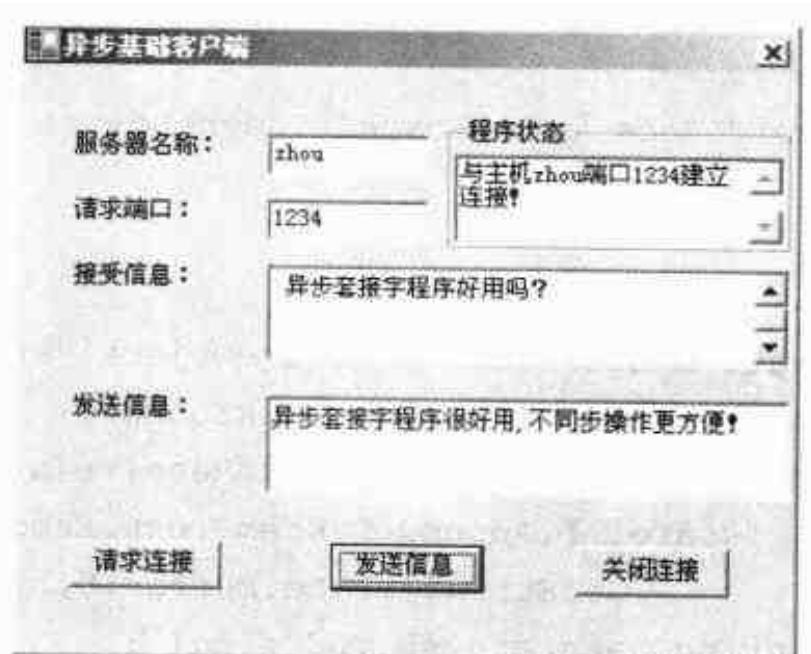


图 3-6 异步客户端已连接

3.3 TcpClient 基础客户端开发

本节首先介绍如何用 System.Net 命名空间下的 TcpClient 类与服务器建立连接，然后利用 TcpClient 类开发一个客户端实例。

3.3.1 建立连接

在 System.Net 命名空间下，TcpClient 类封装了客户端的功能。用 TcpClient 类开发的客户端，可以有两种方法实现与服务器的连接。第一种方法是使用 TcpClient 类的构造方法，下面两个构造方法均可自动实现与服务器的连接。

原型一：

```
public TcpClient(
    IPEndPoint localEP
)
```

在上述方法中，只要指定特定的 IPPEndPoint 对象，就可与该对象建立连接。下列代码演示与主机“111.111.111.111”的 12345 端口建立连接的过程：

```
IPAddress myIP = IPAddress.Parse("111.111.111.111");
IPEndPoint MyServer=new IPEndPoint(myIP,12345);
TcpClient myClient=new TcpClient(MyServer);
```

原型二：

```
public TcpClient(
    string hostname,
    int port
)
```

该方法有两个参数，第一个参数是主机名称，第二个参数是端口号。下列代码演示该方法的用法。

```
TcpClient myClient=new TcpClient("www.263.net", "80");
```

除了 TcpClient 类的构造方法可以连接主机外，该类的 Connect 方法也可与服务器建立连接。

原型一：

```
public void Connect(
    IPEndPoint remoteEP
)
```

该方法只有一个参数，就是 IPPEndPoint 对象。下列代码演示该方法的用法：

```
TcpClient myClient=new TcpClient();
IPAddress myIP = IPAddress.Parse("111.111.111.111");
IPEndPoint MyServer=new IPEndPoint(myIP,12345);
client.Connect(MyServer);
```

原型二：

```
public void Connect(
    IPAddress address,
    int port
)
```

该方法有两个参数，第一个参数是 IP 地址，第二个参数是端口号。下列代码演示该方法的用法：

```
IPAddress myIP = IPAddress.Parse("111.111.111.111");
TcpClient myClient=new TcpClient();
myClient.Connect(myIP,12345);
```

原型三：

```
public void Connect(
    string hostname,
```

```

    int port
)

```

该方法有三个参数，第一个参数是主机名称，第二个参数是端口号。下列代码演示该方法的用法。

```

TcpClient myClient=new TcpClient();
myClient.Connect( "www.aaa.bbb" ,12345);

```

3.3.2 发送数据与接收数据

发送数据与接收数据的方法与第 2 章前两节介绍的方法相同，具体可参考本章 2.1.4、2.1.5、2.2.2 和 2.2.3 节，这里不再赘述。

3.3.3 基础客户端开发实例

下面开发一个 TcpClient 基础客户端。该程序使用 TcpClient 类的 Connect 方法与服务器建立连接，使用 NetworkStream 类的 Read 方法读取数据，使用 Write 方法发送数据。为了避免程序等待，使用 Thread 类的 Start 方法实现线程同步。另外，为了能够收发中文，文本编码使用 BigEndianUnicode 吗。下面是该程序的界面（图 3-7）：

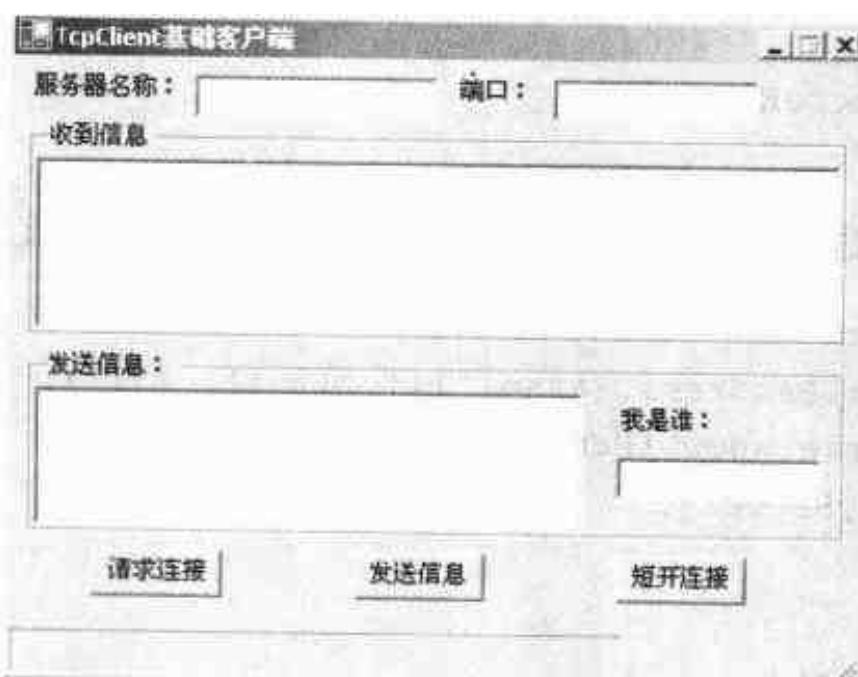


图 3-7 TcpClient 客户端

在上述界面中，“监听端口”文本框是 textBox1，“我是谁”文本框是 textBox2，“收到信息”文本框是 richTextBox1，“发送信息”文本框是 richTextBox2。“请求连接”按钮是 button1，“发送信息”按钮是 button2，“关闭连接”按钮是 button3。另外，在窗体的下面有一个状态条 statusBar1，该控件的 Panels 属性为 statusBarPanel1。下面开始编码。

```

using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

```

```
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace 网络电话
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.RichTextBox richTextBox1;

        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Label label1;
        private System.ComponentModel.IContainer components;
        private TcpClient client;
        private NetworkStream netStream;
        private bool control=false;
        private System.Windows.Forms.StatusBar statusBar1;
        private System.Windows.Forms.StatusBarPanel statusBarPanel1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.RichTextBox richTextBox2;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Label label2;
        public Form1()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();

            //
            // TODO: Add any constructor code after InitializeComponent call
            //
        }

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
```

```
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        int port=Int32.Parse(textBox2.Text);
        client =new TcpClient();
        client.Connect(textBox1.Text,port);
        statusBarPanel1.Text="可以通话";
        Thread thread1=new Thread(new ThreadStart(receive));
        thread1.Start();
    catch{MessageBox.Show("无法连接!");}
}
private void receive(){
    while(!control)
    {
        netStream=client.GetStream();
        byte[] messsageByte=new Byte[64];
        netStream.Read(messsageByte,0,messsageByte.Length);
        string readNessage=System.Text.Encoding.BigEndianUnicode
```

```
.GetString(messsageByte);
    richTextBox1.AppendText(readNessage);
}
}

private void button2_Click(object sender, System.EventArgs e)
{
    netStream=client.GetStream();
    string send=textBox3.Text+">>>>"+richTextBox2.Text+"\r\n";
    byte[] messsageByte=System.Text.Encoding.BigEndianUnicode
        .GetBytes(send.ToCharArray());
    netStream.Write(messsageByte,0,messsageByte.Length);
    netStream.Flush();
}

private void button3_Click(object sender, System.EventArgs e)
{
    try{
        client.Close();
    }
    catch{}
}
}
```

到此为止，本程序开发完毕，这是一个基础服务器，在此基础上加以改造，可以开发实用的网络服务器。

3.3.4 演示

(1) 启动 2.3.3 的程序，输入适当的端口，单击“开始监听”按钮，启动服务器。

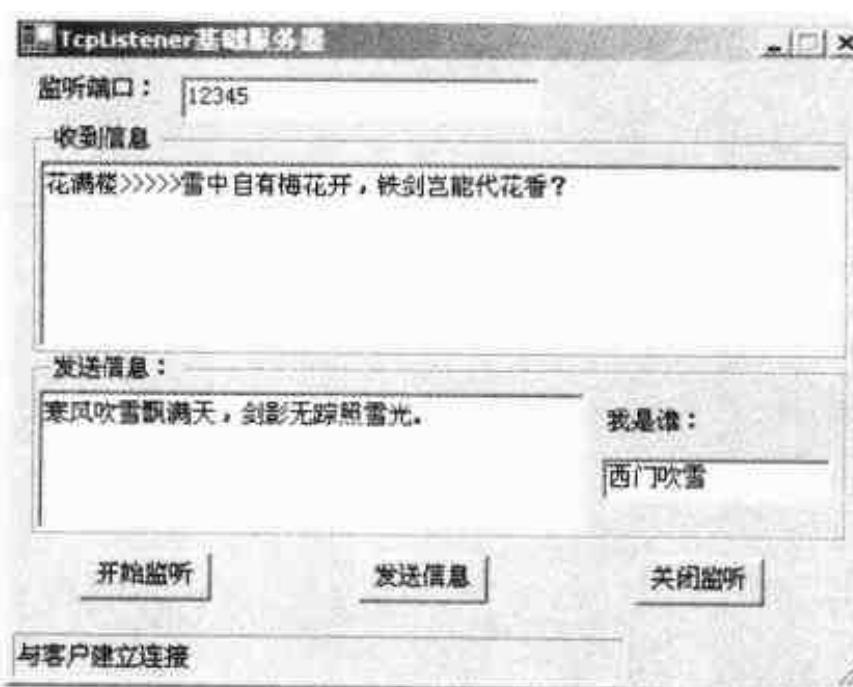


图 3-8 TcpClient 服务器已连接

(2) 启动 3.3.3 的程序，输入适当的服务器名称和端口，然后单击“请求连接”按钮，建立与服务器的连接。

(3) 开始通话。图 3-8 是服务器的运行情况。

图 3-9 是客户端运行情况。

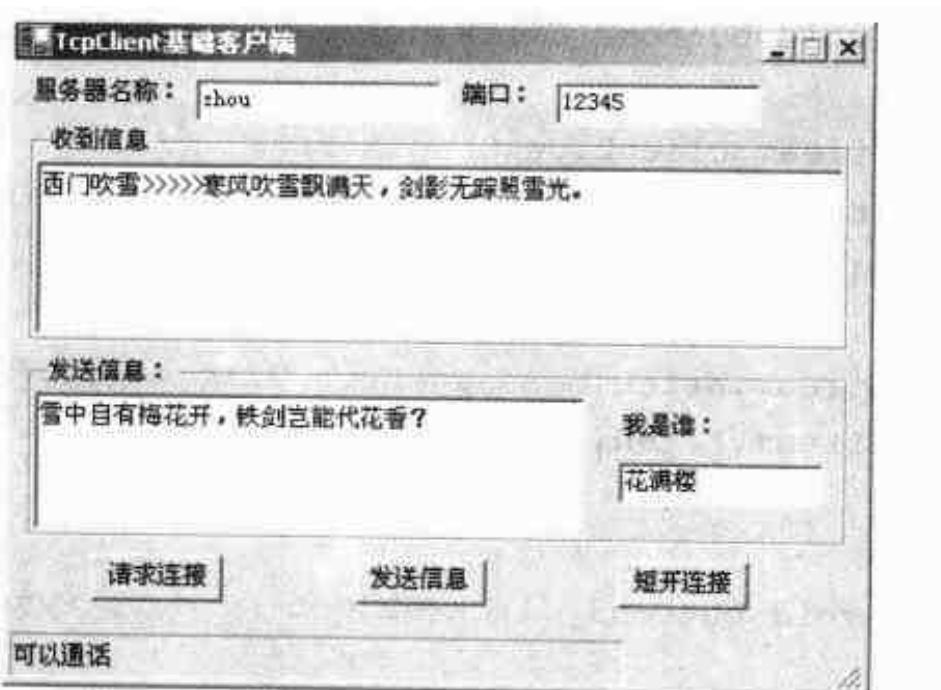


图 3-9 TcpClient 客户端已连接

本章小结

本章详细介绍了同步套接字、异步套接字以及用 `TcpClient` 类开发基础客户端的方法。了解了这些方法后，可根据具体的应用层协议开发实用的客户端程序。

第4章 FTP 协议开发

FTP 是英文 File Transfer Protocol 的缩写，意为文件传输协议，其默认端口号是 21。本章首先介绍 FTP 协议规范，然后利用 TcpListener 类和 TcpClient 类开发一个 FTP 服务器和客户端。

4.1 FTP 协议规范

要开发 FTP 协议程序，必须首先了解 FTP 协议规范，本节重点介绍一下 FTP 命令，了解这些命令后，开发 FTP 程序并不会是一件难事。

4.1.1 FTP 命令格式

下面是 FTP 协议常用命令的格式，其中 `username` 代表用户名，`password` 代表口令，`pathname` 代表路径名，`host-port` 代表主机端口，`account-information` 代表账户信息，`typecode` 代表类型代码，`decimal-integer` 代表十进制整数，`marker` 代表标记，`string` 代表字符串，`<` 代表必须带的参数，`[]` 代表可以选的参数，`SP` 代表空格，`CRLF` 代表回车换行符，即 C# 语法里的“`\r\n`”。FTP 每一个命令都以回车换行符结束，在 C# 中，只要遇到“`\r\n`”就认为命令行结束。

```
USER <SP> <username> <CRLF>
PASS <SP> <password> <CRLF>
ACCT <SP> <account-information> <CRLF>
CWD <SP> <pathname> <CRLF>
CDUP <CRLF>
SMNT <SP> <pathname> <CRLF>
QUIT <CRLF>
REIN <CRLF>
PORT <SP> <host-port> <CRLF>
PASV <CRLF>
TYPE <SP> <type-code> <CRLF>
STRU <SP> <structure-code> <CRLF>
MODE <SP> <mode-code> <CRLF>
RETR <SP> <pathname> <CRLF>
STOR <SP> <pathname> <CRLF>
STOU <CRLF>
APPE <SP> <pathname> <CRLF>
ALLO <SP> <decimal-integer>
```

```
[<SP> R <SP> <decimal-integer>] <CRLF>
REST <SP> <marker> <CRLF>
RNFR <SP> <pathname> <CRLF>
RNTO <SP> <pathname> <CRLF>
ABOR <CRLF>
DELE <SP> <pathname> <CRLF>
RMD <SP> <pathname> <CRLF>
MKD <SP> <pathname> <CRLF>
PWD <CRLF>
LIST [<SP> <pathname>] <CRLF>
NLST [<SP> <pathname>] <CRLF>
SITE <SP> <string> <CRLF>
SYST <CRLF>
STAT [<SP> <pathname>] <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>
```

4.1.2 FTP 命令参数

下面是 FTP 常用命令的参数格式：

<username> 为 <string>
<password> 为 <string>
<account-information> 为 <string>
<string> 为 <char> 或 <string>
<char> 为 除<CR>和<LF>外的所有 ASCII 字符，即除了回车换行符以外所有的 ASCII 码
<marker> 为 <pr-string>
<pr-string> 为 <pr-char> 或 <pr-string>，即打印字符或字符串
<pr-char> 为 可打印 ASCII 字符，从 33 到 126
<byte-size> 为 <number>
<host-port> 为 <host-number>, <port-number>
<host-number> 为 <number>, <number>, <number>, <number>
<port-number> 为 <number>, <number>
<number> 为 从 1 到 255 的十进制整数

<form-code> 为 N 或 T 或 C，其中 N 代表 NON PRINT，即非打印格式，T 代表 TELNET 格式，C 代表 CONTROL，即打印机控制格式，blank 表示向下移动 1 行；“0”表示向下移动 2 行；“1”表示移动至下一页；“+”表示不移动。

<type-code> 为 A [<sp> <form-code>] 或 E [<sp> <form-code>] 或 I 或 L <sp> <byte-size>，其中 A 代表 ASCII，E 代表 EBCDIC，I 代表 IMAGE，L 代表本地字节大小。

<structure-code> 为 F 或 R 或 P，其中 F 表示文件，R 表示记录，P 表示页。

<mode-code> 为 S 或 B 或 C，其中 S 代表流，B 代表块，C 代表压缩。

<pathname> 为 <string>

<decimal-integer> 为 任何十进制整数

4.1.3 FTP 命令

- **USER** (用户名)

参数是表明用户身份的 Telnet 串。参数是访问服务器必需的，此命令通常在连接后首先发出，个别主机要求账户和口令。服务器可以在任何时间接收新的 USER 命令以改变访问控制或账户信息，重新开始登录过程。

- **ACCT** (账号)

参数是表明用户账户的 Telnet 串。此命令不需要与 USER 相关，一些站点可能需要账户用于登录，另一些可以限制账户的权限。在限制访问权限的情况下，该命令可在任何时候发送。应答的不同可以区别不同的情况：当登录需要账户信息时，对 PASS 命令的响应是 332；如果不需要账户信息，对 PASS 的响应是 230；如果在以后需要账户信息，服务器会返回 332 或 532。

- **PASS** (口令)

参数是标记用户口令的 Telnet 串。该命令紧跟 USER 命令，在一些站点是必需的。

- **CWD** (改变工作目录)

该命令使用户可以在不同的目录或数据集下工作而不用改变它的登录或账户信息。传输参数也不变。参数一般是目录名或与系统相关的文件集合。

- **CDUP** (回到上一层目录)

该命令要求系统回到上一层目录。

- **SMNT** (结构加载)

该命令使用户在不改变登录或账户信息的情况下加载另一个文件系统数据结构。传输参数也不变。参数是文件目录或与系统相关的文件集合。

- **REIN** (重新初始化)

该命令终止 USER，重置所有参数，可以再次开始 USER 命令。

- **QUIT** (退出登录)

该命令终止 USER，如果没有数据传输，服务器关闭控制连接；如果有数据传输，在得到传输响应后服务器关闭控制连接。如果用户进程正在向不同的 USER 传输数据，不希望对每个 USER 关闭然后再打开，可以使用 REIN。

- **SITE** (站点参数)

该命令用于获取或提供站点信息。

- **SYST** (系统)

该命令用于获取服务器上运行的操作系统。

- **STAT** (状态)

该命令用于获取连接状态，它可以在文件传送过程中发送，服务器返回操作进行的状态。也可以在文件传送之间发送，这时命令有参数，参数是路径名，服务器以文件名或与说明相关的属性返回；如没有参数，服务器返回服务器 FTP 进程的状态信息，包括传输参数的当前值和连接状态。

- **HELP** (帮助)

该命令用于获取帮助信息，响应类型是 211 或 214。用户使用 USER 命令前使用该命令可以获取具体服务器的特殊信息。

- NOOP (等待)

此命令不产生什么实际动作，它仅使服务器返回 OK。

- LIST (列表)

该命令用于获取文件列表。如果路径指定一个目录或许多文件，返回指定路径下的文件列表。如果路径名指定一个文件，服务器返回文件的当前信息，参数为空表示用户当前的工作目录或默认目录。数据传输在 ASCII 或 EBCDIC 下进行，用户必须确认这一点，否则可能得到一堆乱码。

- NLST (名字列表)

该命令用于将服务器的目录表名传送到用户，路径名应指定目录，空参数指当前目录。服务器返回文件名数据流，以 ASCII 或 EBCDIC 形式传送，并以<CRLF>或<NL>分隔。这里返回的信息有时可以供程序进行进一步处理。

- MKD (创建目录)

该命令在指定路径下创建新目录，参数是标示特定目录的字符串。

- PWD (打印工作目录)

该命令用于请求服务器返回当前工作目录，参数是标示特定目录的字符串。

- RMD (删除目录)

该命令用于删除目录指定目录，参数是目录字符串，参数是标示特定目录的字符串。

- RETR (获得文件)

该命令用于请求服务器将指定路径内的文件复本到服务器或客户端，即下载。

- STOR (保存)

该命令用于向服务器上传文件。如果文件已存在，原文件将被覆盖；如果文件不存在，则新建文件。

- STOU (惟一保存)

该命令和 STOR 相似，该命令要求在特定目录下的文件名是惟一的。

- APPE (附加)

该命令与 STOR 的功能相似，但是如果文件在指定路径内已存在，则把数据附加到原文件尾部；如果不存在，则新建文件。

- ALLO (分配)

该命令用于特定主机上为新传送的文件分配足够的存储空间。

- REST (重新开始)

该命令重新开始特定命令，参数表示服务器要重新开始的位置。该命令并不传送文件，而是忽略指定点后的数据，该命令后应该跟其他要求文件传输的 FTP 命令。

- RNFR (重命名)

该命令用于重新命名文件，后面跟 rename to 指定新的文件名。

- RNTO (重命名为)

该命令和 RNFR 命令共同完成对文件的重命名，紧跟在 RNFR 之后。

- ABOR (放弃)

该命令中止特定的 FTP 命令和与之相关的数据传输。如果先前的操作已经完成，则没有动作，返回 226。如果没有完成，返回 426，然后再返回 226。关闭控制连接，数据连接不关闭。

- DELE (删除)

该命令用于删除指定路径下的文件。

- PORT (数据端口)

该命令用于指定数据传输的端口，因为许多服务器同时可以打开多个端口。参数是端口号，通常情况下对此不需要命令响应。

- PASV (被动)

该命令使服务器在指定的数据端口进行侦听，进入被动接收请求的状态。参数是主机和端口地址。

- TYPE (表示类型)

该命令用于指定文件类型。参数是类型标示符。有些类型需要第二个参数，第一个参数由单个 Telnet 字符定义，第二个参数是十进制整数，指定字节大小，参数间以<SP>分隔。

- STRU (文件结构)

该命令用于指示文件结构，参数是一个 Telnet 字符代码，F 代表文件，R 代表记录结构，P 代表页结构。

- MODE (传输模式)

该命令用于指定传输模式，参数是一个 Telnet 字符代码，S 代表流，B 代表块，C 代表压缩。

4.1.4 FTP 应答

表 4-1 列出 FTP 应答的性质、状态码及其意义。

表 4-1 FTP 应答的性质、状态码及其意义

性 质	状态码	意 义
正在准备	120	服务在 xxx 分钟内准备好
	125	准备传送
命令成功	110	重新启动标记应答
	150	文件状态良好，打开数据连接
	200	命令成功
	220	对新用户服务准备好
	221	服务关闭控制连接，可以退出登录
	226	关闭数据连接，请求的文件操作成功
	230	用户登录
	250	请求的文件操作完成
	257	创建目录
当前状态	211	系统状态或系统帮助响应
	212	目录状态
	213	文件状态
	214	帮助信息，信息仅对用户有用
	215	系统类型
	225	数据连接打开，无传输正在进行
	227	进入被动模式

续表

性 质	状态码	意 义
需要更多信息	532	存储文件需要账户信息
	331	用户名正确，需要口令
	332	登录时需要账户信息
	350	请求的文件操作需要进一步命令
命令未完成	421	不能提供服务，关闭控制连接
	425	不能打开数据连接
	426	关闭连接，中止传输
	450	请求的文件操作未执行
	451	中止请求的操作：有本地错误
	452	未执行请求的操作：系统存储空间不足
	500	格式错误，命令不可识别
	501	参数语法错误
	502	命令未实现
	503	命令顺序错误
	504	此参数下的命令功能未实现
	202	命令未实现
	550	未执行请求的操作
	551	请求操作中止：页类型未知
	552	请求的文件操作中止，存储分配溢出
	530	未登录
	532	存储文件需要账户信息
	553	文件名不合法

4.1.5 FTP 实例

◆ 注意：下面的“客户端”是指FTP客户端或暂以客户端身份工作的FTP服务器，“|”代表或。

1. 连接

客户端：请求连接

服务器：120 | 220 | 421（因为发回响应时，已经建立控制连接，发回 421，关闭控制连接）

2. 登录

客户端：USER

服务器：230 | 530 | 500 | 501 | 421 | 331 | 332 | 220

客户端：ACCT

服务器：230 | 530 | 202 | 500 | 501 | 503 | 421 | 220

客户端：PASS

服务器：230 | 202 | 530 | 500 | 501 | 503 | 421 | 332 | 220

3. 目录操作

客户端: CWD

服务器: 200 | 250 | 500 | 501 | 502 | 421 | 550 | 530

客户端: CDUP

服务器: 530 | 200 | 500 | 501 | 502 | 421 | 550

客户端: RMD

服务器: 250 | 500 | 501 | 502 | 421 | 530 | 550

客户端: SMNT

服务器: 202 | 250 | 500 | 501 | 502 | 421 | 530 | 550

客户端: MKD

服务器: 257 | 500 | 501 | 502 | 421 | 530 | 550

客户端: PWD

服务器: 257 | 500 | 501 | 502 | 421 | 550

4. 传输参数

客户端: PASV

服务器: 227 | 500 | 501 | 502 | 421 | 530

客户端: PORT

服务器: 200 | 500 | 501 | 421 | 530 | 504

客户端: MODE

服务器: 200 | 500 | 501 | 504 | 421 | 530

客户端: TYPE

服务器: 200 | 500 | 501 | 504 | 421 | 530

客户端: STRU

服务器: 200 | 500 | 501 | 504 | 421 | 530

5. 文件操作命令

客户端: LIST

服务器: 125 | 150 | 226 | 250 | 425 | 426 | 451 | 450 | 500 | 501 | 502 | 421 | 530

客户端: NLST

服务器: 125 | 150 | 226 | 250 | 425 | 426 | 451 | 450 | 500 | 501 | 502 | 421 | 530

客户端: ALLO

服务器: 200 | 202 | 500 | 501 | 504 | 421 | 530

客户端: REST

服务器: 500 | 501 | 502 | 421 | 530 | 350

客户端: STOR

服务器: 125 | 150 | 226 | 250 | 425 | 426 | 451 | 500 | 501 | 421 | 530 | 551 |
552 | 532 | 553 | 450 | 452

客户端: STOU

服务器: 125 | 150 | 226 | 250 | 425 | 426 | 451 | 500 | 501 | 421 | 530 | 551 | 552 | 532 | 553 | 450 | 452

客户端: RETR

服务器: 125 | 150 | 226 | 250 | 425 | 426 | 450 | 451 | 550 | 500 | 501 | 421 | 530

客户端: APPE

服务器: 125 | 150 | 226 | 250 | 425 | 426 | 451 | 551 | 552 | 532 | 450 | 550 | 452 | 553 | 500 | 501 | 502 | 421 | 530

客户端: RNTO

服务器: 250 | 532 | 553 | 500 | 501 | 502 | 503 | 421, 530

客户端: RNFR

服务器: 450 | 550 | 500 | 501 | 502 | 421 | 530 | 350

客户端: DELE

服务器: 250 | 450 | 550 | 500 | 501 | 502 | 421 | 530

客户端: ABOR

服务器: 426 | 225 | 226 | 500 | 501 | 502 | 421

6. 获取服务器信息

客户端: SYST

服务器: 215 | 500 | 501 | 502 | 421

服务器: HELP

客户端: 211 | 214 | 500 | 501 | 502 | 421

客户端: STAT

服务器: 211 | 212 | 213 | 450 | 500 | 501 | 502 | 421 | 530

客户端: SITE

服务器: 200 | 202 | 500 | 501 | 530

7. 等待

客户端: NOOP

服务器: 200 | 500 | 421

8. 重新登录

客户端: REIN

服务器: 120 | 220 | 421 | 500 | 502

9. 推出登录

客户端: QUIT

服务器: 221 | 500 | 502

4.1.6 文件传输的特别要求

1. 字节大小

在 FTP 中字节大小有两个：逻辑字节大小和用于传输的字节大小。后者通常是 8 位，而前者没有限制。

2. 数据表示与保存

实际应用中，由于两个系统的数据存储方式不相同，常须要对数据存储格式进行转换，在传送文本时会有 ASCII 表示的问题，在进行二进制传送的时候会有不同系统对字节长度规定不同的问题，有的系统是 7 位，有的系统是 32 位，这也需要进行转换。在这方面，FTP 并不能实现所有功能，需要用户自己完成。

3. 建立数据连接

在实际应用中，服务器进程默认数据端口和控制连接端口相邻。传输字节大小是 8 位字节，其中 6 位字节是实际传输字节，并不代表主机内的数据表示。被动数据传输进程在数据端口接收数据，FTP 请求命令决定数据传输的方向（上传还是下载）。服务器在接到请求以后，将初始化端口的数据连接。当连接建立后，传输在数据传送连接之间进行，服务器通过控制连接对用户返回应答。所以，在传输文件时，最好打开两个以上 FTP 端口，默认端口为控制连接，非默认端口为数据传输连接。当然，只用一个端口也可实现数据传输，但往往不能正确结束文件传输，出错率会高一些。

4. 结束文件传输

结束文件传输有两种，一种是用户命令，即通过发送中止传输的命令，强迫中止传输。另一种是发送 EOF 结束。需要注意的是，所有数据传输必须以一个 EOF 结束，它可以显式给出，也可以通过关闭数据传输连接隐式给出（不是关闭控制连接）。通过关闭数据传输隐式给出 EOF 比较合适，不容易出错。对于记录文件，所有 EOF 是显式的，包括最后一个记录。对于以页结构传送的文件，使用“最后一页”表示结束。为了进行标准化传送，发送数据的主机必须把行结束或记录结束的内部表示转化为传输模式和文件结构指定的形式传送，接收方则进行相反的工作。

5. 端口设置

所有 FTP 服务器都必须支持默认端口。只有用户才能要求使用非默认数据端口，用户可以使用 PORT 命令指定非默认端口，它要求服务器方以 PASV 确定非默认数据端口。

6. 如何连续传输多个文件

在使用流式数据传输模型时，文件结束通过关闭连接指示。如果要传送多个文件，就会出麻烦。解决的方法有两个，一个是确定非默认端口，另一个是使用另一种传输模式。就传输模式而言，流传输模式是不安全的，因此无法确定连接是暂时还是永久关闭。其他传输模式不通过关闭连接表示文件结构，它们可以通过 FTP 命令决定传送结构。因此使用这些传输模式可以在保持连接的情况下传送多个文件。

7. 本章术语

- 控制连接：传输 FTP 命令和服务器应答的连接。
- 数据连接：传输文件流的连接。

4.2 FTP 服务器开发

本节主要讲解如何用 C# 接收与发送命令，如何解读命令，如何发送与接收文件。工作原理如下：FTP 服务器首先在默认端口进行监听（因笔者服务器 21 端口已经占用，本节 FTP

服务器程序以 1234 端口作为默认端口）。当连接建立后，客户端可以发送 RETR 命令直接传输文件，这时，服务器以默认端口的下一个端口为数据传输端口。客户端也可以发送 PASV 命令，要求服务器进入被动状态，然后发送 PORT 命令要求服务器在指定的端口进行监听。数据传输连接建立后，即开始数据传输，数据传输结束后，服务器关闭数据传输连接。本服务器为匿名服务器，不需要 USER、PASS、ACCT 命令，本服务器仅可识别 LIST、PORT、PASV、RETR、HELP、QUIT 命令，对于其他命令发出不识别应答。

本服务器以流进行传输文件，而且仅支持下载（上传和下载原理完全一样）。传输文件时，服务器首先建立一个文件流，然后将文件流赋值给网络流，客户端接收到网络流后，构建一个文件流，然后将网络流赋值给文件流。这样，该 FTP 系统可以传输任何类型的文件，并且不会出错。

从上面的简介中可以看出，服务器与客户机之间至少有两个连接，一个是控制连接，另一个是数据连接。服务器与客户端的控制连接，在传输目录时，为了支持中文文件名和目录，使用 BigEndianUnicode 码，其他时候使用 ASCII 码。数据连接因使用文件流和网络流，没有使用文本流，因此不存在编码的问题。

4.2.1 命令的接收与解读

命令接收直接用网络流即可，将网络流赋值给字节数组，然后解码即可。为了确定命令字符串的长度，可以用 string 类的 IndexOf 方法。

下列代码实现解读命令的过程，首先用 int x=ss.IndexOf("\r\n",0) 代码确定命令字符串的长度，然后用 int y=ss.IndexOf(" ",0) 确定命令与参数之间的分隔符——空格所在位置。

```
string ss=System.Text.Encoding.ASCII.GetString(byteMessage);
int x=ss.IndexOf("\r\n",0);
int y=ss.IndexOf(" ",0);
string commMessage=System.Text.Encoding.ASCII.GetString(byteMessage,0,x);
if(y!=-1)
{
    command=commMessage.Substring(0,y);
    else{command=commMessage.Substring(0,x);}
    command=command.ToLower();
    if(y!=-1)
    {
        parameter=commMessage.Substring(y+1,x-y-1);
        parameter=parameter.ToLower();
    }
    else{parameter="";
}
```

4.2.2 响应码的发送

响应码的发送可以用网络流实现。下列代码实现发送响应码的功能：

```

private void WriteToNetStream(ref NetworkStream NetStream, string ToSend)
{
    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.ASCII.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}

```

4.2.3 发送目录

为了识别中文文件名和文件夹，发送目录使用 BigEndianUnicode 码。下列代码实现发送目录的功能：

```

private void sendDir(ref NetworkStream NetStream, string Dir)
{
    string stringToSend = Dir + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.BigEndianUnicode.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}

```

4.2.4 发送文件

发送文件使用网络流和文件流。由于文件流里封装了文件结构，故不用声明文件类型即可正确传输。下列代码实现文件传输的功能：

```

private void transfer(ref NetworkStream stream, string path)
{
    filestream=new FileStream(path, FileMode.Open, FileAccess.Read);
    int number;
    //定义缓冲区
    byte[] bb=new byte[8];
    //循环读文件
    // NetworkStream stream=new NetworkStream(newClient);
    while((number=filestream.Read(bb,0,8))!=0)
    {
        //向客户端发送流
        stream.Write(bb,0,8);
        //刷新流
        stream.Flush();
        bb=new byte[8];
    }
    filestream.Close();
    //newClient.Close();
    stream.Close();
}

```

4.2.5 接听命令并响应

接听命令并响应是代码的关键部分。服务器启动后，必须循环接听数据；一旦接收到命令，就要作出响应。下列代码实现接听命令的功能：

```
private void recieve()
{
    newClient=listener.AcceptSocket();
    if(newClient.Connected)
    {
        NetworkStream netStream=new NetworkStream(newClient);
        WriteToNetStream(ref netStream,"220 FTP Server Ready!");
        //循环接受命令 * * * * *
        while(!control)
        {
            byte[] byteMessage=new byte[1024];
            int i=newClient.Receive(byteMessage,byteMessage.Length,0);
            string ss=System.Text.Encoding.ASCII.GetString(byteMessage);
            int x=ss.IndexOf("\r\n",0);
            int y=ss.IndexOf(" ",0);
            string commMessage=System.Text.Encoding.ASCII.GetString
                (byteMessage,0, x);
            if(y!=-1)
            {
                command=commMessage.Substring(0,y);
                else{command=commMessage.Substring(0,x);}
                command=command.ToLower();
                if(y!=-1)
                {
                    parameter=commMessage.Substring(y+1,x-y-1);
                    parameter=parameter.ToLower();
                }
                else{parameter="";}
                richTextBox1.AppendText ("command="+command+","
                    +"parameter="+parameter+"\r\n");
                commMessage.Remove(0,commMessage.Length);
                if(command=="list")
                {
                    ...
                }
                else if(command=="retr")
                {
                    ...
                    ...
                }
                else if(command=="pasv")
                {
```

```
...
}
else if(command=="port")
{
...
}
else if(command=="help")
{
...
}
else if(command=="quit")
{
...
}
else
{
    WriteToNetStream(ref netStream,
                      "500 unrecognized command");
}
}
}
}
```

4.2.6 FTP服务器开发

图 4-1 是本例的程序界面，“监听端口”文本框是 textBox1， “客户信息”文本框是 richTextBox1，“开始服务”按钮文本框是 button1，“关闭服务”文本框是 button2。

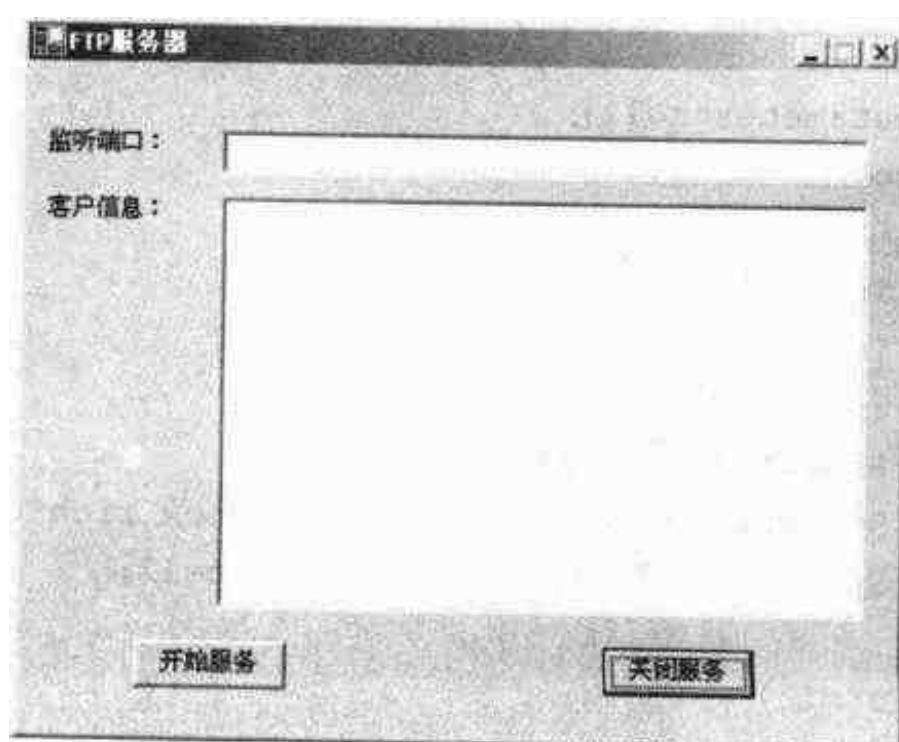


图 4-1 FTP 服务器的程序界面

下面是该服务器的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
namespace accep_so
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Button button2;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
        private bool control = false;
        private bool pasv=false;
        private bool pasvPort=false;
        private int port;
        private Socket newClient;

        string parameter=null;
        string command=null;

        private TcpListener listener;
        private TcpListener newListener;
        private int newPort;
        private FileStream filestream;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.Label label2;

        public Form1()
        {
            //
```

```
// Required for Windows Form Designer support
//
InitializeComponent();
//
// TODO: Add any constructor code after InitializeComponent call
//
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
#endregion
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        port =Int32.Parse(textBox1.Text);
    }
    catch{MessageBox.Show("您输入的格式不对！请输入正整数。");}
}

try
{
    listener=new TcpListener(port);
    listener.Start();
    Thread thread=new Thread(new ThreadStart(recieve));
}
```

```
        thread.Start();
    }
    catch(Exception ee){MessageBox.Show(ee.Message);}
}
private void button2_Click(object sender, System.EventArgs e)
{
    try
    {
        listener.Stop();
    }
    catch{MessageBox.Show("监听还未开始，关闭无效。");}
}

private void recieve()
{
    while(true)
    {
        newClient=listener.AcceptSocket();
        if(newClient.Connected)
        {
            Thread thread=new Thread(new ThreadStart(round));
            thread.Start();
        }
    }
}

private void round()
{
    NetworkStream netStream=new NetworkStream(newClient);
    WriteToNetStream(ref netStream,"220 FTP Server Ready!");
    while(true){
        //接受信息+ + +
        byte[] byteMessage=new byte[1024];
        int i=newClient.Receive(byteMessage,byteMessage.Length,0);
        string ss=
            System.Text.Encoding.ASCII.GetString(byteMessage);
        int x=ss.IndexOf("\r\n",0);
        int y=ss.IndexOf(" ",0);
        string commMessage=System.Text.Encoding.ASCII.GetString
            (byteMessage,0,x);
        if(y!=-1)
        {
            command=commMessage.Substring(0,y);
        }
        else{command=commMessage.Substring(0,x);}
        command=command.ToLower();
        if(y!=-1)
```

```
{  
    parameter=commMessage.Substring(y+1,x-y-1);  
    parameter=parameter.ToLower();  
}  
else{parameter="";}  
richTextBox1.AppendText ("command="+command  
+", "+parameter+"\r\n");  
commMessage.Remove(0,commMessage.Length);  
if(command=="list")  
{  
    try  
{  
        string[] str=new string[1024];  
        string dir=null;  
        for(int k=0;  
            k<Directory.GetDirectories(parameter).Length;  
            k++)  
        {  
            str[k]=  
                Directory.GetDirectories(parameter)[k];  
            dir=dir+"目录: "+str[k]+"\r\n";  
        }  
        for(int k=0;  
            k<Directory.GetFiles(parameter).Length;k++)  
        {  
            str[k]=Directory.GetFiles(parameter)[k];  
            dir=dir+"文件: "+str[k]+"\r\n";  
        }  
        WriteToNetStream(ref netStream,  
            "125 the files and directory is here");  
        sendDir(ref netStream,dir);  
    }  
    catch{WriteToNetStream(ref netStream,  
        "502 unsuccessful,try again ");}  
}  
else if(command=="retr")  
{  
    try  
{  
        if((pasv==false)&&(pasvPort==false))  
        {  
            string path=parameter;  
            newListener=  
                new TcpListener(Int32.Parse(textBox1.Text)  
                +1);  
            newListener.Start();  
        }  
    }  
}
```

```
        WriteToNetStream(ref netStream,
                          "150 data link listening");
        Socket nextLink=newListener.AcceptSocket();
        if(nextLink.Connected)
        {
            WriteToNetStream(ref netStream,
                              "125 now the file's data will be sended");
            NetworkStream stream=
                new NetworkStream(nextLink);
            transfer(ref stream,path);
            stream.Close();
            nextLink.Close();
            newListener.Stop();
        }
    }
    else if((pasv==true)&&(pasvPort==true))
    {
        string path=parameter;
        newListener=new TcpListener(newPort);
        newListener.Start();
        WriteToNetStream(ref netStream,
                          "150 data link listening");
        Socket nextLink=newListener.AcceptSocket();
        if(nextLink.Connected)
        {
            WriteToNetStream(ref netStream,
                              "125 now the file's data will be sended");
            NetworkStream stream=
                new NetworkStream(nextLink);
            transfer(ref stream,path);
            stream.Close();
            nextLink.Close();
            newListener.Stop();
            pasv=false;
            pasvPort=false;
        }
    }
    else
    {
        WriteToNetStream(ref netStream,
                          "350 PASV OR PORT command needed ");
    }
}
catch(WriteToNetStream(ref netStream,
                      "502 performed unsuccessful,try again "));}
```

```
else if(command=="pasv")
{
    try
    {
        pasv=true;
        WriteToNetStream(ref netStream,
                          "227 now the server pasv");
    }
    catch(WriteToNetStream(ref netStream,
                          "502 performed unsuccessful,try again"));
}

else if(command=="port")
{
    try
    {
        if(pasv==true)
        {
            pasvPort=true;
            newPort=Int32.Parse(parameter);
            WriteToNetStream(ref netStream,
                              "200 new data link listen begin");
        }
        else{WriteToNetStream(ref netStream,
                             "350 PASV command needed");}
    }
    catch(WriteToNetStream(ref netStream,
                          "504 the command can't be performed with
                          the parameter "));
}

else if(command=="help")
{
    try
    {
        WriteToNetStream(ref netStream,"215 the FTP system
                                         ready,it can  the comman such as LIST, PORT, PASV,
                                         RETR, HELP, QUIT.before PORT command,
                                         PASV command fistly.we are sorry that the FTP
                                         system can't other comand.");
    }
    catch(WriteToNetStream(ref netStream,
                          "502 performed unsuccessful,try again"));
}

else if(command=="quit")
{
    try
    {
```

```
        WriteToNetStream(ref netStream,
                           "221 connection closed");
        newClient.Close();
    }
    catch(WriteToNetStream(ref netStream,
                           "502 performed unsuccessful,try again"));
}
else{WriteToNetStream(ref netStream,
                      "500 unrecognized command");}
}
}

private void transfer(ref NetworkStream stream,string path)
{
    filestream=new FileStream(path,
                             FileMode.Open, FileAccess.Read);
    int number;
    //定义缓冲区
    byte[] bb=new byte[8];
    //循环读文件
    // NetworkStream stream=new NetworkStream(newClient);
    while((number=filestream.Read(bb,0,8))!=0)
    {//向客户端发送流
        stream.Write(bb,0,8);
        //刷新流
        stream.Flush();
        bb=new byte[8];
    }
    filestream.Close();
    //newClient.Close();
    stream.Close();
}

private void WriteToNetStream(ref NetworkStream NetStream,
                             string ToSend)
{
    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend=System.Text.Encoding.ASCII.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}

private void sendDir(ref NetworkStream NetStream, string Dir)
{
    string stringToSend = Dir + "\r\n";
    byte[] arrayToSend =
        System.Text.Encoding.BigEndianUnicode.GetBytes
```

```
        (stringToSend.ToCharArray()));

        NetStream.Write(arrayToSend, 0, arrayToSend.Length);
        NetStream.Flush();
    }
}
}
```

到此为止，本程序开发完毕。这是一个符合 FTP 规范的服务器，在此基础上可以开发出符合商业规范的 FTP 服务器。

4.3 FTP 客户端开发

本节开发一个 FTP 客户端，该客户端可以用 LIST 命令浏览服务器文件目录，可以用 RETR 命令下载服务器文件。在默认情况下，使用默认端口的下一个端口作为数据连接的端口。本节的 FTP 客户端还可以用 PASV 和 PORT 命令自定义端口，建立与服务器的数据连接。

4.3.1 发送命令

本节 FTP 客户端，与商业 FTP 客户端一样，使用 ASCII 码传输 FTP 命令。下列代码实现 ASCII 码命令的传输：

```
private void WriteToNetStream(ref NetworkStream NetStream,
    string sendMessage)
{
    string stringToSend = sendMessage + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.ASCII.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}
```

4.3.2 接收服务器应答

服务器应答包括两种：一种是普通应答码，比如 125 now the file's data will be sended，接收这类应答码只要是用 ASCII 码即可；另一种是服务器文件列表，需要使用 BigEndianUnicode。

下列方法可接受第一种服务器应答：

```
private string ReadFromNetStream(ref NetworkStream NetStream)
{
    byte[] feedback=new byte[1024];
    NetStream.Read(feedback, 0, feedback.Length);
```

```

        string readFeedback=System.Text.Encoding.ASCII.GetString(feedback);
        return readFeedback;
    }
}

```

下列代码可接受服务器的文件列表应答：

```

private string ReadDir(ref NetworkStream NetStream)
{
    byte[] feedDir=new byte[1024];
    NetStream.Read(feedDir,0,feedDir.Length);
    string Dir=System.Text.Encoding.BigEndianUnicode.GetString(feedDir);
    return Dir;
}

```

4.3.3 检查服务器应答码

客户端必须可以判断服务器应答码的性质，从而可以了解服务器状态，进而发出适当的下一步命令。比如客户端发出 RETR 命令后，如果接收到“150”，表示文件状态良好，通知客户打开（或建立）连接；如果收到“125”，表示服务器将要发送文件数据了，如果接收到“500”，则表示命令发错了，服务器无法识别。下列代码用于检测应答码。该方法有两个参数，第一个参数为待检查的字符串，第二个参数为待检查的应答码；返回值是 bool 型值，如果字符串中存在该应答码，就返回 true，反之为 false。

```

private bool checkFeedback(string strMessage,string check)
{
    if (strMessage.IndexOf(check) != -1) {return true;}
    else{return false;}
}

```

4.3.4 文件传输方法

下列方法实现接收文件传输的功能。

```

private void down(ref NetworkStream stream)
{
    int readNumber=0;
    byte[] bye=new byte[8];
    //下行循环读取网络流并写进文件
    while((readNumber=stream.Read(bye,0,8))>0)
    {
        filestream.Write(bye,0,readNumber);
        filestream.Flush();
    }
    filestream.Close();
    MessageBox.Show("下载完毕！");
}

```

4.3.5 下载功能

本程序有两个方法完成下载功能，一个方法无需使用 PASV 和 PORT 命令而直接使用 RETR 命令，另一个方法需要使用 PASV 和 PORT 命令。

下面是不需要使用 PASV 和 PORT 命令而直接使用 RETR 命令的下载方法。

```
private void download()
{
    //获取服务器网络流
    string downString="retr "+textBox5.Text;
    WriteToNetStream(ref netStream,downString);
    string feedback=ReadFromNetStream(ref netStream);
    bool check=checkFeedback(feedback,"150");
    //*****
    if(check==true)
    {
        TcpClient newClient=new TcpClient(textBox1.Text,
            Int32.Parse(textBox2.Text)+1);
        feedback=ReadFromNetStream(ref netStream);
        bool checkLink=checkFeedback(feedback,"125");
        if(checkLink==true)
        {
            NetworkStream str=newClient.GetStream();
            down(ref str);
            newClient.Close();
        }
        int rounReceive=0;
        while(checkLink!=true&&rounReceive<=5){
            newClient=new TcpClient(textBox1.Text,
                Int32.Parse(textBox2.Text)+1);
            feedback=ReadFromNetStream(ref netStream);
            checkLink=checkFeedback(feedback,"125");
            rounReceive++;
            if(checkLink==true)
            {
                NetworkStream str=newClient.GetStream();
                down(ref str);
                newClient.Close();
            }
        }
        //*****
        int round=0;
        while(check!=true&&round<=5)
        {
            WriteToNetStream(ref netStream,downString);
        }
    }
}
```

```

feedback=ReadFromNetStream(ref netStream);
check=checkFeedback(feedback, "150");
round++;
if(check==true)
{
    TcpClient newClient=new TcpClient(textBox1.Text,
        Int32.Parse(textBox2.Text)+1);
    feedback=ReadFromNetStream(ref netStream);
    bool checkLink=checkFeedback(feedback, "125");
    if(checkLink==true)
    {
        NetworkStream str=newClient.GetStream();
        down(ref str);
        newClient.Close();
    }
    int rounReceive=0;
    while(checkLink!=true&&rounReceive<=5)
    {
        newClient=new TcpClient(textBox1.Text,
            Int32.Parse(textBox2.Text)+1);
        feedback=ReadFromNetStream(ref netStream);
        checkLink=checkFeedback(feedback, "125");
        rounReceive++;
        if(checkLink==true)
        {
            NetworkStream str=newClient.GetStream();
            down(ref str);
            newClient.Close();
        }
    }
}
}
}

```

下面是需要使用 PASV 和 PORT 命令的下载方法。

```

private void newdownload()
{
    //获取服务器网络流
    string downString="retr "+textBox5.Text;
    WriteToNetStream(ref netStream,downString);
    string feedback=ReadFromNetStream(ref netStream);
    bool check=checkFeedback(feedback, "150");
    //***** ****
    if(check==true)
    {
        TcpClient newClient=new TcpClient(textBox1.Text,

```

```
    Int32.Parse(textBox3.Text));
feedback=ReadFromNetStream(ref netStream);
bool checkLink=checkFeedback(feedback,"125");
if(checkLink==true)
{
    NetworkStream str=newClient.GetStream();
    down(ref str);
    newClient.Close();
}
int rounReceive=0;
while(checkLink!=true&&rounReceive<=5)
{
    newClient=new TcpClient(textBox1.Text,
        Int32.Parse(textBox3.Text));
    feedback=ReadFromNetStream(ref netStream);
    checkLink=checkFeedback(feedback,"125");
    rounReceive++;
    if(checkLink==true)
    {
        NetworkStream str=newClient.GetStream();
        down(ref str);
        newClient.Close();
    }
}
//*****
int round=0;
while(check!=true&&round<=5)
{
    WriteToNetStream(ref netStream,downString);
    feedback=ReadFromNetStream(ref netStream);
    check=checkFeedback(feedback,"150");
    round++;
    if(check==true)
    {
        TcpClient newClient=new TcpClient(textBox1.Text,
            Int32.Parse(textBox2.Text)+1);
        feedback=ReadFromNetStream(ref netStream);
        bool checkLink=checkFeedback(feedback,"125");
        if(checkLink==true)
        {
            NetworkStream str=newClient.GetStream();
            down(ref str);
            newClient.Close();
        }
        int rounReceive=0;
```

```

        while(checkLink!=true&&rounReceive<=5)
        {
            newClient=new TcpClient(textBox1.Text,
                Int32.Parse(textBox2.Text)+1);
            feedback=ReadFromNetStream(ref netStream);
            checkLink=checkFeedback(feedback,"125");
            rounReceive++;
            if(checkLink==true)
            {
                NetworkStream str=newClient.GetStream();
                down(ref str);
                newClient.Close();
            }
        }
    }
}

```

4.3.6 FTP 客户端开发

图 4-2 是 FTP 客户端的界面。

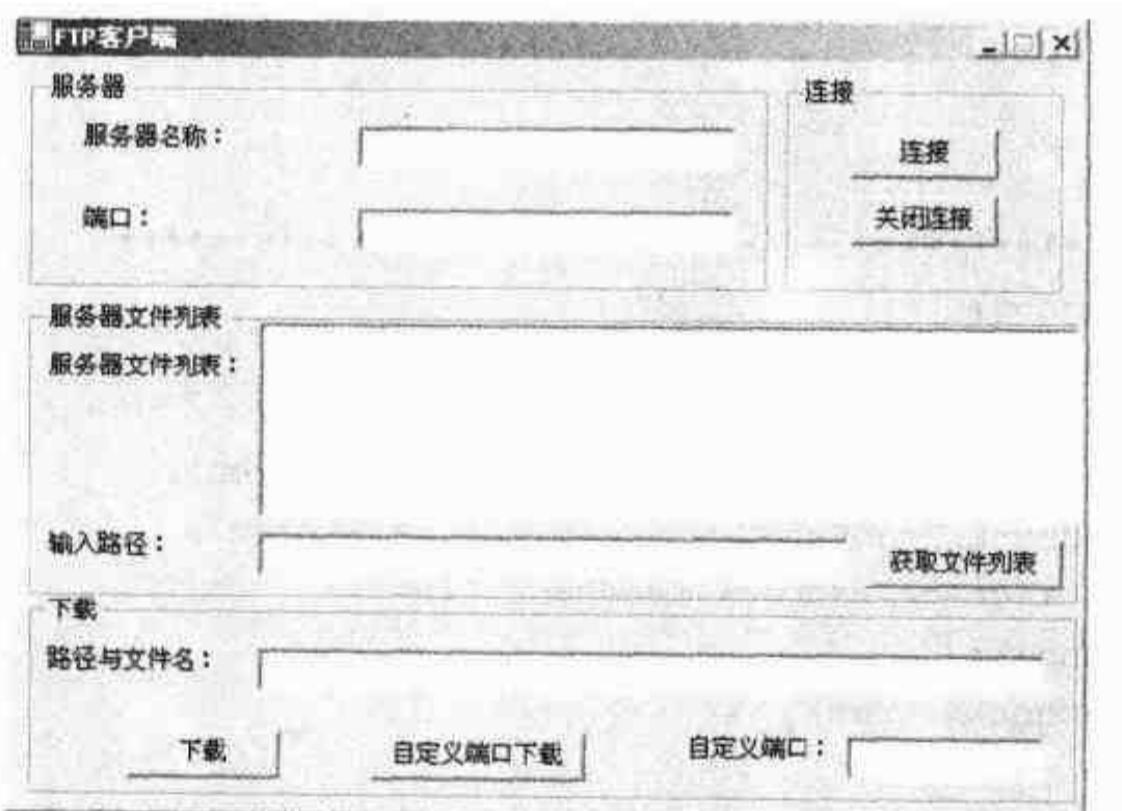


图 4-2 FTP 客户端的界面

在上述界面中，“服务器名称”文本框是 textBox1，“端口”文本框是 textBox2，“自定义端口”文本框是 textBox3，“输入路径”文本框是 textBox4，“路径与文件名”文本框是 textBox5。“连接”按钮是 button1，“关闭连接”按钮是 button2，“获取文件列表”是 button3，“下载”按钮是 button4，“自定义端口下载”按钮是 button5。

下面是该程序的代码：

```

using System;
using System.Drawing;

```

```
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

namespace connect
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private TcpClient client;

        private NetworkStream netStream;
        private FileStream filestream=null;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.Button button5;
        private System.Windows.Forms.Label label4;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
```

```
public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();

    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    .
    if( disposing )
    {
        if (components != null)
        {
            .
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    int port=2100;

    client =new TcpClient();
```

```
try{
    port=Int32.Parse(textBox2.Text);
}
catch{MessageBox.Show("请输入整数。");}
try
{
    client.Connect(textBox1.Text, port);
    netStream=client.GetStream();
    string hello=ReadFromNetStream(ref netStream);
    richTextBox1.AppendText(hello+"\r\n");
}
catch(Exception ee){MessageBox.Show(ee.Message);}
}

private void button2_Click(object sender, System.EventArgs e)
{
    try
    {

        WriteToNetStream(ref netStream, "QUIT");
        string feedback=ReadFromNetStream(ref netStream);
        bool check=checkFeedback(feedback, "221");
        if(check==true)
        {
            client.Close();
        }
        int round=0;
        while(check!=true&&round<=5){
            WriteToNetStream(ref netStream, "QUIT");
            feedback=ReadFromNetStream(ref netStream);
            check=checkFeedback(feedback, "221");
            round++;
        }
    }
    catch(Exception ee){MessageBox.Show(ee.Message);}
}

private void button3_Click(object sender, System.EventArgs e)
{
    string list="list "+textBox4.Text;
    WriteToNetStream(ref netStream, list);
    string feedback=ReadFromNetStream(ref netStream);
    bool check=checkFeedback(feedback, "125");
    if(check==true)
    {
        string dir=ReadDir(ref netStream);
```

```
richTextBox1.AppendText(dir);
}

int round=0;
while(check!=true&&round<=5)
{
    WriteToNetStream(ref netStream,list);
    feedback=ReadFromNetStream(ref netStream);
    check=checkFeedback(feedback,"125");
    round++;
    if(check==true)
    {
        string dir=ReadDir(ref netStream);
        richTextBox1.AppendText(dir);
    }
}

private void button4_Click(object sender, System.EventArgs e)
{
    if(saveFileDialog1.ShowDialog()==DialogResult.OK)
    {
        filestream=new FileStream(saveFileDialog1.FileName,
            FileMode.OpenOrCreate, FileAccess.Write);
        Thread thread=new Thread(new ThreadStart(download));
        thread.Start();
    }
}

private void newdownload()
{
    //获取服务器网络流
    string downString="retr "+textBox5.Text;
    WriteToNetStream(ref netStream,downString);
    string feedback=ReadFromNetStream(ref netStream);
    bool check=checkFeedback(feedback,"150");
    //***** ****
    if(check==true)
    {
        TcpClient newClient=new TcpClient(textBox1.Text,
            Int32.Parse(textBox3.Text));
        feedback=ReadFromNetStream(ref netStream);
        bool checkLink=checkFeedback(feedback,"125");
        if(checkLink==true)
        {
            NetworkStream str=newClient.GetStream();
            down(ref str);
            newClient.Close();
        }
    }
}
```

```
        }

        int rounReceive=0;
        while(checkLink!=true&&rounReceive<=5)
        {
            newClient=new TcpClient(textBox1.Text,
                Int32.Parse(textBox3.Text));
            feedback=ReadFromNetStream(ref netStream);
            checkLink=checkFeedback(feedback,"125");
            rounReceive++;
            if(checkLink==true)
            {
                NetworkStream str=newClient.GetStream();
                down(ref str);
                newClient.Close();
            }
        }
    //*****
    int round=0;
    while(check!=true&&round<=5)
    {
        WriteToNetStream(ref netStream,downString);
        feedback=ReadFromNetStream(ref netStream);
        check=checkFeedback(feedback,"150");
        round++;
        if(check==true)
        {
            TcpClient newClient=new TcpClient(textBox1.Text,
                Int32.Parse(textBox2.Text)+1);
            feedback=ReadFromNetStream(ref netStream);
            bool checkLink=checkFeedback(feedback,"125");
            if(checkLink==true)
            {
                NetworkStream str=newClient.GetStream();
                down(ref str);
                newClient.Close();
            }
            int rounReceive=0;
            while(checkLink!=true&&rounReceive<=5)
            {
                newClient=new TcpClient(textBox1.Text,
                    Int32.Parse(textBox2.Text)+1);
                feedback=ReadFromNetStream(ref netStream);
                checkLink=checkFeedback(feedback,"125");
                rounReceive++;
                if(checkLink==true)
```

```
        {
            NetworkStream str=newClient.GetStream();
            down(ref str);
            newClient.Close();
        }
    }
}

private void download()
{
    //获取服务器网络流
    string downString="retr "+textBox5.Text;
    WriteToNetStream(ref netStream,downString);
    string feedback=ReadFromNetStream(ref netStream);
    bool check=checkFeedback(feedback,"150");
    //***** ****
    if(check==true)
    {
        TcpClient newClient=new TcpClient(textBox1.Text,
            Int32.Parse(textBox2.Text)+1);
        feedback=ReadFromNetStream(ref netStream);
        bool checkLink=checkFeedback(feedback,"125");
        if(checkLink==true)
        {
            NetworkStream str=newClient.GetStream();
            down(ref str);
            newClient.Close();
        }
        int rounReceive=0;
        while(checkLink!=true&&rounReceive<=5)
        {
            newClient=new TcpClient(textBox1.Text,
                Int32.Parse(textBox2.Text)+1);
            feedback=ReadFromNetStream(ref netStream);
            checkLink=checkFeedback(feedback,"125");
            rounReceive++;
            if(checkLink==true)
            {
                NetworkStream str=newClient.GetStream();
                down(ref str);
                newClient.Close();
            }
        }
    }
}
```

```
*****  
int round=0;  
while(check!=true&&round<=5)  
{  
    WriteToNetStream(ref netStream,downString);  
    feedback=ReadFromNetStream(ref netStream);  
    check=checkFeedback(feedback,"150");  
    round++;  
    if(check==true)  
    {  
        TcpClient newClient=new TcpClient(textBox1.Text,  
            Int32.Parse(textBox2.Text)+1);  
        feedback=ReadFromNetStream(ref netStream);  
        bool checkLink=checkFeedback(feedback,"125");  
        if(checkLink==true)  
        {  
            NetworkStream str=newClient.GetStream();  
            down(ref str);  
            newClient.Close();  
        }  
        int rounReceive=0;  
        while(checkLink!=true&&rounReceive<=5)  
        {  
            newClient=new TcpClient(textBox1.Text,  
                Int32.Parse(textBox2.Text)+1);  
            feedback=ReadFromNetStream(ref netStream);  
            checkLink=checkFeedback(feedback,"125");  
            rounReceive++;  
            if(checkLink==true)  
            {  
                NetworkStream str=newClient.GetStream();  
                down(ref str);  
                newClient.Close();  
            }  
        }  
    }  
}  
private void down(ref NetworkStream stream)  
{  
    int readNumber=0;  
    byte[] bye=new byte[8];  
    //下行循环读取网络流并写进文件  
    while((readNumber=stream.Read(bye,0,8))>0)  
    {  
        filestream.Write(bye,0,readNumber);  
    }  
}
```

```
        filestream.Flush();
    }
    filestream.Close();
    MessageBox.Show("下载完毕！");
}

private void button5_Click(object sender, System.EventArgs e)
{
    if(saveFileDialog1.ShowDialog()==DialogResult.OK)
    {
        //构造新的文件流
        string command="pasv";
        WriteToNetStream(ref netStream,command);
        string feedback=ReadFromNetStream(ref netStream);
        bool check=checkFeedback(feedback,"227");
        if(check==true)
        {
            command="port "+textBox3.Text;
            WriteToNetStream(ref netStream,command);
            feedback=ReadFromNetStream(ref netStream);
            bool listenCheck=checkFeedback(feedback,"200");
            if(listenCheck==true)
            {
                filestream=
                    new FileStream(saveFileDialog1.FileName,
                    FileMode.OpenOrCreate, FileAccess.Write);
                Thread thread=
                    new Thread(new ThreadStart(newdownload));
                thread.Start();
            }
            int listenRount=0;
            while((listenCheck==false)&&(listenRount<=5))
            {
                listenRount++;
                WriteToNetStream(ref netStream,command);
                feedback=ReadFromNetStream(ref netStream);
                listenCheck=checkFeedback(feedback,"200");
                if(listenCheck==true)
                {
                    filestream=
                        new FileStream(saveFileDialog1.FileName,
                        FileMode.OpenOrCreate, FileAccess.Write);
                    Thread thread=
                        new Thread(new ThreadStart(newdownload));
                    thread.Start();
                }
            }
        }
    }
}
```

```
int round=0;
while(check!=true&&round<=5)
{
    WriteToNetStream(ref netStream,command);
    feedback=ReadFromNetStream(ref netStream);
    check=checkFeedback(feedback,"227");
    if(check==true)
    {
        command="port "+textBox3.Text;
        WriteToNetStream(ref netStream,command);
        feedback=ReadFromNetStream(ref netStream);
        bool listenCheck=checkFeedback(feedback,"200");
        if(listenCheck==true)
        {
            Thread thread=
                new Thread(new ThreadStart(newdownload));
            thread.Start();
        }
        int listenRound=0;
        while((listenCheck==false)&&(listenRound<=5))
        {
            listenRound++;
            WriteToNetStream(ref netStream,command);
            feedback=ReadFromNetStream(ref netStream);
            listenCheck=checkFeedback(feedback,"200");
            if(listenCheck==true)
            {
                Thread thread=
                    new Thread(new ThreadStart
                    (newdownload));
                thread.Start();
            }
        }
    }
}

private void WriteToNetStream(ref NetworkStream NetStream,
    string sendMessage)
{
    string stringToSend = sendMessage + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.ASCII.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
```

```

        }

private string ReadFromNetStream(ref NetworkStream NetStream)
{
    byte[] feedback=new byte[1024];
    NetStream.Read(feedback,0,feedback.Length);
    string readFeedback=
        System.Text.Encoding.ASCII.GetString(feedback);
    return readFeedback;
}

private string ReadDir(ref NetworkStream NetStream)
{
    byte[] feedDir=new byte[1024];
    NetStream.Read(feedDir,0,feedDir.Length);
    string Dir=
        System.Text.Encoding.BigEndianUnicode.GetString(feedDir);
    return Dir;
}

private bool checkFeedback(string strMessage,string check)
{
    if (strMessage.IndexOf(check) != -1) {return true;}
    else{return false;}
}
}
}

```

4.3.7 演示

(1) 启动服务器程序，在“监听端口”文本框里输入适当的端口，然后单击“开始服务”按钮，开始监听端口。

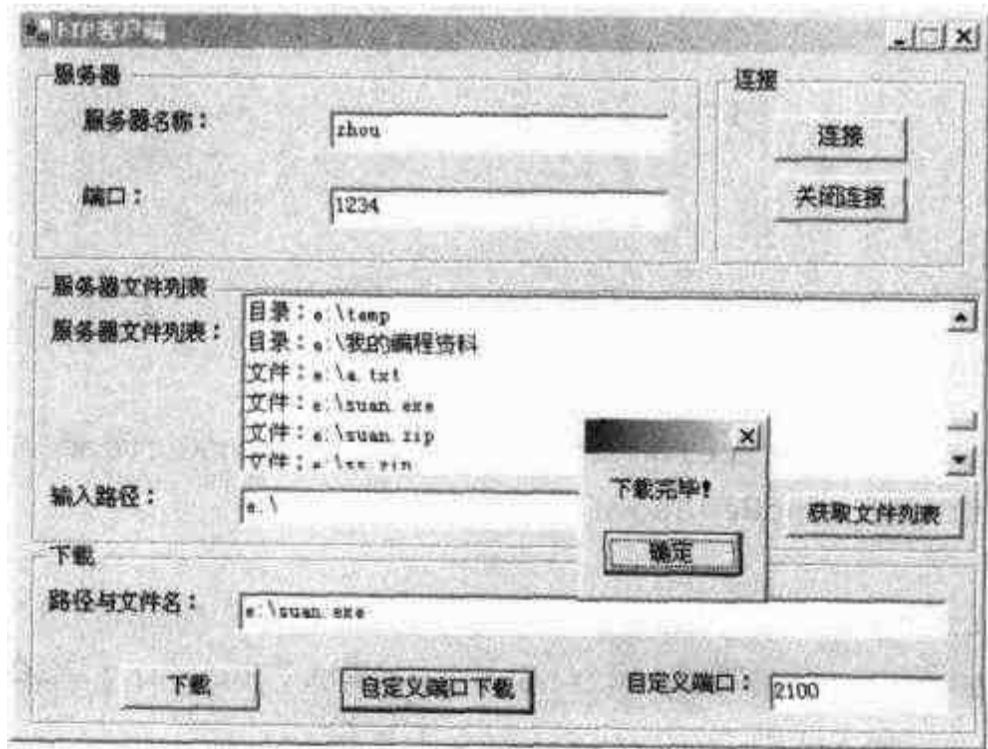


图 4-3 FTP 客户端的运行情况

(2) 启动客户端程序，在“服务器名称”里输入适当的名称，在“端口”文本框里输入适当的端口，然后单击“连接”按钮，建立与服务器的连接。

(3) 在客户端“输入路径”文本框里输入路径并单击“获取文件列表”，可以查询服务器文件列表。在客户端“路径与文件名”文本框里输入适当的路径和文件名，单击“下载”按钮可以下载文件。在“自定义端口”文本框里输入适当的端口并单击“自定义端口下载”按钮，可以从自定义端口下载文件。

图 4-3 是客户端运行情况。

图 4-4 是服务器的运行情况。

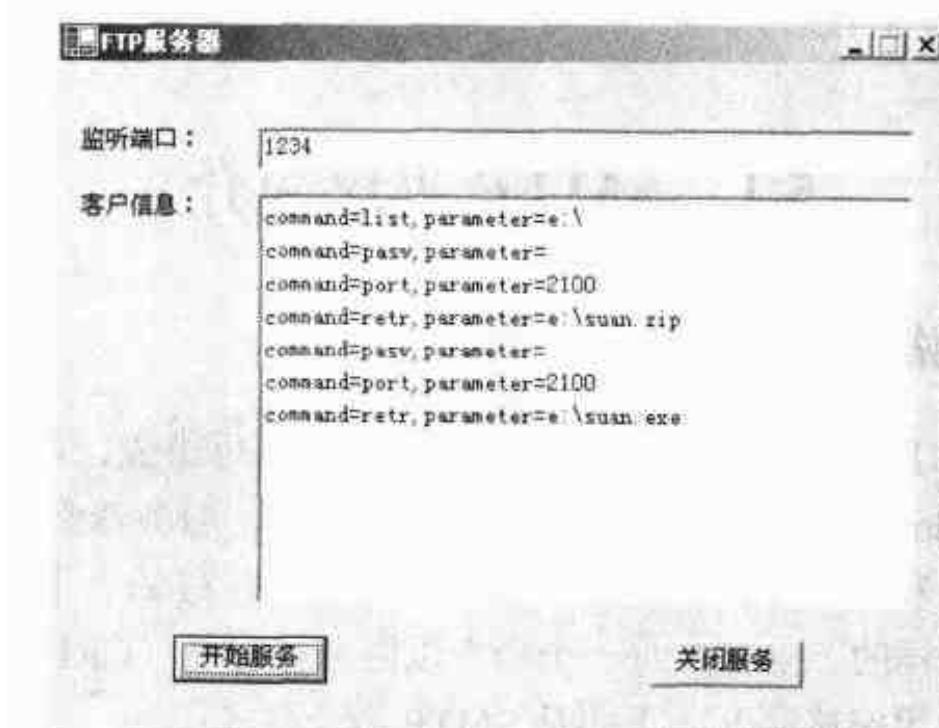


图 4-4 FTP 服务器的运行情况

本章小结

本章首先介绍了 FTP 协议规范，然后根据 FTP 协议开发了 FTP 服务器和客户端。本章程序基本符合商业规范，用户可以在本章程序基础上，开发出符合商业需要的软件来。

第 5 章 SMTP 协议开发

SMTP 是英文 Simple Mail Transfer Protocol 的缩写，意为简单邮件传输协议，默认端口为 25。该协议构成了邮件传输的基本标准，在此基础上，为了减少垃圾邮件，又升级为 ESMTP。本章主要根据 SMTP 协议介绍邮件服务器的开发方法。在介绍 SMTP 协议的同时，也简要介绍一下 ESMTP 协议。

5.1 SMTP 协议简介

5.1.1 SMTP 命令格式

命令是由命令码和其后的参数组成。命令码由四个字母组成，不区别大小写。下面的代码 HELO、helO、hElo、HeLo、heLo 等含义相同，服务器都应能识别。

但是，回复路径和转发路径中的参数是区别大小写的，比如 `aaa@xxx.com` 与 `AAA@xxx.com` 是不一样的。SMTP 每一个命令以回车换行符（CRLF）结束。命令与参数之间、参数与参数之间用空格隔开。下面是 SMTP 命令格式：

```
HELO <SP> <domain> <CRLF>
MAIL <SP> FROM:<reverse-path> <CRLF>
RCPT <SP> TO:<forward-path> <CRLF>
DATA <CRLF>
RSET <CRLF>
SEND <SP> FROM:<reverse-path> <CRLF>
SOML <SP> FROM:<reverse-path> <CRLF>
SAML <SP> FROM:<reverse-path> <CRLF>
VRFY <SP> <string> <CRLF>
EXPN <SP> <string> <CRLF>
HELP [<SP> <string>] <CRLF>
NOOP <CRLF>
QUIT <CRLF>
TURN <CRLF>
```

5.1.2 SMTP 命令参数格式

```
<reverse-path> 同 <path>
<forward-path> 同 <path>
<path> 为 "<" [ <a-d-1> ":" ] <mailbox> ">"
<a-d-1> 为<at-domain> | <at-domain> "," <a-d-1>
<at-domain> 为 "@" <domain>
```

<domain> 为 <element> | <element> "." <domain>
 <element> 为 <name> | "#" <number> | "[" <dotnum> "]"
 <mailbox> 为 <local-part> "@" <domain>
 <local-part> 为 <dot-string> | <quoted-string>
 <name> 为 <a> <ldh-str> <let-dig>
 <ldh-str> 为 <let-dig-hyp> | <let-dig-hyp> <ldh-str>
 <let-dig> 为 <a> | <d>
 <let-dig-hyp> 为 <a> | <d> | "-"
 <dot-string> 为 <字符串> | <字符串> "." <dot-string>
 <字符串> 为 <字符> | <字符> <字符串>
 <quoted-string> 为 "" <qtext> ""
 <qtext> 为 "\" <x> | "\"" <x> <qtext> | <q> | <q> <qtext>
 <字符> 为 <c> | "\\" <x>
 <dotnum> 为 <snum> "," <snum> "," <snum> "," <snum>
 <number> 为 <d> | <d> <number>
 <CRLF> 为 <CR> <LF>
 <CR> 为 回车符 (ASCII 码 13)
 <LF> 为 (ASCII 码 10)
 <SP> 为 空格 (ASCII 码 32)
 <snum> 为由一个、两个或三个数字组成的介于 0~255 之间的数字
 <a> 为 所有 A~Z 的 52 个大小写英文字母
 <c> 为 128 个 ASCII 字符，但不包括空格和特殊字符
 <d> 为 0~9 数字
 <q> 为不包括<CR>、<LF>或\的 128 个 ASCII 字符
 <x> 为所有 128 个 ASCII 字符
 <special> 为 "<" | ">" | "(" | ")" | "[" | "]" | "\" | "." | "," | ";" | ":"
 | "@" | "" 或控制字符

"\"为转意符，程序开发人员对这个转意符都很熟悉。它表示在其后的字符代表另外的意义。有时候名称的转变机制可能不知道主机，这就造成了通信的阻塞。为了解决这个问题，可以采取两种方法：一种方法是在#"后加入一个十进制数表示主机地址；另一种方法是在其后加入 32 位的 IP 地址，IP 地址的形式是由句号分隔的四个介于 0~255 之间的十进制数。时间戳行和返回路径行的格式通常由下面定义：

<return-path-line> 为 "Return-Path:" <SP><reverse-path><CRLF>
 <time-stamp-line> 为 "Received:" <SP> <stamp> <CRLF>
 <stamp> 为 <from-domain> <by-domain> <opt-info> ";" <daytime>
 <from-domain> 为 "FROM" <SP> <域> <SP>
 <by-domain> 为 "BY" <SP> <域> <SP>
 <opt-info> 为 [<via>] [<with>] |<id>] [<for>]
 <via> 为 "VIA" <SP> <连接> <SP>
 <with> 为 "WITH" <SP> <协议> <SP>
 <id> 为 "ID" <SP> <字符串> <SP>
 <for> 为 "FOR" <SP> <路径> <SP>
 <连接> 为在网络信息中心注册的连接的标准名称
 <协议> 为在网络中心注册的协议的名称
 <daytime> 为 <SP> <日期> <SP> <时间>
 <日期> 为 <日> <SP> <月> <SP> <年>
 <时间> 为 <小时> ":" <分> ":" <秒> <SP> <时区>

<dd> 为由一个或两个数字组成的每月 1~31 日
<月> 为 "JAN" | "FEB" | "MAR" | "APR" | "MAY" | "JUN" | "JUL" | "AUG" | "SEP"
| "OCT" | "NOV" | "DEC"
<年> 为由两位数字表示本世界的年代 00~99
<小时> 为每天的 24 小时，由 0~24
<分> 为每小时的分钟数 0~59
<秒> 为每分钟的秒数 0~59
<时区> 为全球标准时区

5.1.3 SMTP 命令

● HELO

该命令用于向接收 SMTP 确认发送 SMTP。参数域包括发送 SMTP 的主机名。接收 SMTP 通过连接确认命令来向发送 SMTP 确认接收 SMTP。该命令和 OK 响应确认发送和接收 SMTP 进入了初始状态，也就是说，没有操作正在执行，所有状态表和缓冲区已经被清除。

● MAIL

该命令用于开始将邮件发送到一个或多个邮箱中。参数域包括回复路径。返回路径中包括可选的主机和发送者邮箱列表。当有主机列表时，它是一个回复路径源，它说明此邮箱是由在表中的主机——传递发送（第一个主机是最后一个接收到此邮件的主机）过来的。此表也有向发送者返回非传递信号的源路径。比如从 aaa@xxx.com 发送邮件，可用如下命令：

```
mail from:aaa@xxx.com
```

该命令清除回复路径缓冲区。转发路径缓冲区和邮件内容缓冲区，并且将此命令的回复路径信息插入到回复路径缓冲区中。

● RCPT (RECIPIENT)

该命令用于确定邮件内容的惟一接收者；多个接收者将由多个该命令指定。转发路径中包括一个可选的主机和一个必需的目的邮箱。当出现主机列表时，这就是一个源路径，它指明邮件必须向列表中的上一个主机发送。

● DATA

接收者将跟在命令后的行作为邮件内容。该命令导致此命令后的邮件内容加入邮件内容缓冲区。邮件内容可以包括所有 128 个 ASCII 码字符。邮件内容由只包括一个句号的行结束，也就是字符序列<CRLF>.<CRLF>，它表示邮件的结束。邮件内容结束时，SMTP 服务器要处理并保存邮件内容。如果操作成功，接收者必须返回 OK 应答；如果失败，也必须返回失败应答。当 SMTP 服务器接收到一条信息时，无论是用作转发还是此邮件已经到达目的地，都必须在邮件内容的开始处加上时间戳这一行，这一行指示了接收到邮件主机和发出此邮件主机的标志，以及接收到邮件内容的时间和日期。转发的信件将有多行这样的时间戳。当接收 SMTP 作最后一站的传送时，它将返回路径信息行插入邮件中。此行包括发送命令中的<reverse-path>的信息。在这里，最后一站的传送的意思是邮件将被送到目的用户手中，但在一些情况下，邮件可能需要进一步加工并由另外的邮件系统传送。

- **SEND**

该命令用于开始一个发送命令，将邮件发送到一个或多个终端上。参数域包括一个回复路径，此命令如果成功，就将邮件发送到终端上了。回复路径包括一个可选的主机列表和发送者邮箱。当出现主机列表时，表示这是一个传送路径，邮件就是经过这个路径上的每个主机发送到这里的（列表上第一个主机是最后经手的主机）。此表用于返回非传递信号到发送者。

- **SOML (SEND OR MAIL)**

该命令用于开始一个邮件操作，并将邮件内容传送到一个或多个终端上，或者传送到邮箱中。对于每个接收者，如果接收者终端打开，邮件内容将被传送到接收者的终端上，否则就送到接收者的邮箱中。参数域包括回复路径，如果成功地将信息送到终端或邮箱中，则此命令成功。回复路径包括一个可选的主机列表和发送者邮箱。当出现主机列表时，表示这是一个传送路径，邮件就是经过这个路径上的每个主机发送到这里的（列表上第一个主机是最后经手的主机）。此表用于返回非传递信号到发送者。

- **SAML (SEND AND MAIL)**

该命令用于开始一个邮件操作，将邮件内容传送到一个或多个终端上，并传送到邮箱中。如果接收者终端打开，邮件内容将被传送到接收者的终端上和接收者的邮箱中。参数域包括回复路径。如果成功地将信息送到邮箱中，此命令成功。回复路径包括一个可选的主机列表和发送者邮箱。当出现主机列表时，表示这是一个传送路径，邮件就是经过这个路径上的每个主机发送到这里的（列表上第一个主机是最后经手的主机）。此表用于返回非传递信号到发送者。

- **RSET (RESET)**

该命令指示当前邮件操作将被放弃。任何保存的发送者、接收者和邮件内容应该被抛弃，所有缓冲区和状态表应该被清除，接收方必须返回OK应答。

- **VRFY (VERIFY)**

该命令要求接收者确认参数是一个用户。如果这是（已经知道的）用户名，返回用户的全名和指定的邮箱。此命令对回复路径缓冲区、转发路径缓冲区和邮件内容缓冲区没有影响。

- **EXPN (EXPAND)**

该命令要求接收者确认参数指定了一个邮件发送列表。如果是一个邮件发送列表，就返回表中的成员。如果是（已经知道的）用户名，返回用户的全名和指定的邮箱。此命令对回复路径缓冲区、转发路径缓冲区和邮件内容缓冲区没有影响。

- **HELP**

该命令导致接收者向 HELP 命令的发送者发出帮助信息。此命令可以带参数，并返回特定的信息作为应答。此命令对回复路径缓冲区、转发路径缓冲区和邮件内容缓冲区没有影响。在实际应用中，许多 SMTP 服务器对该命令不识别。

- **NOOP**

该命令只是说明没有任何操作，也不影响任何参数和已经发出的命令。此命令对回复路径缓冲区、转发路径缓冲区和邮件内容缓冲区没有影响。

- QUIT

该命令指示接收方必须发送 OK 应答然后关闭传送信道。接收方在接到 QUIT 命令并做出响应之前不应该关闭通信信道。发送方在发送 QUIT 命令和接收到响应之前也不应该关闭信道。

- TURN (TURN)

该命令用于指定接收方要么发送 OK 应答并改变角色为发送 SMTP, 要么发送拒绝信息并保持自己的角色。如果程序 A 现在是发送 SMTP, 它发出 TURN 命令后接收到 OK (250) 应答, 它就变成了接收 SMTP。程序 A 就进入初始状态, 好像通信信道刚打开一样, 这时它发送 220 准备好服务信号。如果程序 B 现在是接收 SMTP, 它发出 TURN 命令后接收到 OK (250) 应答, 它就变成了发送 SMTP。程序 A 就进入初始状态, 好像通信信道刚打开一样, 这时它准备接收 220 准备好服务信号。若要拒绝改变角色, 接收方可以发送 502 应答。

5.1.4 SMTP 应答码

作为接收方的 SMTP 服务器, 在接到任何命令后都要作出应答。表 5-1 列出 SMTP 应答码的性质及其意义。

表 5-1 SMTP 应答码的性质和意义

性 质	应答码	意 义
操作成功	220	<domain> 服务就绪
	221	<domain> 服务关闭传输信道
	421	<domain> 服务未就绪, 关闭传输信道
	250	要求的邮件操作完成
	251	用户非本地, 将转发向<forward-path>
	354	开始邮件输入, 以<CRLF>,<CRLF>结束
帮助信息	211	系统状态或系统帮助响应
	214	帮助信息
操作失败	450	要求的邮件操作未完成, 邮箱不可用 (例如, 邮箱忙)
	451	放弃要求的操作: 处理过程中出错
	452	系统存储不足, 要求的操作未执行
	500	格式错误, 命令不可识别
	501	参数格式错误
	502	命令不可实现
	503	错误的命令序列
	504	命令参数不可实现
	550	要求的邮件操作未完成, 邮箱不可用 (例如, 邮箱未找到, 或不可访问)
	552	过量的存储分配, 要求的操作未执行
	553	邮箱名不可用, 要求的操作未执行
	554	操作失败

5.1.5 SMTP示例

下面是一个SMTP的最小实现：

```
sender: HELO <domain> <CRLF>
receiver: 250 <domain><CRLF>

sender: MAIL FROM: aaa@xxx.com
receiver: 250 sender OK

sender: RCPT TO: bbb@xxx.com
receiver: 250 OK

sender: DATA
receiver: 354 input: end with <CRLF>. <CRLF>

sender: .....
sender: .....
sender: .....
receiver: 250 ok

sender: QUIT
receiver: 221 connection closed
```

5.1.6 ESMTP

ESMTP，顾名思义，就是SMTP的扩展（EXTENSION）。到目前为止，大多数SMTP服务器都升级成ESMTP了。ESMTP主要扩展了以下几个方面。

5.1.6.1 关于EHLO命令

在ESMTP里，不适用HELO，而使用EHLO命令，该命令格式如下：

```
ehlo-cmd ::= "EHLO" <SP> domain <CR LF>
```

该命令成功时，服务器返回代码250；如果失败，返回代码550；如果出错，返回500, 501, 502, 504或421。比起SMTP协议标准，EHLO命令可以出现在任何HELO命令出现的地方，在成功发送一个HELO或EHLO命令后再次发送它会使服务器返回503。客户这时不能缓存服务器返回的任何信息。注意，每次开始SMTP扩展服务会话的时候，必须发送EHLO命令。如果服务器能够处理EHLO命令，它会返回代码250。这样，服务器和客户就处于初始状态，也就是说，所有的状态表和缓冲区已经准备完毕。通常这种响应是多行的，每行响应包括一个关键字，如果说有的话，还有一个或多个参数。响应的语法如下：

```
ehlo-ok-rsp ::= "250" domain [ SP greeting ] CR LF
/ ("250-" domain [ SP greeting ] CR LF
*( "250-" ehlo-line CR LF )
"250" SP ehlo-line CR LF )
```

```

greeting ::= 1*<除了 CR 或 LF 的所有字符>
ehlo-line ::= ehlo-keyword *( SP ehlo-param )
ehlo-keyword ::= (字母/数字) *(字母/数字/-)
ehlo-param ::= 1*<除了空格和控制字符外的字符>
ALPHA ::= <大写 A 到 Z, 小写 a 到 z>
DIGIT ::= <0 到 9>
CR ::= <回车, ASCII 码 13>
LF ::= <换行, ASCII 码 10>
SP ::= <空格, ASCII 码 32>

```

需要注意的是, ESMTP 对大小写是敏感的, 这与 SMTP 的规定不同。如果服务器不能列出它所支持的服务扩展, 就返回代码 554。在接收到这个代码后, 客户要么发送 HELO, 要么发送 QUIT 命令。有时候服务器接收到 EHLO 命令, 可是命令参数不可接受, 它就返回代码 501。如果服务器识别了 EHLO, 但对服务器扩展未实现, 则返回代码 502。如果服务器不再提供服务扩展, 则返回代码 421。在接收到这个代码后, 客户要么发送 HELO, 要么发送 QUIT 命令。如果服务器不支持服务扩展, 则返回 500, 服务器保持现有状态。在接收到这个代码后, 客户要么发送 HELO, 要么发送 QUIT 命令。有时候, SMTP 服务器会在接收到 EHLO 命令后因为某种原因关闭连接, 这种情况在原来的 SMTP 协议标准中未涉及。为了处理这种情况, 客户必须能够确认服务器是否能够工作, 他可以重新连接并发送 HELO 或 EHLO 命令。有些服务器在接收到一个 EHLO 命令后会拒绝接收新的 HELO 命令, 这时可以利用 RSET 命令重新启动, 然后再发送 HELO。

5.1.6.2 关于 MAIL FROM 和 RCPT TO 命令

许多 ESMTP 的服务扩展是在 MAIL FROM 和 RCPT TO 命令后加入一些参数来实现的。下面是这两个命令的参数格式:

```

esmtp-cmd ::= inner-esmtp-cmd [SP esmtp-parameters] CR LF
esmtp-parameters ::= esmtp-parameter *(SP esmtp-parameter)
esmtp-parameter ::= esmtp-keyword [= esmtp-value]
esmtp-keyword ::= (字母/数字) *(字母/数字/-)
esmtp-value ::= 1*<除了空格, "=" 和控制字符的所有字符>
inner-esmtp-cmd ::= ("MAIL FROM:" 返回路径) / ("RCPT TO:" 转发路径)

```

如果服务器不能识别或实现一个或多个 MAIL FROM 或 RCPT TO 参数, 它应该返回代码 555。如果这种情况只是暂时的, 服务器返回代码 455。其他返回代码请查阅相关资料, 这里不再详述了。服务器以服务扩展处理时, 它处理的任何信息都应该在包头上加上“服务扩展标记”以示区别。

5.1.6.3 命令流水扩展

凡进行过 SMTP 开发的人员都会有这样的体会: SMTP 虽然好用, 语法也简单, 但要求命令按一定序列发送。如果某个网络需要很长时间进行连接, 那 SMTP 运行的效果可就比较差了, SMTP 的时间就花费在等待一个个命令上, 而且还会产生其他问题, 比如, 在 SMTP 命令失败时清除 TCP 输入缓冲区, 有时这是没有必要的; 对一些命令会不讲道理地判断它为失败。例如一些服务器如果在上一个 REPT TO 失败后会不接收 DATA 命令, 而不管 RCPT

TO 之前的命令是否成功，而有些服务器则可以在 RCPT TO 命令失败后接收 DATA 命令。如果能够使 SMTP 客户端进行命令流水，也就是一次发送许多指令，就会提高运行效率。命令流水扩展框架如下：

- 此服务扩展的名称为流水（Pipelining）；
- 与 EHLO 相关联的扩展值是 PIPELINING；
- PIPELINING EHLO 不带参数；
- MAIL FROM 或 RCPT TO 命令不附加其他参数；
- 没有附加其他 SMTP 命令。

1. 客户使用流水

在客户知道服务器可以支持流水的时候，客户可以传输多个命令（命令组）到服务器，不用发送一条等待一次应答，然后再发下一条，特别的 RSET、MAIL FROM、SEND FROM、SOML FROM、SAML FROM 和 RCPT TO 可以出现在命令组的任何地方。EHLO、DATA、VRFY、EXPN、TURN、QUIT 和 NOOP 只能出现在命令组中的最后位置，因为它们成功与否将改变 SMTP 命令所处的状态。由其他 SMTP 扩展产生的命令只能出现在组中的最后位置。实际传送的命令可以是组中的第一个命令。客户 SMTP 必须检查与组中数据相关的状态。如果 RCPT TO 接收地址未被接受，客户端必须检查 DATA 的状态，客户端不能因为 RCPT TO 没有成功就认为 DATA 一定失败。如果 DATA 命令被正确拒绝，客户端可以发出 RSET；如果 DATA 命令没有被正确拒绝，客户端要发出一个点（dot）。命令所产生的状态必须和分别发出每个命令时的状态相同，必须支持多行（Multiline）响应。客户 SMTP 可以选择在非阻塞状态运行，它在接收到服务器的响应时立即处理，即使还有数据需要发送也不能推迟对响应的处理。如果不支持非阻塞状态，客户端必须检查 TCP 窗口的大小，TCP 窗口的大小必须大于命令组的大小（窗口大小经常是 4KB）。如果不能进行这样的检查，可能会导致死锁。

2. 服务器对流水的支持

服务器应该提供如下服务扩展：

在任何情况下不得将 TCP 输入缓冲区的内容丢弃；

当且仅当接收到一个或多个有效的 RCPT TO 命令时，才对 DATA 命令主动发出响应；

如果因为 DATA 命令没有合法的接收者，结果接收到空信息时，不要再发出消息给任何人；

对成组的 RSET、MAIL FROM、SEND FROM、SOML FROM、SAML FROM 和 RCPT TO 命令的响应先保存起来，然后一起发送；

不允许缓存对 EHLO、DATA、VRFY、EXPN、TURN、QUIT 和 NOOP 的响应；

不允许缓存不可识别命令的响应；

在本地 TCP 输入缓冲区为空时，必须将未发出的响应全部发出；

不允许对未接收到的命令进行猜测，或假设它的存在；

在响应的文本信息中应该表明这是对哪个命令进行的响应。

5.1.6.4 ESMTP 文件格式

ESMTP 支持 MIME 格式，MIME 是 Multimedia Internet Mail Extensions 的缩写，也就是邮件可以传输多媒体文件。邮件分为两部分，一部分为邮件头，一部分为邮件体。邮件体可以不是文本的，但邮件头必须是文本的。下面是一个 MIME 头：

```
Mime-Version: 1.0  
Content-Type: multipart/mixed;  
boundary="====000_Dragon333388350162===="  
Message-Id: <20020419091412.407061D42F8F0@smtp.263.net>
```

5.1.6.5 关于身份认证

SMTP 有一个漏洞，就是可以以虚假身份发送邮件，这就为一些垃圾邮件制造了方便。许多垃圾邮件发送者以别人的名义制造大量的垃圾邮件，使许多 SMTP 成为垃圾邮件的中继站，为互联网的管理带来极大的不便。为了防止一些用户滥发垃圾邮件，SMTP 增添了身份认证功能。身份认证机制采用了加密技术，以防密码被截获。ESMTP 以 AUTH 命令进行身份认证。该命令格式如下：

```
AUTH <认证机制>
```

认证机制有许多种，如 LOGIN、GSSAPI、NTLM、CRAM-MD5 等。大多数 ESMTP 服务器支持 LOGIN 机制。当用户发出 AUTH LOGIN 命令后，ESMTP 服务器返回 334 等待用户输入。当用户输入后，ESMTP 服务器再返回 334 等待用户输入。如果身份认证通过，ESMTP 服务器返回 235；反之返回认证失败的信息。AUTH 命令使用之前，必须先发出 EHLO 命令初始化服务器。

需要注意的是，ESMTP 的 LOGIN 认证机制使用的是 Base64 编码，所以要将用户名和密码转换成 Base64 编码后再发送给服务器。如果将“username:”转换成 Base64 格式，就变成“dXNlcmlhbWU6”；将“password:”转换成 Base64 格式，就变成“cGFzc3dvcmQ6”。

本节为了编程需要，只是简单地介绍了一下 SMTP 协议，如果读者要进一步了解 SMTP，可以参考 RFC0821、RFC1425 等。

5.2 邮件发送程序开发

上一节已经简单地介绍了 SMTP 协议，这一节利用 TcpClient 类开发一个 SMTP 客户端，以发送邮件。该邮件发送程序支持身份认证功能。

5.2.1 身份认证

身份认证时首先要将用户名和密码转换成 Base64 编码，把字节数组转化成 Base64 编码，可以用 System.Security.Cryptography 命名空间下的 ToBase64Transform 类的 TransformBlock 方法。该方法的原型如下：

```
public int TransformBlock(  
    byte[] inputBuffer,
```

```
    int inputOffset,  
    int inputCount,  
    byte[] outputBuffer,  
    int outputOffset  
)
```

该方法有 5 个参数，第一个参数是存放原始数据的字节数组，第二个参数是开始转换的位置，第三个参数是转换的总字节数，第四个参数是存放输出数据的字节数组，第五个参数是输出的起始位置。

需要注意的是，该方法每次可以转换 3 个 ASCII 码字节，并输出 4 个目标字节，所以每一个超过三位的字符串，首先要分割成以 3 为单位的数个字符串，然后将这些字符串转化成字节数组，再用 TransformBlock 方法转化成 Base64 码，最后再将这些输出数据加起来即可。如果最后剩下的字符串不满 3 位，可以添加空字符（即“\0”）。下面代码演示了如何将用户名字符串转换成以 Base64 表示的 ASCII 字符串：

```
string username=null;  
byte[] originalByte=new Byte[1024];  
byte[] aimByte=new Byte[1024];  
  
int x=textBox7.Text.Length;  
string originalString1(textBox7.Text);  
  
if(x%3==1)  
{  
    originalString1=textBox7.Text+"\0\0";  
}  
if(x%3==2)  
{  
    originalString1=textBox7.Text+"\0";  
}  
int y=0;  
int z=originalString1.Length;  
  
while(z!=0)  
{  
    string originalString2=originalString1.Substring(y*3,3);  
    y++;  
    z=z-3;  
    originalByte=System.Text.Encoding.ASCII.GetBytes(originalString2);  
    ToBase64Transform aa=new ToBase64Transform();  
    aa.TransformBlock(originalByte,0,originalByte.Length,aimByte,0);  
    string ss=System.Text.Encoding.ASCII.GetString(aimByte,0,aimByte.Length);  
    ss=ss.Substring(0,4);  
}
```

```

    username=username+ss;
}

```

将密码字符串转换成以 Base64 表示的 ASCII 字符串，可参考上面代码进行。

5.2.2 发送命令

SMTP 服务器一般都识别 UTF8 码，所以发送命令使用 UTF8 编码（也可使用 ASCII 码）。每一个命令均以回车加换行符结束。下列代码实现命令的发送功能。

```

private void WriteToNetStream(ref NetworkStream NetStream, string Command)
{
    string stringToSend = Command + "\r\n";
    Byte[] arrayToSend = System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
}

```

5.2.3 应答码的接受

SMTP 服务器对每一个命令都发出响应。下列代码实现接受应答码的功能。

```

private string ReadFromNetStream(ref NetworkStream NetStream)
{
    byte[] bb=new byte[512];
    NetStream.Read(bb, 0, bb.Length);
    string read=System.Text.Encoding.UTF8.GetString(bb);
    return read;
}

```

5.2.4 发送邮件

客户端发出 DATA 命令，在服务器作出 354 应答后，即可开始邮件内容的发送。邮件以<CRLF>.<CRLF>结束。下面代码实现了邮件发送功能：

```

private void sendMail(ref NetworkStream NetStream, string message)
{
    Byte[] arrayToSend =
        System.Text.Encoding.UTF8.GetBytes(message.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
}

```

 注意：在结束邮件发送时，只要将上述方法的第二个参数设置为<CRLF>.<CRLF>（即"\r\n.\r\n"）。

5.2.5 应答码检查

发送邮件客户端必须要检查应答码，以判断服务器是否已经正确执行了命令。如果服务器没有正确执行命令，则重新发送命令或采取其他措施。下列代码可以检查服务器是否正确地执行了命令。

```
private string checkForError(string strMessage, string check)
{
    if (strMessage.IndexOf(check) == -1)
    {
        return "err";
    }
    else
    {
        return "correct";
    }
}
```

上述方法的第一个参数是服务器返回信息，第二个参数是要检查的应答码。如果服务器返回的信息中不存在第二个参数的字符串，上述方法将返回“err”（错误），表明服务器没有正确执行命令。

5.2.6 邮件发送程序开发

首先建立一个新的C#项目，然后如图5-1所示，在窗体上拖放四个GroupBox控件、六个TextBox控件、四个Button控件和两个RichTextBox控件。其中“服务器”文本框为textBox1，“端口”文本框为textBox2，“邮件主题”文本框为textBox3，“发件人”文本框为textBox4，“收件人”文本框为textBox5，“抄送”文本框为textBox6。“连接”按钮为button1，“发送”按钮为button2，“保存应答信息”按钮为button3，“关闭连接”按钮为button4。“服务器应答信息”文本框为richTextBox1，“信件内容”文本框为richTextBox2。

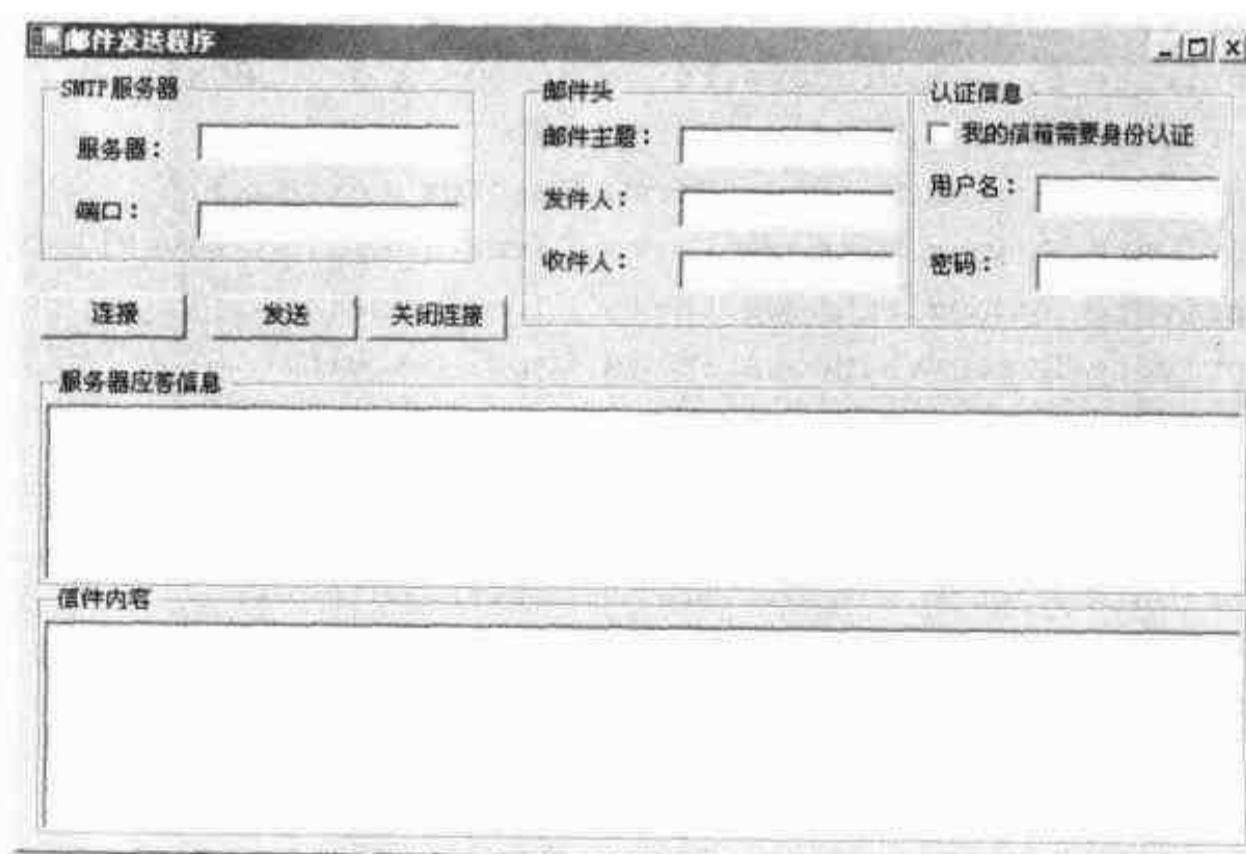


图5-1 邮件发送程序界面

下面是该程序的代码。

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
using System.Security.Cryptography;

namespace sendmail
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private TcpClient client;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.RichTextBox richTextBox2;
        private System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.GroupBox groupBox5;
        private System.Windows.Forms.CheckBox checkBox1;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
```

```
private System.Windows.Forms.TextBox textBox6;
private System.Windows.Forms.TextBox textBox7;
/// <summary>
/// 必需的设计器变量
/// </summary>
private System.ComponentModel.Container components = null;
public Form1()
{
    // Windows 窗体设计器支持所必需的
    InitializeComponent();
    textBox1.Text="smtp.263.net";
    textBox2.Text="25";
    textBox3.Text="这是试验邮件";
    // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
}
/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}
//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
#endregion
//上面系统自动产生的代码已经删除      /// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}
private void button2_Click(object sender, System.EventArgs e)
{
    if(checkBox1.Checked){
        Thread thread=new Thread(new ThreadStart(send1));
        thread.Start();
    }
}
```

```
        }

        else{
            Thread thread=new Thread(new ThreadStart(send2));
            thread.Start();
        }
    }

    private void send1()
    {
        string come=null;
        string check=null;
        NetworkStream stream=client.GetStream();
        string hostName=Dns.GetHostName();

        WriteToNetStream(ref stream,"EHLO "+hostName);
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("EHLO 应答: "+come+"\r\n");
        check=checkForError(come,"250");
        int round=0;
        while(check=="err"&&round<5)
        {
            round++;
            WriteToNetStream(ref stream,"EHLO "+hostName);
            come=ReadFromNetStream(ref stream);
            richTextBox1.AppendText ("EHLO 应答: "+come+"\r\n");
            check=checkForError(come,"250");
        }
        //*****将用户名转换为 Base-64 码*****
        string username=null;
        byte[] originalByte=new Byte[1024];
        byte[] aimByte=new Byte[1024];
        int x=textBox6.Text.Length;
        string originalString1(textBox6.Text;
        if(x%3==1)
        {
            originalString1=textBox6.Text+"\0\0";
        }
        if(x%3==2)
        {
            originalString1=textBox6.Text+"\0";
        }
        int y=0;
        int z=originalString1.Length;

        while(z!=0)
        {
            string originalString2=originalString1.Substring(y*3,3);

```

```
        y++;
        z=z-3;
        originalByte=
            System.Text.Encoding.ASCII.GetBytes(originalString2);
        ToBase64Transform aa=new ToBase64Transform();
        aa.TransformBlock(originalByte,0,
            originalByte.Length,aimByte,0);
        string ss=
            System.Text.Encoding.ASCII.GetString
            (aimByte,0,aimByte.Length);
        ss=ss.Substring(0,4);
        username=username+ss;
    }

//*****将密码转换为Base-64码*****
string password=null;
originalByte=new Byte[1024];
aimByte=new Byte[1024];
x=textBox7.Text.Length;
originalString1(textBox7.Text);
if(x%3==1)
{
    originalString1=textBox7.Text+"\0\0";
}
if(x%3==2)
{
    originalString1=textBox7.Text+"\0";
}
y=0;
z=originalString1.Length;
while(z!=0)
{
    string originalString2=originalString1.Substring(y*3,3);
    y++;
    z=z-3;
    originalByte=
        System.Text.Encoding.ASCII.GetBytes(originalString2);
    ToBase64Transform aa=new ToBase64Transform();
    aa.TransformBlock(originalByte,0,
        originalByte.Length,aimByte,0);
    string ss=
        System.Text.Encoding.ASCII.GetString(aimByte,
        0,aimByte.Length);
    ss=ss.Substring(0,4);
    password=password+ss;
}

WriteToNetStream(ref stream,"AUTH LOGIN");
```

```
come=ReadFromNetStream(ref stream);
richTextBox1.AppendText ("AUTH LOGIN 应答: "+come+"3\r\n");
check=checkForError(come, "334");
round=0;
while(check=="err"&&round<5)
{
    round++;
    WriteToNetStream(ref stream, "AUTH LOGIN");
    come=ReadFromNetStream(ref stream);
    richTextBox1.AppendText ("EHLO 应答: "+come+"\r\n");
    check=checkForError(come, "334");
}
WriteToNetStream(ref stream,username);
come=ReadFromNetStream(ref stream);
richTextBox1.AppendText ("用户名应答: "+come+"\r\n");
check=checkForError(come, "334");
round=0;
while(check=="err"&&round<5)
{
    round++;
    WriteToNetStream(ref stream,username);
    come=ReadFromNetStream(ref stream);
    richTextBox1.AppendText ("用户名应答: "+come+"\r\n");
    check=checkForError(come, "334");
}
WriteToNetStream(ref stream,password);
come=ReadFromNetStream(ref stream);
richTextBox1.AppendText ("密码应答: "+come+"3\r\n");
check=checkForError(come, "235");
round=5;
while(check=="err"&&round<5)
{
    round++;
    WriteToNetStream(ref stream,"EHLO "+hostName);
    come=ReadFromNetStream(ref stream);
    richTextBox1.AppendText ("EHLO 应答: "+come+"\r\n");
    check=checkForError(come, "250");
    int round0=0;
    while(check=="err"&&round<5)
    {
        round0++;
        WriteToNetStream(ref stream, "EHLO "+hostName);
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("EHLO 应答: "+come+"\r\n");
        check=checkForError(come, "250");
    }
}
```

```
        WriteToNetStream(ref stream, "AUTH LOGIN");
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("AUTH LOGIN 应答: "+come+"3\r\n");
        WriteToNetStream(ref stream,username);
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("用户名应答: "+come+"3\r\n");
        WriteToNetStream(ref stream,password);
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("密码应答: "+come+"3\r\n");
        check=checkForError(come, "235");
    }
    //*****以上是准备阶段*****
    WriteToNetStream(ref stream, "MAIL FROM:<" + textBox4.Text + ">");
    come=ReadFromNetStream(ref stream);
    richTextBox1.AppendText ("MAIL 应答: "+come+"\r\n");
    check=checkForError(come, "250");
    round=0;
    while(check=="err"&&round<5)
    {
        round++;
        WriteToNetStream(ref stream,
            "MAIL FROM:<" + textBox4.Text + ">");
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("MAIL 应答: "+come+"\r\n");
        check=checkForError(come, "250");
    }
    WriteToNetStream(ref stream, "RCPT TO:<" + textBox5.Text + ">");
    come=ReadFromNetStream(ref stream);
    richTextBox1.AppendText ("RCPT 应答: "+come+"\r\n");
    check=checkForError(come, "250");
    round=0;
    while(check=="err"&&round<5)
    {
        round++;
        WriteToNetStream(ref stream,
            "RCPT TO:<" + textBox5.Text + ">");
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("RCPT 应答: "+come+"\r\n");
        check=checkForError(come, "250");
    }
    WriteToNetStream(ref stream, "DATA");
    come=ReadFromNetStream(ref stream);
    richTextBox1.AppendText ("DATA 应答: "+come+"\r\n");
    check=checkForError(come, "354");
    round=0;
    while(check=="err"&&round<5)
```

```
{  
    round++;  
    WriteToNetStream(ref stream, "DATA");  
    come=ReadFromNetStream(ref stream);  
    richTextBox1.AppendText ("DATA 应答: "+come+"\r\n");  
    check=checkForError(come, "354");  
}  
sendMail(ref stream, "Subject: "+textBox3.Text+"\n"  
    +richTextBox2.Text+"\r\n.\r\n");  
come=ReadFromNetStream(ref stream);  
richTextBox1.AppendText ("信已发完, 服务器应答: "+come+"\r\n");  
check=checkForError(come, "250");  
round=0;  
while(check=="err"&&round<5)  
{  
    round++;  
    sendMail(ref stream, "Subject: "+textBox3.Text+"\n"  
        +richTextBox2.Text+"\r\n.\r\n");  
    come=ReadFromNetStream(ref stream);  
    richTextBox1.AppendText ("信已发完, 服务器应答: "  
        +come+"\r\n");  
    check=checkForError(come, "250");  
}  
}  
private void send2()  
{  
    string come=null;  
    string check=null;  
    NetworkStream stream=client.GetStream();  
    string hostName=Dns.GetHostName();  
    WriteToNetStream(ref stream, "HELO "+hostName);  
    come=ReadFromNetStream(ref stream);  
    richTextBox1.AppendText ("HELO 应答: "+come+"\r\n");  
    check=checkForError(come, "250");  
    int round=0;  
    while(check=="err"&&round<5)  
{  
        round++;  
        WriteToNetStream(ref stream, "HELO zhou868");  
        come=ReadFromNetStream(ref stream);  
        richTextBox1.AppendText ("HELO 应答: "+come+"\r\n");  
        check=checkForError(come, "250");  
    }  
    WriteToNetStream(ref stream, "MAIL FROM:<"+textBox4.Text+">");  
    come=ReadFromNetStream(ref stream);  
    richTextBox1.AppendText ("MAIL 应答: "+come+"\r\n");  
}
```

```
check=checkForError(come,"250");
round=0;
while(check=="err"&&round<5)
{ round++;
  WriteToNetStream(ref stream,"MAIL FROM:<"  

    +textBox4.Text+">>");
  come=ReadFromNetStream(ref stream);
  richTextBox1.AppendText("MAIL 应答: "+come+"\r\n");
  check=checkForError(come,"250");
}  

WriteToNetStream(ref stream,"RCPT TO:<" +textBox5.Text+">>");
come=ReadFromNetStream(ref stream);
richTextBox1.AppendText("RCPT 应答: "+come+"\r\n");
check=checkForError(come,"250");
round=0;
while(check=="err"&&round<5)
{ round++;
  WriteToNetStream(ref stream,"RCPT TO:<"  

    +textBox5.Text+">>");
  come=ReadFromNetStream(ref stream);
  richTextBox1.AppendText("RCPT 应答: "+come+"\r\n");
  check=checkForError(come,"250");
}  

WriteToNetStream(ref stream,"DATA");
come=ReadFromNetStream(ref stream);
richTextBox1.AppendText("DATA 应答: "+come+"\r\n");
check=checkForError(come,"354");
round=0;
while(check=="err"&&round<5)
{ round++;
  WriteToNetStream(ref stream,"DATA");
  come=ReadFromNetStream(ref stream);
  richTextBox1.AppendText("DATA 应答: "+come+"\r\n");
  check=checkForError(come,"354");
}  

sendMail(ref stream,"Subject: "+textBox3.Text+"\n"
  +richTextBox2.Text+"\r\n.\r\n");
come=ReadFromNetStream(ref stream);
richTextBox1.AppendText("信已发完，服务器应答: "+come+"6\r\n");
check=checkForError(come,"250");
round=0;
while(check=="err"&&round<5)
{ round++;
  sendMail(ref stream,"Subject: "+textBox3.Text+"\n"
```

```
        +richTextBox2.Text+"\r\n.\r\n");
        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText("信已发完，服务器应答：“
        "+come+"\r\n");
        check=checkForError(come, "250");
    }
}

private void WriteToNetStream(ref NetworkStream NetStream,
    string Command)
{
    string stringToSend = Command + "\r\n";
    Byte[] arrayToSend =
        System.Text.Encoding.Default.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
}

private void sendMail(ref NetworkStream NetStream, string message){
    Byte[] arrayToSend =
        System.Text.Encoding.Default.GetBytes
        (message.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
}

private string ReadFromNetStream(ref NetworkStream NetStream)
{
    byte[] bb=new byte[512];
    NetStream.Read(bb,0,bb.Length);
    string read=System.Text.Encoding.Default.GetString(bb);
    return read;
}

private string checkForError(string strMessage,string check)
{
    if (strMessage.IndexOf(check) == -1)
    {
        return "err";
    }
    else
    {
        return "correct";
    }
}

private void button3_Click(object sender, System.EventArgs e)
{
    string come=null;
    NetworkStream stream=client.GetStream();
    WriteToNetStream(ref stream,"QUIT");
```

```

        come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("QUIT应答: "+come+"\r\n");
    }
    private void button1_Click(object sender, System.EventArgs e)
    {
        client=new TcpClient(textBox1.Text,
        Int32.Parse(textBox2.Text));
        NetworkStream stream=client.GetStream();
        string come=ReadFromNetStream(ref stream);
        richTextBox1.AppendText ("连接应答: "+come+"\r\n");
        string check=checkForError(come, "220");
        while(check=="err")
        {
            client.Connect("smtp.263.net",25);
            come=ReadFromNetStream(ref stream);
            richTextBox1.AppendText ("连接应答: "+come+"\r\n");
            check=checkForError(come, "220");
        }
    }
}

```

5.2.7 演示

下面用 smtp.263.net 向 zhou868@263.net 发送一封邮件。执行结果如图 5-2 所示。

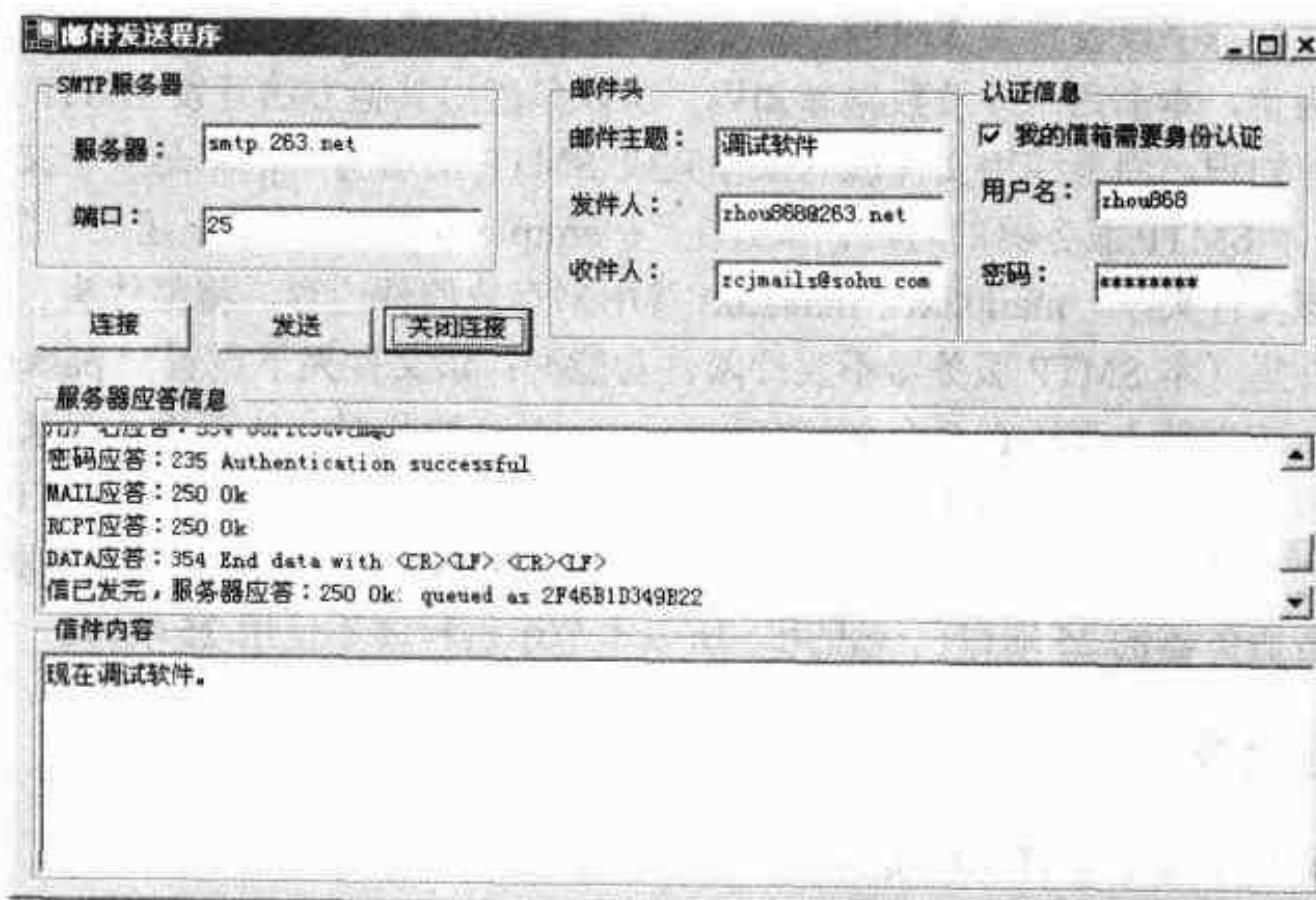


图 5-2 发送邮件

图 5-3 是用 FOXMAIL 接收的全部邮件内容。

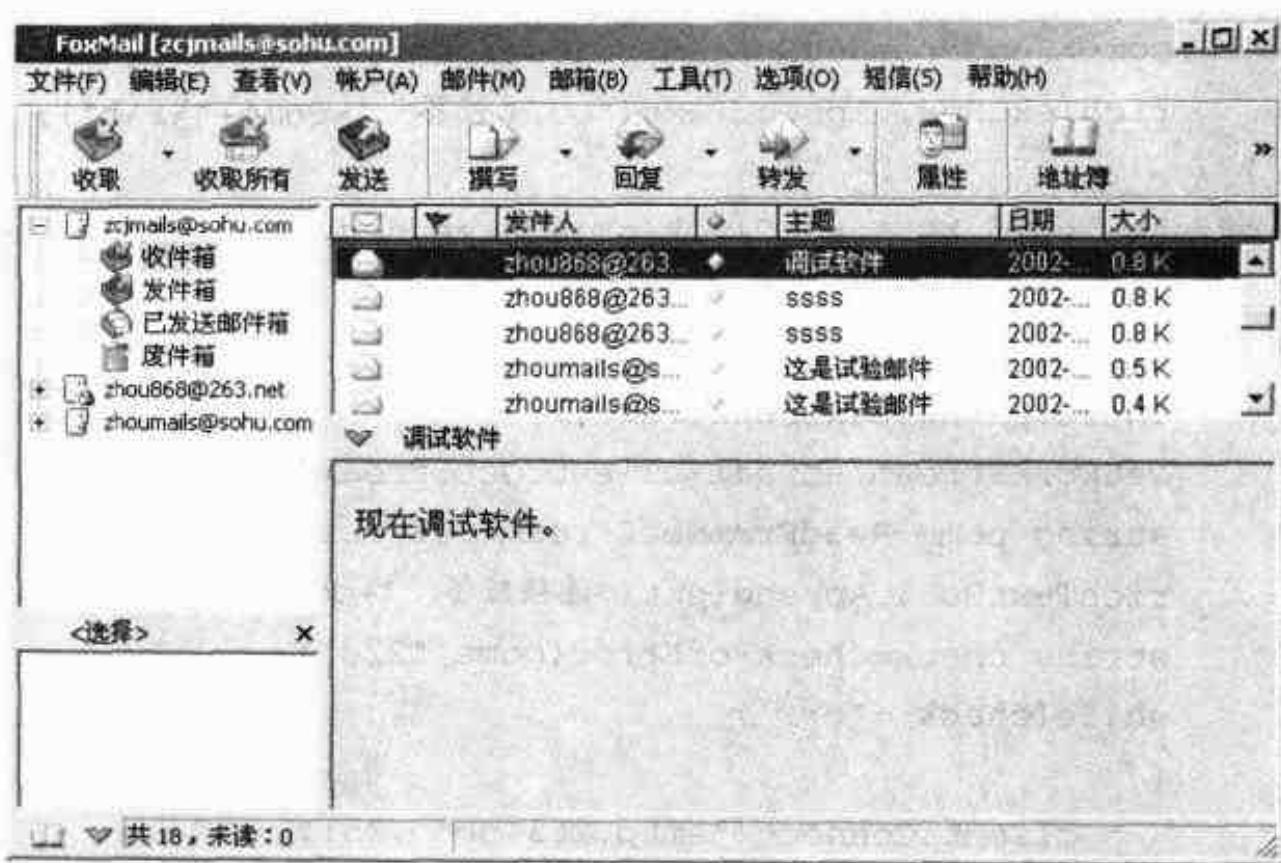


图 5-3 用 FOXMAIL 接收邮件

5.3 SMTP 服务器开发

一般而言，使用关系数据库开发 SMTP 服务器比较方便，可以把发件人的信箱、收件人的信箱、信件文件所处的位置、邮件状态等信息存储在数据库里。服务器在接到命令时，只要根据数据库的信息进行特定操作就可以了。比如，在数据库里设置一字段标明信件状态，1 为新信件，2 为已读信件，3 为删除信件。如果 POP3 用户想删除一特定信件，只要把数据库的信件状态字段设置为 3 即可，而不用真正删除信件。

到目前为止，本书还未涉及数据库知识，所以只能用其他方法开发 SMTP 服务器。本节不用数据库知识，而单纯用文件读写操作完成 SMTP 的服务。信箱是一个文件夹，比如 zhousunjie 是本 SMTP 服务器的用户，那么在“e:\smtp\mailbox”目录下建立一个 zhousunjie 文件夹。在该文件夹下，mail0.txt、mail1.txt 等用来存放信件内容，将邮件头、邮件内容均存放在该文件里（本 SMTP 服务器不支持附件功能）；该文件夹下设置一 pass.txt，用以存放账户密码；该文件夹下还有一个文件名称为 record.txt 的临时文件，用于记录 SMTP 客户端的操作，用户操作完后，删除该临时文件。该服务器不支持转发到其他 SMTP 服务器的功能。如果要支持该功能，可以把 5.2 节的程序与本节程序合并即可。本服务器使用 UTF8 码。由于笔者服务器的 25 端口已经占用，所以本节示例程序不使用 25 端口。

5.3.1 读取命令

下列代码用于读取来自客户端的全部信息：

```
private string readCommand(ref NetworkStream NetStream)
{
    byte[] byteMessage=new byte[1024];
    NetStream.Read(byteMessage, 0, byteMessage.Length);
```

```
    string command=System.Text.Encoding.UTF8.GetString(byteMessage);
    return command;
}
```

下列代码用于删除客户端信息的末尾空字符，截取来自客户端的有效命令：

```
private string bigCommand(string commString)
{
    int x=commString.IndexOf("\r\n");
    string command=commString.Substring(0,x);
    return command;
}
```

下列代码用于删除客户命令的参数，获取命令码：

```
private string splitCommand(string aimString)
{
    string[] aim=new string[2];
    int x=aimString.IndexOf(" ");
    if(x!=-1)
    {
        char[] a=new char[] {' '};
        aim=aimString.Split(a);
        return aim[0];
    }
    else{return aimString;}
}
```

下列代码用于删除客户命令的主命令码，只获取参数：

```
private string splitParameter(string aimString)
{
    string[] aim=new string[2];
    int x=aimString.IndexOf(" ");
    if(x!=-1)
    {
        char[] a=new char[] {' '};
        aim=aimString.Split(a);
        return aim[1];
    }
    else{return "";}
}
```

5.3.2 发送反馈

下列代码用于向客户端发送反馈信息：

```
private void sendFeedback(ref NetworkStream NetStream, string ToSend)
{
```

```

    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}

```

5.3.3 读取邮件内容

当客户端发送<CRLF>.<CRLF>时，就认为邮件内容已发送完毕。下列代码用于接收客户端发送的邮件内容（该方法与 5.3.1 节的代码基本相似，可以互换。为了编程时便于区别，故写成两个方法）：

```

private string readMail(ref NetworkStream NetStream)
{
    byte[] byteArray=new Byte[1024];
    NetStream.Read(byteArray,0,byteArray.Length);
    string mailMessage=System.Text.Encoding.UTF8.GetString(byteArray);
    return mailMessage;
}

```

5.3.4 获取邮箱字符串中的服务器名称

邮箱以 xxx@xxx.xxx 格式出现，“@”后为服务器名称，许多商业软件在发送 MAIL 和 RCPT 命令时，常以下面的形式出现：

```

MAIL FROM: <xxx@xxx.xxx>
RCPT TO: <xxx@xxx.xxx>

```

所以服务器必须能够识别<>内的字符串。下列代码用于获取服务器名称：

```

private string splitServer(string aimString)
{
    string[] aim=new string[2];
    char[] a=new char[] {'@'};
    aim=aimString.Split(a);
    int y=aim[1].IndexOf(">");
    if(y!=-1)
    {
        string server=aim[1].Substring(0,aim[1].Length-1);
        return server;
    }
    else{return aim[1];}
}

```

5.3.5 获取邮箱字符串中的邮箱名称

下列方法用于获取邮箱字符串中的邮箱名称，参数是 rcpt 命令的参数。rcpt 命令常以下列格式出现：rcpt to:zhou868@263.net，RCPT TO：<zhou868@263.net>，参数是 to:zhou868@263.net 或 To:<zhou868@263.net>。将该参数输入下列方法，可以获取字符串“zhou868”。

```
private string splitMailbox(string aimString)
{
    string[] aim=new string[2];
    char[] a=new char[] {'@'};
    aim=aimString.Split(a);
    int y=aim[0].IndexOf("<");
    if(y!=-1)
    {
        string mailbox=aim[0].Substring (y+1,aim[0].Length-y-1);
        return mailbox;
    }
    else
    {
        int x=aim[0].IndexOf(":");
        string mailbox=aim[0].Substring (x+1,aim[0].Length-x-1);
        return mailbox;
    }
}
```

5.3.6 SMTP 服务器开发

如图 5-4 所示，“监听端口”文本框是 textBox1，“客户信息”文本框是 richTextBox1。“开始服务”按钮为 button1，“关闭服务”按钮是 button2。

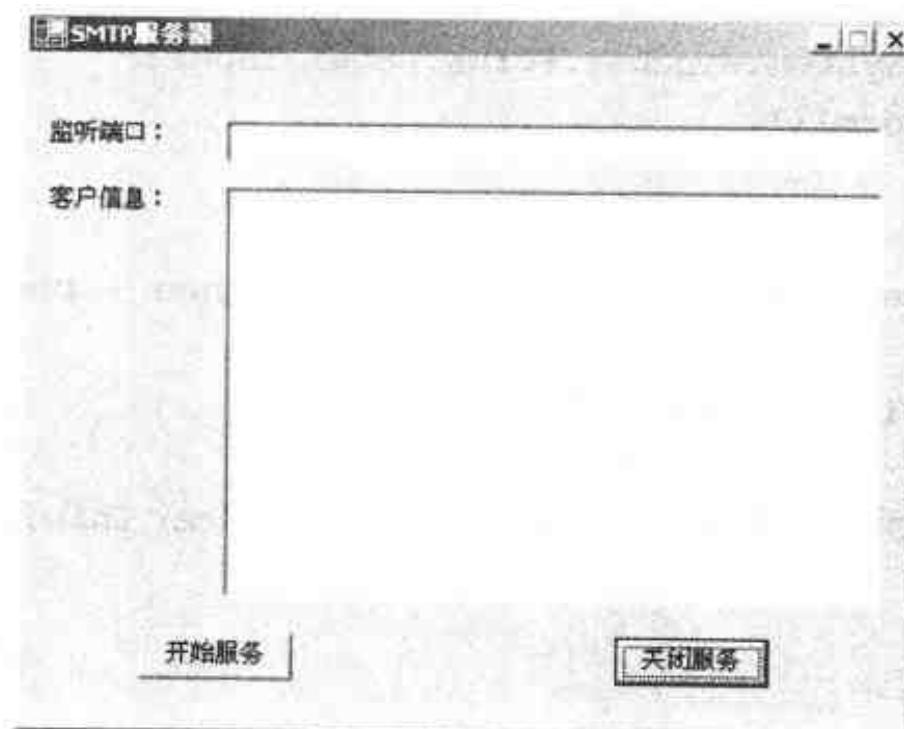


图 5-4 SMTP 服务器界面

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
namespace accep_so
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Button button2;
        /// <summary>
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.Container components = null;
        private bool control = false;
        Thread thread;
        private int port;
        private TcpListener listener;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.Label label2;
        public Form1()
        {
            //
            // Required for Windows Form Designer support
            //
            InitializeComponent();
            //
            // TODO: Add any constructor code after InitializeComponent call
            //
        }

        /// <summary>
        /// Clean up any resources being used.
    }
}
```

```
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        port =Int32.Parse(textBox1.Text);
    }
    catch{MessageBox.Show("您输入的格式不对! 请输入正整数。");}
}

try
{
    listener=new TcpListener(port);
    listener.Start();
    thread=new Thread(new ThreadStart(recieve));
    thread.Start();
}
catch(Exception ee){MessageBox.Show(ee.Message);}
}

private void button2_Click(object sender, System.EventArgs e)
{
    try
    {
        listener.Stop();
    }
}
```

```
        }

        catch(Exception ee){MessageBox.Show(ee.Message);}

    }

    private void recieve()
    {
        while(true)
        {
            newClient=listener.AcceptSocket();
            if(!newClient.Connected)
            {
                Thread thread=new Thread(new ThreadStart(round));
                thread.Start();
            }
        }
    }

    private void round()
    {
        bool mailfrom=false;
        bool rcptto=false;
        bool getMail=false;
        StreamWriter mail=null;
        int queue=0;
        string commString=null;
        string bigComm=null;
        string command=null;
        string parameter=null;
        string hostName=null;
        string mailbox=null;
        NetworkStream netStream=new NetworkStream(newClient);
        string from=null;
        sendFeedback(ref netStream,"220 SMTP Server Ready!");

        while(true)
        {
            //接受信息+ + +
            if(getMail==false)
            {
                commString=readCommand(ref netStream);
                bigComm=bigCommand(commString);
                command=splitCommand(bigComm);
                parameter=splitParameter(bigComm);
                command=command.ToLower();
                parameter=parameter.ToLower();
                hostName=Dns.GetHostName();
            }
        }
    }
}
```

```
richTextBox1.AppendText ("command:" +command+
    " "+ "parameter:" +parameter+"\r\n");
}

if(command=="hello")
{
    try
    {
        sendFeedback(ref netStream,"250 "+hostName);
        mailfrom=false;
        rcptto=false;
    }
    catch
    {
        sendFeedback(ref netStream,
            "421 unsuccessful,connection will be Closed");
        newClient.Close();
    }
}

else if(command=="mail")
{
    try
    {
        mailfrom=true;
        from=parameter;
        rcptto=false;
        sendFeedback(ref netStream,
            "250 "+parameter+"....sender OK!");
    }
    catch(sendFeedback(ref netStream,
        "502 performed unsuccessful"));
}

else if(command=="rcpt")
{
    bool round =true;
    int i=0;
    try
    {
        if(mailfrom==true)
        {
            rcptto=true;
            string host=splitServer(parameter);
            mailbox=splitMailbox(parameter);
            if(host==hostName)
            {
                bool mailboxExists=Directory.Exists
                    ("e:\\\\smtp\\\\mailbox\\\\"+mailbox);
            }
        }
    }
}
```

```
if(mailboxExists==true)
{
    while((round==true)&&(i<1024))
    {
        bool mailExists=
            File.Exists("e:\\smtp
                \\mailbox\\\"+mailbox
                +"\\\"+"mail"
                +i.ToString()+" .txt");
        if(mailExists==false)
        {
            queue=i;
            mail=new StreamWriter
                ("e:\\smtp\\\"+mailbox\\\"+
                +mailbox+"\\\"+
                +"mail"+queue.ToString()
                +" .txt",false,
                System.Text.Encoding.UTF8);
            mail.WriteLine(from);
            mail.WriteLine("\r\n");
            mail.Close();
            round=false;
        }
        i++;
    }
    sendFeedback(ref netStream,
        "250 "+parameter);
    if(round==true)
    {
        sendFeedback(ref netStream,
            "452 "+parameter
            +"space insufficient");
    }
}
else
{
    sendFeedback(ref netStream,
        "550 "+parameter+"does not exist");
}
}
else
{
    //下面可添加代码调动转发功能
    //上面可添加代码调用转发功能
}
}
```

```
        else
        {
            sendFeedback(ref netStream,
                "503 mail from command needed");
        }
    }
    catch(sendFeedback(ref netStream,
        "502 performed unsuccessful"));
}
else if(command=="data"||getMail==true)
{
    try
    {
        if((mailfrom==true)&&(rcptto==true))
        {
            bool round=true;
            if(getMail==false)
            {
                sendFeedback(ref netStream,
                    "354 input: end with <CR><LF>.<CR><LF>");
                getMail=true;
            }
            while(round==true)
            {
                string message=readMail(ref netStream);
                int checkEnd=message.IndexOf("\r\n.\r\n");
                if(checkEnd==-1)
                {
                    mail=new StreamWriter
                        ("e:\\smtp\\mailbox\\"
                        +mailbox+"\\\"+mail"
                        +queue.ToString()+" .txt",
                        true,System.Text.Encoding.UTF8);
                    mail.Write(message);
                    mail.Close();
                }
                else
                {
                    mail=new StreamWriter
                        ("e:\\smtp\\mailbox\\"
                        +mailbox+"\\\""
                        +"mail"+queue.ToString()
                        +" .txt",true,
                        System.Text.Encoding.UTF8);
                    mail.Write(message);
                    mail.Close();
                }
            }
        }
    }
}
```

```
        round=false;
        getMail=false;
    }
}
sendFeedback(ref netStream,"250 OK");
}
else{sendFeedback(ref netStream,
    "503 MAIL or RCPT command needed");}
}
catch{sendFeedback(ref netStream,
    "502 performed unsuccessful");}
}
else if(command=="help")
{
    thread.Resume();
    try
    {
        sendFeedback(ref netStream,"211 free smtp server
            ...and other help message.");
    }
    catch{sendFeedback(ref netStream,"502 performed
        unsuccessful,try again");}
}
else if(command=="quit")
{
    try
    {
        control=true;
        sendFeedback(ref netStream,
            "211 connection closed");
        newClient.Close();
    }
    catch{sendFeedback(ref netStream,
        "502 performed unsuccessful,try again");}
}
else if(getMail==false)
{
    sendFeedback(ref netStream,"500 unrecognized command");
}
}
private string readCommand(ref NetworkStream NetStream)
{
    byte[] byteMessage=new byte[1024];
```

```
NetStream.Read(byteMessage, 0, byteMessage.Length);
string command=
    System.Text.Encoding.UTF8.GetString(byteMessage);
return command;
}
private void sendFeedback(ref NetworkStream NetStream,
    string ToSend)
{
    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend =
        System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}
private string readMail(ref NetworkStream NetStream)
{
    byte[] byteArray=new Byte[1024];
    NetStream.Read(byteArray, 0, byteArray.Length);
    string mailMessage=
        System.Text.Encoding.UTF8.GetString(byteArray);
    return mailMessage;
}
private string splitServer(string aimString){
string[] aim=new string[2];
char[] a=new char[] {'@'};
aim=aimString.Split(a);
int y=aim[1].IndexOf(">");
if(y!=-1){
    string server=aim[1].Substring (0,aim[1].Length-1);
    return server;
}
else{
    return aim[1];
}
}
private string splitMailbox(string aimString)
{
    string[] aim=new string[2];
    char[] a=new char[] {'@'};
    aim=aimString.Split(a);
    int y=aim[0].IndexOf("<");
    if(y!=-1)
    {
        string mailbox=aim[0].Substring (y+1,aim[0].Length-y-1);
```

```
        return mailbox;
    }
    else{
        int x=aim[0].IndexOf(":");
        string mailbox=aim[0].Substring (x+1,aim[0].Length-x-1);
        return mailbox;
    }
}
private string splitCommand(string aimString)
{
    string[] aim=new string[2];
    int x=aimString.IndexOf(" ");
    if(x!=-1)
    {
        char[] a=new char[] {' '};
        aim=aimString.Split(a);
        return aim[0];
    }
    else{return aimString;}
}

private string splitParameter(string aimString)
{
    string[] aim=new string[2];
    int x=aimString.IndexOf(" ");
    if(x!=-1)
    {
        string para=aimString.Substring(x+1,aimString.Length -x-1);
        return para;
    }
    else{return "";}
}

private string bigCommand(string commString){
    int x=commString.IndexOf("\r\n");
    string command=commString.Substring(0,x);
    return command;
}
}
```

5.3.7 演示

启动本节开发的服务器，在“监听端口”文本框里输入适当的端口，然后单击“开始服务”按钮，开始监听端口。然后启动5.2节的示例程序，开始发送邮件。图5-5是服务器端的运行情况。



图 5-5 SMTP 服务器端的运行情况

图5-6是客户端运行的情况。

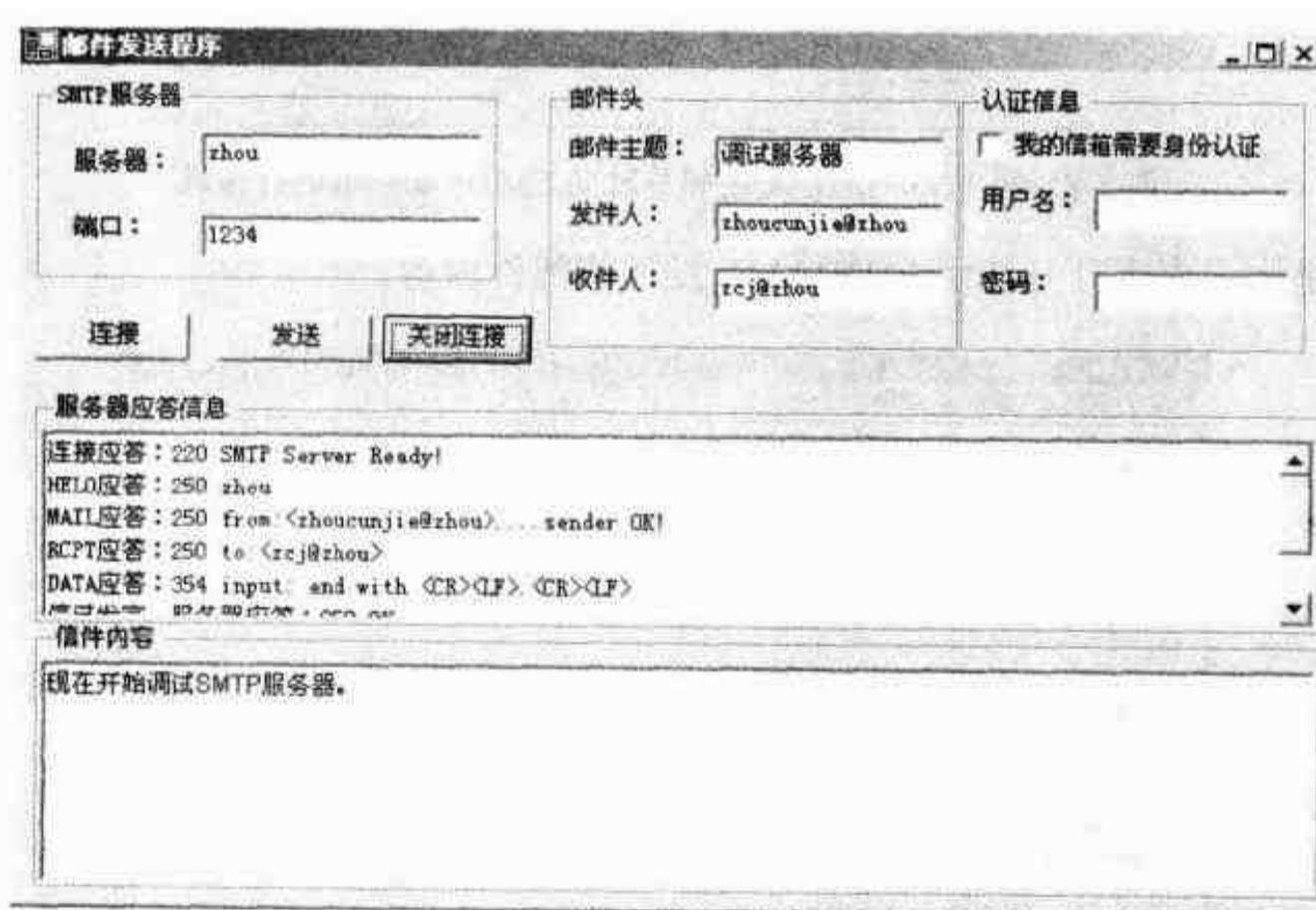


图 5-6 邮件接收客户端的运行情况

图5-7是用写字板打开的邮件内容。

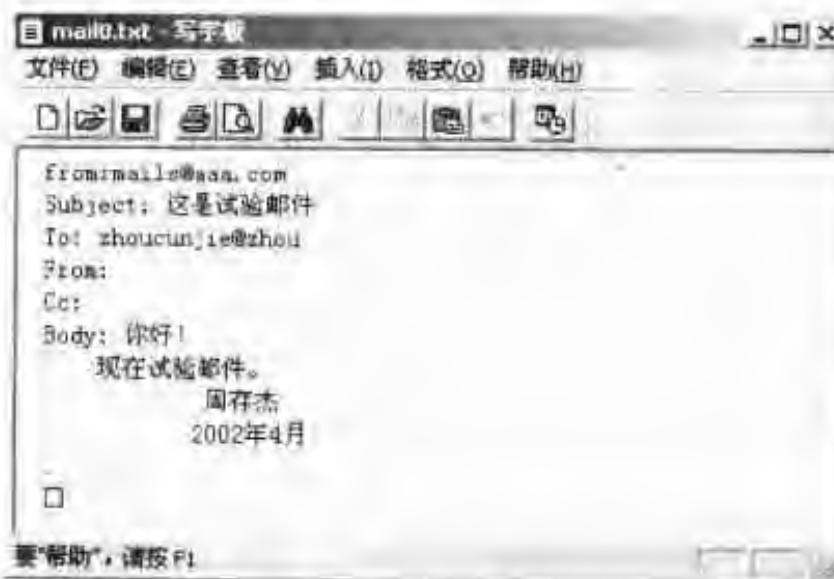


图 5-7 用写字板打开的邮件内容

图 5-8 是用 FOXMAIL 发送邮件时服务器的情形。



图 5-8 用 FOXMAIL 发送邮件时的 SMTP 服务器运行情况

图 5-9 是用记事本打开的用 FOXMAIL 发送的邮件内容。



图 5-9 用记事本打开的用 FOXMAIL 发送的邮件内容

5.3.8 改进意见

在学习本书数据库知识后，利用 SQL Server 数据库开发一个安全易用的 SMTP 服务器。

本章小结

本章在介绍 SMTP 协议的基础上，开发了一个邮件发送程序和 SMTP 服务器。在这两个程序的基础上加以改进，可以成为一个完善的商业 SMTP 系统。如果进一步结合 SQL Server 数据库，则可开发出安全、易用的 SMTP 服务系统。

第 6 章 POP3 协议开发

POP3 是 Post Office Protocol 3 的缩写，意为邮局协议 3，默认端口为 110。该协议用于处理邮件（从信箱读取邮件并发送给客户端，接收邮件，中转邮件等），就像一个普通邮局一样。

6.1 POP3 协议简介

6.1.1 POP3 协议命令格式

命令是由命令码和其后的参数组成，不区别大小写，每一个命令以回车换行符（CRLF）结束。命令与参数之间、参数与参数之间用空格隔开。下面是 POP3 命令格式：

- USER <SP> <name> <CRLF>
- PASS <SP> <password> <CRLF>
- STAT <CRLF>
- LIST <SP> [number] <CRLF>，比如 LIST 1<CRLF>
- RETR <SP> <number> <CRLF>
- DELE<SP> <number> <CRLF>
- NOOP<CRLF>
- RSET<CRLF>
- TOP<SP> <number> <SP> <number><CRLF>
- UIDL<SP> [number] <CRLF>
- APOP <SP><name><SP>< digest><CRLF>
- QUIT<CRLF>

6.1.2 POP3 命令参数

- USER <SP> <name> <CRLF>的参数为用户名。比如有一个邮箱为 `aaa@xxx.com`，那么“aaa”为用户名。
- PASS <SP> <password> <CRLF>的参数为用户密码，明码形式。
- LIST <SP> [number] <CRLF>的参数为信件序列号，即第 xx 封信。
- RETR <SP> <number> <CRLF>的参数为信件序列号，即第 xx 封信。
- DELE<SP> <number> <CRLF>的参数为信件序列号，即第 xx 封信。
- TOP<SP> <number> <SP> <number><CRLF>的第一个参数是信件序列号，第二个参数是非负数，即邮件的前 xx 行。

- UIDL<SP>[number]<CRLF>的参数是信件序列号。
- APOP <SP><name><SP>< digest><CRLF>的第一个参数是信箱名称，第二个参数是摘要。

6.1.3 POP3协议命令

- USER <SP><name><CRLF>

用于通知POP3服务器当前用户的用户名（信箱名）。限制：仅在USER和PASS命令失败后或在“确认”状态中使用。返回值：+OK表示有效邮箱；-ERR表示无效邮箱。

- PASS <SP><password><CRLF>

用于通知POP3服务器当前用户的口令（密码）。限制：仅在USER命令成功后使用。返回值：+OK表示邮件锁住并已经准备好；-ERR表示无效口令或无法使用排他锁锁住邮件。

- STAT <CRLF>

该命令用于通知POP3服务器开始准备接收邮件。限制：必须在USER、PASS命令成功后使用。返回值：+OK: xx yy，其中xx为信件总数，yy为信件总大小。

- LIST <SP>[number]<CRLF>

该命令用于通知服务器，客户端需要获取邮件列表。如果给出了参数，服务器返回“+OK”和指定邮件的序列号和大小。如果没有参数，服务器返回“+OK”确认响应和所有邮件的序列号和大小。命令执行失败，则返回“-ERR”。注意：被标记为删除的信件不在此列。

- RETR <SP><number><CRLF>

该命令用于通知服务器，客户端需要接收指定的邮件。命令执行成功，服务器返回“+OK”和邮件内容；命令执行失败，则返回“-ERR”。

- DELE<SP><number><CRLF>

该命令用于通知服务器，客户端需要删除指定的邮件。命令执行成功，服务器返回“+OK”和邮件内容；命令执行失败，则返回“-ERR”。

- NOOP<CRLF>

该命令只是说明没有任何操作，也不影响任何参数和已经发出的命令。命令执行成功，服务器返回“+OK”和邮件内容；命令执行失败，则返回“-ERR”。

- RSET<CRLF>

该命令指示当前邮件操作将被放弃。任何保存的发送者、接收者和邮件内容应该被抛弃，所有缓冲区和状态表应该被清除，接收方必须返回OK应答。

- TOP<SP><number><SP><number><CRLF>

该命令用于通知服务器，客户端要求获取指定邮件的前xx行。命令如果成功，返回“+OK”和制定信息；命令失败，则返回“-ERR”。

- UIDL<SP>[number]<CRLF>

该命令用于通知服务器，客户端要求获取邮件的“独立-ID表”。如果命令不带参数，返回所有邮件的“独立-ID表”；如果命令带有邮件，返回指定邮件的“独立-ID表”。

- APOP <SP><name><SP><digest><CRLF>

该命令用于加密认证用户身份。一般而言，POP3 会话多以 USER、PASS 开始，这导致用户名和口令在网络上的显式传送，可能造成泄密。用 APOP 命令可以解决这一问题。实现 APOP 命令的服务器包括一个标记确认的时间戳。例如：在 UNIX 上使用 APOP 命令的语法为：process-ID.clock@hostname，其中进程-ID 是进程的十进制的数，时钟是系统时钟的十进制表示，主机名与 POP3 服务器名一致。客户记录下此时间戳，然后发送 APOP 命令。name 语法和 USER 命令一致。Digest 是采用 MD5 算法产生的包括时间戳和共享密钥的字串。此密钥是客户和服务器共知的，应该注意保护此密钥，如果泄密，任何人都能够以用户身份进入服务器。如果服务器接到 APOP 命令，它验证 digest。如果正确，服务器返回“+OK”，进入“操作”状态；否则，给出“-ERR”并停留在“确认”状态。

- QUIT<CRLF>

该命令结束会话，命令成功，服务器返回“+OK”；如果命令失败，返回“-ERR”。

6.1.4 POP3 简单示例

建立连接……

客户端：USER xxxxxx

服务器：+OK

客户端：PASS yyyyyy

服务器：+OK

客户端：STAT

服务器：+OK

客户端：LIST

服务器：+OK 2 2048

服务器：1 1024

服务器：1 1024

客户端：RETR 1

服务器：+OK

服务器：信件内容

服务器：.....

服务器：.....

服务器：<CRLF>.<CRLF>

客户端：RETR 2

服务器：+OK

服务器: 信件内容
服务器:
服务器:
服务器: <CRLF>,<CRLF>

客户端: DELE 1
服务器: +OK

客户端: DELE 2
服务器: +OK

客户端: QUIT
服务器: +OK

6.2 邮件接收程序

本节利用 POP3 协议开发一个邮件接收程序，该程序将邮件全部内容都接收下来，不分开邮件头和邮件体。本节程序使用 USER、PASS、STAT、LIST、RETR、DELE、QUIT 命令。下面开始开发过程。

6.2.1 接收服务器应答

一般而言，接收服务器应答，既可使用 ASCII 码，也可使用 UTF8 码，这里使用 ASCII 码。下列方法用于接受服务器的应答。

```
private string ReadFromNetStream(ref NetworkStream NetStream)
{
    byte[] bb=new byte[1024];
    NetStream.Read(bb,0,bb.Length);
    string read=System.Text.Encoding.ASCII.GetString(bb);
    return read;
}
```

6.2.2 发送命令码

发送命令码，既可使用 ASCII 码，也可使用 UTF8 码，这里使用 ASCII 码。下列方法用于向服务器发送命令码。

```
private void WriteToNetStream(ref NetworkStream NetStream, string Command)
{
    string stringToSend = Command + "\r\n";
    Byte[] arrayToSend = System.Text.Encoding.ASCII.GetBytes
```

```

        (stringToSend.ToCharArray());
        NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    }
}

```

6.2.3 接收邮件

接收邮件使用 UTF8 码。当遇到<CRLF><CRLF>，则结束读取数据。下列代码用于接收邮件。

```

private void ReadMail(ref NetworkStream NetStream,int number)
{
    int k=0;
    bool check=false;
    byte[] bb=new byte[6400];
    while(!check)
    {
        k=NetStream.Read(bb,0,bb.Length);
        string read=System.Text.Encoding.UTF8.GetString(bb,0,k);
        int x= read.IndexOf("\r\n.\r\n");
        if(x!=-1)
        {
            check=true;
        }
        richTextBox2.AppendText(read);
        WriteToNetStream(ref NetStream,"DELE "+number.ToString());
        string back=ReadFromNetStream(ref NetStream );
        richTextBox1.AppendText("DELE "+number.ToString()
            +"命令应答: "+back+"\r\n");
    }
}

```

6.2.4 检查应答码

程序应能检查应答码，如果有错误，可以及时处理。下列代码用于检查是否存在第二个参数的字符串。如果存在，返回"correct"；如果不存在，返回"err"。

```

private string checkForError(string strMessage,string check)
{
    if (strMessage.IndexOf(check) == -1)
    {return "err";}
    else
    {return "correct";}
}

```

6.2.5 获取邮件总数

下列代码用于检查 LIST 命令应答码，并获取应答码中关于邮件总数的信息。

```
private int checkmail(string back)
{
    char[] a=new char[] {' '};
    string[] mess=back.Split(a);
    int mailNumber=Int32.Parse(mess[1]);
    return mailNumber;
}
```

6.2.6 邮件接收程序开发

下面开发一个邮件接收程序，该程序可以从任何POP3服务器接收邮件。该程序将所有邮件内容全部接下來，不分离邮件头和邮件体。

新建一个项目，如图6-1所示，“服务器”文本框是 textBox1，“端口”文本框是 textBox2，“账号”文本框是 textBox3，“密码”文本框是 textBox4，“服务器应答信息”文本框是 richTextBox1，“邮件全部内容”文本框是 richTextBox2。

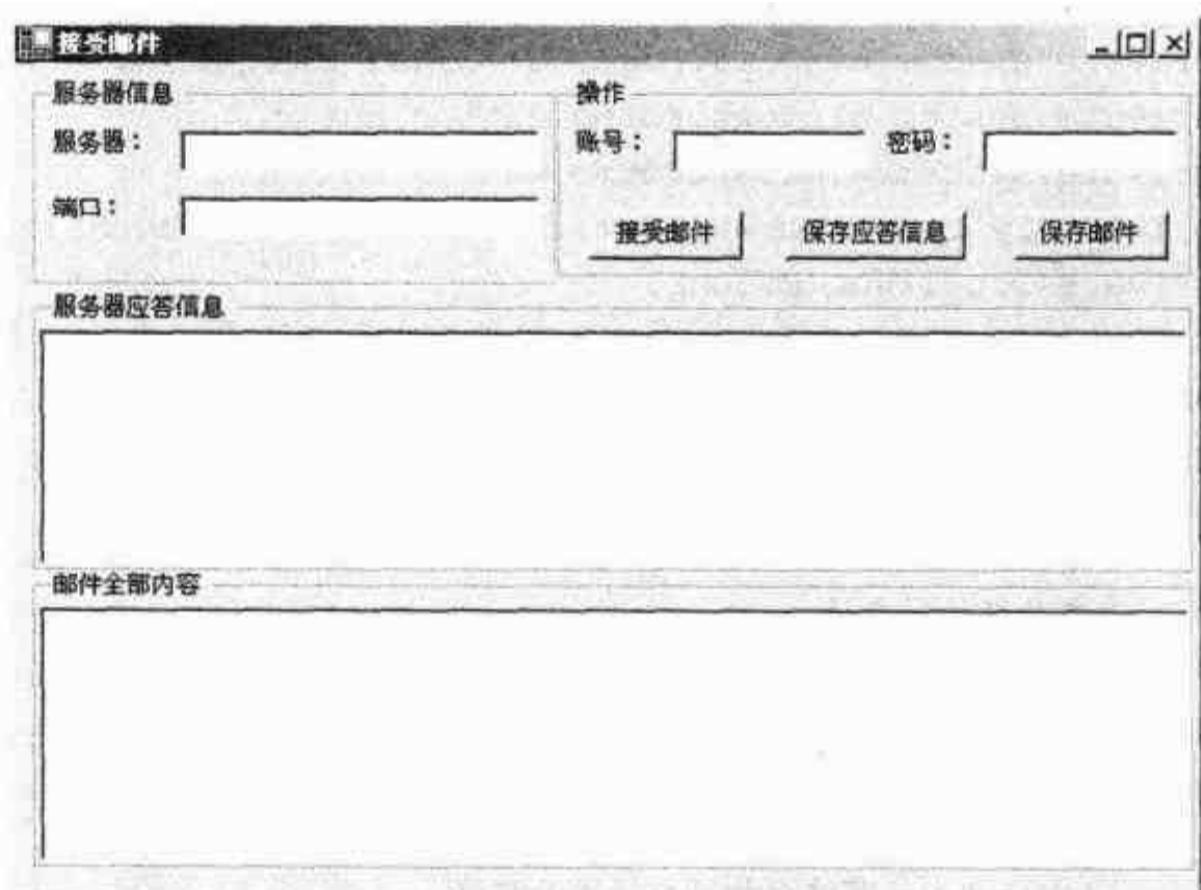


图6-1 邮件接收程序界面

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;
```

```
namespace getmails
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.RichTextBox richTextBox2;
        private System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
        private TcpClient client;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.TextBox textBox4;
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        public Form1()
        {
            // Windows 窗体设计器支持所必需的
            InitializeComponent();
            // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
        }
        /// <summary>
        /// 清理所有正在使用的资源。
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
        }
    }
}
```

```
        }
        base.Dispose( disposing );
    }

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除

/// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    Thread thread=new Thread( new ThreadStart(receive));
    thread.Start();
}
private void receive()
{
    client=new TcpClient(textBox1.Text,
        Int32.Parse(textBox2.Text));
    NetworkStream stream=client.GetStream();
    string back=ReadFromNetStream(ref stream);
    richTextBox1.AppendText("连接应答: "+back+"\r\n");

    //*****写入命令*****
    WriteToNetStream(ref stream,"USER "+textBox3.Text);
    back=ReadFromNetStream(ref stream );
    richTextBox1.AppendText("user命令应答: "+back);
    string check=checkForError(back,"+OK");
    int round=0;
    while(check=="err"&&round<5)
    {
        round++;
        WriteToNetStream(ref stream,"USER "+textBox3.Text);
        back=ReadFromNetStream(ref stream );
        richTextBox1.AppendText("user命令应答: "+back);
        check=checkForError(back,"+OK");
    }
    //*****写入密码*****
    WriteToNetStream(ref stream,"PASS "+textBox4.Text);
    back=ReadFromNetStream(ref stream );
    richTextBox1.AppendText("pass命令应答: "+back);
```

```
check=checkForError(back, "+OK");
round=0;
while(check=="err"&&round<5)
{ round++;
  WriteToNetStream(ref stream, "PASS "+textBox4.Text);
  back=ReadFromNetStream(ref stream );
  richTextBox1.AppendText("pass 命令应答: "+back);
  check=checkForError(back, "+OK");
}
//*****
WriteToNetStream(ref stream, "STAT");
back=ReadFromNetStream(ref stream );
richTextBox1.AppendText("stat 命令应答: "+back+"\r\n");
check=checkForError(back, "+OK");
round=0;
while(check=="err"&&round<5)
{ round++;
  WriteToNetStream(ref stream, "STAT");
  back=ReadFromNetStream(ref stream );
  richTextBox1.AppendText("stat 命令应答: "+back);
  check=checkForError(back, "+OK");
}
//*****
WriteToNetStream(ref stream, "LIST");
back=ReadFromNetStream(ref stream );
richTextBox1.AppendText("list 命令应答: "+back+"\r\n");
check=checkForError(back, "+OK");
round=0;
while(check=="err"&&round<5)
{ round++;
  WriteToNetStream(ref stream, "LIST");
  back=ReadFromNetStream(ref stream );
  richTextBox1.AppendText("list 命令应答: "+back);
  check=checkForError(back, "+OK");
}

//*****
int nn=checkmail(back);
for(int i=1;i<=nn;i++)
{
  WriteToNetStream(ref stream, "RETR "+i.ToString());

  back=ReadFromNetStream(ref stream );
  richTextBox1.AppendText("RETR "+i.ToString()
    +"命令应答: "+back+"\r\n");
  check=checkForError(back, "+OK");
  if(check!="err")
```

```
{  
    richTextBox2.AppendText("第"+i.ToString()  
        +"封信内容为: \r\n");  
    ReadMail(ref stream,i);  
}  
else  
{  
    richTextBox2.AppendText("第"+i.ToString()  
        +"封信没有内容。 \r\n");  
}  
}  
WriteToNetStream(ref stream,"QUIT");  
back=ReadFromNetStream(ref stream);  
richTextBox1.AppendText("QUIT命令应答: "+back+"\r\n");  
check=checkForError(back,"+OK");  
round=0;  
while(check=="err"&&round<5)  
{  
    round++;  
    WriteToNetStream(ref stream,"QUIT");  
    back=ReadFromNetStream(ref stream);  
    richTextBox1.AppendText("QUIT命令应答: "+back+"\r\n");  
    check=checkForError(back,"+OK");  
}  
}  
//*****  
private string ReadFromNetStream(ref NetworkStream NetStream)  
{  
    byte[] bb=new byte[1024];  
    NetStream.Read(bb,0,bb.Length);  
    string read=System.Text.Encoding.ASCII.GetString(bb);  
    return read;  
}  
//*****  
private void ReadMail(ref NetworkStream NetStream,int number)  
{  
    int k=0;  
    bool check=false;  
    byte[] bb=new byte[6400];  
    while(!check)  
    {  
        k=NetStream.Read(bb,0,bb.Length);  
        string read=System.Text.Encoding.UTF8.GetString(bb,0,k);  
        int x= read.IndexOf("\r\n.\r\n");  
        if(x!=-1){  
            check=true;  
        }  
        richTextBox2.AppendText(read);  
    }  
}
```

```
        WriteToNetStream(ref NetStream, "DELE "+number.ToString());
        string back=ReadFromNetStream(ref NetStream );
        richTextBox1.AppendText ("DELE "+number.ToString()
            +"命令应答: "+back+"\r\n");
    }
//*****
private void WriteToNetStream(ref NetworkStream NetStream,
    string Command)
{
    string stringToSend = Command + "\r\n";
    Byte[] arrayToSend = System.Text.Encoding.ASCII.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
}
//*****
private string checkForError(string strMessage,string check)
{
    if (strMessage.IndexOf(check) == -1)
    {return "err";}
    else
    {return "correct";}
}
//*****
private int checkmail(string back)
{
    char[] a=new char[] {' '};
    string[] mess=back.Split(a);
    int mailNumber=Int32.Parse(mess[1]);
    return mailNumber;
}
private void button2_Click(object sender, System.EventArgs e)
{
    StreamWriter sw=null;
    if(saveFileDialog1.ShowDialog()==DialogResult.OK)
    {
        try
        {
            sw=new StreamWriter(saveFileDialog1.FileName,false,
                System.Text.Encoding.Unicode);
            sw.WriteLine(richTextBox1.Text);
        }
        catch(Exception excep){MessageBox.Show(excep.Message);}
        finally{if(sw!=null){sw.Close();}}
    }
}
```

```
}

private void button3_Click(object sender, System.EventArgs e)
{
    StreamWriter sw=null;
    if(saveFileDialog1.ShowDialog() == DialogResult.OK)
    {
        try
        {
            sw=new StreamWriter(saveFileDialog1.FileName, false,
                System.Text.Encoding.Unicode);
            sw.Write(richTextBox2.Text);
        }
        catch(Exception excep){MessageBox.Show(excep.Message);}
        finally{if(sw!=null){sw.Close();}}
    }
}

private void button4_Click(object sender, System.EventArgs e)
{
    client.Close();
}
}
```

6.2.7 演示

下面用本节程序演示一下从“263.net”接收邮件。本书5.2节开发了一个邮件发送程序，zcj268@263.net是笔者目前的一个免费信箱，先用5.2节的程序向该信箱发送一封邮件，然后用本节的程序接收下来。图6-2是本节程序的运行情况。

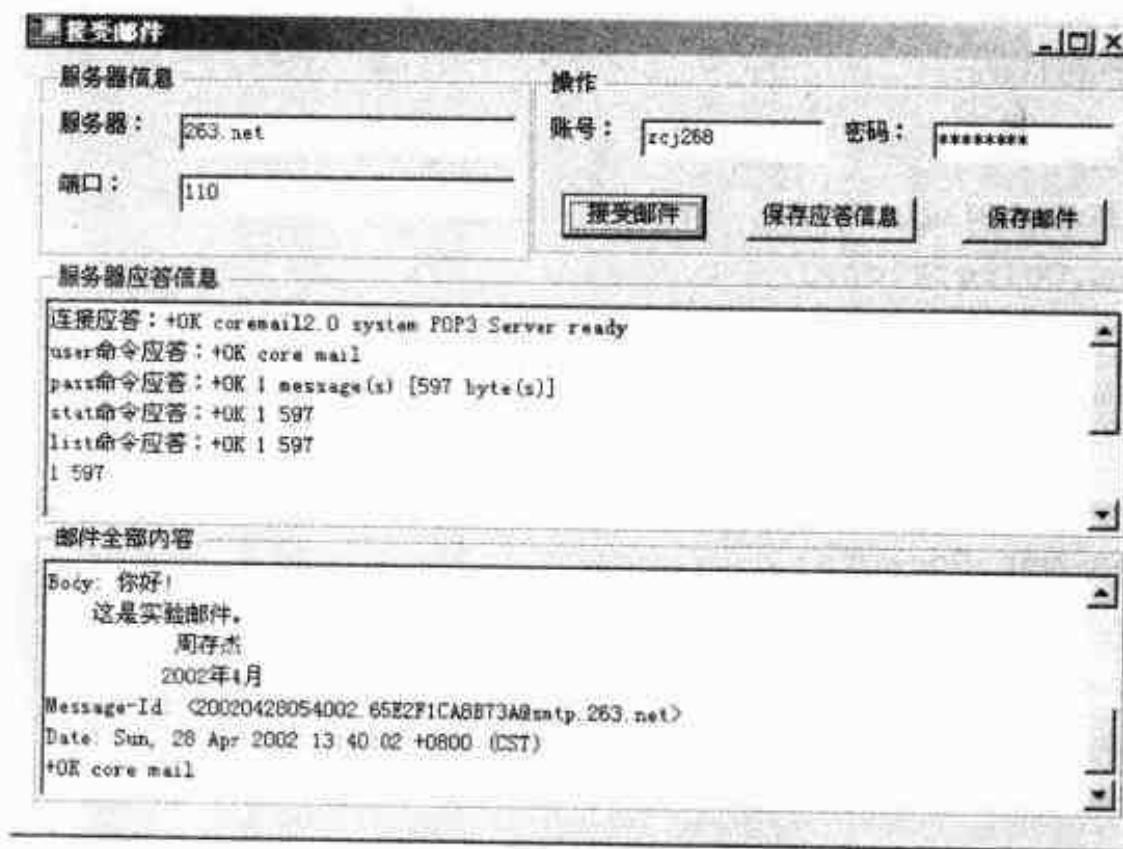


图 6-2 接收邮件

6.3 POP3 服务器开发

就像开发 SMTP 服务器一样，利用数据库开发 POP3 服务器也很方便。但现在本书还未涉及数据库的应用，所以只能用第 5 章的方法，使用文件读写操作来完成 POP 服务。熟悉本节内容后，对今后开发许多软件都有所帮助。本节使用同步套接字技术开发一个 POP3 服务器。

6.3.1 POP3 服务器开发

如图 6-3 所示，“服务器 IP”文本框是 textBox1，“监听端口”文本框是 textBox2，“服务器状态”文本框是 textBox3，“接收信息”文本框是 richTextBox1。“开始监听”按钮为 button1，“停止监听”按钮为 button2。

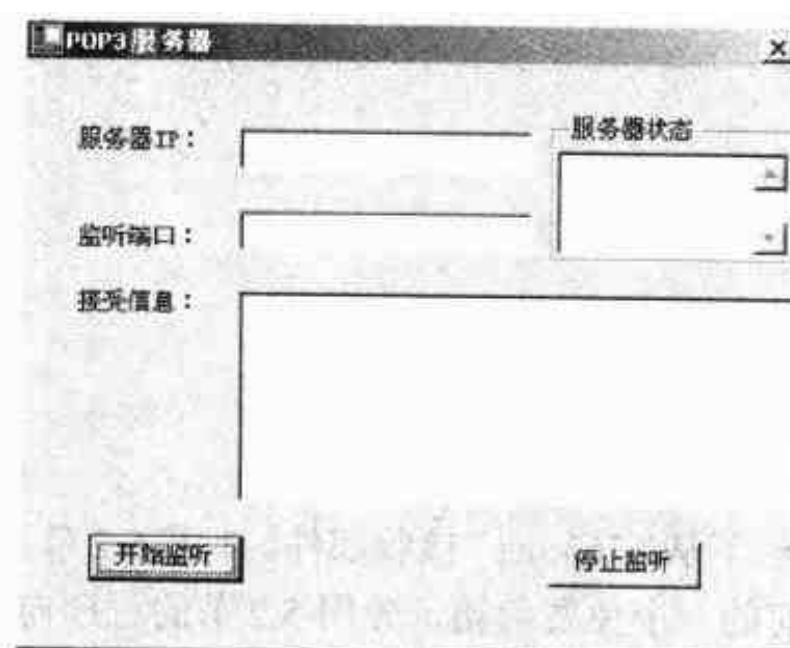


图 6-3 POP3 服务器界面

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.IO;

namespace Socket_Bin
{
    /// <summary>
    /// Summary description for Form1.
```

```
/// </summary>
public class Form1 : System.Windows.Forms.Form
{
    private System.Windows.Forms.Button button1;
    private System.Windows.Forms.Label label1;
    private System.Windows.Forms.TextBox textBox1;
    private System.Windows.Forms.Label label2;
    private System.Windows.Forms.TextBox textBox2;
    private IPAddress myIP=IPAddress.Parse("127.0.0.1");
    private IPEndPoint MyServer;
    private Socket sock;

    private bool control=false;
    private Socket accSock;

    private System.Windows.Forms.Button button2;
    private System.Windows.Forms.Label label3;
    private System.Windows.Forms.RichTextBox richTextBox1;
    private System.Windows.Forms.TextBox textBox3;
    private System.Windows.Forms.GroupBox groupBox1;

    /// <summary>
    /// Required designer variable.
    /// </summary>
    private System.ComponentModel.Container components = null;

    public Form1()
    {
        //
        // Required for Windows Form Designer support
        //
        InitializeComponent();

        //
        // TODO: Add any constructor code after InitializeComponent call
        //
    }

    /// <summary>
    /// Clean up any resources being used.
    /// </summary>
    protected override void Dispose( bool disposing )
    {
        if( disposing )
        {
            if (components != null)
```

```
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}
#endregion Windows Form Designer generated code

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        myIP = IPAddress.Parse(textBox1.Text);
    }
    catch{MessageBox.Show("您输入的 IP 地址格式不正确，请重新输入！");}
    try
    {
        Thread thread=new Thread(new ThreadStart(accp));
        thread.Start();
    }
    catch(Exception ee){textBox3.AppendText(ee.Message);}
}

private void accp()
{
    MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
    sock =new Socket(AddressFamily.InterNetwork,
                      SocketType.Stream,ProtocolType.Tcp);
    sock.Bind(MyServer);
    sock.Listen(50);
    textBox3.AppendText("主机"+textBox1.Text+"端口"
                       +textBox2.Text+"开始监听.....\r\n");
    while(true)
    {
        accSock=sock.Accept();
        if(accSock.Connected)
        {
            Thread thread=new Thread(new ThreadStart(round));
        }
    }
}
```

```
        thread.Start();
    }
}

private void round()
{
    bool USER=false;
    bool PASS=false;
    int LOCK=0;
    string commString=null;
    string bigComm=null;
    string command=null;
    string parameter=null;
    string hostName=null;
    string mailbox=null;
    NetworkStream netStream=new NetworkStream(accSock);
    sendFeedback(ref netStream,"+OK POP3 Server Ready!");
    while(true)
    {
        //接受信息 + + +
        commString=readCommand(ref netStream);
        bigComm=bigCommand(commString);
        command=splitCommand(bigComm);
        parameter=splitParameter(bigComm);
        command=command.ToLower();
        parameter=parameter.ToLower();
        hostName=Dns.GetHostName();
        richTextBox1.AppendText("command:" + command + " "
            +"parameter:" + parameter + "\r\n");
        if(command=="user")
        {
            try
            {
                bool checkUser=Directory.Exists
                    ("e:\\smtp\\mailbox\\" + parameter);
                if(checkUser==true)
                {
                    if(mailbox!=parameter)
                    {
                        USER=true;
                        mailbox=parameter;
                        sendFeedback(ref netStream,
                            "+OK username correct!");
                        LOCK=0;
                    }
                    else
                }
            }
        }
    }
}
```

```
        {
            sendFeedback(ref netStream,
                "+OK username correct!but locked");
        }
    }
    else{sendFeedback(ref netStream,"-ERR sorry,
        no mailbox for"+parameter+" here");}
    PASS=false;
}
catch
{
    sendFeedback(ref netStream,"-ERR Server problem,
        connection will be Closed");
    accSock.Close();
}
}
else if(command=="pass")
{
    try
    {
        if(USER==true)
        {
            FileStream passFile=null;
            passFile=File.OpenRead("e:\\\\smtp\\\\mailbox\\\\"
                +mailbox+"\\\\pass.txt");
            StreamReader readPass=
                new StreamReader(passFile);
            string password=readPass.ReadToEnd();
            readPass.Close();
            passFile.Close();
            if((password==parameter)&&(LOCK==0))
            {
                PASS=true;
                LOCK++;
                sendFeedback(ref netStream,
                    "+OK welcome to "+mailbox);
            }
            else if((password==parameter)&&(LOCK!=0))
            {
                LOCK++;
                sendFeedback(ref netStream,"-ERR "+mailbox
                    +" locked or password error! ");
            }
        }
    }
    catch
```

```
{  
    sendFeedback(ref netStream,  
        "-ERR performed unsuccessful");  
}  
}  
else if(command=="stat")  
{  
    bool round =true;  
    int i=0;  
    int mailNumber=0;  
    try  
{  
        if(USER==true&&PASS==true)  
        {  
            if(LOCK==1)  
            {  
                while(round==true&&i<1024)  
                {  
                    bool exist=  
                        File.Exists("e:\\smtp\\mailbox\\\\"  
                            +mailbox+"\\mail"+i.ToString()  
                            +".txt");  
                    if(exist==true)  
                    {  
                        mailNumber++;  
                    }  
                    i++;  
                }  
                sendFeedback(ref netStream, "+OK "  
                    +mailNumber.ToString());  
            }  
            else{ sendFeedback(ref netStream,  
                "-ERR mailbox locked");}  
        }  
        else  
{  
            sendFeedback(ref netStream,  
                "-ERR USER OR PASS command needed");  
        }  
    }  
    catch{sendFeedback(ref netStream,  
        "-ERR performed unsuccessful");}  
}  
else if(command=="list")  
{  
    int mailNumber=0;
```

```
    bool round =true;
    int i=0;
    string feedback=null;
    try
    {
        if(USER==true&&PASS==true)
        {
            if(LOCK==1)
            {
                while(round==true&&i<1024)
                {
                    bool exist=
                        File.Exists("e:\\smtp\\mailbox\\"
                        +"mailbox+"\\mail"+i.ToString()
                        +".txt");
                    if(exist==true)
                    {
                        mailNumber++;
                        FileStream mailStream=null;
                        mailStream=new FileStream
                            ("e:\\smtp\\mailbox\\"
                            +"mailbox+"\\mail"+i.ToString()
                            +".txt", FileMode.Open,
                            FileAccess.Read);
                        int number=0;
                        int mailSize=0;
                        byte[] bb=new byte[8];
                        while((number=
                            mailStream.Read(bb, 0, 8))!=0)
                        {
                            mailSize=mailSize+number;
                        }
                        mailStream.Close();
                        feedback=feedback+mailNumber
                            +" "+mailSize+"\r\n";
                        StreamWriter recorder=
                            new StreamWriter
                            ("e:\\smtp\\mailbox\\"
                            +"mailbox+"\\record.txt",
                            true,
                            System.Text.Encoding.UTF8);
                        recorder.Write("mail"
                            +i.ToString()+" .txt\\");
                        recorder.Close();
                        mailSize=0;
                    }
                }
            }
        }
    }
```

```
i++;
}
sendFeedback(ref netStream,
    "+OK "+mailNumber+"\r\n"+feedback);
}
else(sendFeedback(ref netStream,
    "-ERR mailbox locked"));
}
else
{
    sendFeedback(ref netStream,
    "-ERR USER OR PASS command needed");
}
}
catch(sendFeedback(ref netStream,
    "-ERR performed unsuccessful"));
}
else if(command=="retr")
{
    try
    {
        if((USER==true)&&(PASS==true)&&(LOCK==1))
        {
            FileStream Record=null;
            Record=File.OpenRead
                ("e:\\smtp\\mailbox\\\"+mailbox
                +"\\record.txt");
            StreamReader readRecord=new StreamReader(Record);
            string recordString=readRecord.ReadToEnd();
            Record.Close();
            readRecord.Close();
            char[] a=new Char[]{'\\'};
            string[] splitRecord=recordString.Split(a);
            string readFileName=
                splitRecord[Int32.Parse(parameter)-1];
            FileStream MailFile=null;
            MailFile=File.OpenRead
                ("e:\\smtp\\mailbox\\"
                +mailbox+"\\\"+readFileName);
            StreamReader readMail=new StreamReader(MailFile);
            string MailMessage=readMail.ReadToEnd();
            MailFile.Close();
            readMail.Close();
            sendFeedback(ref netStream,"+OK "+parameter);
            sendFeedback(ref netStream,MailMessage);
        }
    }
}
```

```
        else
        {
            sendFeedback(ref netStream, "-ERR USER OR PASS
                command need OR mailbox locked");
        }
    }
    catch
    {
        sendFeedback(ref netStream,
            "-ERR performed unsuccessful");
    }
}
else if(command=="delete")
{
    try
    {
        FileStream Record=null;
        Record=File.OpenRead
            ("e:\\smtp\\mailbox\\"
            +mailbox+"\\record.txt");
        StreamReader readRecord=new StreamReader(Record);
        string recordString=readRecord.ReadToEnd();
        Record.Close();
        readRecord.Close();
        char[] a=new Char[]{'\\'};
        string[] splitRecord=recordString.Split(a);
        string readFileName=
            splitRecord[Int32.Parse(parameter)-1];
        File.Delete("e:\\smtp\\mailbox\\"
            +mailbox+"\\\"+readFileName);
        sendFeedback(ref netStream, "+OK delete");
    }
    catch
    {
        sendFeedback(ref netStream,
            "-ERR performed unsuccessful,try again");
    }
}
else if(command=="quit")
{
    try
    {
        File.Delete("e:\\smtp\\mailbox\\"
            +mailbox+"\\record.txt");
        sendFeedback(ref netStream,
            "+OK connection closed");
    }
}
```

```
        control=true;
        accSock.Close();
    }
    catch{sendFeedback(ref netStream,
        "502 performed unsuccessful,try again");}
}
else
{
    sendFeedback(ref netStream,
        "-ERR unrecognized command");
}
}
private void button2_Click(object sender, System.EventArgs e)
{
    try
    {
        sock.Close();
        textBox3.AppendText("主机"+textBox1.Text+"端口"
            +textBox2.Text+"监听停止! \r\n");
    }
    catch{MessageBox.Show("监听尚未开始,关闭无效!");}
}
private string readCommand(ref NetworkStream NetStream)
{
    byte[] byteMessage=new byte[1024];
    NetStream.Read(byteMessage, 0, byteMessage.Length);
    string command=
        System.Text.Encoding.ASCII.GetString(byteMessage);
    return command;
}
private void sendFeedback(ref NetworkStream NetStream,string ToSend)
{
    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}
private string readMail(ref NetworkStream NetStream)
{
    byte[] byteArray=new Byte[1024];
    NetStream.Read(byteArray, 0, byteArray.Length);
    string mailMessage=
        System.Text.Encoding.UTF8.GetString(byteArray);
```

```
        return mailMessage;
    }
    private string splitMailbox(string aimString)
    {
        string[] aim=new string[2];
        char[] a=new char[] {'@'};
        aim=aimString.Split(a);
        int y=aim[0].IndexOf("<");
        if(y!=-1)
        {
            string mailbox=aim[0].Substring (y+1,aim[0].Length-y-1);
            return mailbox;
        }
        else
        {
            int x=aim[0].IndexOf(":");
            string mailbox=aim[0].Substring (x+1,aim[0].Length-x-1);
            return mailbox;
        }
    }
    private string splitServer(string aimString)
    {
        string[] aim=new string[2];
        char[] a=new char[] {'@'};
        aim=aimString.Split(a);
        int y=aim[1].IndexOf(">");
        if(y!=-1)
        {
            string server=aim[1].Substring (0,aim[1].Length-1);
            return server;
        }
        else
        {
            return aim[1];
        }
    }
    private string splitCommand(string aimString)
    {
        string[] aim=new string[2];
        int x=aimString.IndexOf(" ");
        if(x!=-1)
        {
            char[] a=new char[] {' '};
            aim=aimString.Split(a);
            return aim[0];
        }
        else{return aimString;}
    }
```

```
private string splitParameter(string aimString)
{
    string[] aim=new string[2];
    int x=aimString.IndexOf(" ");
    if(x!=-1)
    {
        char[] a=new char[] {' '};
        aim=aimString.Split(a);
        return aim[1];
    }
    else{return "";}
}

private string bigCommand(string commString)
{
    int x=commString.IndexOf("\r\n");
    string command=commString.Substring(0,x);
    return command;
}
```

6.3.2 演示

启动本节程序，输入适当的 IP 地址和端口，打开服务功能。然后启动 6.2 节的示例程序，开始接收邮件。图 6-4 是客户端的运行情况。

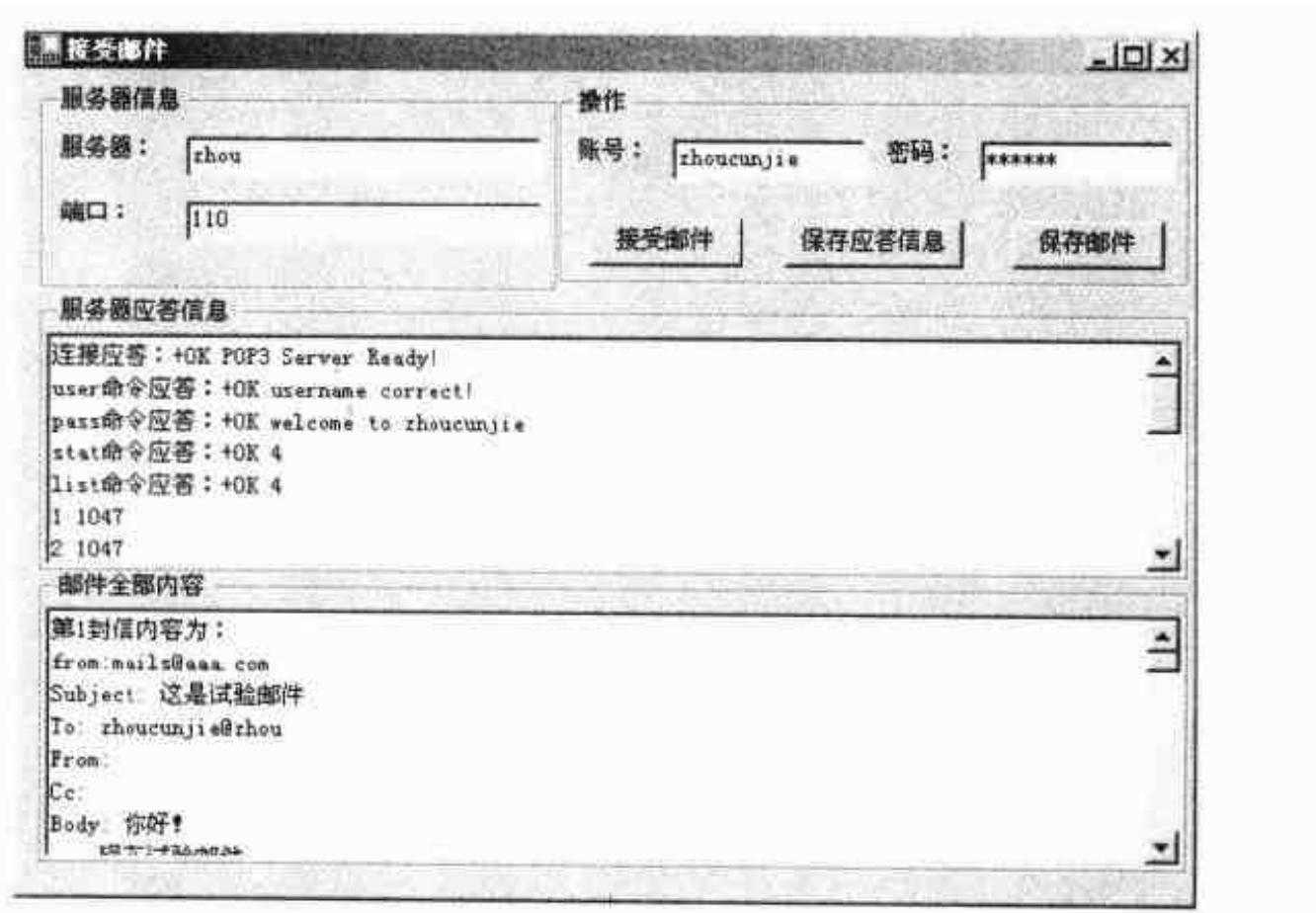


图 6-4 邮件接收客户端运行情况

图 6-5 是服务器的运行情况。

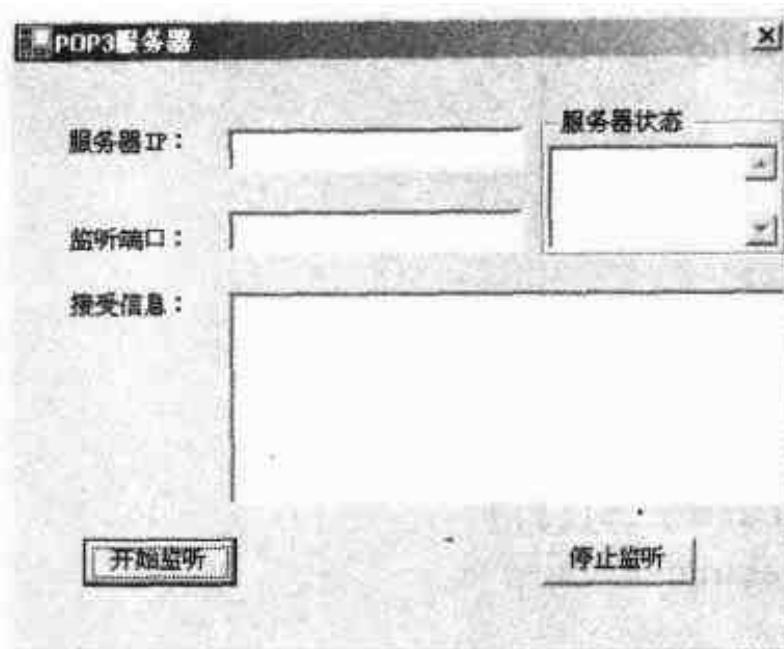


图 6-5 POP3 服务器运行情况

6.3.3 改进建议

在学习本书数据库知识后，利用 SQL Server 数据库开发一个安全易用的 POP3 服务器。

本章小结

本章在介绍 POP3 协议的基础上，开发了一个邮件发送程序和 POP3 服务器。在这两个程序的基础上加以改进，可以成为一个完善的商业 POP3 系统。如果进一步结合 SQL Server 数据库，可以开发出安全、易用的 POP3 服务系统。

第7章 远程控制开发

远程控制是网络开发的一个非常重要的方面，应用非常广泛。远程控制对底层协议没有要求，无论是使用 TCP 还是使用 UDP 或者其他协议，都可以完成远程控制的目的。TCP 协议应用最为广泛。UDP 协议在远程控制方面也有一定的应用，可以开发木马，UDP 木马的控制端不容易被发现。本节利用同步套接字的技术，开发一个远程控制工具。该远程控制工具分为两端——服务器和控制端。远程控制工具的功能如下(由于在第 4 章里已经讲过如何传输文件，所以这里不再赘述)：

- 获取服务器文件目录
- 文件移动
- 文件删除
- 文件拷贝
- 文件重命名
- 发送短信
- 创建注册表新键
- 删除注册表子键
- 设置注册表键值
- 获取注册表键值
- 删除注册表键值

7.1 服务端开发

服务器与客户端约定：服务器对每一个命令均作出答复，如果能正确执行，返回<OK>；如果执行期间发生错误，返回<ER>。下面是命令与用途的对照：

LIST —— 获取服务器文件列表；

MOVE —— 移动文件；

DELE —— 删除文件；

RENA —— 文件重命名；

COPY —— 复制文件；

MESS —— 通知短信；

RGCR —— 创建注册表新键；

RGDE —— 删除注册表子键；

RGSV —— 设置注册表键值；

RGGV —— 获取注册表键值；

RGDV —— 删除注册表键值；

QUIT —— 关闭连接。

命令的格式如下：

命令码 <SP>参数<SP>参数<CRLF>

命令可以带多个参数，可以带一个参数，也可以不带参数。

7.1.1 获取客户发送的信息

下列方法可以获取客户发送的信息，并将其转化成字符串。

```
private string readFromClient(ref Socket socket)
{
    byte[] byteMessage=new byte[1024];
    socket.Receive(byteMessage);
    string command=System.Text.Encoding.ASCII.GetString(byteMessage);
    int x=command.IndexOf("\r\n");
    command=command.Substring(0,x);
    return command;
}
```

7.1.2 获取用户命令

下列方法把用户信息分为几部分，然后返回用户命令码。

```
private string getCommand(string aimString)
{
    int x=aimString.IndexOf(" ");
    if(x!=-1)
    {
        string comm=aimString.Substring(0,x);
        return comm;
    }
    else{return aimString;}
}
```

7.1.3 获取命令参数

下列方法可以把用户信息分成几部分，然后返回命令参数。

```
private string getParameter(string aimString)
{
    int x=aimString.IndexOf(" ");
    if(x!=-1)
    {
        string para=aimString.Substring(x+1,aimString.Length -x-1);
        return para;
    }
}
```

```
        }
        else{return "";}
    }
```

7.1.4 发送反馈信息

下列方法可以发送反馈信息。第一个参数是正在连接的套接字对象，第二个参数是要发送的字符串。

```
private void sendFeedback(ref Socket socket, string ToSend)
{
    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
    socket.Send(arrayToSend);
}
```

7.1.5 服务器开发

一般而言，作为远程控制的服务器要后台运行。后台运行可通过设置程序窗体属性来实现。打开主窗体（如果这样称呼合适的话）的属性窗口，将其 `ShowInTaskbar` 属性设为 `False`，将其 `WindowState` 属性设为 `Minimized`，程序即可变成后台运行程序。

除了让程序后台运行以外，还要让程序启动时就自动监听端口，等候客户的连接请求。要使程序自动启动服务功能，可以靠下面的办法完成：在代码编辑窗口找到代码“`InitializeComponent();`”，在该代码下面添加启动服务的代码即可。下面是服务端的代码。

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Microsoft.Win32;
using System.IO;

namespace Socket_Bin
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.RichTextBox richTextBox1;
```

```
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;

//*****
IPEndPoint MyServer=null;
Socket sock=null;
Socket accSock=null;
bool control=false;
string commString;
IPHostEntry hostEntry=new IPHostEntry();
string hostName=Dns.GetHostName();
string command=null;
string parameter=null;

public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();
    Thread thread=new Thread(new ThreadStart(accp));
    thread.Start();
    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code
```

```
#endregion

/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void accp(){
    hostEntry=Dns.GetHostByName(hostName);
    myIP=hostEntry.AddressList[0];
    MyServer=new IPEndPoint(myIP,2525);
    sock =new Socket(AddressFamily.InterNetwork,
        SocketType.Stream,ProtocolType.Tcp);
    sock.Bind(MyServer);
    sock.Listen(50);
    while(true)
    {
        accSock=sock.Accept();
        if(accSock.Connected)
        {
            Thread thread=new Thread(new ThreadStart(round));
            thread.Start();
        }
    }
}

private void round()
{
    sendFeedback(ref accSock,"<OK> connection established");
    while(true){
        //接受信息+ + +
        commString=readFromClient(ref accSock);
        command=getCommand(commString);
        parameter=getParameter(commString);
        richTextBox1.AppendText("命令: "+command+" "+"参数: "+
            +parameter+"\r\n");
        if(command=="LIST")
        {
            try
            {
                string[] str=new string[1024];
                string dir=null;
```

```
        for(int k=0;
            k<Directory.GetDirectories(parameter).Length;
            k++)
        {
            str[k]=
                Directory.GetDirectories(parameter)[k];
            dir=dir+" 目录: "+str[k]+"\r\n";
        }
        for(int k=0;
            k<Directory.GetFiles(parameter).Length;k++)
        {
            str[k]=Directory.GetFiles(parameter)[k];
            dir=dir+" 文件: "+str[k]+"\r\n";
        }
        sendFeedback(ref accSock,<OK> DIR send\r\n"+dir);
    }
    catch(Exception ee){sendFeedback(ref accSock,
        "<ER> "+ee.Message);}
}
else if(command=="MOVE")
{
    try
    {
        char[] a=new Char[]{' '};
        string[] spl=parameter.Split(a);
        string para1=spl[0];
        string para2=spl[1];
        File.Move(para1,para2);
        sendFeedback(ref accSock,<OK> file moved");
    }
    catch(Exception ee){sendFeedback(ref accSock,
        "<ER> "+ee.Message);}
}
else if(command=="DELETE")
{
    try
    {
        File.Delete(parameter);
        sendFeedback(ref accSock,<OK> file deleted");
    }
    catch{sendFeedback(ref accSock,
        "<ER> performed unsuccessful");}
}
else if(command=="RENAME")
{
    try
```

```
{  
    char[] a=new Char[]{' '};  
    string[] spl=parameter.Split(a);  
    string para1=spl[0];  
    string para2=spl[1];  
    File.Copy(para1,para2,true);  
    File.Delete(para1);  
    sendFeedback(ref accSock,<OK> file renamed");  
}  
catch(Exception ee){sendFeedback(ref accSock,  
    "<ER> "+ee.Message);}  
}  
else if(command=="COPY")  
{  
    try  
{  
        char[] a=new Char[]{' '};  
        string[] spl=parameter.Split(a);  
        string para1=spl[0];  
        string para2=spl[1];  
        File.Copy(para1,para2,true);  
        sendFeedback(ref accSock,<OK> file copied");  
    }  
    catch(Exception ee){sendFeedback(ref accSock,  
        "<ER> "+ee.Message);}  
}  
else if(command=="MESS")  
{  
    try  
{  
        MessageBox.Show(parameter);  
        sendFeedback(ref accSock,"Message Received");  
    }  
    catch  
{  
        sendFeedback(ref accSock,  
            "<ER> performed unsuccessful");  
    }  
}  
else if(command=="RGCR")  
{  
    try  
{  
        RegistryKey key=null;  
        char[] a=new Char[]{'\\'};  
        string[] mainKeySplit=parameter.Split(a);  
    }  
}
```

```
        string mainKey=mainKeySplit[0];
        if(mainKey=="HKEY_LOCAL_MACHINE")
        {
            key1=Registry.LocalMachine;
        }
        if(mainKey=="HKEY_CLASSES_ROOT")
        {
            key1=Registry.ClassesRoot;
        }
        if(mainKey=="HKEY_CURRENT_USER")
        {
            key1=Registry.CurrentUser;
        }
        if(mainKey=="HKEY_CURRENT_CONFIG")
        {
            key1=Registry.CurrentConfig;
        }
        if(mainKey=="HKEY_USERS")
        {
            key1=Registry.Users;
        }
        string subKey=
            parameter.Substring(mainKey.Length+1,
            parameter.Length-mainKey.Length-1);
        RegistryKey key2=key1.CreateSubKey(subKey);
        sendFeedback(ref accSock,<OK> subkey create");
    }
    catch(Exception ee){sendFeedback(ref accSock,
        "<ER> "+ee.Message);}
}
else if(command=="RGDE")
{
    try
    {
        RegistryKey key1=null;
        char[] a=new Char[]{'\\'};
        string[] mainKeySplit=parameter.Split(a);
        string mainKey=mainKeySplit[0];
        if(mainKey=="HKEY_LOCAL_MACHINE")
        {
            key1=Registry.LocalMachine;
        }
        if(mainKey=="HKEY_CLASSES_ROOT")
        {
            key1=Registry.ClassesRoot;
        }
```

```
if(mainKey=="HKEY_CURRENT_USER")
{
    key1=Registry.CurrentUser;
}
if(mainKey=="HKEY_CURRENT_CONFIG")
{
    key1=Registry.CurrentConfig;
}
if(mainKey=="HKEY_USERS")
{
    key1=Registry.Users;
}
string subKey=
    parameter.Substring(mainKey.Length+1,
    parameter.Length-mainKey.Length-1);
key1.DeleteSubKey(subKey);
sendFeedback(ref accSock,<OK> subkey delete");
}
catch(Exception ee){sendFeedback(ref accSock,
    "<ER> "+ee.Message);}
}
else if(command=="RGSV")
{
try
{
    RegistryKey key1=null;
    char[] a=new Char[]{'\\'};
    string[] mainKeySplit=parameter.Split(a);
    string mainKey=mainKeySplit[0];
    if(mainKey=="HKEY_LOCAL_MACHINE")
    {
        key1=Registry.LocalMachine;
    }
    if(mainKey=="HKEY_CLASSES_ROOT")
    {
        key1=Registry.ClassesRoot;
    }
    if(mainKey=="HKEY_CURRENT_USER")
    {
        key1=Registry.CurrentUser;
    }
    if(mainKey=="HKEY_CURRENT_CONFIG")
    {
        key1=Registry.CurrentConfig;
    }
    if(mainKey=="HKEY_USERS")
```

```
{  
    key1=Registry.Users;  
}  
int x=parameter.IndexOf(" ");  
string subKey=  
    parameter.Substring(mainKey.Length+1,  
        x-mainKey.Length-1);  
RegistryKey key2=key1.OpenSubKey(subKey,true);  
char[] b=new Char[]{' '};  
string[] para=parameter.Split(b);  
if(para[3]=="0")  
{  
    key2.SetValue(para[1],para[2]);}  
else{key2.SetValue(para[1],  
    Int32.Parse(para[2]));}  
sendFeedback(ref accSock,<OK> subkey value set");  
}  
catch(Exception ee){sendFeedback(ref accSock,  
    "<ER> "+ee.Message);}  
}  
else if(command=="RGGV")  
{  
    try  
{  
        RegistryKey key1=null;  
        char[] a=new Char[]{'\\'};  
        string[] mainKeySplit=parameter.Split(a);  
        string mainKey=mainKeySplit[0];  
        if(mainKey=="HKEY_LOCAL_MACHINE")  
{  
            key1=Registry.LocalMachine;  
        }  
        if(mainKey=="HKEY_CLASSES_ROOT")  
{  
            key1=Registry.ClassesRoot;  
        }  
        if(mainKey=="HKEY_CURRENT_USER")  
{  
            key1=Registry.CurrentUser;  
        }  
        if(mainKey=="HKEY_CURRENT_CONFIG")  
{  
            key1=Registry.CurrentConfig;  
        }  
        if(mainKey=="HKEY_USERS")  
{  
    }
```

```
        key1=Registry.Users;
    }
    int x=parameter.IndexOf(" ");
    string subKey=
        parameter.Substring(mainKey.Length+1,
        x-mainKey.Length-1);
    RegistryKey key2=key1.OpenSubKey(subKey,true);
    char[] b=new Char[]{' '};
    string[] para=parameter.Split(b);
    string backToClient=
        key2.GetValue(para[1]).ToString();
    sendFeedback(ref accSock,<OK> subkey value send");
    sendFeedback(ref accSock,backToClient);
}
catch(Exception ee){sendFeedback(ref accSock,
    "<ER> "+ee.Message);}
}
else if(command=="RGDV")
{
    try
    {
        RegistryKey key1=null;
        char[] a=new Char[]{'\\'};
        string[] mainKeySplit=parameter.Split(a);
        string mainKey=mainKeySplit[0];
        if(mainKey=="HKEY_LOCAL_MACHINE")
        {
            key1=Registry.LocalMachine;
        }
        if(mainKey=="HKEY_CLASSES_ROOT")
        {
            key1=Registry.ClassesRoot;
        }
        if(mainKey=="HKEY_CURRENT_USER")
        {
            key1=Registry.CurrentUser;
        }
        if(mainKey=="HKEY_CURRENT_CONFIG")
        {
            key1=Registry.CurrentConfig;
        }
        if(mainKey=="HKEY_USERS")
        {
            key1=Registry.Users;
        }
        int x=parameter.IndexOf(" ");
    }
```

```
        string subKey=
            parameter.Substring(mainKey.Length+1,
                x-mainKey.Length-1);
        RegistryKey key2=key1.OpenSubKey(subKey,true);
        char[] b=new Char[]{' '};
        string[] para=parameter.Split(b);
        key2.DeleteValue(para[1]);
        sendFeedback(ref accSock,
            "<OK> subkey value deleted");
    }
    catch(Exception ee){sendFeedback(ref accSock,
        "<ER> "+ee.Message);}
}
else if(command=="QUIT")
{
    try
    {
        sendFeedback(ref accSock,"<OK> connection closed");
        accSock.Close();
    }
    catch(Exception ee){sendFeedback(ref accSock,
        "<ER> "+ee.Message);}
}
else
{
    sendFeedback(ref accSock,"<ER> unrecognized command");
}
}

private string readFromClient(ref Socket socket)
{
    byte[] byteMessage=new byte[1024];
    socket.Receive(byteMessage);
    string command=
        System.Text.Encoding.ASCII.GetString(byteMessage);
    int x=command.IndexOf("\r\n");
    command=command.Substring(0,x);
    return command;
}
private void sendFeedback(ref Socket socket, string ToSend)
{
    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend =
        System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
}
```

```
        socket.Send(arrayToSend);
    }

    private string getCommand(string aimString)
    {
        int x=aimString.IndexOf(" ");
        if(x!=-1)
        {
            string comm=aimString.Substring(0,x);
            return comm;
        }
        else{return aimString;}
    }

    private string getParameter(string aimString)
    {
        int x=aimString.IndexOf(" ");
        if(x!=-1)
        {
            string para=aimString.Substring(x+1,
                aimString.Length -x-1);
            return para;
        }
        else{return "";}
    }
}
```

7.2 控制端开发

客户端的主要功能就是发送命令码并判断应答码。当程序启动时，就启动线程，循环接受来自服务器的应答信息。下面开始开发。

1. 界面设计

本程序有两个窗体，图 7-1 是主窗体界面 Form1。其中“服务器 IP”文本框是 textBox1，“端口”文本框是 textBox2，“命令参数”文本框是 textBox3，“服务器反馈”文本框是 richTextBox1。“连接”按钮是 button1，“关闭连接”按钮是 button2，“发送命令”按钮是 button3。“控制目标”栏里的单选按钮从上到下，从左至右分别为 radioButton1，radioButton2，……radioButton11，“控制目标”栏里的复选框为 checkBox1。

图 7-2 是子窗体 Form2，该窗体上只有一个控件，即 label1，该控件的 Text 属性为“首先建立与服务器的连接，然后发送命令码即可。服务器收到命令后，会发出反馈。比如，在‘控制目标’栏里选中‘文件移动’单选按钮，然后在‘命令参数’栏里输入‘C:\AA.EXE E:\BB.EXE’。然后单击‘发送命令’按钮，服务器即可把‘C:\AA.EXE’移动到‘E:\BB.EXE’。

参数之间用空格隔开。在进行注册表操作时，如果注册表键值为数值类型值，请选中‘注册表键值为数值’复选框。”。

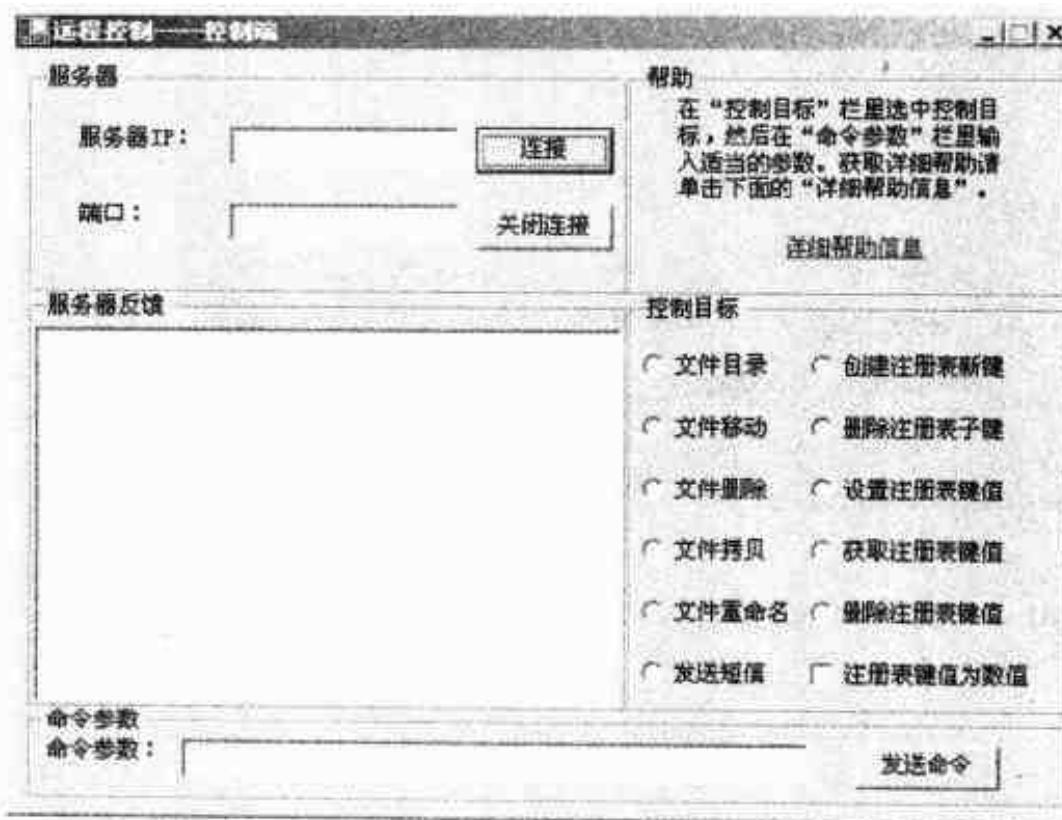


图 7-1 远程控制程序控制端界面

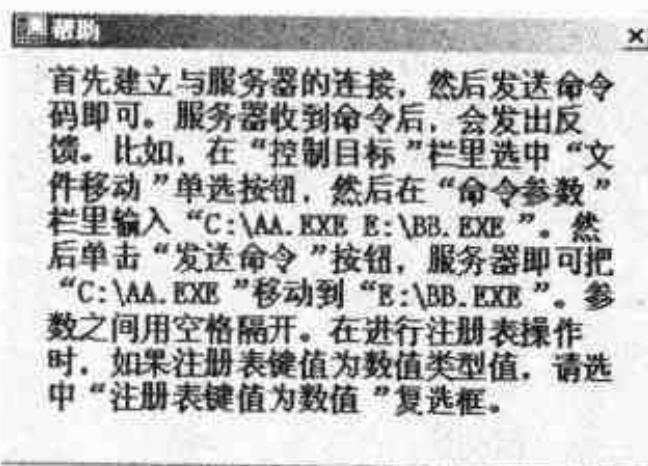


图 7-2 帮助子窗体界面

2. 编写代码

下列代码实现用 UTF8 码向服务器发送命令的功能。

```
private void WriteToSever(ref Socket sock, string sendMessage)
{
    string stringToSend = sendMessage + "\r\n";
    byte[] arrayToSend = System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
    sock.Send(arrayToSend);
}
```

下列代码实现用 UTF8 码从服务器接受命令的功能。

```
private string ReadFromServer(ref Socket sock)
{
    byte[] feedback=new byte[1024];
    sock.Receive(feedback);
```

```
string readFeedback=System.Text.Encoding.UTF8.GetString(feedback);
readFeedback=readFeedback+"\r\n";
return readFeedback;
}
```

下列是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;
namespace connect
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private bool crlRound=false;
        private Socket sock;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.SaveFileDialog saveFileDialog1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.GroupBox groupBox3;
        private System.Windows.Forms.GroupBox groupBox4;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.GroupBox groupBox5;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.LinkLabel linkLabel1;
        private System.Windows.Forms.RadioButton radioButton1;
        private System.Windows.Forms.RadioButton radioButton2;
        private System.Windows.Forms.RadioButton radioButton3;
```

```
private System.Windows.Forms.RadioButton radioButton4;
private System.Windows.Forms.RadioButton radioButton5;
private System.Windows.Forms.RadioButton radioButton6;
private System.Windows.Forms.RadioButton radioButton7;
private System.Windows.Forms.RadioButton radioButton8;
private System.Windows.Forms.RadioButton radioButton9;
private System.Windows.Forms.RadioButton radioButton10;
private System.Windows.Forms.CheckBox checkBox1;
private System.Windows.Forms.RadioButton radioButton11;
/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;

public Form1()
{
    //
    // Required for Windows Form Designer support.
    //
    InitializeComponent();
    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

#region Windows Form Designer generated code

#endregion
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
```

```
{  
    Application.Run(new Form1());  
}  
private void button1_Click(object sender, System.EventArgs e)  
{  
    IPAddress myIP=null;  
    try  
{  
        myIP =IPAddress.Parse(textBox1.Text);  
    }  
    catch{MessageBox.Show("您输入的 IP 地址格式不正确, 请重新输入!");}  
    try  
{  
        IPEndPoint MyServer=new IPEndPoint(myIP,  
            Int32.Parse(textBox2.Text));  
        sock =new Socket(AddressFamily.InterNetwork,  
            SocketType.Stream,ProtocolType.Tcp);  
        sock.Connect(MyServer);  
        richTextBox1.AppendText ("连接应答: \r\n");  
        readFeedback();  
    }  
    catch(Exception ee){MessageBox.Show(ee.Message);}  
}  
  
private void button2_Click(object sender, System.EventArgs e)  
{  
    try  
{  
        WriteToSever(ref sock,"QUIT");  
        richTextBox1.AppendText ("关闭应答: \r\n");  
        readFeedback();  
        sock.Close();  
    }  
    catch(Exception ee){MessageBox.Show(ee.Message);}  
}  
  
private void button3_Click(object sender, System.EventArgs e)  
{  
    if(radioButton1.Checked)  
{  
        string command="LIST "+textBox3.Text;  
        WriteToSever(ref sock,command);  
        richTextBox1.AppendText ("服务器文件列表: \r\n");  
    }  
    if(radioButton2.Checked){  
        string command="MOVE "+textBox3.Text;
```

```
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("文件移动: \r\n");
    }
    if(radioButton3.Checked)
    {
        string command="DELE "+textBox3.Text;
        WriteToSever(ref sock,command);

        richTextBox1.AppendText("文件删除: \r\n" );
    }
    if(radioButton4.Checked)
    {
        string command="COPY "+textBox3.Text;
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("文件拷贝\r\n" );
    }
    if(radioButton5.Checked)
    {
        string command="RENA "+textBox3.Text;
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("文件重命名: \r\n");
    }
    if(radioButton6.Checked)
    {
        string command="MESS "+textBox3.Text;
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("短信: \r\n");
    }
    if(radioButton7.Checked)
    {
        string command="RGCR "+textBox3.Text;
        WriteToSever(ref sock,command);

        richTextBox1.AppendText("创建注册表新键: \r\n");
    }
    if(radioButton8.Checked)
    {
        string command="RGDE "+textBox3.Text;
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("删除注册表子键: \r\n");
    }
    if(radioButton9.Checked)
    {
        string command=null;
        if(checkBox1.Checked)
        {
            command="RGSV "+textBox3.Text+" 1";
        }
    }
}
```

```
        else{
            command="RGSV "+textBox3.Text+" 0";
        }
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("设置注册表键值: \r\n");
    }
    if(radioButton10.Checked)
    {
        string command="RGGV "+textBox3.Text;
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("获取注册表键值: \r\n");
    }
    if(radioButton11.Checked)
    {
        string command="RGDV "+textBox3.Text;
        WriteToSever(ref sock,command);
        richTextBox1.AppendText("删除注册表键值: \r\n");
    }
    readFeedback();
}
private void readFeedback(){
    string feedback=ReadFromServer(ref sock);
    richTextBox1.AppendText(feedback);
}

private void WriteToSever(ref Socket sock, string sendMessage)
{
    string stringToSend = sendMessage + "\r\n";
    byte[] arrayToSend =
        System.Text.Encoding.UTF8.GetBytes
        (stringToSend.ToCharArray());
    sock.Send(arrayToSend);
}
private string ReadFromServer(ref Socket sock)
{
    byte[] feedback=new byte[1024];
    sock.Receive(feedback);
    string readFeedback=
        System.Text.Encoding.UTF8.GetString(feedback);
    readFeedback=readFeedback+"\r\n";
    return readFeedback;
}

private void linkLabel1_LinkClicked(object sender,
    System.Windows.Forms.LinkLabelLinkClickedEventArgs e)
{
```

```
Form2 form2=new Form2();
form2.Show();

}

}

}
```

7.3 演示

启动服务器程序，服务器自动监听 2525 端口。运行客户端程序，输入服务器 IP 地址和端口，然后建立与服务器的连接。图 7-3 是程序的运行情况。

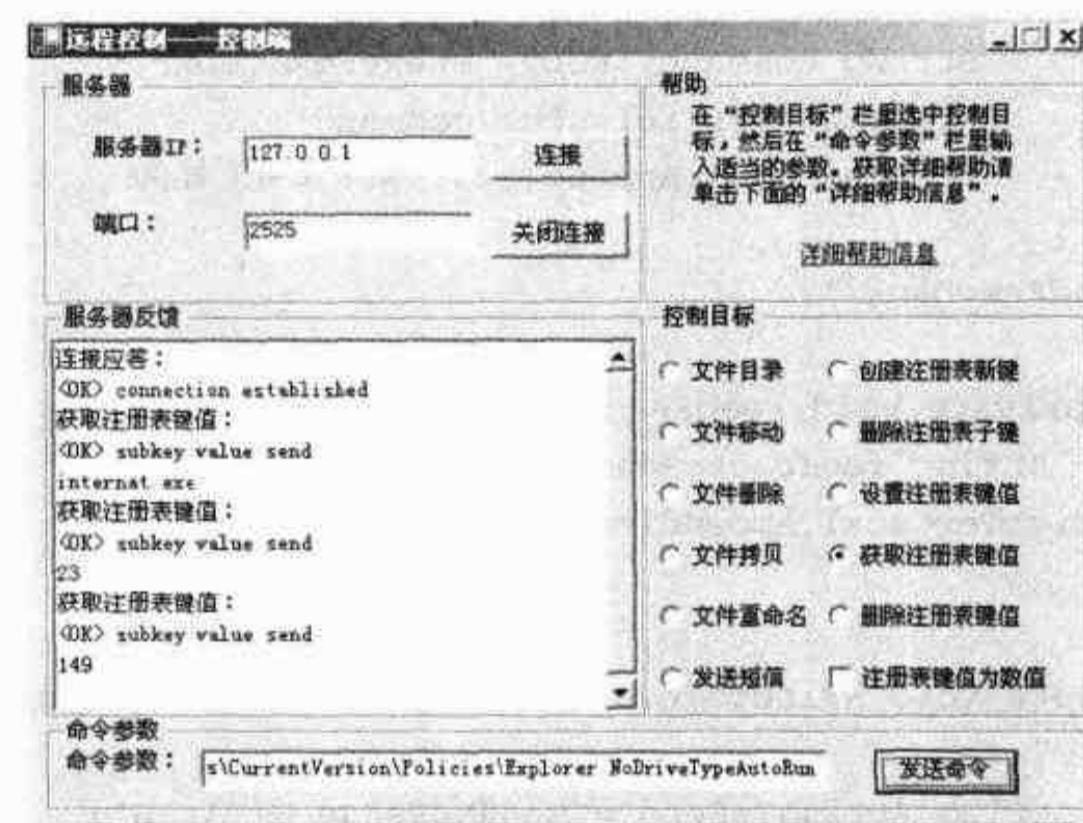


图 7-3 远程控制演示

本章小结

本章使用同步套接字技术，利用 TCP 协议开发了一个远程控制实例。通过该实例的学习，可以帮助读者在实际应用中迅速构建适合需要的远程控制工具，同时也有利于对套接字技术的进一步掌握。

第8章 网络组件开发

组件(component)是指实现 System.ComponentModel.IComponent 接口的一个类，或从实现该接口的类中直接或间接导出的类。如果想要使自己开发的组件或控件在其他编程语言中可以使用，则必须符合公共语言规范 (CLS) 并确保所有公共和保护的成员都符合 CLS。.NET 框架的 SDK 为四种符合 CLS 的语言：Visual Basic .NET、C#、C++ 的托管扩展和 JScript .NET 提供编译器。

控件是提供（或实现）用户界面 (UI) 功能的组件。.NET 框架为控件提供两个基类：一个用于客户端 Windows 窗体控件，另一个用于 ASP.NET 服务器控件。它们分别是 System.Windows.Forms.Control 和 System.Web.UI.Control。.NET 框架类库中的所有控件都是直接或间接从这两个类导出的。System.Windows.Forms.Control 从 Component 导出，它本身提供 UI 功能。System.Web.UI.Control 实现 IComponent，它还提供结构，在这个结构上可以很方便地添加 UI 功能。需要注意的是：每个控件都是一个组件，但并不是每个组件都是控件。

无论是组件还是控件，都必须实现容器或站点。所谓容器是一个实现 System.ComponentModel.IContainer 接口的类，或从实现该接口的类导出的类。容器在逻辑上包含一个或多个组件，这些组件叫做容器的子组件。所谓站点是一个实现 System.ComponentModel.ISite 接口的类，或从实现该接口的类导出的类。站点由容器提供，用来管理其子组件及与子组件进行通信。通常，容器和站点作为一个单元来实现。如果开发人员正在为 Windows 窗体或 Web 窗体页 (ASP.NET 页) 开发组件和控件，则不需要主动实现容器或站点，因为 Windows 窗体和 Web 窗体的设计器就是 Windows 窗体和 ASP.NET 服务器控件的容器。容器向放置在其中的组件和控件提供服务。在设计时，控件放置在设计器中并从设计器获得服务。在了解了这些基本概念以后，下面开始网络组件和控件的开发。

 注意：组件与控件的属性的使用方法在本章 8.4 节讲解。所以这样安排，是因为属性属于组件与控件开发的高级内容，读者在了解了组件与控件开发的基本方法后，再学习属性的开发与使用就顺理成章了。

8.1 网络组件的开发基础

本节首先用 VS 7.0 开发几个简单的 Windows 组件，使读者熟悉组件的开发方法，然后在此基础上开发网络组件。

8.1.1 第一个组件的开发

单击菜单“文件”→“新建”→“项目”，打开“新建项目”对话框，在“项目类型”里选择“Visual C# 项目”，在“模板”里选择“类库”，然后输入适当的文件名（本例为 ClassLibrary2）并选择适当的路径，完成上述操作后单击“确定”，进入代码编辑窗口。在代码编辑窗口里系统自动产生的代码如下：

```
using System;
namespace ClassLibrary2
{
    /// <summary>
    /// Class1 的摘要说明。
    /// </summary>
    public class Class1
    {
        public Class1()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }
        //
        //在这里可以添加方法，以实现特定功能（笔者按）。
        //
    }
}
```

本例在上述代码中添加如下代码：

```
public string hello()
{
    return "你好！欢迎阅读本书，谢谢！";
}
```

全部代码变为：

```
using System;
namespace ClassLibrary2
{
    /// <summary>
    /// Class1 的摘要说明。
    /// </summary>
    ///
    public class Class1
    {
        public Class1()
        {
```

```

    //
    // TODO: 在此处添加构造函数逻辑
    //

}

public string hello(){
    return "你好!欢迎阅读本书, 谢谢!";
}

}

```

单击菜单“生成”→“生成”，将上述代码编译成 ClassLibrary2.dll。下面可以使用该组件了。

新建一个 Windows 应用程序项目（本例项目名称为 WindowsApplication10），在窗体上拖放一个 TextBox 控件和一个 Button 控件。然后单击菜单“项目”→“添加引用”，打开图 8-1 窗口，单击“浏览”按钮，将 ClassLibrary2.dll 加入进去，然后单击“确定”即可将该组件加入到项目中。

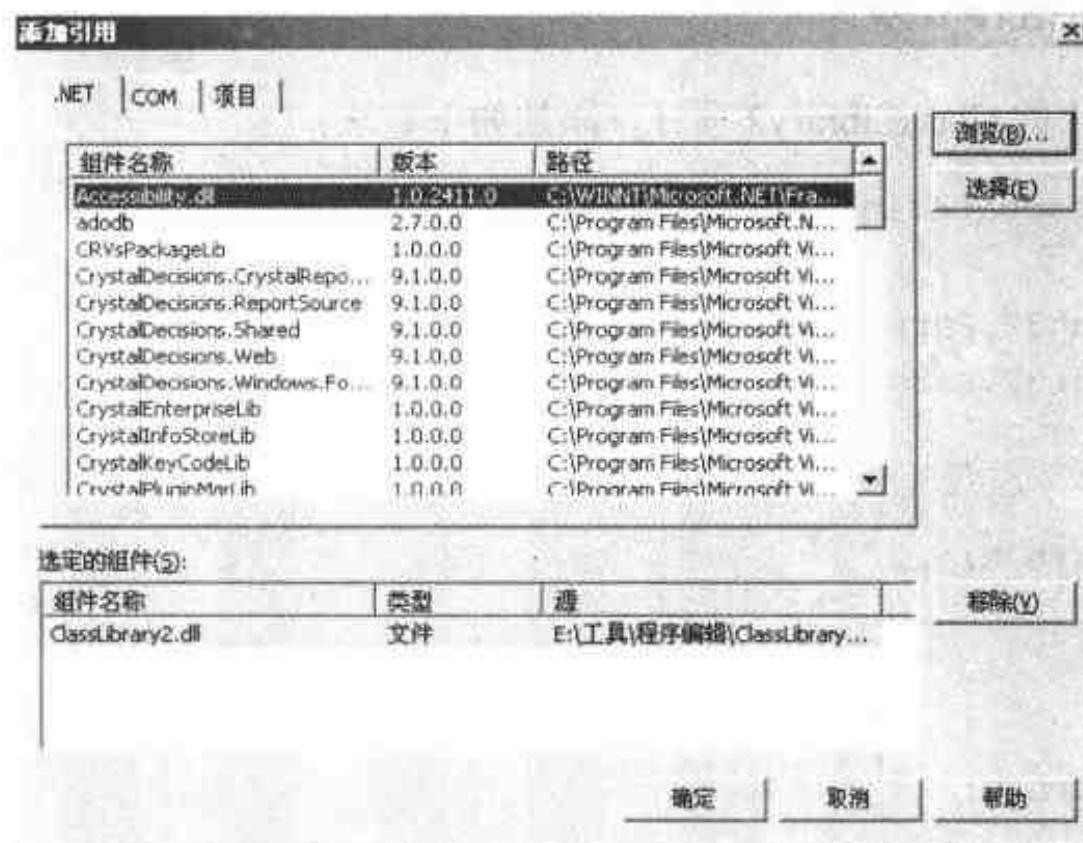


图 8-1 添加引用

打开代码窗口，添加如下代码：

```
using ClassLibrary2;
```

然后添加私有成员：

```
private ClassLibrary2.Class1 myclass;
```

回到窗体设计窗口，找到窗体构造方法 InitializeComponent()，在该代码下面构造组件类“myclass”，代码为：

```
myclass=new ClassLibrary2.Class1();
```

然后双击 button1 控件，为该控件添加 Click 事件。下面是该事件的全部代码：

```
private void button1_Click(object sender, System.EventArgs e)
{
    myclass=new ClassLibrary2.Class1();
    textBox1.Text=myclass.hello();
}
```

编译并执行程序，单击 button1 控件，执行结果如图 8-2 所示。



图 8-2 使用组件

8.1.2 带参数的组件开发

重新打开 4.1.1 的 ClassLibrary2 项目，添加如下方法：

```
public int compare(int aa,int bb)
{
    int x=Int32.Parse(aa);
    int y=Int32.Parse(bb);
    if(x>y)
    {
        return x;
    }
    if(x<y)
    {
        return y;
    }
    return x;
}
```

注意：最后一句是 return x，读者也许觉得该句没什么用。其实系统要求所有路径都有返回值，如果去掉该句，当 x 等于 y 时就没有返回值了，在编译项目时就会出错。

ClassLibrary2 项目的全部代码变为：

```
using System;

namespace ClassLibrary2
{
    /// <summary>
    /// Class1 的摘要说明。
}
```

```
/// </summary>
///

public class Class1
{
    public Class1()
    {
        //
        // TODO: 在此处添加构造函数逻辑
        //
    }

    public string hello()
    {
        return "你好!欢迎阅读本书, 谢谢!";
    }

    public int compare(string aa, string bb)
    {
        int x=Int32.Parse(aa);
        int y=Int32.Parse(bb);
        if(x>y){
            return x;
        }
        if(x<y){
            return y;
        }
        return x;
    }
}
```

单击菜单“生成”→“生成”，重新生成 ClassLibrary2.dll。然后重新打开上例项目 WindowsApplication10，如图 8-3 所示，在窗体上再拖放两个 Label 控件、两个 TextBox 控件和一个 Button 控件，其属性如图所示。

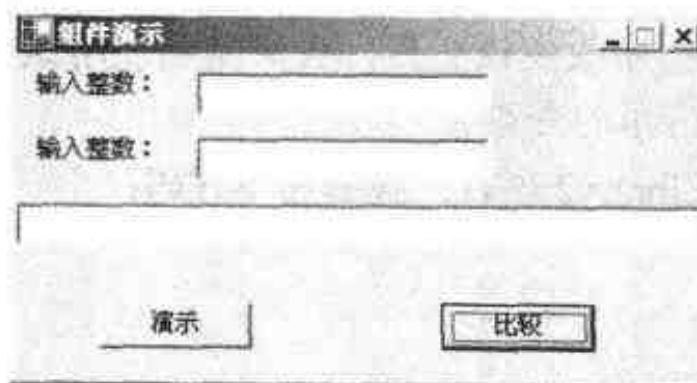


图 8-3 使用带参数的组件程序的界面

双击“比较”按钮，为该控件添加 Click 事件。下面是该事件的全部代码：

```
private void button2_Click(object sender, System.EventArgs e)
{
```

```

myclass=new ClassLibrary2.Class1();
int i=0;
try
{
    i=myclass.compare(textBox2.Text, textBox3.Text);
    textBox1.Text="较大的数值为: "+i.ToString();
}
catch{MessageBox.Show("请输入整数! ");}
}

```

编译并执行程序，随便输入两个整数，单击“比较”，执行结果如图 8-4 所示。

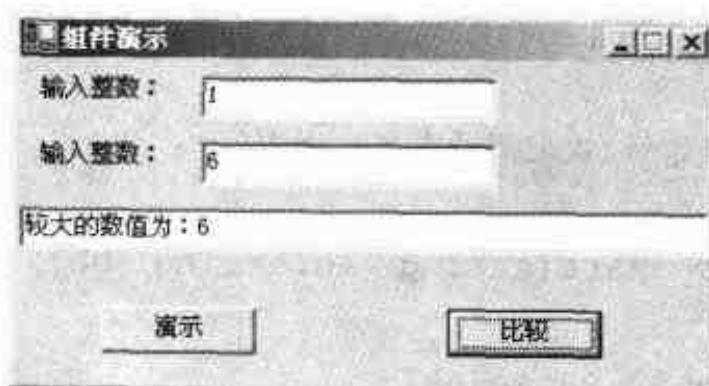


图 8-4 比较不相等参数的运行结果

如果输入的值相等，执行结果如图 8-5 所示。

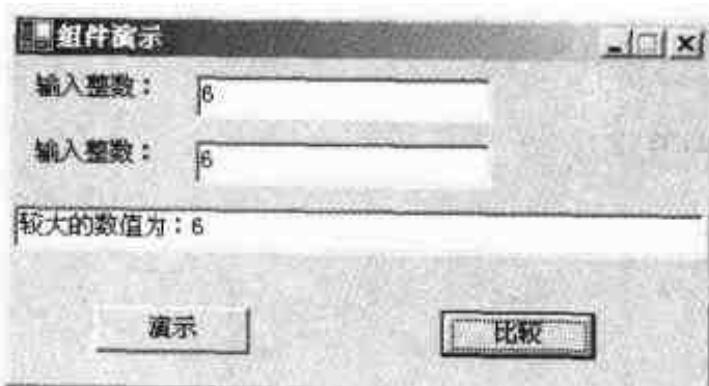


图 8-5 比较相等参数的运行结果

8.1.3 如何定义全局变量

有时候，一个变量不仅要在在一个方法里使用，而且还要在其他方法里使用，甚至在其他程序里使用。一个方法或程序改变该变量，会影响到使用该变量的另一个方法或程序。下面定义一个可在整个组件应用的变量。

重新打开 8.1.1 的 ClassLibrary2 项目，找到如下代码：

```

using System;

namespace ClassLibrary2
{
    /// <summary>
    /// Class1 的摘要说明。
    /// </summary>
    ///

```

```
public class Class1
{
```

在上述代码的下面添加如下代码：

```
public string str="";
```

然后把 4.1.1 中的 hello()方法代码修改为：

```
public string hello()
{
    return str;
}
```

现在组件 ClassLibrary2 的全部代码变为：

```
using System;

namespace ClassLibrary2
{
    /// <summary>
    /// Class1 的摘要说明
    /// </summary>
    ///

    public class Class1
    {
        public string str="";
        public Class1()
        {
            //
            // TODO：在此处添加构造函数逻辑
            //
        }
        public string hello()
        {
            return str;
        }
        public int compare(string aa,string bb){
            int x=Int32.Parse(aa);
            int y=Int32.Parse(bb);
            if(x>y){
                return x;
            }
            if(x<y){
                return y;
            }
            return x;
        }
    }
}
```

```
    }
}
}
```

重新打开项目 WindowsApplication10，将“演示”按钮的 Click 事件修改为：

```
private void button1_Click(object sender, System.EventArgs e)
{
    myclass=new ClassLibrary2.Class1();
    myclass.str=textBox2.Text;
    textBox3.Text=myclass.hello();
}
```

编译并执行程序，在 textBox2 里输入“您好，欢迎学习 C#”，然后单击“演示”按钮，在 textBox3 里将输出“您好，欢迎学习 C#”。

Windows 组件开发就讲到这里。下面开始尝试开发网络组件。

8.1.4 TcpListener 基础服务器组件开发

新建一个组件库，添加如下引用：

```
using System.Net;
using System.Net.Sockets;
```

然后添加如下成员：

```
private TcpListener listener;
public int port;
```

其中，`listener` 用于该组件类内部使用，`port` 供外部程序赋值。下面开始开发具有特定功能的公共方法。

下列方法用于监听端口：

```
public void listen()
{
    listener=new TcpListener(port);
    listener.Start();
}
```

下列方法用于接收客户端的连接并返回已经连接的套接字对象。

```
public Socket accept()
{
    Socket mySock=listener.AcceptSocket();
    return mySock;
}
```

下列方法用于从客户端接收数据。

◆ 注意：.NET SDK 要求每个路径都有返回值，否则在编译时就通不过。

```
public string receive(Socket mySock)
{
    if(mySock.Connected)
    {
        NetworkStream netStream=new NetworkStream(mySock);
        byte[] messageByte=new Byte[64];
        netStream.Read(messageByte,0,messageByte.Length);
        string readMessage=System.Text.Encoding.
            BigEndianUnicode.GetString(messageByte);
        return readMessage;
    }
    else{return "连接未建立!";}
}
```

下列代码用于关闭监听。方法很简单，只使用 TcpListener 类的 Stop 方法。

```
public void closeListen()
{
    listener.Stop();
}
```

到此为止，一个可实现监听和从客户端接收数据的组件开发成功，以后在开发服务器程序时就可以使用。下面是该组件的全部代码。

```
using System;
using System.Net;
using System.Net.Sockets;

namespace TCPListen
{
    /// <summary>
    /// Class1 的摘要说明
    /// </summary>
    public class Class1
    {

        private TcpListener listener;
        public int port;
        public Class1()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }
        public void listen()
        {
            listener=new TcpListener(port);
        }
    }
}
```

```

        listener.Start();
    }

    public Socket accept(){
        Socket mySock=listener.AcceptSocket();
        return mySock;
    }

    public string receive(Socket mySock){

        if(mySock.Connected)
        {
            NetworkStream netStream=new NetworkStream(mySock);
            byte[] messageByte=new Byte[64];
            netStream.Read(messageByte,0,messageByte.Length);
            string readMessage=System.Text.Encoding
                .BigEndianUnicode.GetString(messageByte);
            return readMessage;
        }
        else{return "连接未建立！";}
    }

    public void closeListen(){
        listener.Stop();
    }
}
}

```

8.1.5 使用基础服务器的组件

使用网络组件与使用普通 Windows 组件没有区别。新建一个 Windows 程序项目（本例文件名为 useListenClass）。下面在该程序中使用 8.1.4 节开发的网络组件。

(1) 添加引用。

```

using System.Net;
using System.Net.Sockets;
using System.Threading;

```

然后单击菜单“项目”→“添加引用”，将 TCPListen.dll 加入进来，然后添加如下代码：

```
using TCPListen;
```

(2) 添加私有成员。

```
private TCPListen.Class1 myClass;
```

(3) 构造组件 TCPListen 类的成员。

在构造方法“InitializeComponent();”下添加如下方法：

```
myClass=new TCPListen.Class1();
```

(4) 设计窗体。

如图 8-6 所示，在窗体上拖放两个 Label 控件、两个 Button 控件，一个 TextBox 控件、一个 RichTextBox 控件。其中“监听端口”文本框是 textBox1，“客户信息”文本框是 textBox2，“开始监听”按钮是 button1，“关闭监听”按钮是 button2。

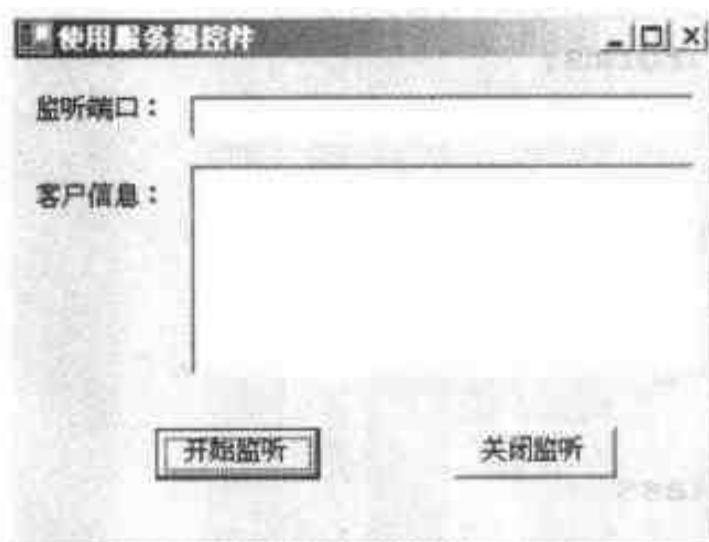


图 8-6 使用 TcpListener 基础服务器组件程序的界面

(5) “开始监听”按钮的 Click 事件代码。

```
private void button1_Click(object sender, System.EventArgs e)
{
    int port=0;
    port=Int32.Parse(textBox1.Text);
    myClass.port=port;
    myClass.listen();
    Thread thread=new Thread(new ThreadStart(receive));
    thread.Start();
}
```

下面是线程同步方法 receive 的代码：

```
private void receive()
{
    Socket sock=myClass.accept();
    while(!round)
    {
        string clientMessage=myClass.receive(sock);
        richTextBox1.AppendText(clientMessage+"\r\n");
    }
}
```

(6) “关闭监听”按钮的 Click 事件的代码。

```
private void button2_Click(object sender, System.EventArgs e)
{
    round=true;
    myClass.closeListen();
}
```

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using TCPListen;
using System.Threading;

namespace useListenClass
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private System.Windows.Forms.Label label2;
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        private TCPListen.Class1 myClass;
        private bool round=false;
        public Form1()
        {
            //
            // Windows 窗体设计器支持所必需的
            //
            InitializeComponent();
            myClass=new TCPListen.Class1();
            //
            // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
            //
        }
        /// <summary>
        /// 清理所有正在使用的资源
        /// </summary>
```

```
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// 应用程序的主入口点。
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    int port=0;
    port=Int32.Parse(textBox1.Text);
    myClass.port=port;
    myClass.listen();
    Thread thread=new Thread(new ThreadStart(receive));
    thread.Start();
}

private void receive(){
    Socket sock=myClass.accept();
    while(!round)
    {
        string clientMessage=myClass.receive(sock);
        richTextBox1.AppendText(clientMessage+"\r\n");
    }
}

private void button2_Click(object sender, System.EventArgs e)
{
    round=true;
    myClass.closeListen();
}
```

```
    }  
}
```

(7) 演示。

启动该程序，输入适当的端口，单击“开始监听”按钮，开始监听，然后启动一个客户端程序。图 8-7 是本程序的运行情况。

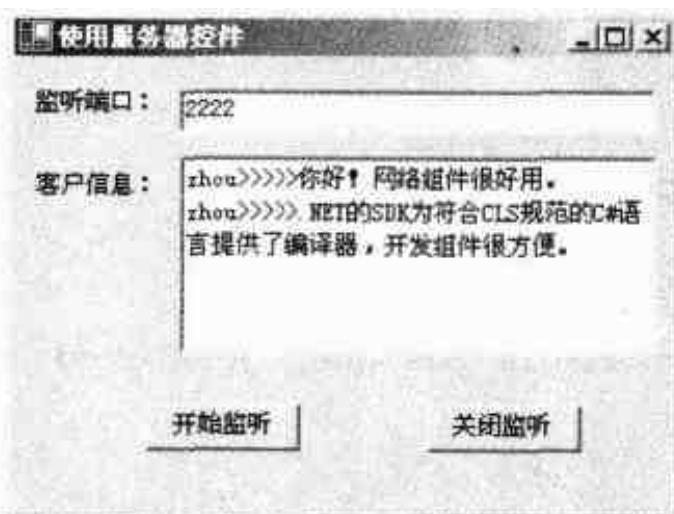


图 8-7 使用 TcpListener 基础服务器组件程序的运行结果

图 8-8 是客户端的运行情况。

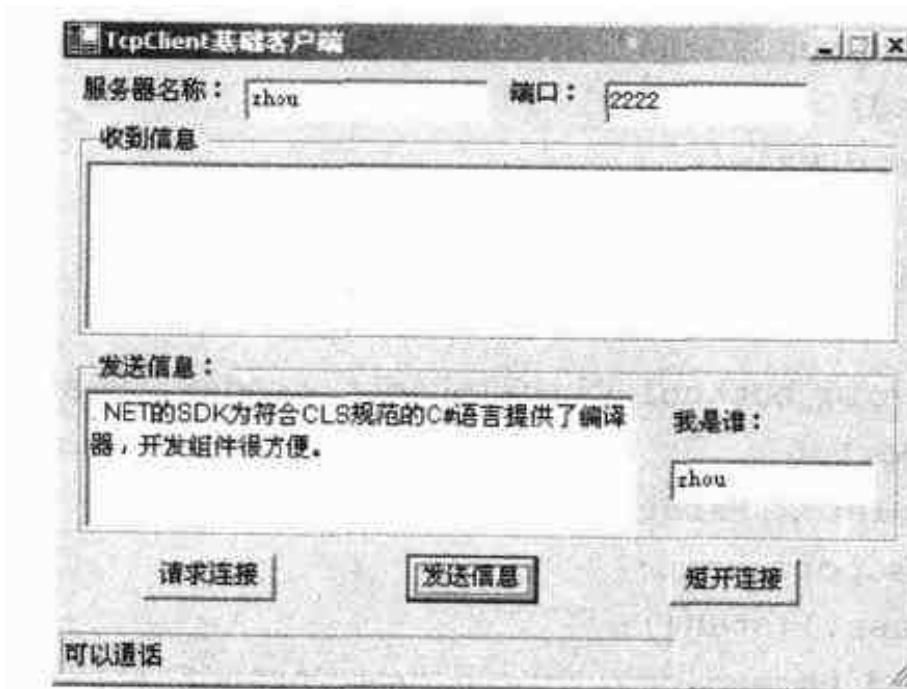


图 8-8 TcpClient 基础客户端运行情况

8.2 FTP 服务器组件开发

文件传输是网络开发的基本应用之一，在许多程序中都要使用，因此，将 FTP 服务器开发成组件具有非常重要的意义，可以在不同程序中反复使用。本节将本书第 4 章开发的 FTP 服务器转化成 FTP 服务器组件。读者可以参考本节内容将应用程序转化成组件，以供程序开发使用。

8.2.1 FTP 服务器组件开发

(1) 新建组件类项目。

新建一个组件类库，本例文件名为 FTPServerClass。

(2) 添加以下成员。

```
private TcpListener listener;
public int port;
private bool pasv=false;
private bool pasvPort=false;
private TcpListener newListener;
private int newPort;
private FileStream filestream;
```

(3) 添加监听端口的方法。

```
public void listen()
{
    listener=new TcpListener(port);
    listener.Start();
}
```

(4) 添加接收客户连接的方法。

```
public Socket accept()
{
    Socket mySock=listener.AcceptSocket();
    return mySock;
}
```

(5) 添加接收数据并传输文件的方法。

```
public string receive(Socket newClient)
{
    string parameter=null;
    string command=null;
    if(newClient.Connected)
    {
        NetworkStream netStream=new NetworkStream(newClient);
        WriteToNetStream(ref netStream,"220 FTP Server Ready!");
        byte[] byteMessage=new byte[1024];
        int i=newClient.Receive(byteMessage,byteMessage.Length,0);
        string ss=System.Text.Encoding.ASCII.GetString(byteMessage);
        int x=ss.IndexOf("\r\n",0);
        int y=ss.IndexOf(" ",0);
        string commMessage=
            System.Text.Encoding.ASCII.GetString(byteMessage,0, x);
        if(y!=-1)
        {
            command=commMessage.Substring(0,y);
        }
        else{command=commMessage.Substring(0,x);}
    }
}
```

```
command=command.ToLower();
if(y!=-1)
{
    parameter=commMessage.Substring(y+1,x-y-1);
    parameter=parameter.ToLower();
}
else{parameter="";
commMessage.Remove(0,commMessage.Length);
if(command=="list")
{
    try
    {
        string[] str=new string[1024];
        string dir=null;
        for(int k=0;
            k<Directory.GetDirectories(parameter).Length;k++)
        {
            str[k]=Directory.GetDirectories(parameter)[k];
            dir=dir+"目录: "+str[k]+"\r\n";
        }
        for(int k=0;k<Directory.GetFiles(parameter).Length;k++)
        {
            str[k]=Directory.GetFiles(parameter)[k];
            dir=dir+"文件: "+str[k]+"\r\n";
        }
        WriteToNetStream(ref netStream,
            "125 the files and directory is here");
        sendDir(ref netStream,dir);
    }
    catch{WriteToNetStream(ref netStream,
        "502 unsuccessful,try again ");}
}
else if(command=="retr")
{
    try
    {
        if((pasv==false)&&(pasvPort==false))
        {
            string path=parameter;
            newListener=new TcpListener(port+1);
            newListener.Start();
            WriteToNetStream(ref netStream,
                "150 data link listening");
            Socket nextLink=newListener.AcceptSocket();
            if(nextLink.Connected)
            {
```

```
        WriteToNetStream(ref netStream,
                          "125 now the file's data will be sended");
        NetworkStream stream=new NetworkStream(nextLink);
        transfer(ref stream,path);
        stream.Close();
        nextLink.Close();
        newListener.Stop();
    }
}
else if((pasv==true)&&(pasvPort==true))
{
    string path=parameter;
    newListener=new TcpListener(newPort);
    newListener.Start();
    WriteToNetStream(ref netStream,
                      "150 data link listening");
    Socket nextLink=newListener.AcceptSocket();
    if(nextLink.Connected)
    {
        WriteToNetStream(ref netStream,
                          "125 now the file's data will be sended");
        NetworkStream stream=new NetworkStream(nextLink);
        transfer(ref stream,path);
        stream.Close();
        nextLink.Close();
        newListener.Stop();
        pasv=false;
        pasvPort=false;
    }
}
else
{
    WriteToNetStream(ref netStream,
                      "350 PASV OR PORT command needed ");
}
}
catch{WriteToNetStream(ref netStream,
                      "502 performed unsuccessful,try again");}
}
else if(command=="pasv")
{
    try
    {
        pasv=true;
        WriteToNetStream(ref netStream,"227 now the server pasv");
    }
}
```

```
        catch(WriteToNetStream(ref netStream,
            "502 performed unsuccessful,try again"));}

    else if(command=="port")
    {
        try
        {
            if(pasv==true)
            {
                pasvPort=true;
                newPort=Int32.Parse(parameter);
                WriteToNetStream(ref netStream,
                    "200 new data link listen begin");
            }
            else{WriteToNetStream(ref netStream,
                "350 PASV command needed");}
        }
        catch(WriteToNetStream(ref netStream,"504 the command can't be
            performed with the parameter ");}
    }

    else if(command=="help")
    {
        try
        {
            WriteToNetStream(ref netStream,"215 the FTP system ready,it
                can the comman such as LIST、PORT、PASV、RETR、HELP、
                QUIT.before PORT command, PASV command fistly.we are
                sorry that the FTP system can't other comand.");
        }
        catch(WriteToNetStream(ref netStream,
            "502 performed unsuccessful,try again"));
    }

    else if(command=="quit")
    {
        try
        {
            WriteToNetStream(ref netStream,"221 connection closed");
            newClient.Close();
        }
        catch(WriteToNetStream(ref netStream,
            "502 performed unsuccessful,try again"));
    }

    else{WriteToNetStream(ref netStream,"500 unrecognized command");}
    string returnString="Command: "+command+" parameter: "
        +parameter+"\r\n";
    return returnString;
```

```
    }
    else{
        return "连接未建立!";
    }
}
```

(6) 添加其他辅助方法。

辅助方法主要是发送反馈、发送目录、读取文件并发送文件流（转化成网络流）等私有方法，和第4章的FTP服务器程序没有区别，这里不再赘述。

下面是本组件的代码：

```
using System;
using System.Net;
using System.Net.Sockets;
using System.IO;

namespace FTPServerClass
{
    /// <summary>
    /// Class1 的摘要说明。
    /// </summary>
    public class Class1
    {
        private TcpListener listener;
        public int port;
        private bool pasv=false;
        private bool pasvPort=false;
        private TcpListener newListener;
        private int newPort;
        private FileStream filestream;
        public Class1()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }
        public void listen()
        {
            listener=new TcpListener(port);
            listener.Start();
        }
        public Socket accept()
        {
            Socket mySock=listener.AcceptSocket();
            return mySock;
        }
    }
}
```

```
}

public string receive(Socket newClient)
{
    string parameter=null;
    string command=null;
    if(newClient.Connected)
    {
        NetworkStream netStream=new NetworkStream(newClient);
        WriteToNetStream(ref netStream,"220 FTP Server Ready!");
        byte[] byteMessage=new byte[1024];
        int i=newClient.Receive(byteMessage,byteMessage.Length,0);
        string ss=
            System.Text.Encoding.ASCII.GetString(byteMessage);
        int x=ss.IndexOf("\r\n",0);
        int y=ss.IndexOf(" ",0);
        string commMessage=
            System.Text.Encoding.ASCII.GetString(byteMessage,
            0, x);
        if(y!=-1)
        {
            command=commMessage.Substring(0,y);
        }
        else{command=commMessage.Substring(0,x);}
        command=command.ToLower();
        if(y!=-1)
        {
            parameter=commMessage.Substring(y+1,x-y-1);
            parameter=parameter.ToLower();
        }
        else{parameter=""}
        commMessage.Remove(0,commMessage.Length);
        if(command=="list")
        {
            try
            {
                string[] str=new string[1024];
                string dir=null;
                for(int k=0;
                    k<Directory.GetDirectories(parameter).Length;
                    k++)
                {
                    str[k]=Directory.GetDirectories
                        (parameter)[k];
                    dir=dir+"目录: "+str[k]+"\r\n";
                }
                for(int k=0;
                    k<Directory.GetFiles(parameter).Length;k++)
                {
                    str[k]=Directory.GetFiles
                        (parameter)[k];
                    dir=dir+"文件: "+str[k]+"\r\n";
                }
            }
            catch{}
        }
    }
}
```

```
{  
    str[k]=Directory.GetFiles(parameter)[k];  
    dir=dir+"文件: "+str[k]+"\r\n";  
}  
WriteToNetStream(ref netStream,  
    "125 the files and directory is here");  
sendDir(ref netStream,dir);  
}  
catch{WriteToNetStream(ref netStream,  
    "502 unsuccessful,try again ");}  
}  
else if(command=="retr")  
{  
    try  
{  
        if((pasv==false)&&(pasvPort==false))  
        {  
            string path=parameter;  
            newListener=new TcpListener(port+1);  
            newListener.Start();  
            WriteToNetStream(ref netStream,  
                "150 data link listening");  
            Socket nextLink=newListener.AcceptSocket();  
            if(nextLink.Connected)  
            {  
                WriteToNetStream(ref netStream,  
                    "125 now the file's data will be sended");  
                NetworkStream stream=  
                    new NetworkStream(nextLink);  
                transfer(ref stream,path);  
                stream.Close();  
                nextLink.Close();  
                newListener.Stop();  
            }  
        }  
        else if((pasv==true)&&(pasvPort==true))  
        {  
            string path=parameter;  
            newListener=new TcpListener(newPort);  
            newListener.Start();  
            WriteToNetStream(ref netStream,  
                "150 data link listening");  
            Socket nextLink=newListener.AcceptSocket();  
            if(nextLink.Connected)  
            {  
                WriteToNetStream(ref netStream,  
                    "125 now the file's data will be sended");  
            }  
        }  
    }  
}
```

```
        NetworkStream stream=
            new NetworkStream(nextLink);
        transfer(ref stream,path);
        stream.Close();
        nextLink.Close();
        newListener.Stop();
        pasv=false;
        pasvPort=false;
    }
}
else
{
    WriteToNetStream(ref netStream,
        "350 PASV OR PORT command needed ");
}
}
catch{WriteToNetStream(ref netStream,
    "502 performed unsuccessful,try again ");}

else if(command=="pasv")
{
    try
    {
        pasv=true;
        WriteToNetStream(ref netStream,
            "227 now the server pasv");
    }
    catch{WriteToNetStream(ref netStream,
        "502 performed unsuccessful,try again");}

else if(command=="port")
{
    try
    {
        if(pasv==true)
        {
            pasvPort=true;
            newPort=Int32.Parse(parameter);
            WriteToNetStream(ref netStream,
                "200 new data link listen begin");
        }
        else{WriteToNetStream(ref netStream,
            "350 PASV command needed");}
    }
    catch{WriteToNetStream(ref netStream,"504 the command
        can't be performed with the parameter ");}
```

```
        }
        else if(command=="help")
        {
            try
            {
                WriteToNetStream(ref netStream,"215 the FTP system
                    ready,it can  the comman such as LIST、PORT、PASV、
                    RETR、HELP、QUIT.before PORT command,
                    PASV command fistly.we are sorry that the FTP
                    system can't  other comand.");
            }
            catch{WriteToNetStream(ref netStream,"502 performed
                unsuccessful,try again");}
        }
        else if(command=="quit")
        {
            try
            {
                WriteToNetStream(ref netStream,
                    "221 connection closed");
                newClient.Close();
            }
            catch{WriteToNetStream(ref netStream,
                "502 performed unsuccessful,try again");}
        }
        else{WriteToNetStream(ref netStream,
            "500 unrecognized command");}
        string returnString="Command: "+command+" parameter: "
            +parameter+"\r\n";
        return returnString;
    }

    else{
        return "连接未建立!";
    }
}

private void transfer(ref NetworkStream stream,string path)
{
    filestream=
        new FileStream(path, FileMode.Open, FileAccess.Read);
    int number;
    //定义缓冲区
    byte[] bb=new byte[8];
    //循环读文件
```

```
// NetworkStream stream=new NetworkStream(newClient);
while((number=filestream.Read(bb,0,8))!=0)
{//向客户端发送流
    stream.Write(bb,0,8);
    //刷新流
    stream.Flush();
    bb=new byte[8];
}
filestream.Close();
stream.Close();
}

private void WriteToNetStream(ref NetworkStream NetStream,
    string ToSend)
{
    string stringToSend = ToSend + "\r\n";
    byte[] arrayToSend =
        System.Text.Encoding.ASCII .GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}

private void sendDir(ref NetworkStream NetStream, string Dir)
{
    string stringToSend = Dir + "\r\n";
    byte[] arrayToSend =
        System.Text.Encoding.BigEndianUnicode.GetBytes
        (stringToSend.ToCharArray());
    NetStream.Write(arrayToSend, 0, arrayToSend.Length);
    NetStream.Flush();
}

public void stop(){
    listener.Stop();
}
}
```

8.2.2 使用 FTP 服务器组件

下面演示如何使用 FTP 服务器组件开发 FTP 服务器。读者可以仿照本节内容，将该组件加入到需要文件传输的程序中。

(1) 新建项目。

新建一个 Windows 程序项目，本例文件名称为 useFTPServerClass。

(2) 界面设计。

如图 8-9 所示，“监听端口”文本框是 textBox1， “客户信息”文本框是 richTextBox1， “开始监听”按钮是 button1， “关闭监听”按钮是 button2。

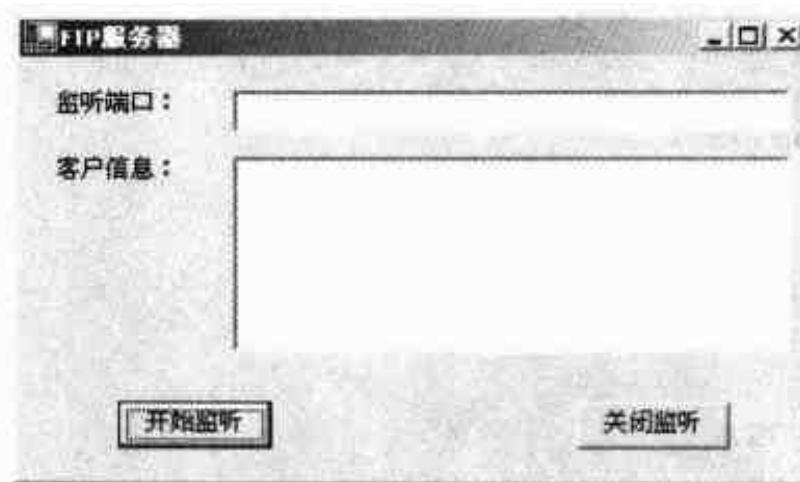


图 8-9 使用 FTP 服务器组件程序的界面

(3) 添加引用。

单击菜单“项目”→“添加引用”，将 FTPServerClass.dll 添加进来，然后添加下列代码：

```
using FTPServerClass;
using System.Net;
using System.Net.Sockets;
using System.Threading;
```

(4) 添加私有成员。

```
private FTPServerClass.Class1 myFTPClass;
private bool round=false;
```

(5) “开始监听”按钮的 Click 事件的代码。

```
private void button1_Click(object sender, System.EventArgs e)
{
    myFTPClass=new FTPServerClass.Class1();
    int port=0;
    port=Int32.Parse(textBox1.Text);
    myFTPClass.port=port;
    myFTPClass.listen();
    Thread thread=new Thread(new ThreadStart(receive));
    thread.Start();
}
```

线程同步方法 receive 方法的代码。

```
private void receive()
{
    Socket sock=myFTPClass.accept();
    while((!round)&&(sock.Connected))
    {
        string getReturn=myFTPClass.receive(sock);
        richTextBox1.AppendText(getReturn);
    }
}
```

(6) “关闭监听”按钮的 Click 事件的代码。

```
private void button2_Click(object sender, System.EventArgs e)
{
    round=true;
    myFTPClass.stop();
}
```

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using FTPServerClass;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace useFTPSeverClass
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label1;
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        private FTPServerClass.Class1 myFTPClass;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private bool round=false;
        public Form1()
        {
            //
            // Windows 窗体设计器支持所必需的
            //
            InitializeComponent();
        }
```

```
//  
// TODO: 在 InitializeComponent 调用后添加任何构造函数代码  
//  
}  
/// <summary>  
/// 清理所有正在使用的资源  
/// </summary>  
protected override void Dispose( bool disposing )  
{  
    if( disposing )  
    {  
        if (components != null)  
        {  
            components.Dispose();  
        }  
    }  
    base.Dispose( disposing );  
}  
//下面系统自动产生的代码已经删除  
#region Windows Form Designer generated code  
...  
...  
#endregion  
//上面系统自动产生的代码已经删除  
/// <summary>  
/// 应用程序的主入口点  
/// </summary>  
[STAThread]  
static void Main()  
{  
    Application.Run(new Form1());  
}  
private void button1_Click(object sender, System.EventArgs e)  
{  
    myFTPClass=new FTPServerClass.Class1();  
    int port=0;  
    port=Int32.Parse(textBox1.Text);  
    myFTPClass.port=port;  
    myFTPClass.listen();  
    Thread thread=new Thread(new ThreadStart(receive));  
    thread.Start();  
}  
private void receive()  
{  
    Socket sock=myFTPClass.accept();  
    while((!round)&&(sock.Connected))  
    {
```

```

        string getReturn=myFTPClass.receive(sock);
        richTextBox1.AppendText (getReturn);
    }
}

private void button2_Click(object sender, System.EventArgs e)
{
    round=true;
    myFTPClass.stop();
}

}
}

```

8.2.3 演示

启动 8.2.2 节的程序，输入适当的端口号，然后开始监听端口。启动 FTP 客户端程序，随意下载几个文件。图 8-10 是客户端的运行情况。

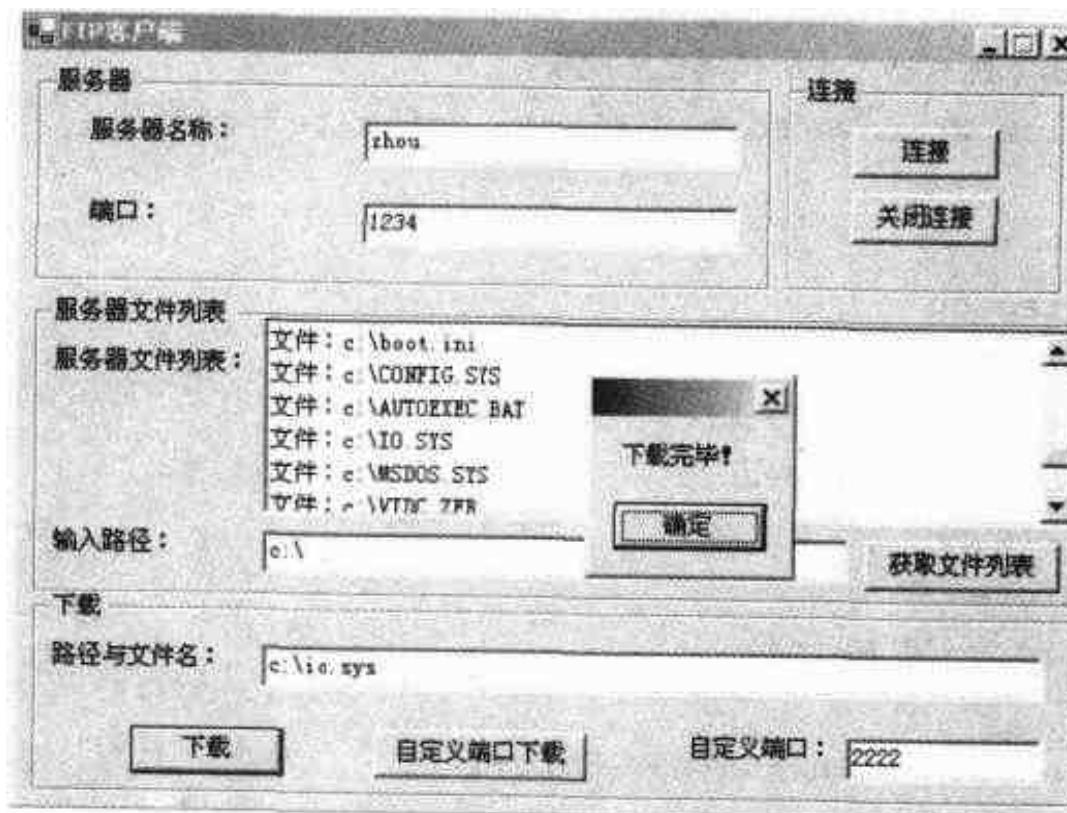


图 8-10 FTP 客户端的运行情况

图 8-11 是 8.2.2 节服务器的运行情况。

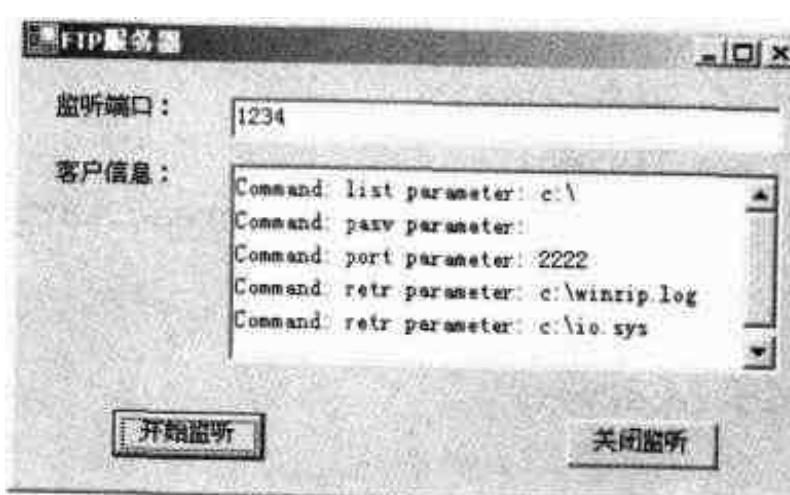


图 8-11 使用 FTP 服务器组件程序的运行情况

8.3 网络控件的开发

控件是具有用户界面的组件，除了界面设计外，控件的开发方法与组件基本相似。下面先开发一个具有打开、保存、打印预览、打印作用的编辑控件，然后再开发一个 TcpClient 客户端控件，最后再讲一下组件与控件必不可少的要素——属性的使用。

8.3.1 编辑控件开发与使用

8.3.1.1 编辑控件开发

单击菜单“文件”→“新建”→“项目”，如图 8-12 所示，在“项目类型”里选择“Visual C# 项目”，在“模板”里选择“Windows 控件库”，然后输入适当的文件名和路径（本例文件名是“文件读写控件”），单击“确定”即可。

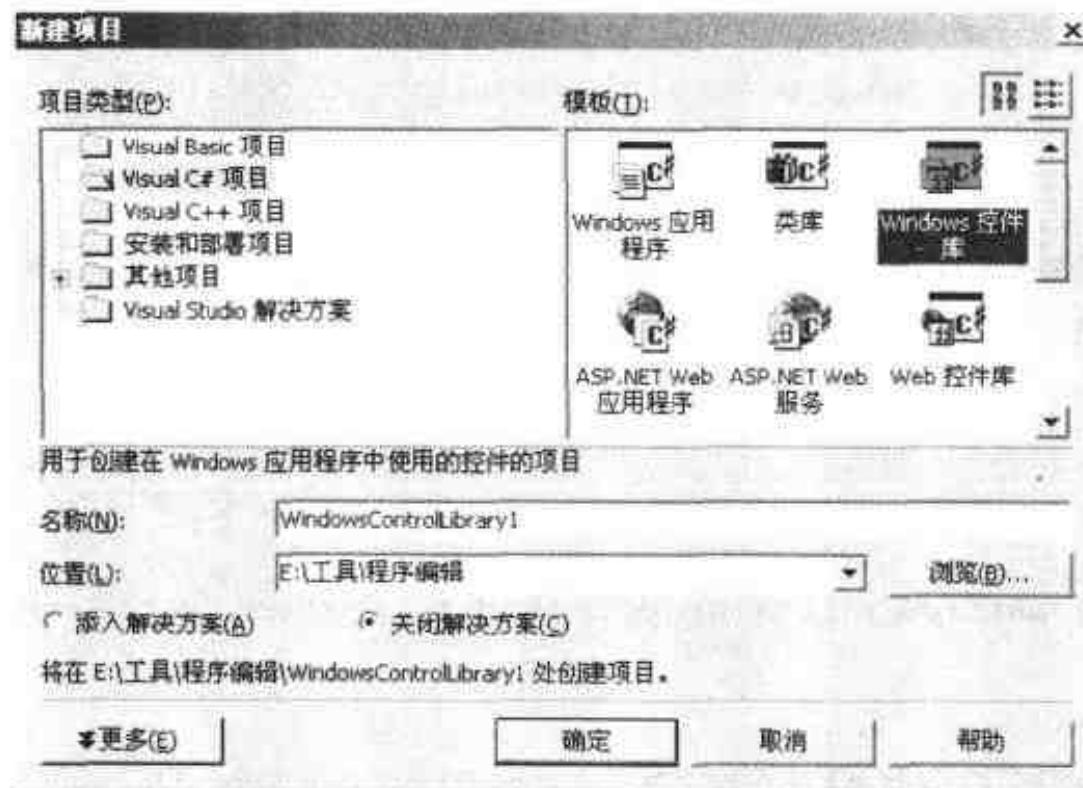


图 8-12 新建 Windows 控件库

完成上述步骤后，即出现如图 8-13 所示的窗体。



图 8-13 控件初始界面

图 8-14 编辑控件界面设计

下面我们就设计这个窗体，如图 8-14 所示，在窗体上拖放一个 RichTextBox1 控件，四个 Button 控件，一个 PrintPreviewDialog 控件，一个 PrintDocument 控件，一个 PrintDialog 控件，一个 OpenFileDialog 控件，一个 SaveFileDialog 控件。将 PrintPreviewDialog 控件的 Document 属性设置为 printDocument1，将 PrintDialog 控件的 Document 属性设为 printDocument1，其他属性如图 8-14 所示。

打开代码窗口，在代码窗口添加引用：using System.IO。

双击“保存”按钮，为该按钮加入 Click 事件，该事件的代码在第 3.1 节已经编好，只要粘贴过来就可以了。为了读者阅读方便，现将代码粘贴如下：

```
private void button1_Click(object sender, System.EventArgs e)
{
    StreamWriter sw=null;
    if(saveFileDialog1.ShowDialog()== DialogResult.OK)
    {
        try
        {
            sw=new StreamWriter(saveFileDialog1.FileName,
                false, System.Text.Encoding.Unicode);
            sw.WriteLine(richTextBox1.Text);
        }
        catch(Exception excep){MessageBox.Show(excep.Message);}
        finally{if(sw!=null){sw.Close();}}
    }
}
```

同理，将“打开”按钮 Click 事件代码粘贴过来，代码如下：

```
private void button2_Click(object sender, System.EventArgs e)
{
    StreamReader sr=null;
    if(openFileDialog1.ShowDialog()== DialogResult.OK)
    {
        try
        {
            sr=new StreamReader(openFileDialog1.FileName,
                System.Text.Encoding.Unicode);
            richTextBox1.Text=sr.ReadToEnd();
        }
        catch(Exception excep){MessageBox.Show(excep.Message);}
        finally{if(sr!=null){sr.Close();}}
    }
}
```

下面是“打印预览”的 Click 事件代码：

```
private void button3_Click(object sender, System.EventArgs e)
{
```

```

StringReader aaa=new StringReader(richTextBox1.Text);
printPreviewDialog1.ShowDialog();
}

```

下面是“打印”的 Click 事件代码：

```

private void button4_Click(object sender, System.EventArgs e)
{
    StringReader aaa=new StringReader(richTextBox1.Text);
    if(printDialog1.ShowDialog()==DialogResult.OK){
        printDocument1.Print();
    }
}

```

到此为止，我们已经完成了该控件的全部编码。单击菜单“生成”→“生成”，即可生成.dll文件。因为本人建立项目时输入的文件名为“文件读写控件”，所以此时生成“文件读写控件.dll”。

8.3.1.2 使用编辑控件

下面我们使用该控件开发一个程序。

(1) 添加引用。新建一个 Windows 应用程序项目，然后单击菜单“项目”→“添加引用”，打开如图 8-15 所示的对话框。

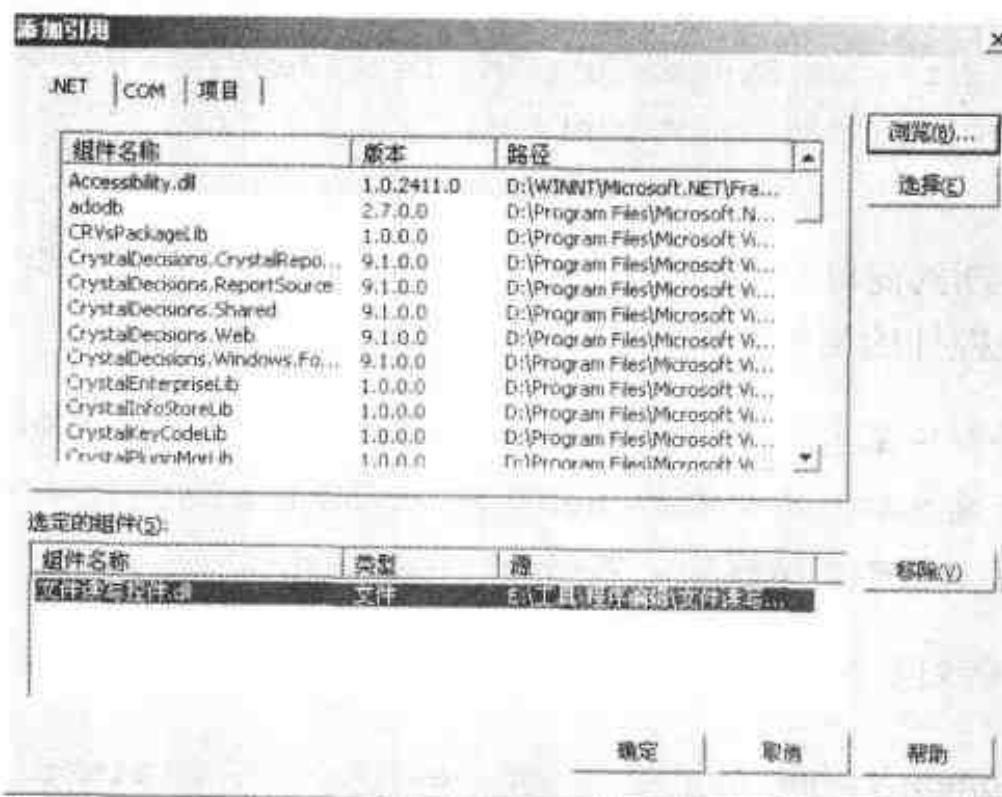


图 8-15 添加控件文件

单击“浏览”按钮，将我们刚刚生成的“文件读写控件.dll”加入到对话框，然后单击“确定”。

完成上述步骤后，打开代码窗口，添加如下代码：“using 文件读写控件;”。

(2) 添加成员。

```
private 文件读写控件.UserControl1 mycontrol;
```

(3) 将控件加到窗体中。

在代码窗口里找到代码：

```

public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();

    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
///

```

在上述代码下面添加如下代码：

```

private void InitializeComponent()
{
    this.components=new System.ComponentModel.Container();
    this.mycontrol=new 文件读写控件.UserControl1 ();
    mycontrol.Size=new System.Drawing.Size(368,200);
    this.Controls.Add(this.mycontrol);
}

```

关闭代码窗口，回到窗体设计窗口，我们发现窗体上添加了该控件（如图 8-16 所示）。现在可以随意修改该控件的属性了。

◆ 注意：代码编辑器窗口里有两个 InitializeComponent()方法，这在编译中是通不过的。您在修改完该控件属性后，将新添的 InitializeComponent()方法改一下名称即可。比如将上面的代码改为（也可直接删除，不影响控件的使用）：

```

private void eee()
{
    this.components=new System.ComponentModel.Container();
    this.mycontrol=new 文件读写控件.UserControl1 ();
    mycontrol.Size=new System.Drawing.Size(368,200);
    this.Controls.Add(this.mycontrol);
}

```

方法名称修改后，编译就可以顺利通过了（窗体中的控件不会消失）。

(4) 演示。编译并执行程序，出现如图 8-17 的窗口，您现在可以进行保存、打开、打印预览、打印等操作了。

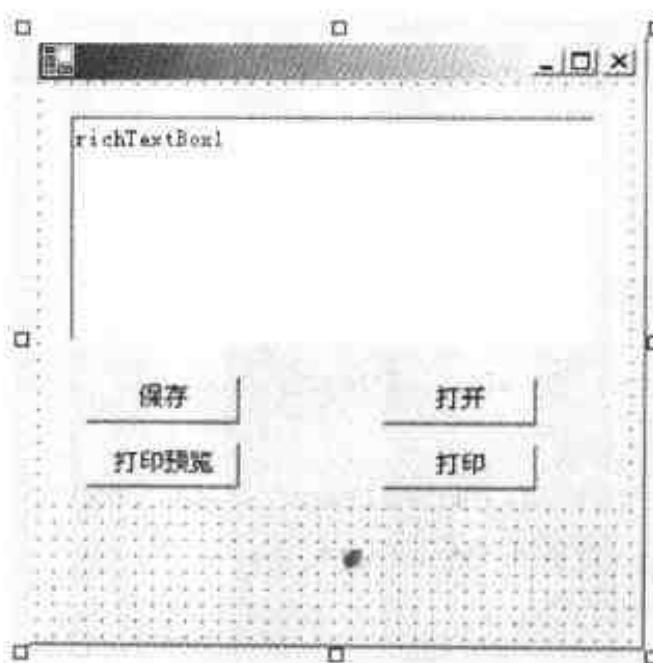


图 8-16 添加控件完成后的窗体



图 8-17 编辑控件编译后的程序界面

上面演示了用户控件的开发方法和使用方法。下一节再开发一个 FTP 客户端控件，以供以后编程使用。

8.3.2 TcpClient 客户端控件开发与使用

8.3.2.1 TcpClient 客户端控件的开发

(1) 新建 Windows 控件库项目。

新建一个 Windows 控件库项目，本例项目名称为 TCPClientControl。

(2) 界面设计。

如图 8-18 所示，“服务器名称”文本框为 textBox1，“端口”文本框为 textBox2，“接收信息”文本框为 richTextBox1，“发送信息”文本框为 richTextBox2。“连接”按钮为 button1，“发送信息”按钮为 button2，“关闭连接”按钮为 button3。

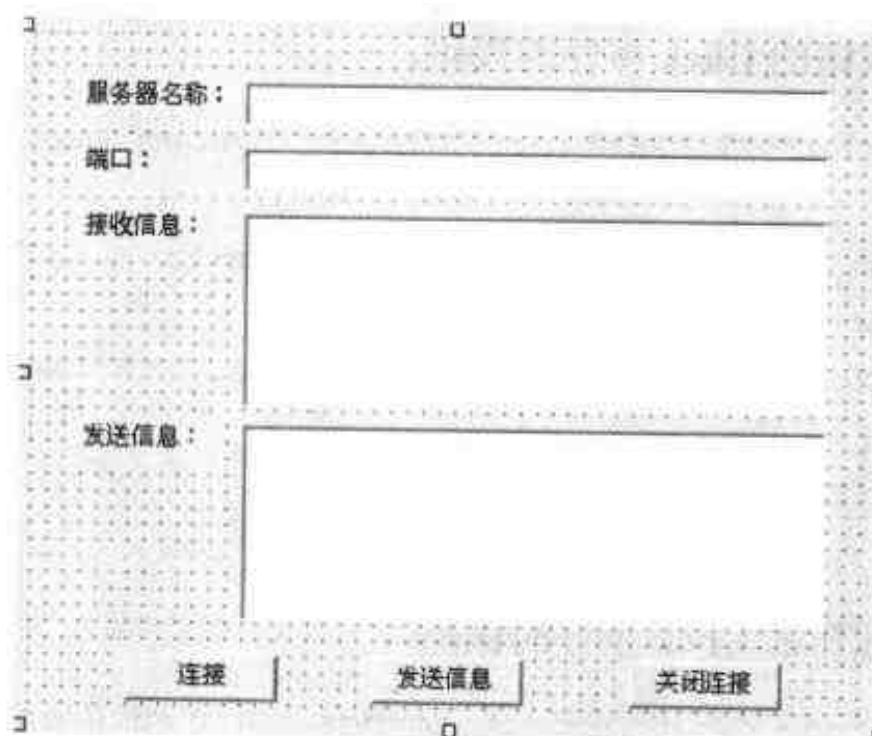


图 8-18 TcpClient 客户端控件界面

(3) 添加引用。

```
using System.Net;
using System.Net.Sockets;
using System.Threading;
```

(4) 添加私有成员。

```
private TcpClient client;
private bool round=false;
```

(5) “连接”按钮的 Click 事件的代码。

```
private void button1_Click(object sender, System.EventArgs e)
{
    client=new TcpClient(textBox1.Text,Int32.Parse(textBox2.Text));
    Thread thread1=new Thread(new ThreadStart(receive));
    thread1.Start();
}
```

线程同步方法 receive 的代码。

```
private void receive()
{
    while(!round)
    {
        NetworkStream netStream=client.GetStream();
        byte[] messageByte=new Byte[1024];
        netStream.Read(messageByte,0,messageByte.Length);
        string readMessage=
            System.Text.Encoding.BigEndianUnicode.GetString
            (messageByte);
        richTextBox1.AppendText(readMessage+"\r\n");
    }
}
```

(6) “发送信息”按钮的 Click 事件的代码。

```
private void button2_Click(object sender, System.EventArgs e)
{
    NetworkStream netStream=client.GetStream();
    byte[] by=new Byte[1024];
    by=System.Text.Encoding.BigEndianUnicode.GetBytes(richTextBox2.Text
        +"\r\n");
    netStream.Write(by,0,by.Length);
}
```

(7) “关闭连接”按钮的 Click 事件的代码。

```
private void button3_Click(object sender, System.EventArgs e)
{
    round=true;
    client.Close();
}
```

下面是该控件的代码：

```
using System;
```

```
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace TCPClientControl
{
    /// <summary>
    /// UserControl1 的摘要说明
    /// </summary>
    public class UserControl1 : System.Windows.Forms.UserControl
    {
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private TcpClient client;
        private System.Windows.Forms.RichTextBox richTextBox2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private bool round=false;
        public UserControl1()
        {
            // 该调用是 Windows.Forms 窗体设计器所必需的
            InitializeComponent();
            // TODO: 在 InitializeComponent 调用后添加任何初始化
        }
        /// <summary>
        /// 清理所有正在使用的资源
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
```

```
        if( components != null )
            components.Dispose();
    }
    base.Dispose( disposing );
}
//下面系统自动产生的代码已经删除
#region Component Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
private void button1_Click(object sender, System.EventArgs e)
{
    client=new TcpClient(textBox1.Text,
        Int32.Parse(textBox2.Text));
    Thread thread1=new Thread(new ThreadStart(receive));
    thread1.Start();
}
private void button2_Click(object sender, System.EventArgs e)
{
    NetworkStream netStream=client.GetStream();
    byte[] by=new Byte[1024];
    by=System.Text.Encoding.BigEndianUnicode.GetBytes
        (richTextBox2.Text+"\r\n");
    netStream.Write(by,0,by.Length);
}
private void receive()
{
    while(!round)
    {
        NetworkStream netStream=client.GetStream();
        byte[] messsageByte=new Byte[1024];
        netStream.Read(messsageByte,0,messsageByte.Length);
        string readNessage=
            System.Text.Encoding.BigEndianUnicode.GetString
            (messsageByte);
        richTextBox1.AppendText(readNessage+"\r\n");
    }
}
private void button3_Click(object sender, System.EventArgs e)
{
    round=true;
    client.Close();
}
}
```

8.3.2.2 TcpClient 客户端控件的使用

(1) 新建一个 Windows 项目。

新建一个 Windows 项目，本例名称为 useTCPClient。

(2) 添加引用。

单击菜单“项目”→“添加引用”，将 TCPClientControl 加入进去。然后添加代码：

```
using TCPClientControl;
```

(3) 添加私有成员。

```
private TCPClientControl.UserControl1 mycontrol;
```

(4) 添加如下构造方法。

```
private void InitializeComponent ()  
{  
    this.components=new System.ComponentModel.Container();  
    this.mycontrol=new TCPClientControl.UserControl1();  
    mycontrol.Size=new System.Drawing.Size(416, 336);  
    this.Controls.Add(this.mycontrol);  
}
```

(5) 进入窗体设计窗口设计窗体。

进入窗体设计窗口设计，修改控件 mycontrol 的大小和颜色等属性。

(6) 删除步骤(4)添加的方法或修改该方法的名称。比如修改成以下方法：

```
private void eee()  
{  
    this.components=new System.ComponentModel.Container();  
    this.mycontrol=new TCPClientControl.UserControl1();  
    mycontrol.Size=new System.Drawing.Size(416, 336);  
    this.Controls.Add(this.mycontrol);  
}
```

(7) 编译并执行程序。

单击菜单“调试”→“开始执行”，系统自动编译并执行程序。

下面是该程序的代码：

```
using System;  
using System.Drawing;  
using System.Collections;  
using System.ComponentModel;  
using System.Windows.Forms;  
using System.Data;  
using TCPClientControl;  
using System.Net;  
using System.Net.Sockets;
```

```
namespace useTCPClient
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        private TCPClientControl.UserControl1 mycontrol;
        public Form1()
        {
            //
            // Windows 窗体设计器支持所必需的
            //
            InitializeComponent();
            //
            // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
            //
        }
        /// <summary>
        /// 清理所有正在使用的资源
        /// </summary>
        ///
        private void eee()
        {
            this.components=new System.ComponentModel.Container();
            this.mycontrol=new TCPClientControl.UserControl1();
            mycontrol.Size=new System.Drawing.Size(416, 336);
            this.Controls.Add(this.mycontrol);
        }
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
                if (components != null)
                {
                    components.Dispose();
                }
            }
            base.Dispose( disposing );
        }
    }
}
```

```
#region Windows Form Designer generated code
/// <summary>
/// 设计器支持所需的方法 - 不要使用代码编辑器修改
/// 此方法的内容。
/// </summary>
private void InitializeComponent()
{
    this.mycontrol = new TCPClientControl.UserControl1();
    this.SuspendLayout();
    //
    // mycontrol
    //
    this.mycontrol.controlRound = false;
    this.mycontrol.myClient = null;
    this.mycontrol.Name = "mycontrol";
    this.mycontrol.Size = new System.Drawing.Size(416, 336);
    this.mycontrol.TabIndex = 0;
    //
    // Form1
    //
    this.AutoScaleBaseSize = new System.Drawing.Size(6, 14);
    this.ClientSize = new System.Drawing.Size(432, 341);
    this.Controls.AddRange(new System.Windows.Forms.Control[]
        {this.mycontrol});
    this.Name = "Form1";
    this.ResumeLayout(false);
}
#endregion
/// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}
}
```

8.3.2.3 演示

启动一个服务器程序，然后监听端口。运行 8.3.2.2 节的程序。图 8-19 是 8.3.2.2 节程序的运行情况。

图 8-20 是服务器的运行情况。

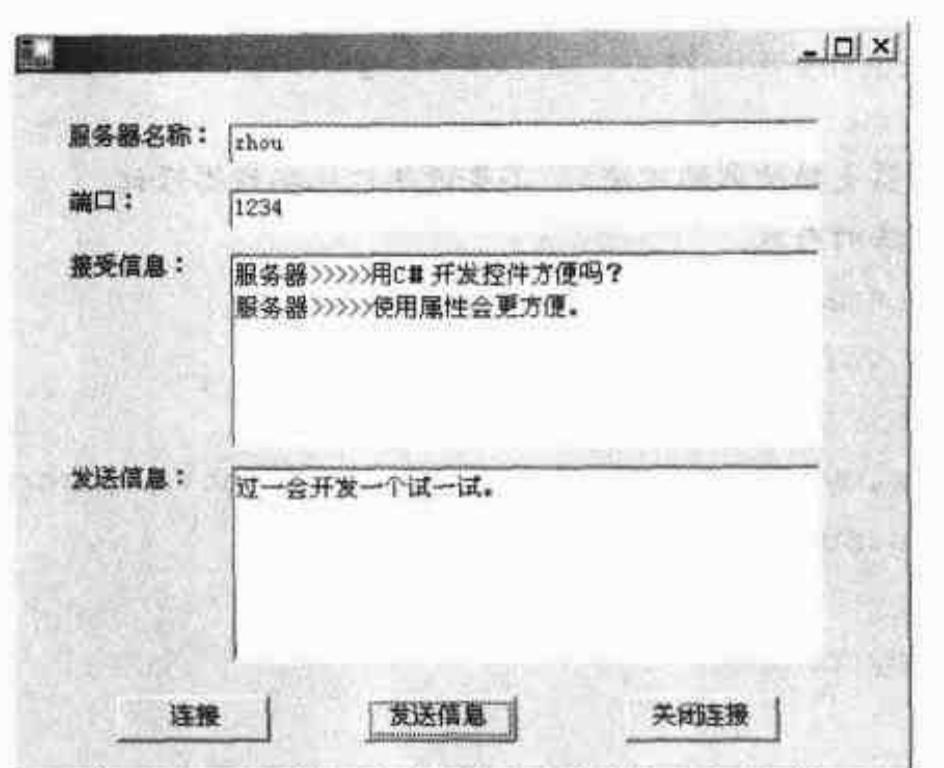


图 8-19 TcpClient 客户端控件使用程序的运行结果

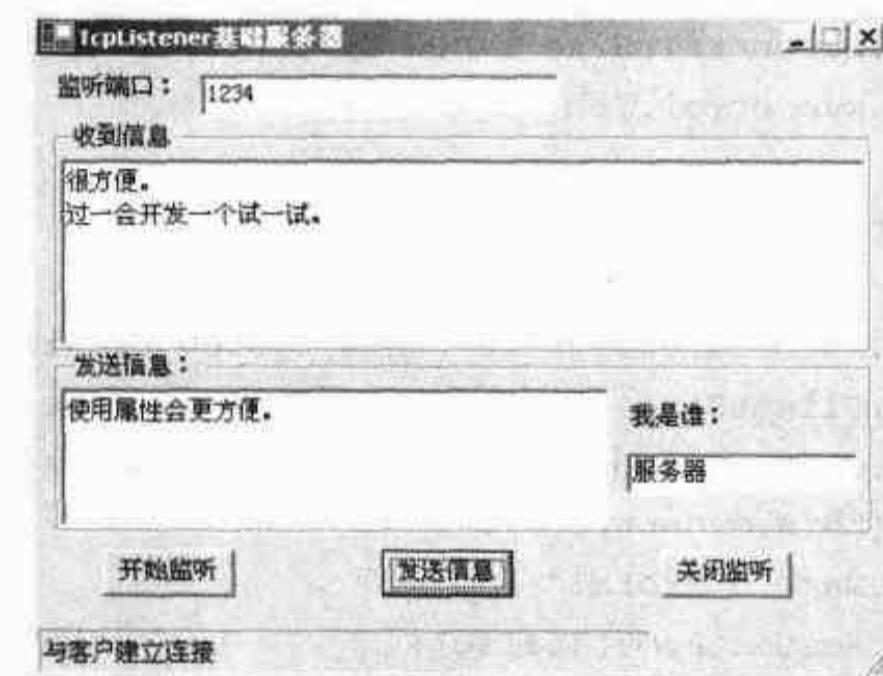


图 8-20 TcpListener 服务器的运行情况

8.4 关于属性

用微软的话说，属性就像智能字段，它通常具有带着访问函数的专用数据成员，在语法上属性作为类的字段被访问。组件应该定义属性而不是公共字段，因为可视化设计器（例如 Visual Studio .NET）在属性浏览器中显示属性，而不显示字段。属性定义通常由以下两部分组成：

1. 专用数据成员的定义

例如：

```
private TcpClient client;
```

2. 使用属性声明语法对公共属性进行的定义

该语法通过 get 和 set 访问函数将专用数据成员和公共属性关联起来。

例如：

```
public TcpClient myClient{  
    get{  
        return client;  
    }  
    set{  
        client=value;  
    }  
}
```

value 这个术语是属性定义语法中的一个关键字。在呼叫代码中，将变量 value 分配给属性。value 的类型必须同分配给的属性的声明类型相同。虽然属性定义中通常包含专用数据成员，但这不是必需的。get 访问器可以不访问私有数据成员就返回一个值。一般说，get 和 set 方法与其他方法没有什么区别。它们可以执行任何程序逻辑、引发异常、被重写以及用编程语言允许的任意修饰符进行声明。但是请注意，属性也可以是静态的。如果属性是静态的，则在 get 和 set 方法可以实现的功能上有一些局限性。在不同的编程语言中，属性语法存在一些差异。例如，术语“属性”不是 C# 中的关键字，但却是 VB.NET 中的关键字。

属性的类型可以是基元类型、基元类型的集合、用户定义类型或用户定义类型的集合。对于所有基元类型，.NET 框架提供实现字符串到值转换的类型转换器。当属性可以使用类型转换器时，在设计器的属性浏览器中会显示属性。如果想自定义属性并让属性浏览器显示它们，则必须实现自定义类型转换器。当属性的数据类型是枚举时，开发环境（例如 VS.NET）在“属性”窗口中将该属性显示为下拉列表。如果属性的数据类型是具有属性的类，这些属性叫作定义属性的子属性。在 VS.NET 的“属性”窗口中，用户可以展开属性以显示其子属性。需要注意的是，向属性添加特性十分重要，这样它们可以在设计时正确地显示在属性浏览器中。组件和控件应该公开属性而不是公共字段，因为对属性可以指定版本，它们允许数据隐藏，并且访问器方法可以执行附加逻辑，而且，由于实时优化，属性不比字段更耗费资源。

8.4.1 在组件中使用属性

8.4.1.1 开发带有属性的组件

下面在开发一个 TcpListener 组件，该组件可以实现监听功能并接收数据。在该组件中使用一个 TcpListener 类型的 myListener 属性，该属性可以获取和设置私有字段 listener（TcpListener 类型）的值。

(1) 新建项目。

新建一个类库，本例项目名称为 TCPListen。

(2) 添加私有字段。

```
private TcpListener listener;
```

(3) 开发属性。

```
public TcpListener myListener  
{
```

```
get{
    return listener;
}
set{
    listener=value;
}
```

下面是该组件类的全部代码：

```
using System;
using System.Net;
using System.Net.Sockets;

namespace TCPListen
{
    /// <summary>
    /// Class1 的摘要说明
    /// </summary>
    public class Class1
    {
        private TcpListener listener;
        public Class1()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }

        public TcpListener myListener{
            get{
                return listener;
            }
            set{
                listener=value;
            }
        }

        public Socket accept(){
            Socket mySock=listener.AcceptSocket();
            return mySock;
        }

        public string receive(Socket mySock){
            if(mySock.Connected)
            {
                NetworkStream netStream=new NetworkStream(mySock);
                byte[] messageByte=new Byte[64];
                netStream.Read(messageByte,0,messageByte.Length);
                string readMessage=
                    System.Text.Encoding.BigEndianUnicode.GetString

```

```
        (messageByte);
    return readMessage;
}
else{return "连接未建立!";
}
public void closeListen(){
    listener.Stop();
}
}
}
```

8.4.1.2 使用组件

(1) 新建项目。

新建一个 Windows 应用程序项目（本例项目名称为 useListenClass），图 8-21 是该程序的界面。

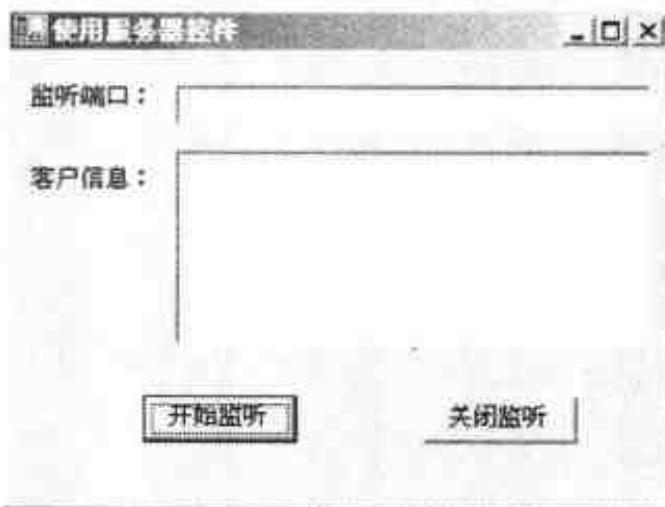


图 8-21 带属性服务器组件的使用程序的界面

其中“监听端口”文本框是 textBox1，“客户信息”文本框是 textBox2，“开始监听”按钮是 button1，“关闭监听”按钮是 button2。

(2) 添加引用。

单击菜单“项目”→“添加引用”，将 TCPListen.dll 加入进来，然后添加如下代码：

```
using System.Net;
using System.Net.Sockets;
using TCPListen;
using System.Threading;
```

(3) 构造组件 TCPListen 类的成员。

在构造方法 InitializeComponent(); 下添加如下方法：

```
myClass=new TCPListen.Class1();
```

(4) 构造私有成员 myClass。

```
myClass=new TCPListen.Class1();
```

(5) 为 myClass 的属性 myListner 赋值。

```
private void button1_Click(object sender, System.EventArgs e)
```

```
{  
    int port=0;  
    port=Int32.Parse(textBox1.Text);  
    TcpListener newListener=new TcpListener(port);  
    newListener.Start();  
    //下行为 myClass 的属性 myListener 赋值  
    myClass.myListener=newListener;  
    Thread thread=new Thread(new ThreadStart(receive));  
    thread.Start();
```

下面是该程序的代码：

```
using System;  
using System.Drawing;  
using System.Collections;  
using System.ComponentModel;  
using System.Windows.Forms;  
using System.Data;  
using System.Net;  
using System.Net.Sockets;  
using TCPListen;  
using System.Threading;  
  
namespace useListenClass  
{  
    /// <summary>  
    /// Form1 的摘要说明  
    /// </summary>  
    public class Form1 : System.Windows.Forms.Form  
    {  
        private System.Windows.Forms.Button button1;  
        private System.Windows.Forms.Button button2;  
        private System.Windows.Forms.Label label1;  
        private System.Windows.Forms.TextBox textBox1;  
        private System.Windows.Forms.RichTextBox richTextBox1;  
        private System.Windows.Forms.Label label2;  
        /// <summary>  
        /// 必需的设计器变量  
        /// </summary>  
        private System.ComponentModel.Container components = null;  
        private TCPListen.Class1 myClass;  
        private bool round=false;  
        public Form1()  
        {  
            //  
            // Windows 窗体设计器支持所必需的  
            //  
            InitializeComponent();
```

```
myClass=new TCPListen.Class1();
//
// TODO: 在 InitializeComponent 调用后添加任何构造函数代码
//
}

/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}
//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    int port=0;
    port=Int32.Parse(textBox1.Text);
    TcpListener newListener=new TcpListener(port);
    newListener.Start();
    myClass.myListener=newListener;
    Thread thread=new Thread(new ThreadStart(receive));
    thread.Start();
}

private void receive(){
    Socket sock=myClass.accept();
    while(!round)
    {
```

```

        string clientMessage=myClass.receive(sock);
        richTextBox1.AppendText(clientMessage+"\r\n");
    }
}

private void button2_Click(object sender, System.EventArgs e)
{
    round=true;
    myClass.closeListen();
}
}
}

```

8.4.1.3 演示

启动 8.4.1.2 节的程序，并监听端口。然后启动一个客户端程序。图 8-22 是 8.4.1.2 节的程序的运行情况。

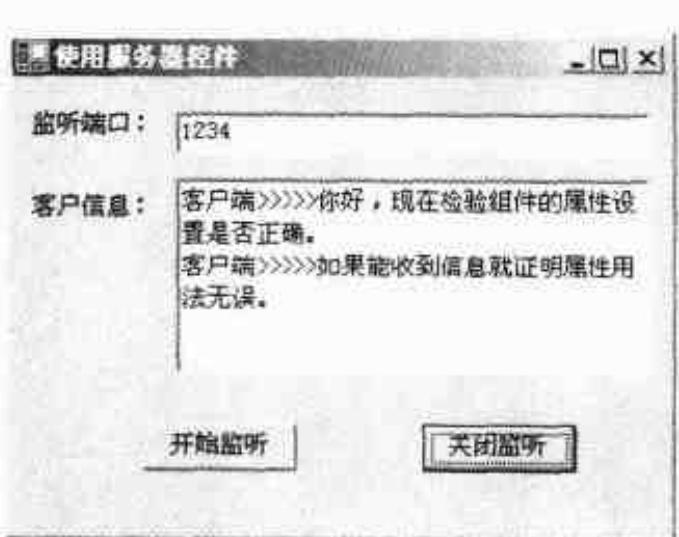


图 8-22 带属性服务器组件的使用程序的运行情况

图 8-23 是客户端的运行情况。

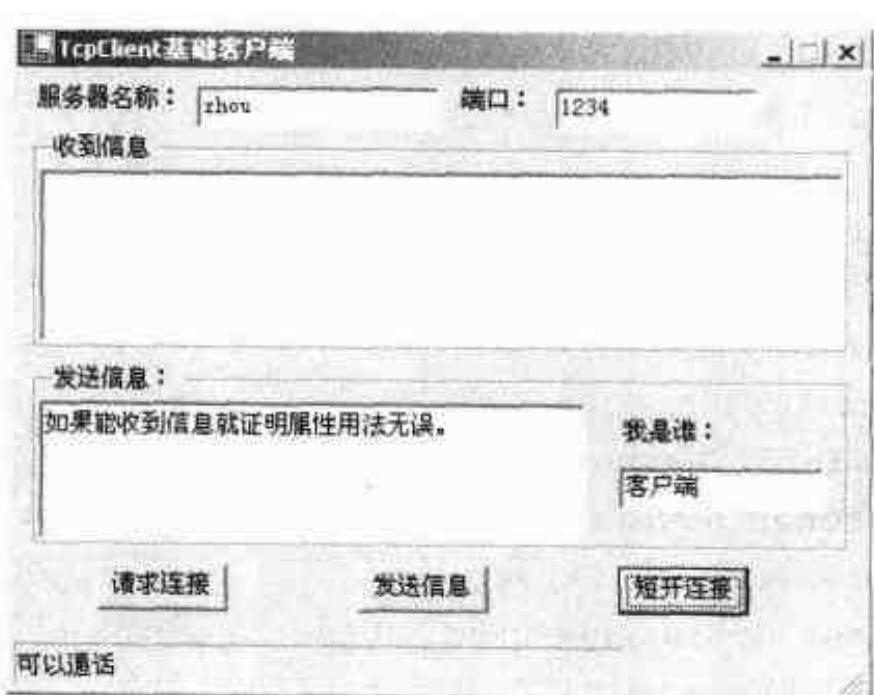


图 8-23 客户端运行情况

8.4.2 在控件中使用属性

在控件中使用属性与在普通组件中使用属性没有区别。下面演示如何在 TcpClient 客户端控件中使用属性。

8.4.2.1 控件开发

新建一个 Windows 控件库，本例的项目名称为 TCPClientControl。该控件界面如图 8-24 所示，其中“接受信息”文本框是 richTextBox1，“发送信息”文本框是 richTextBox2，“发送信息”按钮是 button1，“关闭连接”按钮是 button2。

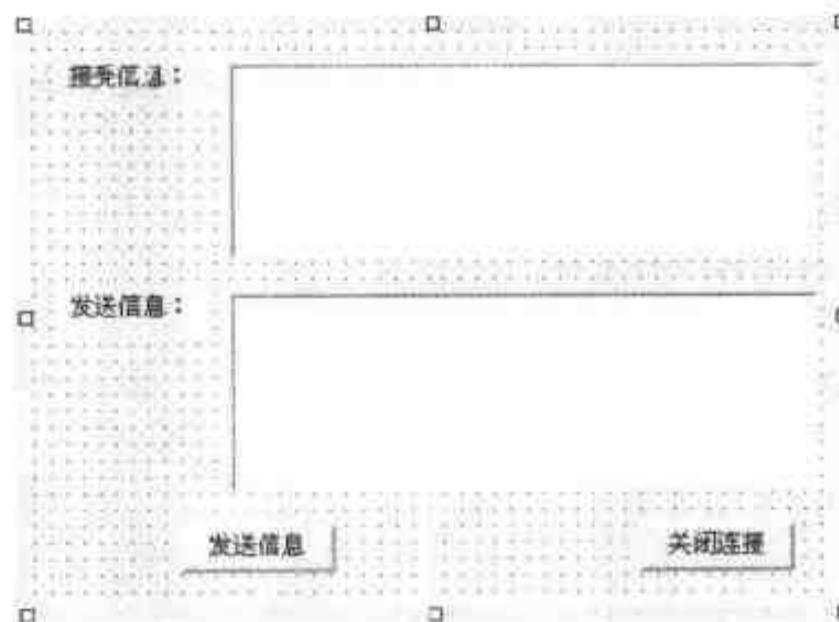


图 8-24 带属性控件的界面

该控件的代码如下：

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Drawing;
using System.Data;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace TCPClientControl
{
    /// <summary>
    /// UserControl1 的摘要说明
    /// </summary>
    public class UserControl1 : System.Windows.Forms.UserControl
    {
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        private System.Windows.Forms.RichTextBox richTextBox1;
        private TcpClient client;
        private System.Windows.Forms.RichTextBox richTextBox2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
```

```
private bool round=false;
public UserControl1()
{
    // 该调用是 Windows.Forms 窗体设计器所必需的
    InitializeComponent();

    // TODO: 在 InitializeComponent 调用后添加任何初始化
}

/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if( components != null )
            components.Dispose();
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
public void recMessage()
{
    //client=new TcpClient(textBox1.Text,
    //    Int32.Parse(textBox2.Text));
    Thread thread=new Thread(new ThreadStart(receive));
    thread.Start();
}

public TcpClient myClient{
    get{
        return client;
    }
    set{client=value;}
}

public bool controlRound
{
    get
    {
        return round;
    }
    set{round=value;}
}
private void button1_Click(object sender, System.EventArgs e)
```

```
{  
    NetworkStream netStream=client.GetStream();  
    byte[] by=new Byte[1024];  
    by=System.Text.Encoding.BigEndianUnicode.GetBytes  
        (richTextBox2.Text+"\r\n");  
    netStream.Write(by,0,by.Length);  
}  
private void receive()  
{  
    while(!round)  
    {  
        NetworkStream netStream=client.GetStream();  
  
        byte[] messsageByte=new Byte[1024];  
        netStream.Read(messsageByte,0,messsageByte.Length);  
        string readNessage=  
            System.Text.Encoding.BigEndianUnicode.GetString  
                (messsageByte);  
        richTextBox1.AppendText(readNessage+"\r\n");  
    }  
}  
private void button2_Click(object sender, System.EventArgs e)  
{  
    round=true;  
    client.Close();  
}  
}  
}
```

8.4.2.2 使用控件

新建一个 Windows 程序项目，本例项目名称为 useTCPClient。先把 8.4.2.1 的控件加到窗体上，然后再往窗体上拖放一个 button 按钮(button1)。该程序的界面如图 8-25 所示。

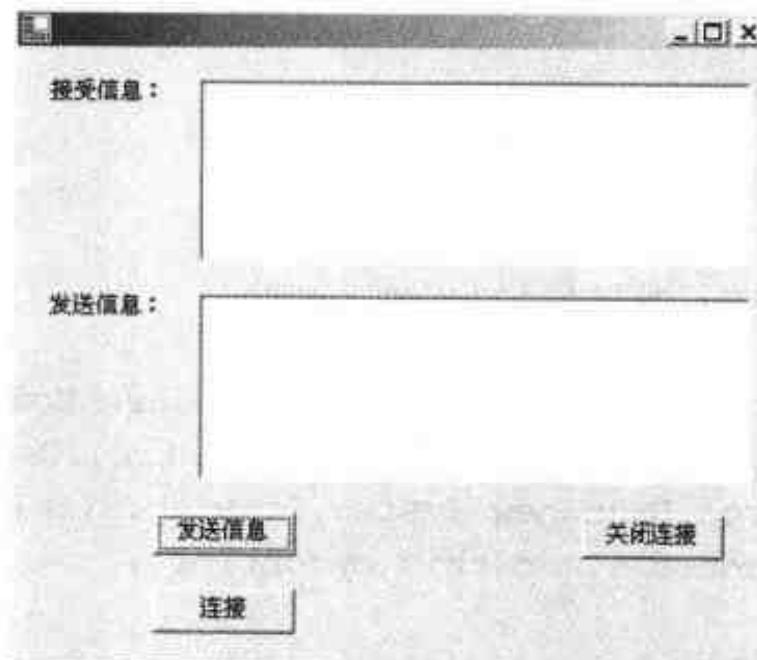


图 8-25 使用带属性控件程序的界面

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using TCPClientControl;
using System.Net;
using System.Net.Sockets;

namespace useTCPClient
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        private System.Windows.Forms.Button button1;
        private TCPClientControl.UserControl1 mycontrol;
        public Form1()
        {
            //
            // Windows 窗体设计器支持所必需的
            //
            InitializeComponent();
            //
            // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
            //
        }
        /// <summary>
        /// 清理所有正在使用的资源
        /// </summary>
        ///
        private void InitializeComponent()
        {
            this.components=new System.ComponentModel.Container();
            this.mycontrol=new TCPClientControl.UserControl1();
            mycontrol.Size=new System.Drawing.Size(416, 336);
            this.Controls.Add(this.mycontrol);
        }
        protected override void Dispose( bool disposing )
        {
            if( disposing )
            {
```

```
if (components != null)
{
    components.Dispose();
}

base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    TcpClient client=new TcpClient("zhou",1234);
    mycontrol.myClient=client;
    mycontrol.recMessage();
}

}
}
}
```

8.4.2.3 演示

启动一个服务器，然后开始监听端口 1234。启动本例的程序。图 8-26 是本例程序的运行情况。

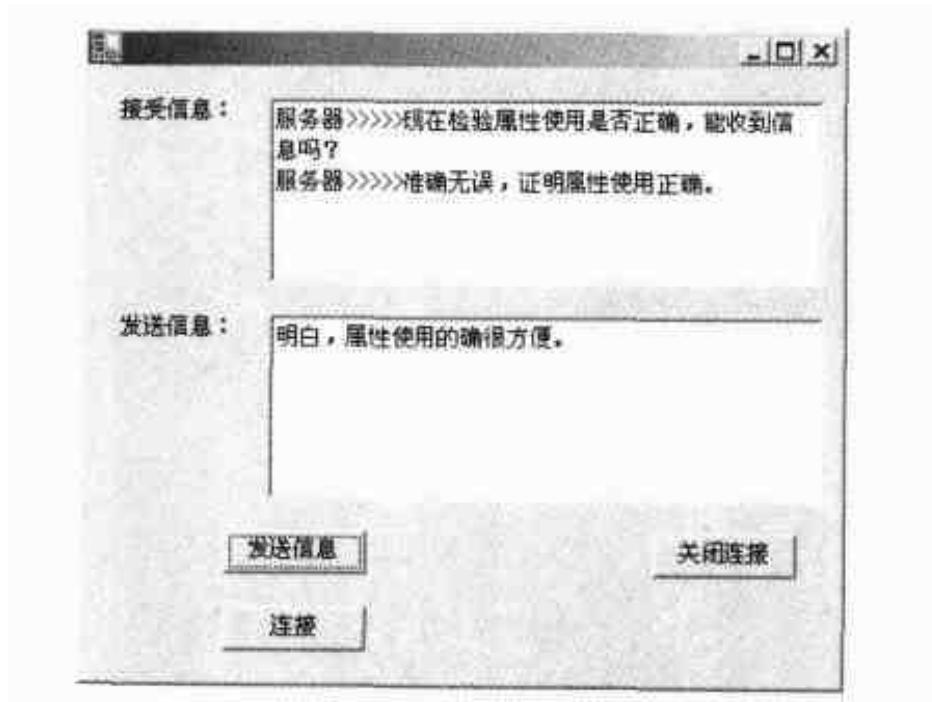


图 8-26 使用带属性控件程序的运行程序

图 8-27 是服务器的运行情况。

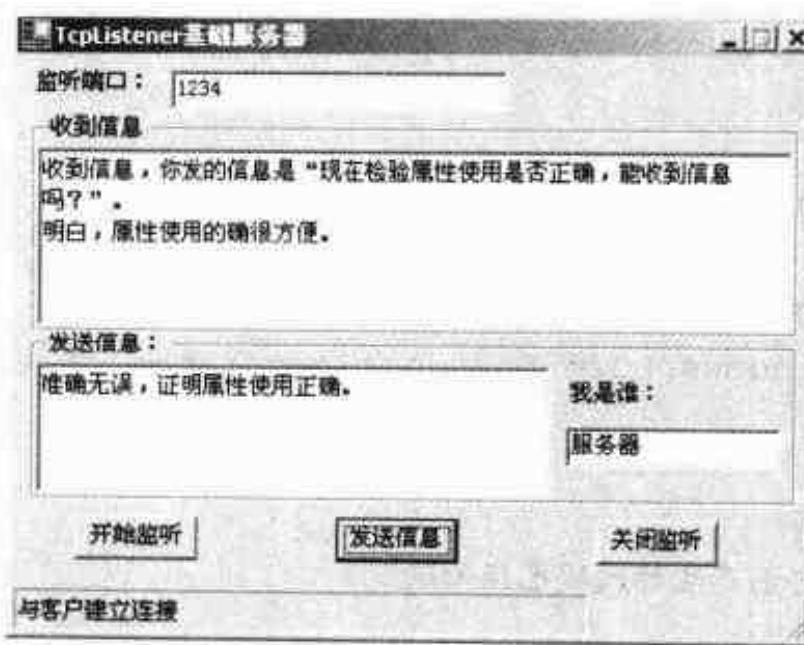


图 8-27 服务器运行情况

本章小结

本章详细讲解了 Win32 网络组件与控件的开发。通过本章的学习，不但可以掌握组件化编程技术，而且有助于迅速构件企业级网络应用程序。

第9章 ADO.NET Web 应用开发

数据库在网络上的应用主要有两种，一种是数据库的 Web 应用，另一种是数据库的 Win32 网络应用。数据库的 Win32 网络应用在下一章讲，本章主要讲解数据库的 Web 应用程序，至于数据库的 Web 服务，将在本书第 11 章介绍。本章除了讲解数据库的 Web 应用程序外，还涉及数据库的一些基础知识。

9.1 数据库建立

9.1.1 用 VS.NET 创建数据库

VS.NET 7.0 本身已经内置了与 SQL Server 连接的引擎，在 VS.NET 7.0 环境里，可以方便地建立数据库。下面用 VS.NET 7.0 建立一个 SQL 数据库。需要注意的是，在建立数据库之前，要确保 VS.NET 已经与 SQL Server 建立了连接。

(1) 打开“服务器资源管理器”。

如图 9-1 所示，打开“服务器资源管理器”，并选中“数据连接”选项。

(2) 如图 9-2 所示，右击“数据连接”选项，在弹出的菜单上单击菜单“创建新 SQL Server 数据库”。



图 9-1 服务器资源管理器



图 9-2 打开快捷菜单

(3) 如图 9-3 所示，在“服务器”文本框里输入 SQL Server 服务器名称，在“新数据库名”里输入新建数据库的名称（本例是“图书馆”）。

注意：要选择正确的身份认证机制（即特定的 SQL Server 所要求的身份认证机制），不然在建立或使用数据库时可能出错。有些 SQL Server 禁止使用 Windows NT 认证机制，这时就要使用 SQL Server 的身份认证机制。

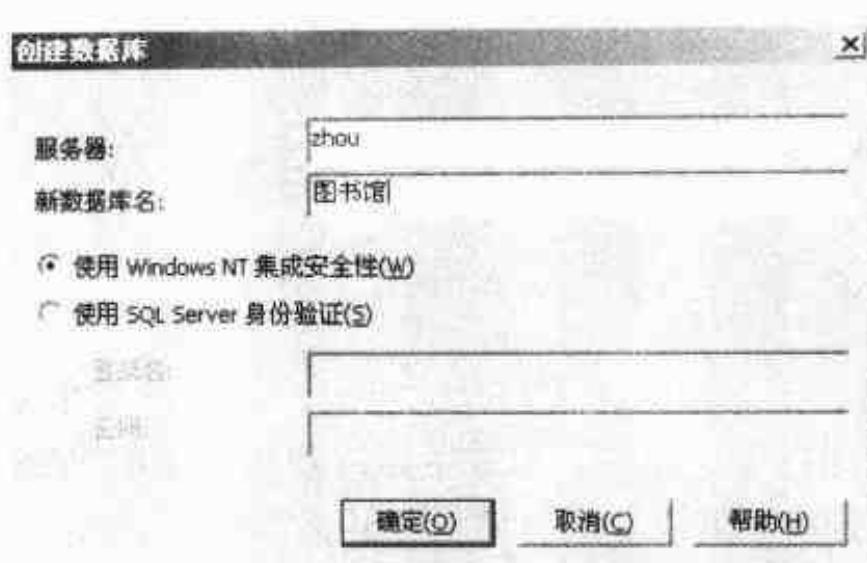


图 9-3 输入服务器的数据库名称

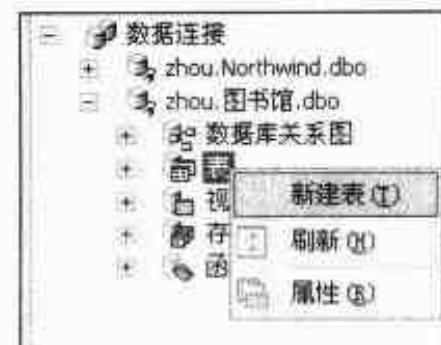


图 9-4 新建表快捷菜单

(4) 如图 9-4 所示，在新建的数据库“图书馆”的“表”选项上，单击鼠标右键，在快捷菜单上单击菜单“新建表”。

(5) 设计表。

如图 9-5 所示，设计列名、数据类型和长度等，设计完后，保存表并关闭表的设计窗口（关闭表时系统自动提示是否保存表）。

列名	数据类型	长度	允许空
ISBN	char	30	
书名	char	100	✓
作者	char	30	✓
出版社	char	50	✓
责任者	char	30	✓
出版日期	char	30	✓
定价	float	8	✓

图 9-5 设计表

(6) 填充原始数据。

如图 9-6 所示，在“表 1”上单击鼠标右键，然后单击快捷菜单“从表中检索数据”，然后在出现图 9-7 的表格里填充数据即可。

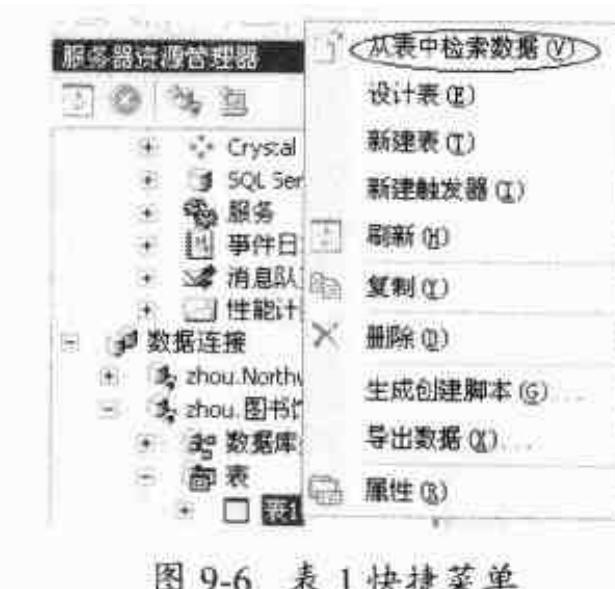


图 9-6 表 1 快捷菜单

MS SQL Server - db.dbo.表1: 图 (zhou.图书馆)					
ISBN	书名	作者	出版社	责任者	出版日期
7-00000-000-1	名师讲读法硕	王新东	新江北出版社	江明	2000
7-00000-000-2	中国古代名著欣赏	张平	浙江画出版社	李小丽	2002
7-00000-000-3	战国风云	张小方	新江东出版社	赵进东	2002

图 9-7 原始数据

到此为止，一个 SQL Server 数据库建立成功。除了 SQL Server 数据库外，VS.NET 还可以支持其他多种数据库。有兴趣的读者，可以用 Access 建立数据库，以供日后编程使用。

用 VS.NET 7.0 新建 SQL Server 数据库，可按如下方法进行，如图 9-8 所示，在“服务器 \ 服务器名称 (zhou) \ SQL Server \”下，任意选中一个数据库实例，然后单击鼠标右键，然后单击快捷菜单“新建数据库”。其他步骤和上面介绍的相同。

9.1.2 用代码创建数据库

本处的“数据库”实际上是指数据库的数据表，不同于数据集中的表格。数据库的建立要用 SQL 的“CREATE”语句。该语句的语法如下：

```
CREATE TABLE 模式名.约束名 [表约束] (
    列名 数据类型 [列约束]
    列名 数据类型 [列约束]
    列名 数据类型 [列约束]
    ...
)
[ PCTFREE int 值 PCTUSED int 值 ]
[ TABLESPACE 表空间名 ]
[ STORAGE (存储子句) ]
[ CLUSTER 聚集名 (列名 [, 列名, ...] ) ]
[ ENABLE 约束名 [, 约束名, ...] ]
[ DISABLE 约束名 [, 约束名, ...] ]
[ AS 子查询 ]
```

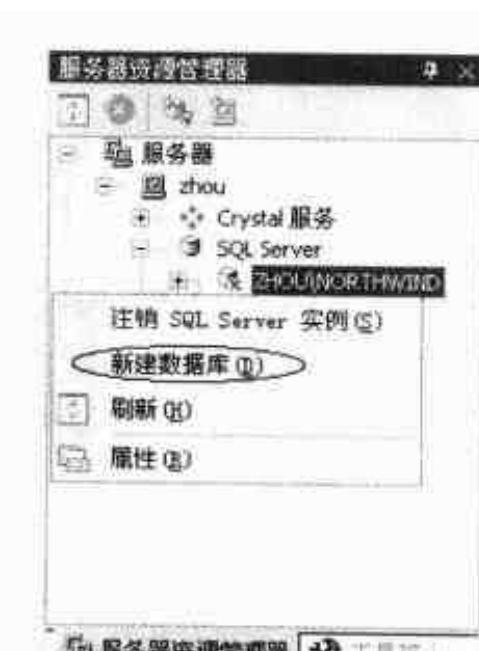


图 9-8 新建数据库快捷菜单

如果要在 Northwind 上建立一个包括“自动编号”、“姓名”、“住址”、“年龄”、“电话”四列，数据类型分别为整型、字符、字符、整型、字符，长度分别为默认、20、50、默认、20，“自动编号”为主键的数据库数据表，可用如下代码实现：

```
private void button1_Click(object sender, System.EventArgs e)
{
    //建立连接
    SqlConnection myConnection = new SqlConnection();
    myConnection.ConnectionString =
        " data source=localhost;
        initial catalog=Northwind;
        persist security info=False;
        user id=sa;
        workstation id=ZHOUE;packet size=4096 ";
    //建立数据库数据表
```

```

SqlCommand createDatatable=
    new SqlCommand("CREATE TABLE 表 2
    (自动编号 int IDENTITY PRIMARY KEY,姓名 char(20),住址 char(50),
    年龄 int,电话 char(20))",myConnection);
myConnection.Open();
createDatatable.ExecuteNonQuery();
myConnection.Close();
}

```

需要注意的是，自动编号属性的设置可以由“IDENTITY”来完成，一个表只能有一列定义为 IDENTITY 属性，而且该列必须以 decimal、int、numeric、smallint、bigint 或 tinyint 数据类型定义。主键的设置可由“PRIMARY KEY”来完成，“PRIMARY”与“KEY”之间有空格。

9.2 数据库连接

本节演示如何与 SQL Server 的数据库、Access 的数据库建立连接。

9.2.1 与 SQL Server 数据库连接

9.2.1.1 用拖放的办法连接数据库

- (1) 如图 9-9 所示，建立一个 ASP.NET Web 应用程序（本例项目名称为 `dataLink`）。

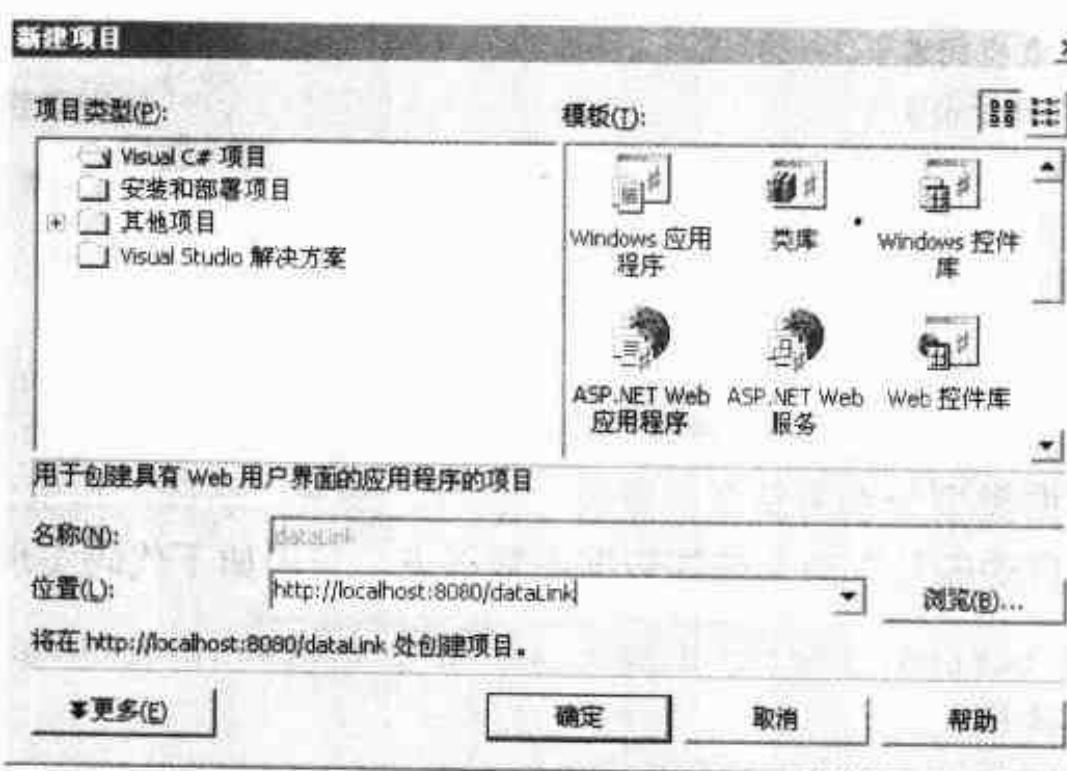


图 9-9 新建 Web 应用程序

- (2) 如图 9-10 所示，在“解决方案资源管理器”，将文件名 `WebForm1.aspx` 修改成 `fistDataProgram.aspx`。

(3) 如图 9-11 所示，将数据库——图书馆的“表 1”拖放到页面上（网页）即可。

- (4) 完成步骤 (3) 后，如图 9-12 所示，页面下面自动添加了 `sqlDataAdapter1` 和 `sqlConnection1`，同时在代码编辑窗体还添加了如下对象：

```
protected System.Data.SqlClient.SqlCommand sqlSelectCommand1;
protected System.Data.SqlClient.SqlCommand sqlInsertCommand1;
protected System.Data.SqlClient.SqlCommand sqlUpdateCommand1;
protected System.Data.SqlClient.SqlCommand sqlDeleteCommand1;
```

开发人员可以根据需要把上述“protected”修改为“public”或“private”。



图 9-10 修改页名称



图 9-11 拖放表 1

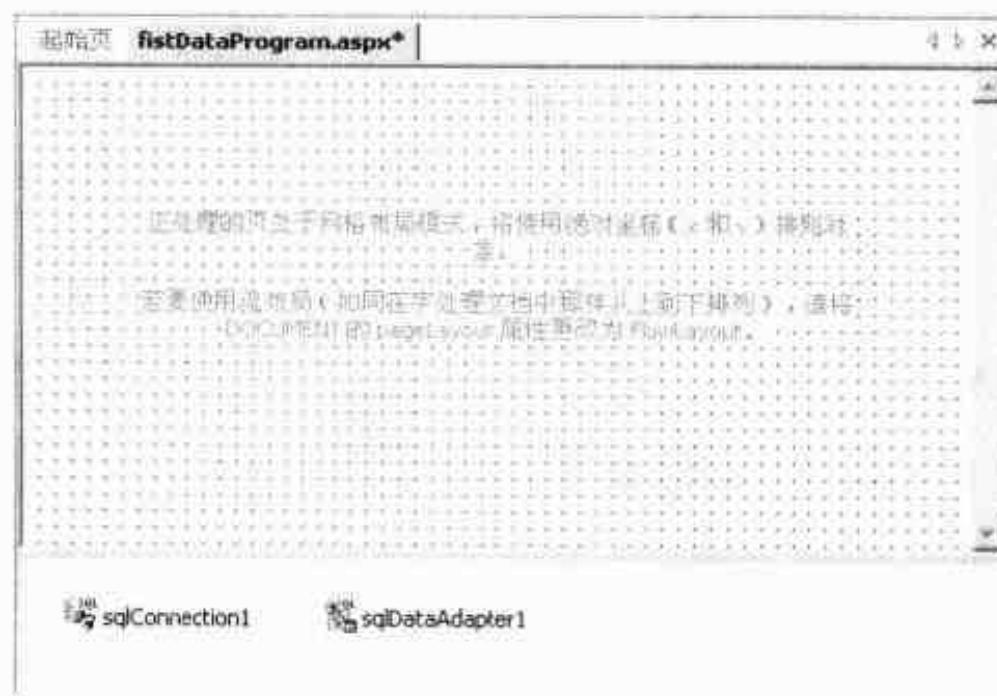


图 9-12 连接完成

(5) 建立数据集。

如图 9-13 所示,在 sqlDataAdapter1 上单击鼠标右键,然后单击快捷菜单“生成数据集”。



图 9-13 生成数据集快捷菜单

(6) 完成步骤(5)后，出现如图 9-14 所示的窗口，在新建文本框里输入适当的数据集名称（本例为 books），然后单击“确定”即可。注意选中窗体上的“将此数据集添加到设计器”复选框。

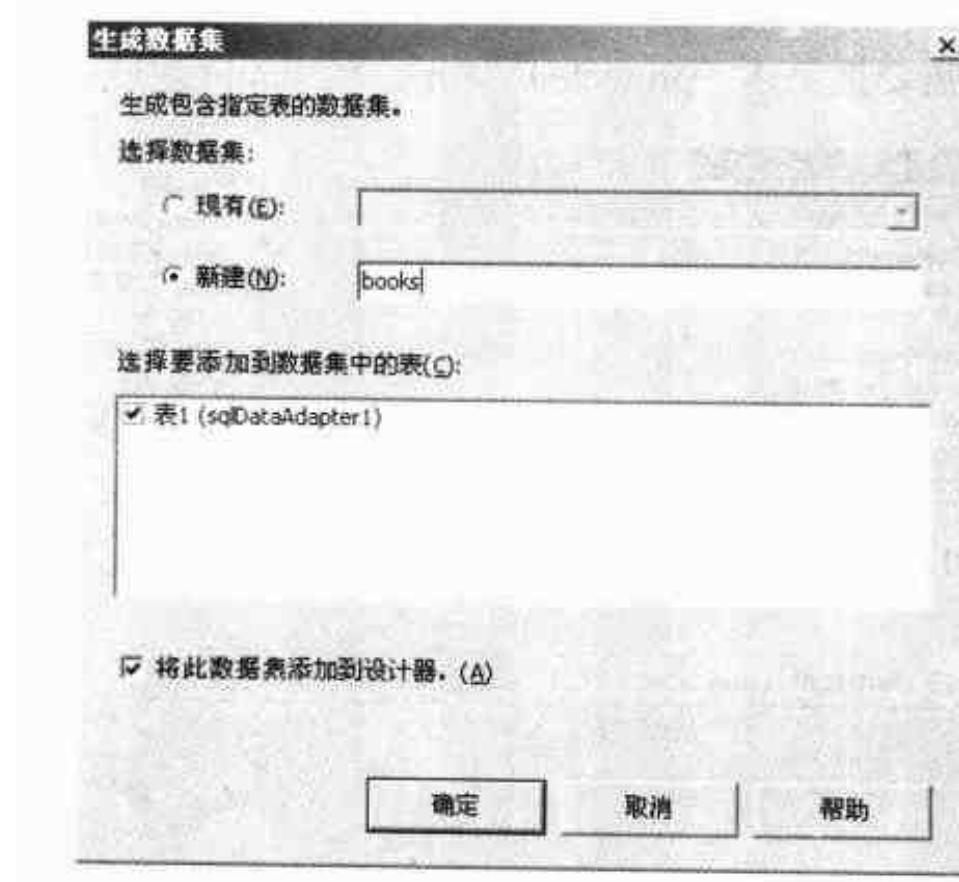


图 9-14 输入数据集名称

完成步骤(6)后，如图 9-15 所示，页面的下面自动添加了数据集对象 books1。

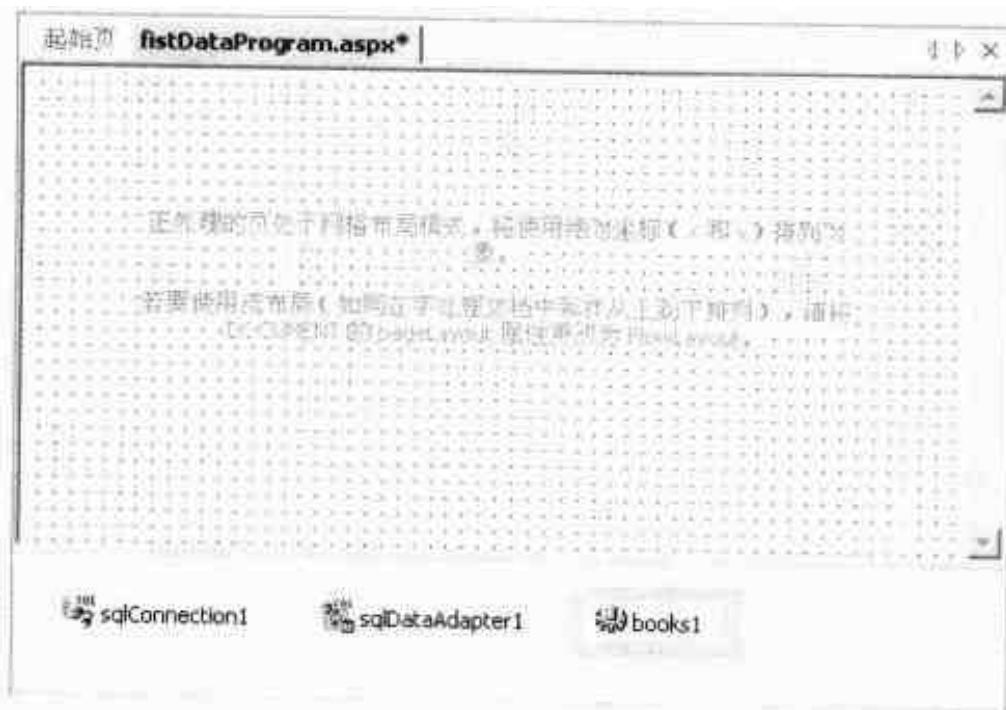


图 9-15 数据集建立完成

9.2.1.2 用数据适配器连接数据库

(1) 将工具箱的 SqlDataAdapter 拖放到页面上，即可出现如图 9-16 所示的窗口，单击图中的“下一步”。

(2) 如图 9-17 所示，单击“新建连接”。

(3) 在图 9-18 的窗口上，输入正确的 SQL 服务器名称、用户名、密码等，确保连接无误后，单击“确定”。

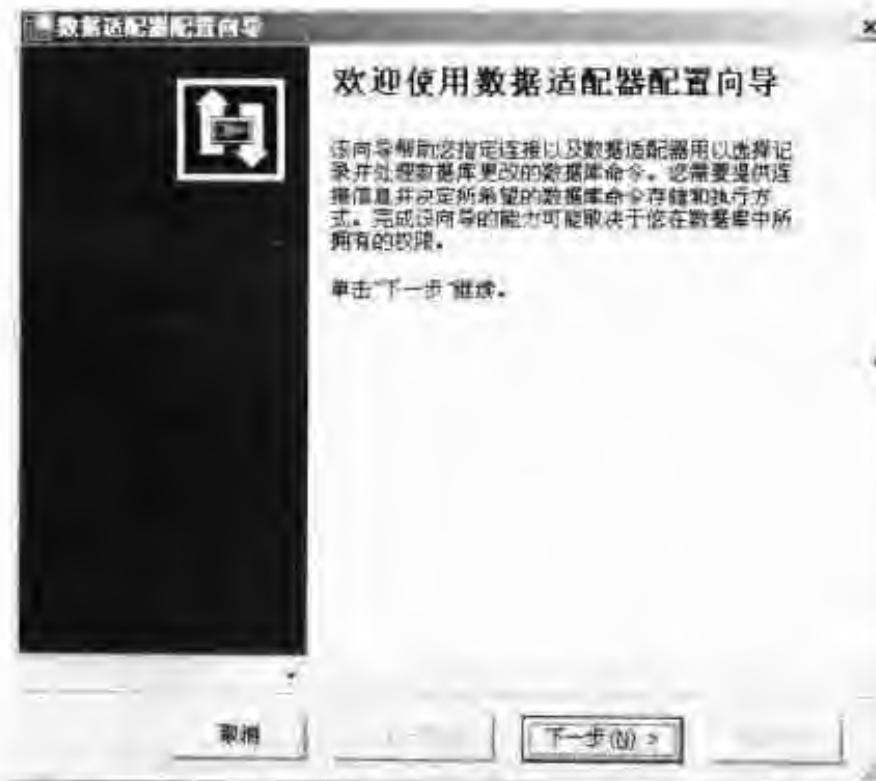


图 9-16 向导第一步

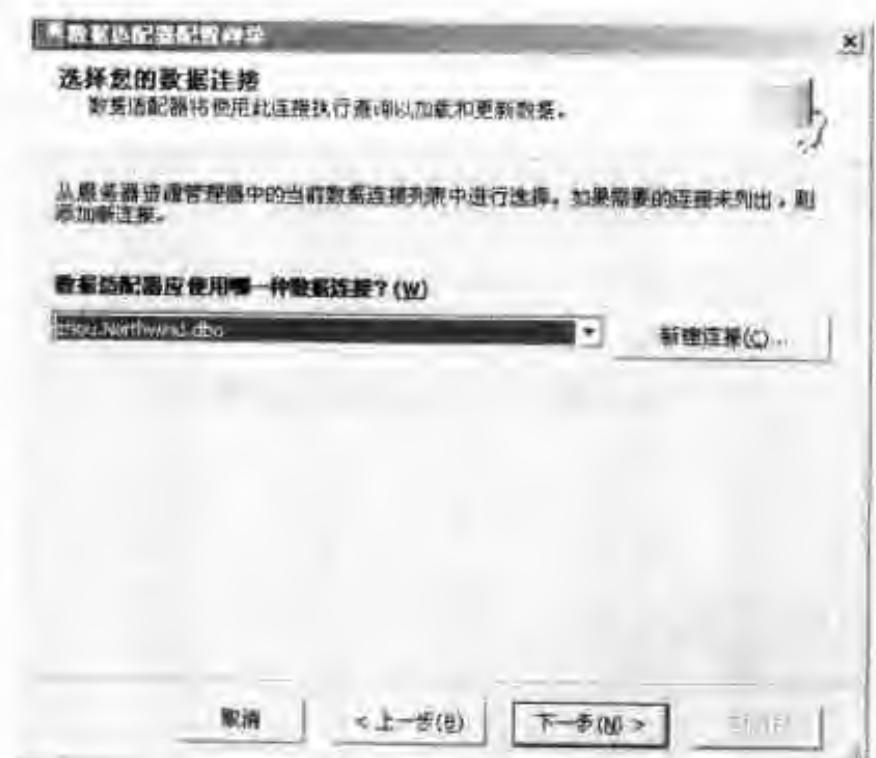


图 9-17 向导第二步

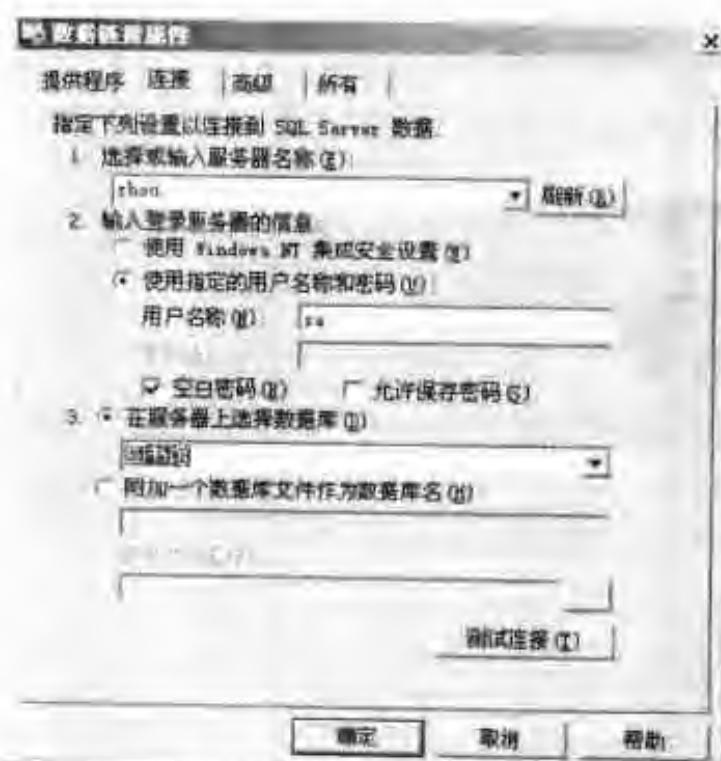


图 9-18 输入连接信息

(4) 按照向导一步步地单击“下一步”，直到出现如图 9-19 所示的窗口。在图中单击“查询生成器”按钮。

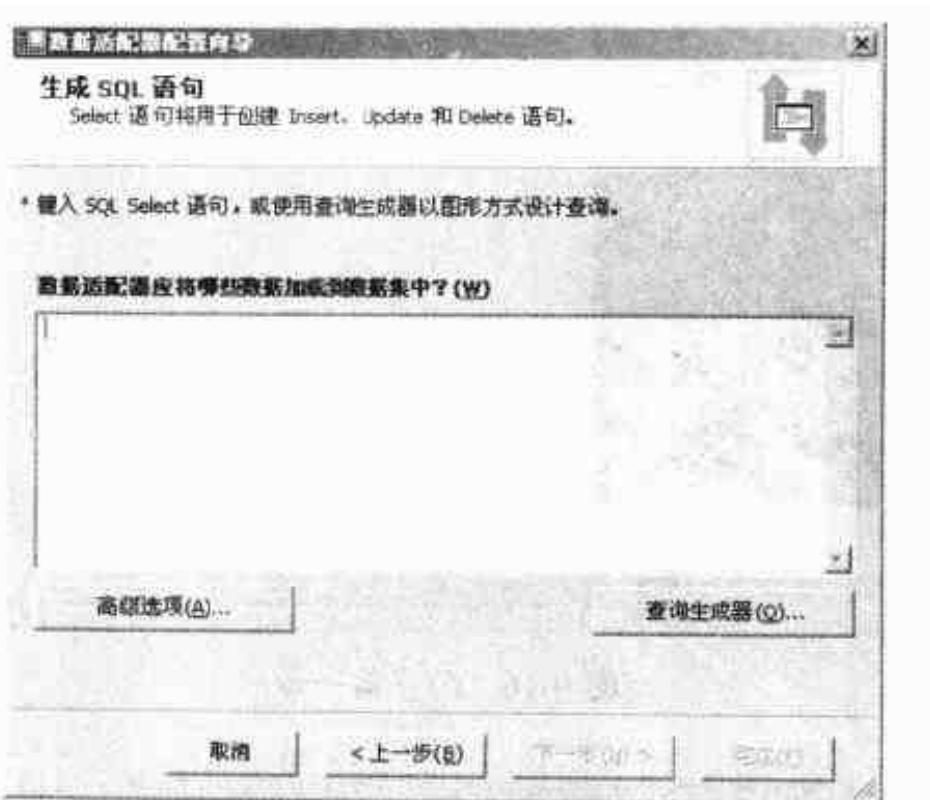


图 9-19 查询生成器

(5) 在图 9-20 的窗口上，选中要连接的数据表，然后单击“添加”，完成后单击“关闭”。



图 9-20 在查询生成器里添加表

(6) 在图 9-21 的窗口上，选择要查询的列，然后单击“确定”。

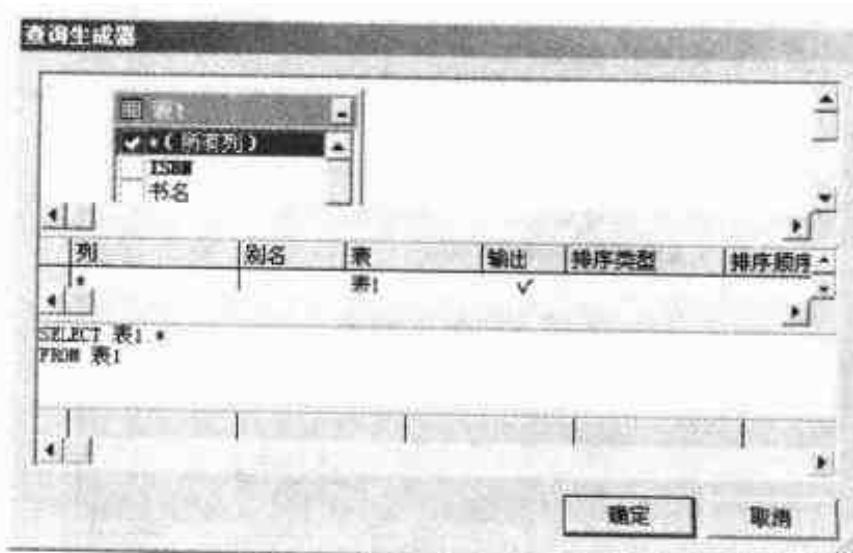


图 9-21 选择查询列

完成上述步骤后，一步步地单击下一步，直到完成时为止。

9.2.2 与非 SQL Server 数据库连接

下面演示如何用 VS.NET 7.0 连接 Access 数据库。

- (1) 如图 9-22 所示，选中“连接到数据库”并单击，即可出现如图 9-23 所示的窗口。



图 9-22 启动连接工具

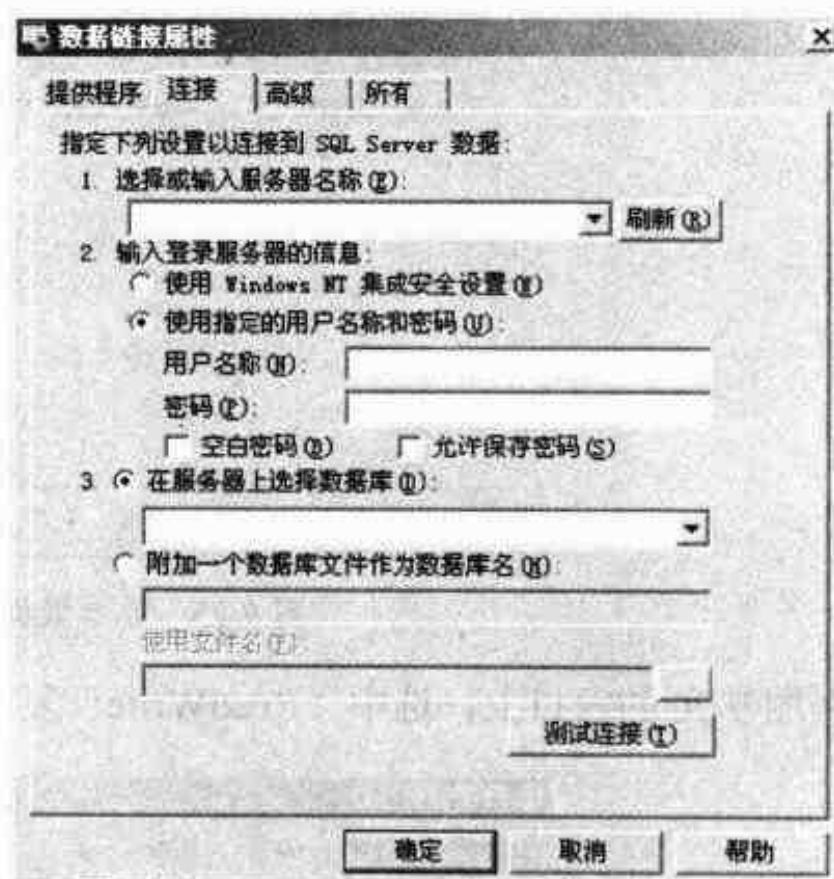


图 9-23 连接参数输入界面

- (2) 在图 9-23 的窗口上单击“提供程序”选项卡，即可出现如图 9-24 所示的窗口。



图 9-24 选择提供程序

- (3) 在图 9-24 所示的窗口上，选中 Microsoft Jet 4.0 OLE DB Provider 并单击“下一步”。

(4) 在图 9-25 所示的窗口上，通过浏览将数据库对象加入进来，然后单击“高级”选项卡。

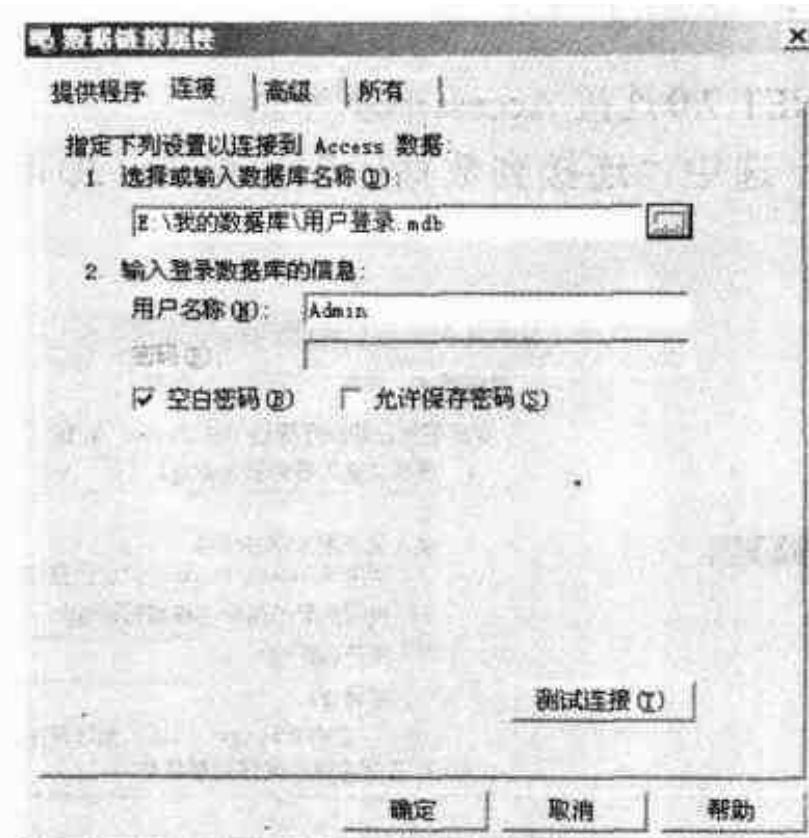


图 9-25 添加数据库

(5) 在图 9-26 的窗口上，选中“ReadWrite”复选框，然后单击“确定”按钮。

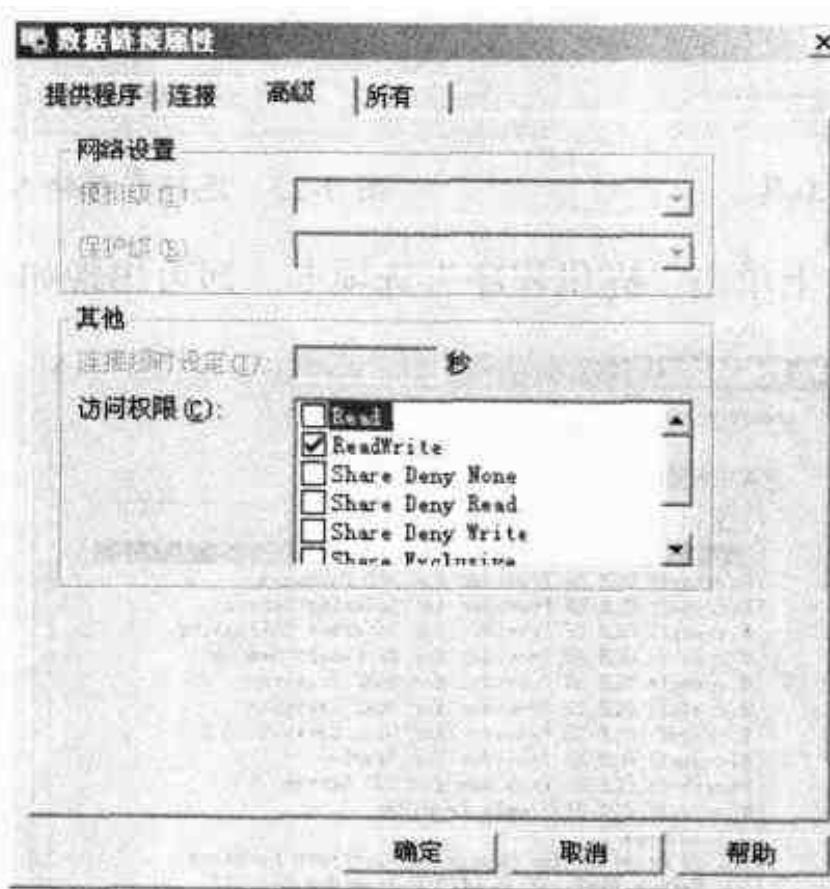


图 9-26 确定访问权限

完成上述步骤后，如图 9-27 所示，在“服务器资源管理器”里的“数据连接”目录下，系统自动添加了“ACCESS.E:\我的数据库\用户登录.mdb.Admin”。

(6) 如图 9-28 所示，将目标数据库的表（本例为“表 1”）拖放到页面（或窗体）上即可。

(7) 如图 9-29 所示，系统自动添加 `oleDbConnection1` 和 `oleDbDataAdapter1` 以及：

```
protected System.Data.OleDb.OleDbCommand oleDbSelectCommand1;
```

```
protected System.Data.OleDb.OleDbCommand oleDbInsertCommand1;
protected System.Data.OleDb.OleDbCommand oleDbUpdateCommand1;
protected System.Data.OleDb.OleDbCommand oleDbDeleteCommand1;
```



图 9-27 “数据连接”目录添加数据库后的情况

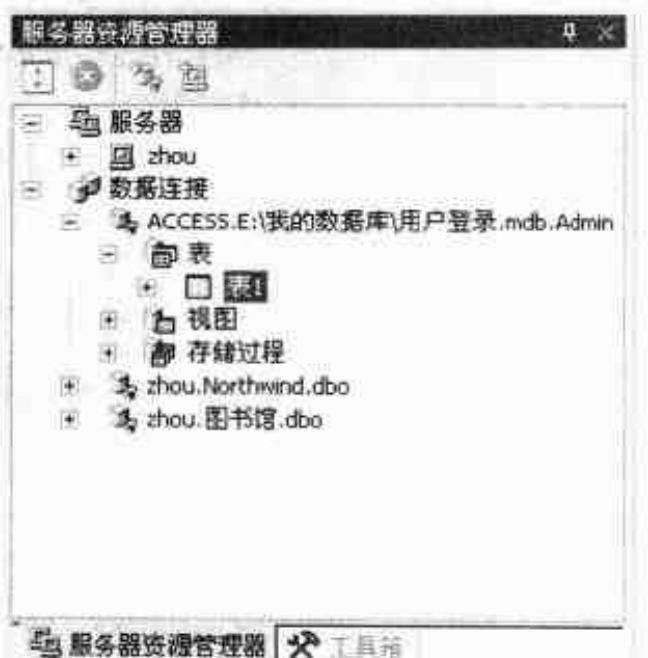


图 9-28 将表1拖放到页面



图 9-29 连接完成

(8) 建立数据集

如图 9-30 所示,在 `oleDbConnection1` 上单击鼠标右键,在快捷菜单上单击“生成数据集”。



图 9-30 生成数据集快捷菜单

(9) 如图 9-31 所示，在“新建”文本框里输入数据集名称（本例为 load），选中窗体上的“将此数据集添加到设计器”复选框，然后单击“确定”即可。

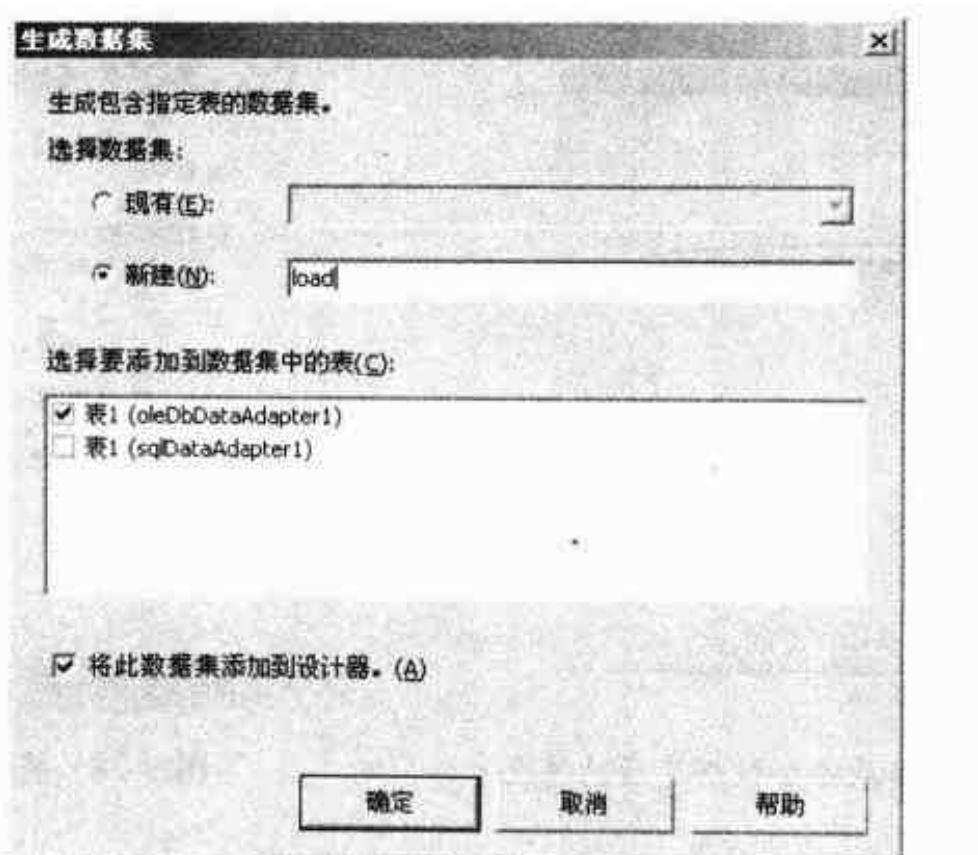


图 9-31 输入数据集名称

完成上述步骤后，页面上自动添加了数据集对象 load1，如图 9-32 所示。



图 9-32 数据集建立完成

需要注意的是，用拖放的方法连接数据库有一个缺点，就是身份认证有可能出错，在连接时发现不了，在使用时就出现了。这时可以使用 SqlDataAdapter 或 OleDbDataAdapter 来连接数据库。用 SqlDataAdapter 或 OleDbDataAdapter 连接数据库虽然麻烦，但要安全一些，使用时一般不会出现身份认证的错误。用数据适配器连接非 SQL 数据库可使用 OleDbDataAdapter，步骤和连接 SQL Server 数据库基本相同，只是在选择数据提供程序时，一般要选择 Microsoft Jet 4.0 OLE DB Provider 或其他程序。下面简单演示一下。

(1) 将 OleDbDataAdapter 拖放到页面上，一步步地单击“下一步”，直到出现如图 9-33 所示的界面。在图中单击“新建连接”。

(2) 进入图 9-34 所示的窗口，选中适当的数据提供程序。本例选择 Microsoft Jet 4.0 OLE DB Provider，然后单击“下一步”。

(3) 在图 9-35 所示的窗口上, 添加目标数据库并测试连接, 测试无误后, 单击“高级”选项卡, 设置操作数据的权限, 一般权限为 ReadWrite。设置完毕后, 单击“确定”, 剩下的步骤与连接 SQL Server 数据库没有区别了。

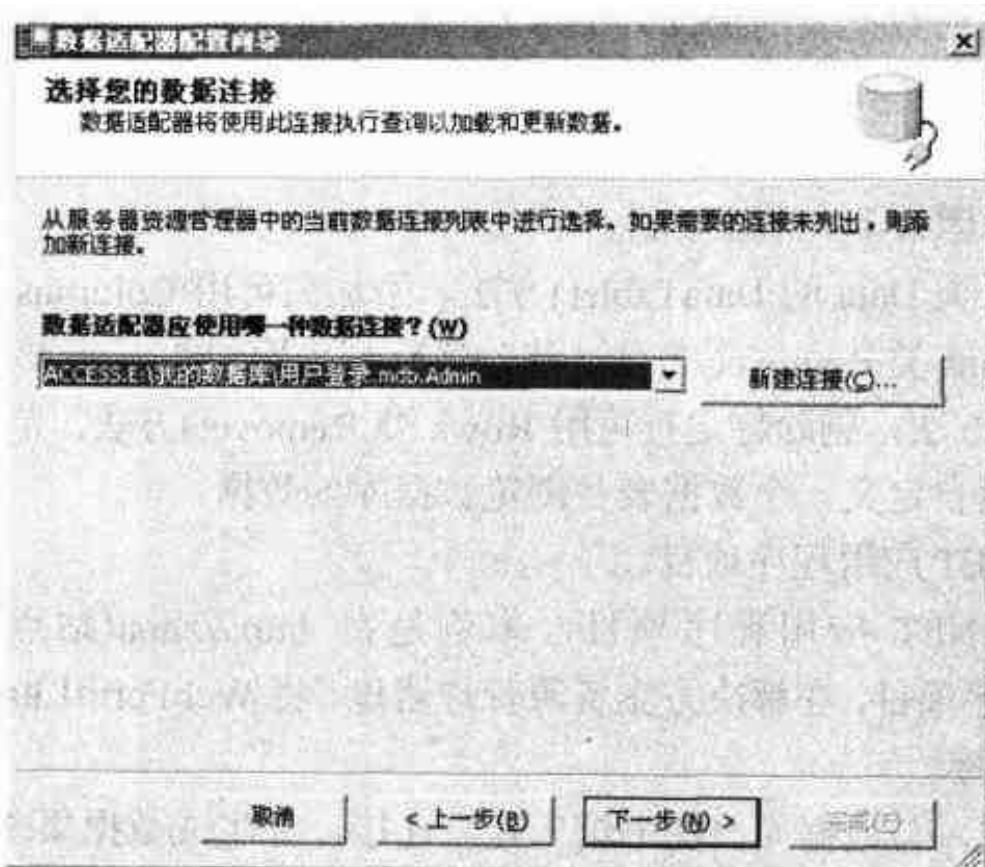


图 9-33 使用数据适配器建立与非 SQL 数据库连接



图 9-34 选择提供程序

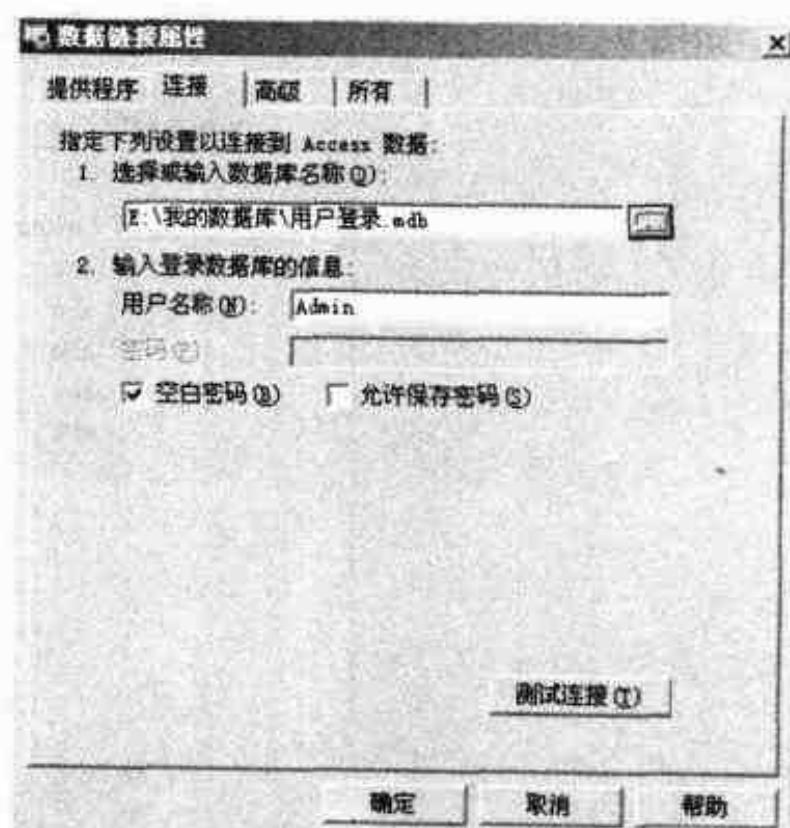


图 9-35 加入数据库

9.3 数据浏览

下面讲的数据表既包括数据库中的数据表, 也包括页面 (或窗体) 上的数据表。本章主要讲解数据库数据表的浏览、在页面上创建数据表等内容。

9.3.1 自定义页面表格

在很多情况下，数据库中的数据并不需要全部浏览，而只需浏览其中的一列或几列。这时候如果将数据库的数据表的全部列都放到页面（窗体）上，显得有一点画蛇添足，这就需要自定义一个表格，在表格里只显示数据库的部分列。到此为止，本书还未涉及数据查找、删除等知识，但这里可能用到一些数据查找的知识。你若看不懂没关系，只要明白如何建表就可以了，因为下一节还要讲。

数据表创建可用类 Data 的 DataTable()方法，添加列可用 Columns 类的 Add()方法，删除特定列可用 Columns 类 Remove()方法，清除所有列可用 Columns 类 Clear()方法。添加行可用 Rows 类 Add()方法，删除特定行可用 Rows 类 Remove()方法，清除所有行可用 Rows 类 Clear()方法。下面自定义一个数据表并浏览数据库的数据。

(1) 新建 ASP.NET 应用程序项目。

新建一个 ASP.NET 应用程序项目，本例是在 <http://zhou/> 站点下建立一个名称为 dataDemo 的应用程序项目，在解决方案资源管理器里，将 WebForm1.aspx 修改为 data.aspx。

(2) 建立数据连接。

用上一节的方法，与数据库“图书馆”建立连接，并生成数据集 book，其对象实例为 book1。

(3) 设计页面。

如图 9-36 所示，在页面上添加一个 DataGrid Web 控件，一个 Button Web 控件。

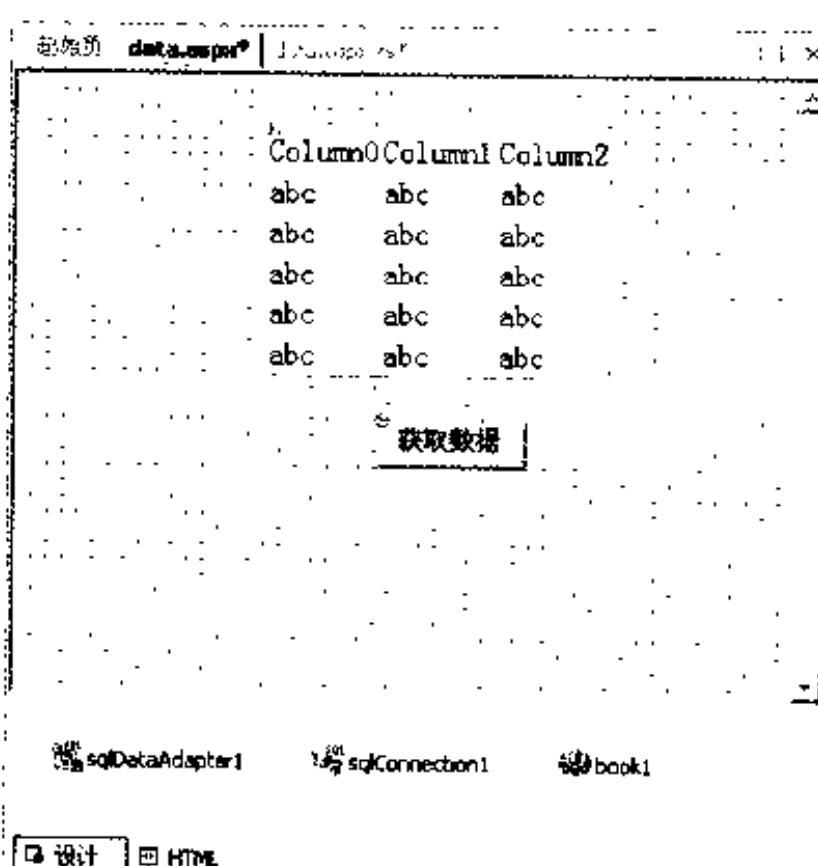


图 9-36 页面设计

(4) 添加引用。

```
using System.Data.SqlClient;
```

(5) 添加私有成员。

```
private DataTable table;
```

```
private DataColumn col;
private DataRow row;
```

(6) 在代码编辑窗口，找到如下方法：

```
private void Page_Load(object sender, System.EventArgs e)
{
    // 在此处放置用户代码以初始化页面
}
```

然后将该方法修改为：

```
private void Page_Load(object sender, System.EventArgs e)
{
    // 在此处放置用户代码以初始化页面
    try
    {
        table=new DataTable();
        catch{}
        try
        {
            col=new DataColumn();
            col.DataType=System.Type.GetType("System.String");
            col.ColumnName="书名";
            col.AutoIncrement=false;
            col.Unique=false;
            col.ReadOnly=false;
            table.Columns.Add(col);
            //*****
            col=new DataColumn();
            col.DataType=System.Type.GetType("System.String");
            col.ColumnName="作者";
            col.AutoIncrement=false;
            col.Unique=false;
            col.ReadOnly=false;
            table.Columns.Add(col);
            //*****
            col=new DataColumn();
            col.DataType=System.Type.GetType("System.String");
            col.ColumnName="责任者";
            col.AutoIncrement=false;
            col.Unique=false;
            col.ReadOnly=false;
            table.Columns.Add(col);
            //*****
            col=new DataColumn();
            col.DataType=System.Type.GetType("System.Double");
            col.ColumnName="定价";
        }
    }
}
```

```

        col.AutoIncrement=false;
        col.Unique=false;
        col.ReadOnly=false;
        table.Columns.Add(col);
        DataGrid1.DataSource=table;
        DataGrid1.DataBind();
    }
    catch{}
}

```

(7) 下面是“获取数据”按钮的 Click 事件代码:

```

private void Button1_Click(object sender, System.EventArgs e)
{
    string sele1="SELECT * FROM 表 1";

    sqlConnection1.Open();
    sqlSelectCommand1=new SqlCommand (sele1,sqlConnection1 );
    SqlDataReader reader1=sqlSelectCommand1.ExecuteReader();
    while(reader1.Read())
    {
        row=table.NewRow();
        row["书名"]=reader1.GetString(1);
        row["作者"]= reader1.GetString(2);
        row["责任者"]= reader1.GetString(4);
        row["定价"]=reader1.GetDouble(6);
        table.Rows.Add(row);
    }
    reader1.Close();
    sqlConnection1.Close();
    DataGrid1.DataSource=table;
    DataGrid1.DataBind();
}

```

下面是该程序的代码:

```

using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

```

```
namespace dataDemo
{
    /// <summary>
    /// WebForm1 的摘要说明
    /// </summary>
    public class WebForm1 : System.Web.UI.Page
    {
        private DataTable table;
        private DataColumn col;
        protected System.Web.UI.WebControls.Button Button1;
        protected System.Web.UI.WebControls.DataGrid DataGrid1;
        protected System.Data.SqlClient.SqlDataAdapter sqlDataAdapter1;
        protected System.Data.SqlClient.SqlCommand sqlSelectCommand1;
        protected System.Data.SqlClient.SqlCommand sqlInsertCommand1;
        protected System.Data.SqlClient.SqlCommand sqlUpdateCommand1;
        protected System.Data.SqlClient.SqlCommand sqlDeleteCommand1;
        protected System.Data.SqlClient.SqlConnection sqlConnection1;
        protected dataDemo.book book1;
        private DataRow row;
        private void Page_Load(object sender, System.EventArgs e)
        {
            // 在此处放置用户代码以初始化页面
            try
            {
                table=new DataTable();
            }
            catch{}
            try
            {
                col=new DataColumn();
                col.DataType=System.Type.GetType("System.String");
                col.ColumnName="书名";
                col.AutoIncrement=false;
                col.Unique=false;
                col.ReadOnly=false;
                table.Columns.Add(col);
                //*****
                col=new DataColumn();
                col.DataType=System.Type.GetType("System.String");
                col.ColumnName="作者";
                col.AutoIncrement=false;
                col.Unique=false;
                col.ReadOnly=false;
                table.Columns.Add(col);
                //*****
            }
        }
    }
}
```

```
        col=new DataColumn();
        col.DataType=System.Type.GetType("System.String");
        col.ColumnName="责任者";
        col.AutoIncrement=false;
        col.Unique=false;
        col.ReadOnly=false;
        table.Columns.Add(col);
        //*****
        col=new DataColumn();
        col.DataType=System.Type.GetType("System.Double");

        col.ColumnName="定价";
        col.AutoIncrement=false;
        col.Unique=false;
        col.ReadOnly=false;
        table.Columns.Add(col);
        DataGrid1.DataSource=table;

        DataGrid1.DataBind();
    }
    catch{}
}

//下面系统自动产生的代码已经删除
#region Web Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
private void Button1_Click(object sender, System.EventArgs e)
{
    string sele1="SELECT * FROM 表 1";

    sqlConnection1.Open();
    sqlSelectCommand1=new SqlCommand (sele1,sqlConnection1 );
    SqlDataReader reader1=sqlSelectCommand1.ExecuteReader();
    while(reader1.Read())
    {
        ...
        row=table.NewRow();
        row["书名"]=reader1.GetString(1);
        row["作者"]= reader1.GetString(2);
        row["责任者"]= reader1.GetString(4);
        reader1.GetDouble(6);
        //注意: 要用 GetDouble() 而不是 GetFloat() 方法获取
        //SQL Sever 的 Float 类型值, 否则会出错。
        row["定价"]=reader1.GetDouble(6);
```

```
        table.Rows.Add(row);
    }
    reader1.Close();
    sqlConnection1.Close();
    DataGrid1.DataSource=table;
    DataGrid1.DataBind();
}
}
```

(8) 演示。

编译并执行程序，图 9-37 是该程序的初始状态。



图 9-37 自定义表格数据加载前的结果

单击“获取数据”按钮，结果如图 9-38 所示。

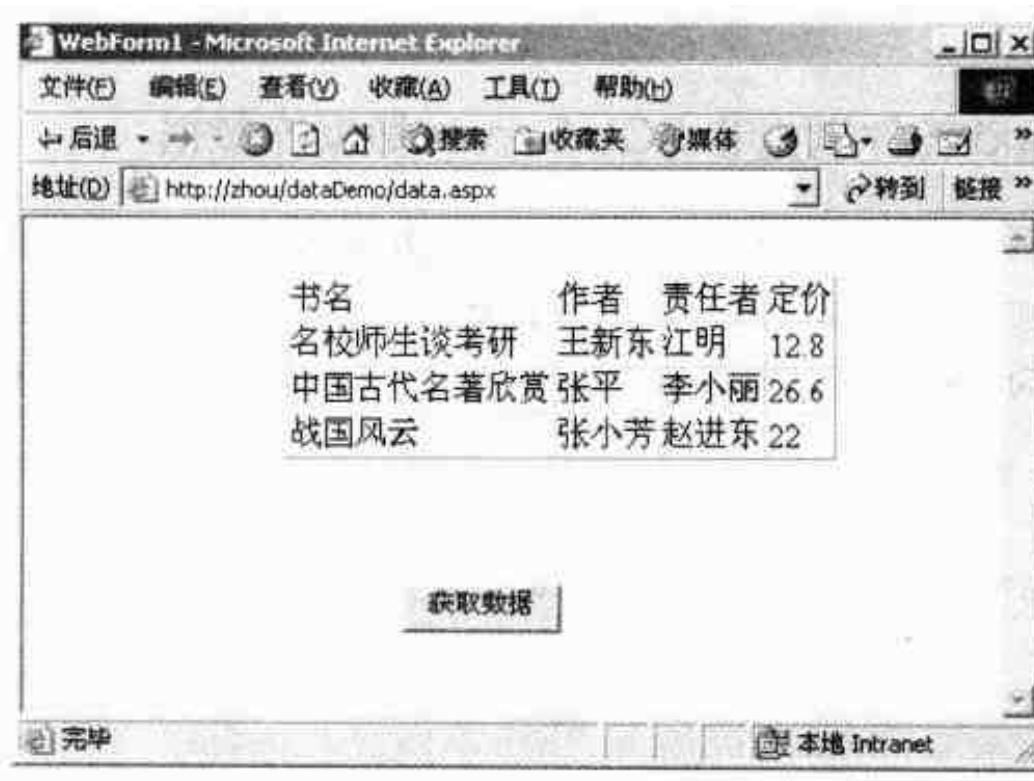


图 9-38 自定义表格数据加载后的结果

9.3.2 用 DataGrid 控件浏览 SQL Server 数据库数据

- (1) 在页面上再添加一个按钮，将其 Text 属性修改为“浏览”。
- (2) “浏览”按钮的 Click 事件的代码下所示。

```
private void Button2_Click(object sender, System.EventArgs e)
{
    sqlDataAdapter1.Fill(book1.表1);
    DataGrid1.DataSource=book1.表1;
    DataGrid1.DataBind();
}
```

编译并执行程序，图 9-39 是浏览数据的结果。

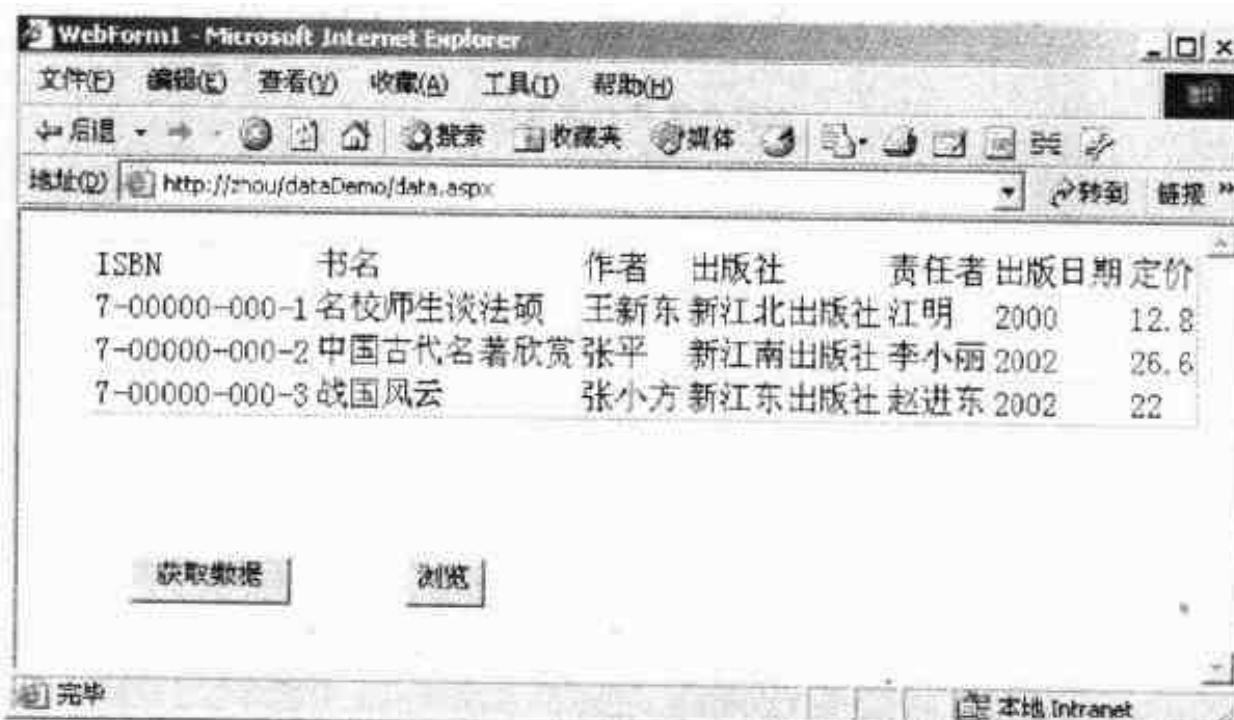


图 9-39 使用向导建立的数据集浏览数据的结果

9.3.3 用 DataGrid 控件浏览非 SQL Server 数据库数据

浏览非 SQL Server 数据库数据的方法与浏览 SQL Server 数据库的数据的方法基本相同。可以在页面上添加一个 Button 控件，然后建立与一个 ACCESS 数据库的连接，并生成适当的数据集（本例为 suer，其对象实例为 user1）。下面是 Button1 的 Click 事件的代码。

```
private void Button3_Click(object sender, System.EventArgs e)
{
    oleDbDataAdapter1.Fill(user1.表1);
    DataGrid1.DataSource=user1.表1;
    DataGrid1.DataBind();
}
```

图 9-40 是浏览的结果。



图 9-40 ACCESS 数据浏览结果

9.4 数据查询、插入、删除和更新

9.4.1 数据查询

数据查询可用 SQL 的 SELECT 语句进行，该语句的语法为：

```

SELECT [DISTINCT|ALL] { * | 模式名 . } [ 表名 | 视图名 |
快照名 ] . * ... | { 表达式 [ 列别名 ] ... } [ , [ 模式名 . ] [ 表名 |
视图名 ] ] . * ... | 表达式 [ 列别名 ] ] ...
FROM [ 模式名 . ] [ 表名 | 视图名 | 快照名 ] [ @ 数据库链名 ] [ 表别名 ]
[ , [ 模式名 . ] [ 表名 | 视图名 | 快照名 ] [ @ 数据库链名 ] ]
[ 表别名 ] ...
[ WHERE 条件 ]
[ START WITH 条件 CONNECT BY 条件 ]
[ GROUP BY 表达式 [ , 表达式 ] ... [ HAVING 条件 ]
[ UNION | UNION ALL | INTERSECT | MINUS ] SELECT 命令
[ ORDER BY { 表达式 | 位置 } [ ASC | DESC ] [ , { 表达式 | 位置 } [ ASC | DESC ] ] ] ...

```

例如，对于 BOOKS 表：

ISBN	书名	定价
0001	XXXXXX	15
0002	YYYYYY	21

- 查询定价为 15 的书名：

```
SELECT 书名 FROM BOOKS WHERE 定价=15;
```

还可以使用 “>=” 、 “<=” 、 “>” 、 “<” 等查询。

- 查询定价在 10 至 50 之间的书名：

```
SELECT 书名 FROM BOOKS WHERE 定价 BETWEEN 10 AND 50;
```

- 查询定价不在 11 至 21 之间的书名:

```
SELECT 书名 FROM BOOKS WHERE 定价 NOT BETWEEN 11 AND 21;
```

- 查询所有书名以 X 开头的图书的书名:

```
SELECT 书名 FROM BOOKS WHERE 书名 LIKE 'B%';
```

- 将所有图书按定价顺序降序排列:

```
SELECT * FROM BOOKS ORDER BY 定价 DESC;
```

- 将所有图书按定价顺序升序排列:

```
SELECT * FROM BOOKS ORDER BY 定价 ASC;
```

下面演示常用的查询方法。

9.4.1.1 数值查询

数值查询一般包括“=”、“>”、“<”、“>=”、“<=”、“BETWEEN”以及求最大值、最小值、平均值、最大值差值与最小值之间差值等查询方法。

1. 等值查询

“>”、“<”、“>=”、“<=”查询与等值查询方法相同，这里不再赘述，只要参考等值查询即可。

如果要查询一条定价为 26.6 的图书，可用如下代码来实现（在 9.2 节程序的基础上修改而得）。

```
private void Button2_Click(object sender, System.EventArgs e)
{
    sqlDataAdapter1.SelectCommand.CommandText=
        "SELECT * FROM 表1 WHERE 定价=26.6";
    sqlDataAdapter1.Fill(book1.表1);
    DataGrid1.DataSource=book1.表1;
    DataGrid1.DataBind();
}
```

演示结果如图 9-41 所示。

2. BETWEEN 查询

下面查询一条定价在 12 以上且 19 以下的数据，代码如下：

```
private void Button2_Click(object sender, System.EventArgs e)
{
    sqlDataAdapter1.SelectCommand.CommandText=
        "SELECT * FROM 表1 WHERE 定价 BETWEEN 12 AND 19";
    sqlDataAdapter1.Fill(book1.表1);
    DataGrid1.DataSource=book1.表1;
    DataGrid1.DataBind();
}
```

演示结果如图 9-42 所示。

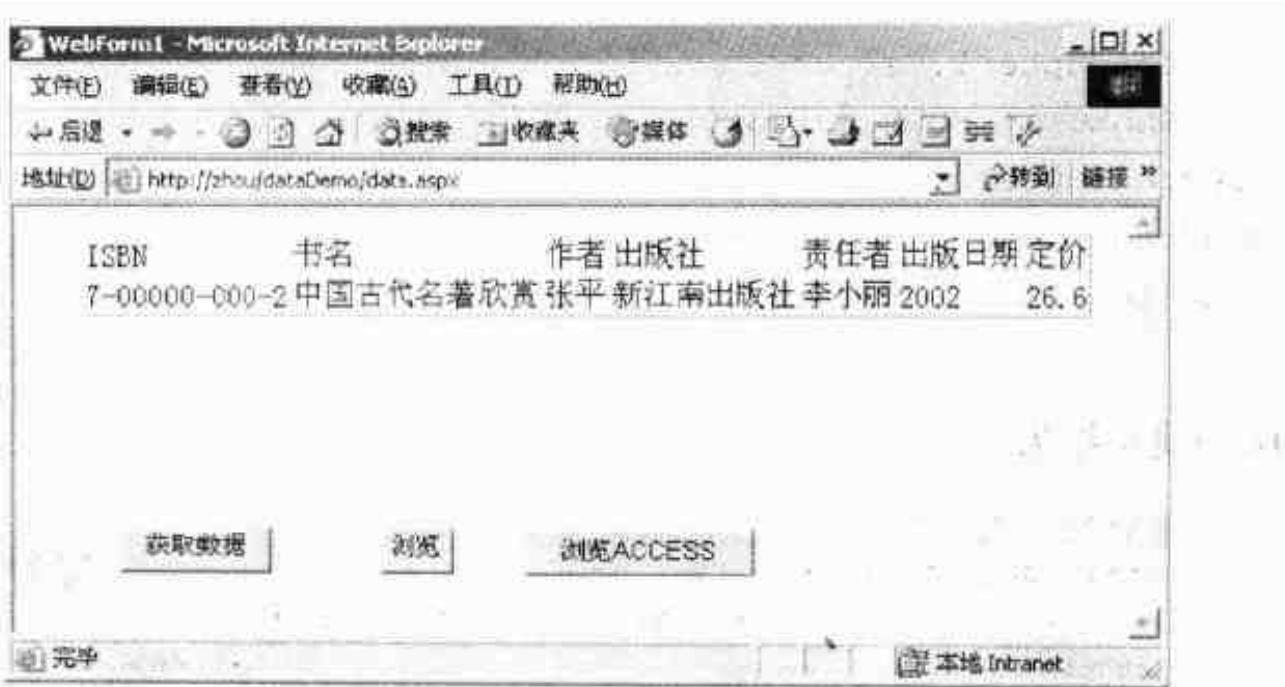


图 9-41 等值查询

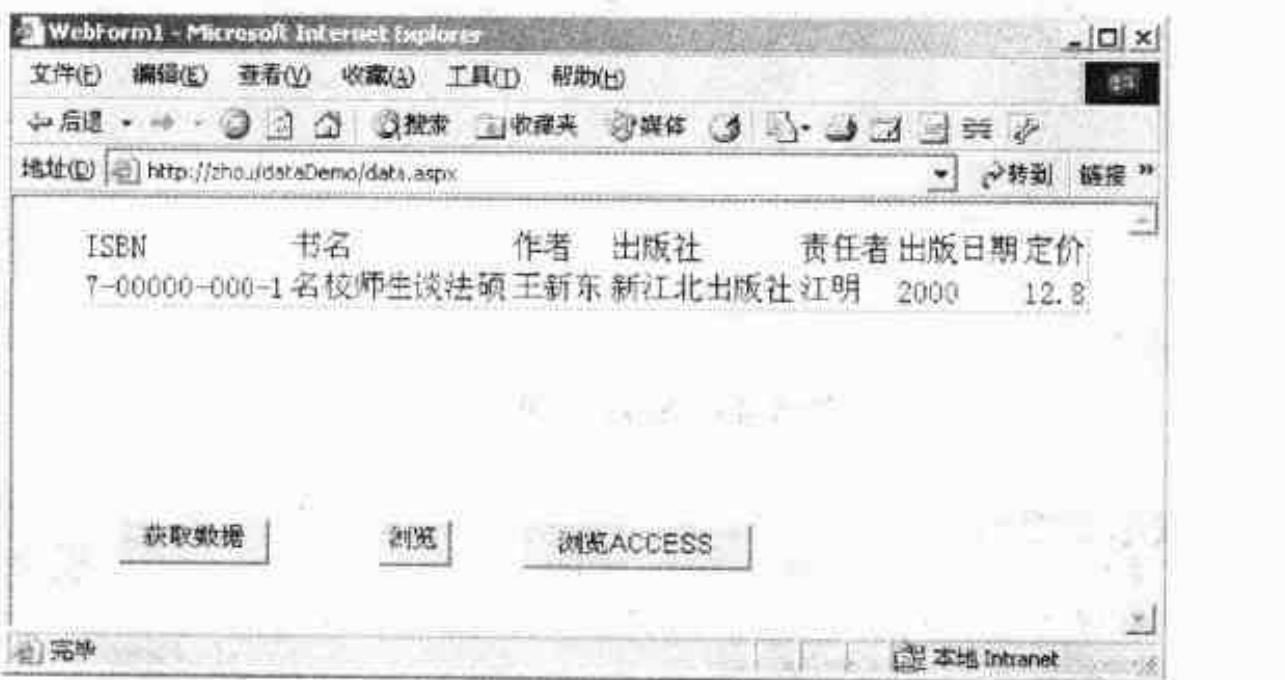


图 9-42 BETWEEN 查询

3. NOT BETWEEN 查询

下面查询定价不在 12 至 19 之间的所有图书，代码如下：

```
private void Button2_Click(object sender, System.EventArgs e)
{
    sqlDataAdapter1.SelectCommand.CommandText=
        "SELECT * FROM 表1 WHERE 定价 NOT BETWEEN 12 AND 19";
    sqlDataAdapter1.Fill(book1.表1);
    DataGrid1.DataSource=book1.表1;
    DataGrid1.DataBind();
}
```

演示结果如图 9-43 所示。

4. 降序查询

下面代码可按降序查询所有图书。

```

private void Button2_Click(object sender, System.EventArgs e)
{
    sqlDataAdapter1.SelectCommand.CommandText=
        "SELECT * FROM 表1 ORDER BY 定价 DESC";
    sqlDataAdapter1.Fill(book1.表1);
    DataGrid1.DataSource=book1.表1;
    DataGrid1.DataBind();
}

```

图 9-44 是演示结果。

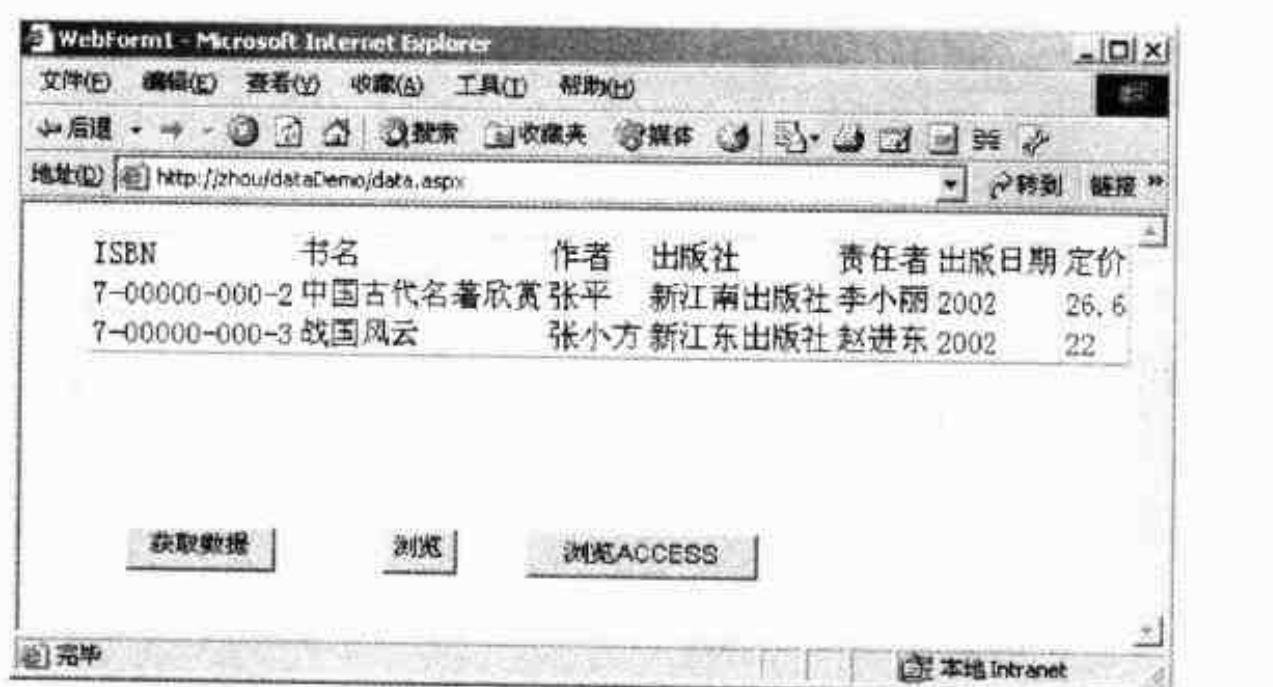


图 9-43 NOT BTWEEN 查询

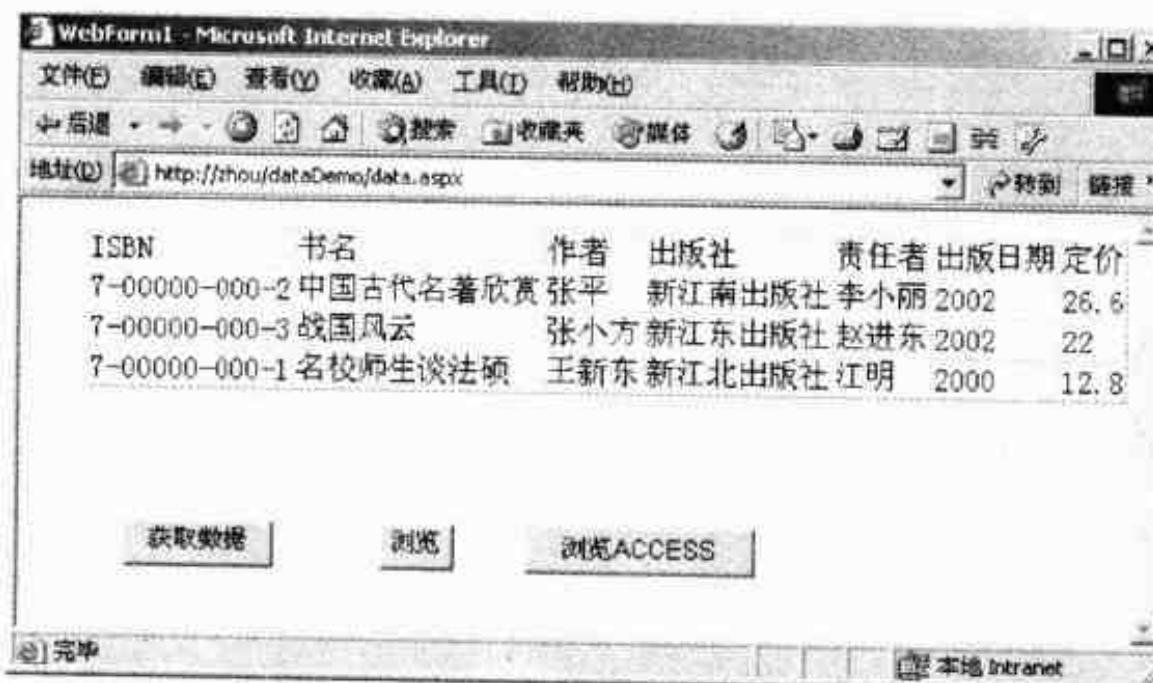


图 9-44 降序查询

如果要升序查询，可将代码修改为：

```

private void Button2_Click(object sender, System.EventArgs e)
{
    sqlDataAdapter1.SelectCommand.CommandText=
        "SELECT * FROM 表1 ORDER BY 定价 ASC ";
    sqlDataAdapter1.Fill(book1.表1);
    DataGrid1.DataSource=book1.表1;
}

```

```
        DataGrid1.DataBind();  
    }  
}
```

9.4.1.2 字符串查询

字符串查询主要包括等值查询、开头字符（串）匹配查询和子字符串查询。下面演示这几种查询的用法。

1. 等值字符串查询

下面代码查询责任者为“李小丽”的数据。

```
private void Button2_Click(object sender, System.EventArgs e)  
{  
    sqlDataAdapter1.SelectCommand.CommandText=  
        "SELECT * FROM 表1 WHERE 责任者='"+""+"李小丽"+"';  
    sqlDataAdapter1.Fill(book1.表1);  
    DataGrid1.DataSource=book1.表1;  
    DataGrid1.DataBind();  
}
```

图 9-45 是演示结果。

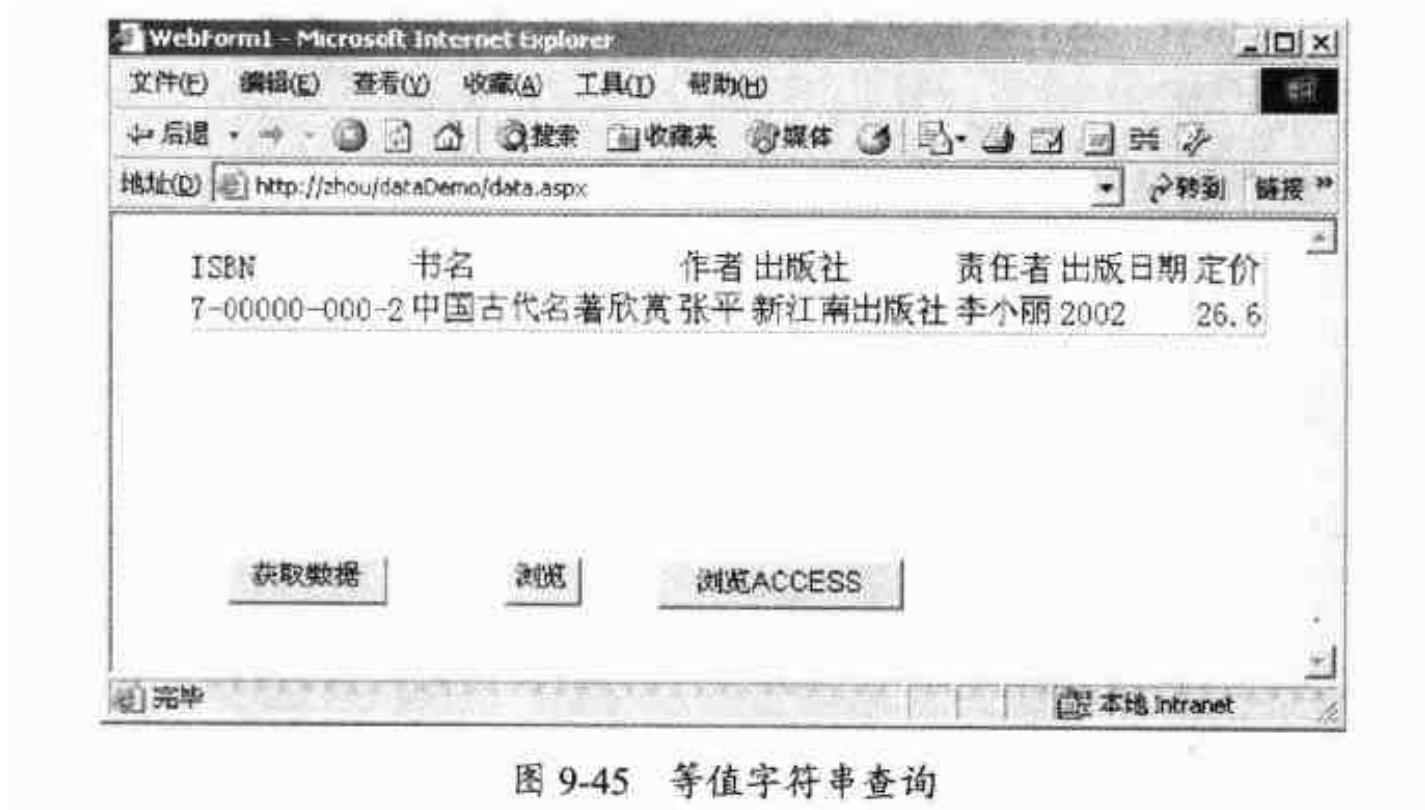


图 9-45 等值字符串查询

2. 开头字符（串）匹配查询

下面代码演示了查询责任者头两个字为“李小”的数据。

```
private void Button2_Click(object sender, System.EventArgs e)  
{  
    //注意“责任者”与 LIKE 之间有空格  
    sqlDataAdapter1.SelectCommand.CommandText=  
        "SELECT * FROM 表1 WHERE 责任者 LIKE '"+%"李小%"+"';  
    sqlDataAdapter1.Fill(book1.表1);  
    DataGrid1.DataSource=book1.表1;  
    DataGrid1.DataBind();  
}
```

图 9-46 是演示结果。

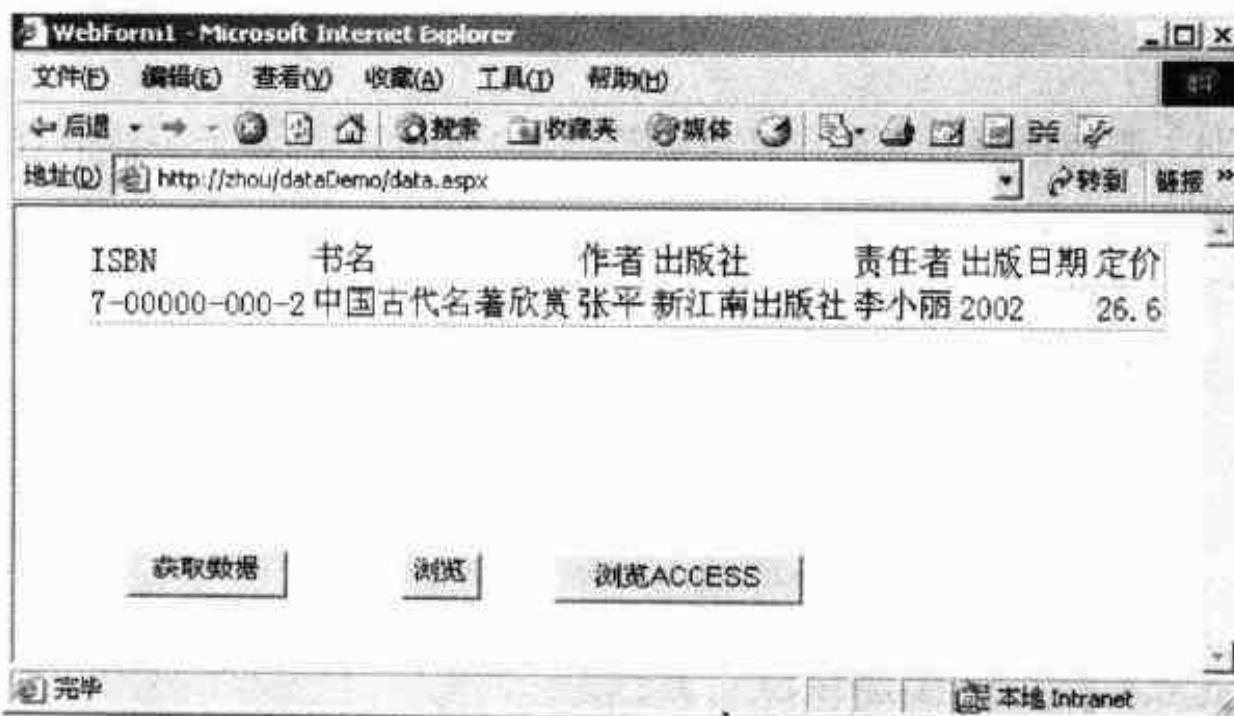


图 9-46 开头字符串查询

3. 子字符串查询

子字符串查询要使用 `SqlDataReader` 类的 `Read` 方法和 `String` 类的 `IndexOf` 方法。下列代码首先建立一个数据表，然后逐条检索数据，遇到书名中包含“名著”二字时，将数据加入到数据表中。

```
private void Button2_Click(object sender, System.EventArgs e)
{
    try
    {
        table=new DataTable();
    }
    catch{}
    try
    {
        //*****
        col=new DataColumn();
        col.DataType=System.Type.GetType("System.String");
        col.ColumnName="ISBN";
        col.AutoIncrement=false;
        col.Unique=false;
        col.ReadOnly=false;
        table.Columns.Add(col);
        //*****
        col=new DataColumn();
        col.DataType=System.Type.GetType("System.String");
        col.ColumnName="书名";
        col.AutoIncrement=false;
        col.Unique=false;
        col.ReadOnly=false;
```

```
table.Columns.Add(col);
//*****
col=new DataColumn();
col.DataType=System.Type.GetType("System.String");
col.ColumnName="作者";
col.AutoIncrement=false;
col.Unique=false;
col.ReadOnly=false;
table.Columns.Add(col);
//*****
col=new DataColumn();

col.DataType=System.Type.GetType("System.String");
col.ColumnName="出版社";
col.AutoIncrement=false;
col.Unique=false;
col.ReadOnly=false;
table.Columns.Add(col);
//*****
col=new DataColumn();
col.DataType=System.Type.GetType("System.String");
col.ColumnName="责任者";
col.AutoIncrement=false;
col.Unique=false;
col.ReadOnly=false;
table.Columns.Add(col);
//*****
col=new DataColumn();
col.DataType=System.Type.GetType("System.String");
col.ColumnName="出版日期";
col.AutoIncrement=false;
col.Unique=false;
col.ReadOnly=false;
table.Columns.Add(col);
//*****
col=new DataColumn();
col.DataType=System.Type.GetType("System.Double");
col.ColumnName="定价";
col.AutoIncrement=false;
col.Unique=false;
col.ReadOnly=false;
table.Columns.Add(col);
DataGrid1.DataSource=table;
DataGrid1.DataBind();
}
catch{}
```

```

string sele1="SELECT * FROM 表 1";
sqlConnection1.Open();
sqlSelectCommand1=new SqlCommand (sele1,sqlConnection1 );
SqlDataReader reader1=sqlSelectCommand1.ExecuteReader();
while(reader1.Read())
{
    string name=reader1.GetString(1);
    int x=name.IndexOf("名著");
    if(x!=-1)
    {
        row=table.NewRow();
        row["ISBN"]=reader1.GetString(0);
        row["书名"]=reader1.GetString(1);
        row["作者"]= reader1.GetString(2);
        row["出版社"]=reader1.GetString(3);
        row["责任者"]= reader1.GetString(4);
        row["出版日期"]=reader1.GetString(5);
        row["定价"]=reader1.GetDouble(6);
        table.Rows.Add(row);
    }
}
reader1.Close();
sqlConnection1.Close();
DataGrid1.DataSource=table;
DataGrid1.DataBind();
}

```

图 9-47 是演示结果。

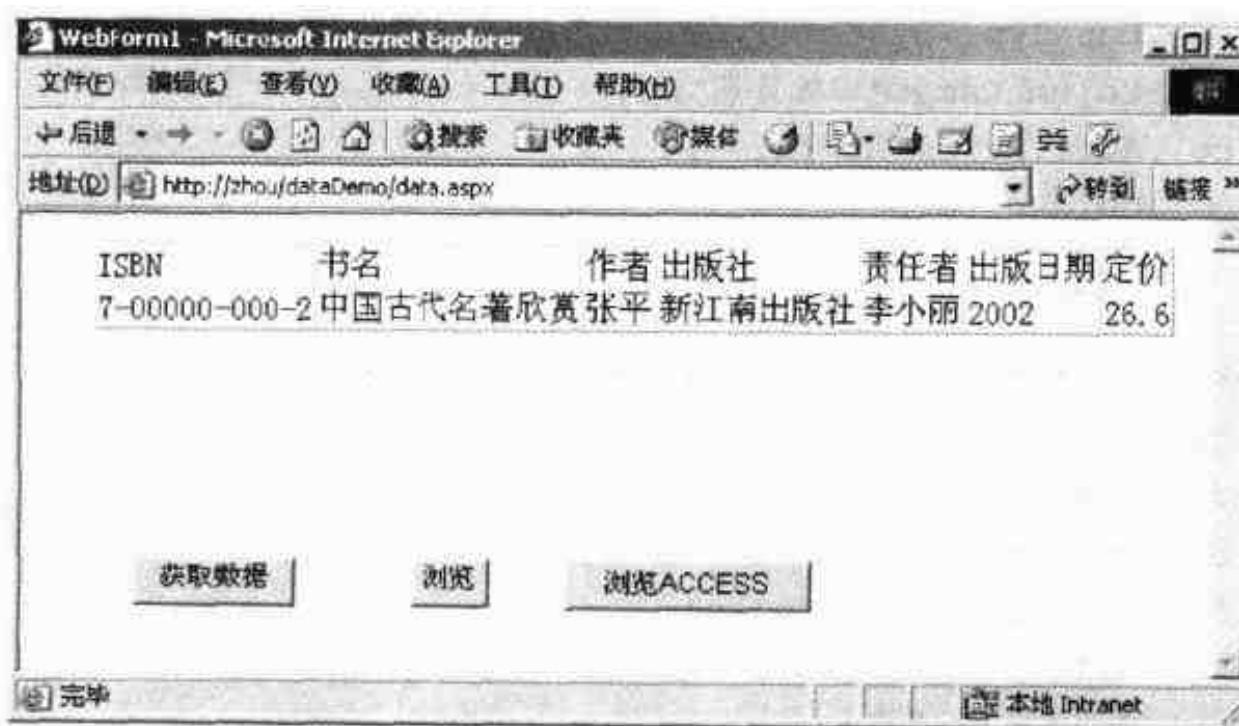


图 9-47 子字符串查询

9.4.2 数据插入

数据的插入可使用 SQL 的 INSERT 语句实现，该语句语法为：

```
INSERT INTO 表名 [(列名 1,...)]
VALUES (值 1, 值 2, ..., 值 n)
[子查询];
```

下列代码演示插入一条 ISBN 为“7-00000-000-4”，书名为“时代呼声”，作者为“周明”，出版社为“新时代出版社”，责任者为“王昕”，出版日期为“2002”，定价为“12.9”的数据，并且将全部数据浏览出来。

```
private void Button4_Click(object sender, System.EventArgs e)
{
    string insert="INSERT INTO 表1(ISBN,书名,作者,出版社,责任者,出版日期,
    定价)VALUES "+("+"+"+"7-00000-000-4"+""+", "+""+"时代呼声"+
    +"", "+""+"周明"+""+", "+""+"新时代出版社"+""+", "+""+
    +"王昕"+""+", "+""+"2002"+""+", "+""+"12.9"+""+");
    book1.AcceptChanges();
    sqlConnection1.Open();
    sqlInsertCommand1=new SqlCommand(insert,sqlConnection1);
    sqlInsertCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表1 ";
    sqlDataAdapter1.Fill(book1.表1);
    DataGrid1.DataSource=book1.表1;
    DataGrid1.DataBind();
}
```

图 9-48 为演示结果。

ISBN	书名	作者	出版社	责任者	出版日期	定价
7-00000-000-1	名校师生谈法硕	王新东	新江北出版社	江明	2000	12.8
7-00000-000-2	中国古代名著欣赏	张平	新江南出版社	李小丽	2002	26.6
7-00000-000-3	战国风云	张小方	新江东出版社	赵进东	2002	22
7-00000-000-4	时代呼声	周明	新时代出版社	王昕	2002	12.9

图 9-48 数据插入

9.4.3 数据删除

数据删除可使用 SQL 的 DELETE 语句实现，该语句语法为：

```
DELETE FROM 表名 WHERE 条件;
```

下列代码演示先删除一条 ISBN 为“7-00000-000-4”的数据，然后浏览全部数据。

```
private void Button5_Click(object sender, System.EventArgs e)
{
    string deleteCommand="DELETE FROM 表 1
    WHERE ISBN='"+""+7-00000-000-4""+'";
    book1.AcceptChanges();
    sqlConnection1.Open();
    sqlDeleteCommand1=new SqlCommand(deleteCommand,sqlConnection1);
    sqlDeleteCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表 1 ";
    sqlDataAdapter1.Fill(book1.表 1);
    DataGrid1.DataSource=book1.表 1;
    DataGrid1.DataBind();
}
```

图 9-49 是演示结果。

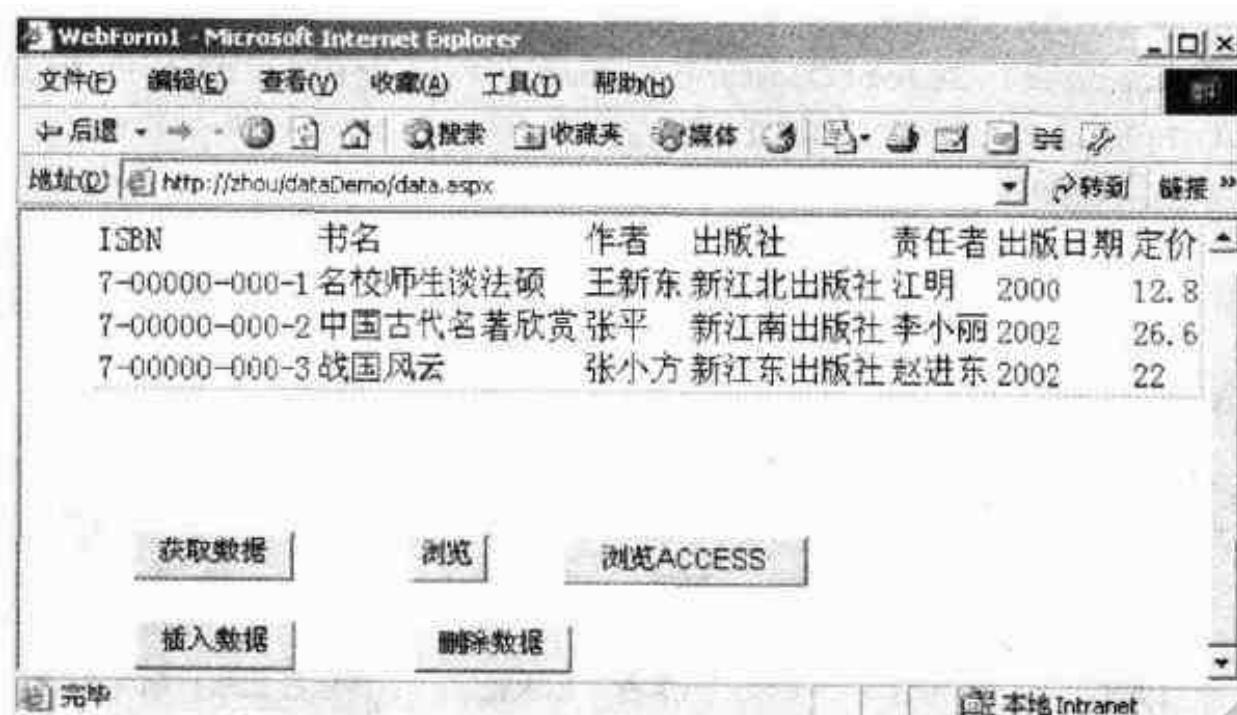


图 9-49 数据删除

9.4.4 数据更新

数据更新（替换）可用 SQL 的 UPDATE 语句实现，该语句语法为：

```
UPDATE 表名 SET 列名 1 = 表达式 1, 列名 2 = 表达式 2, ...
WHERE 条件;
```

下列代码演示将一条 ISBN 为“7-00000-000-2”的数据的“作者”替换为“李小丽”，“责任者”替换为“张平”。

```
private void Button6_Click(object sender, System.EventArgs e)
{
    book1.AcceptChanges();
    string update="UPDATE 表 1 SET "+ " "+ "作者=" + " " +"李小丽" + " " + ", "
    +"责任者=" + " " +"张平" + " " + " WHERE"
    +"
```

```
+ " " + "ISBN=" + " " + "7-00000-000-2" + " ";
sqlConnection1.Open();
sqlDeleteCommand1=new SqlCommand(update,sqlConnection1);
sqlDeleteCommand1.ExecuteNonQuery();
sqlConnection1.Close();
sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表1";
sqlDataAdapter1.Fill(book1.表1);
DataGrid1.DataSource=book1.表1;
DataGrid1.DataBind();
}
```

图 9-50 是上述代码的演示结果。

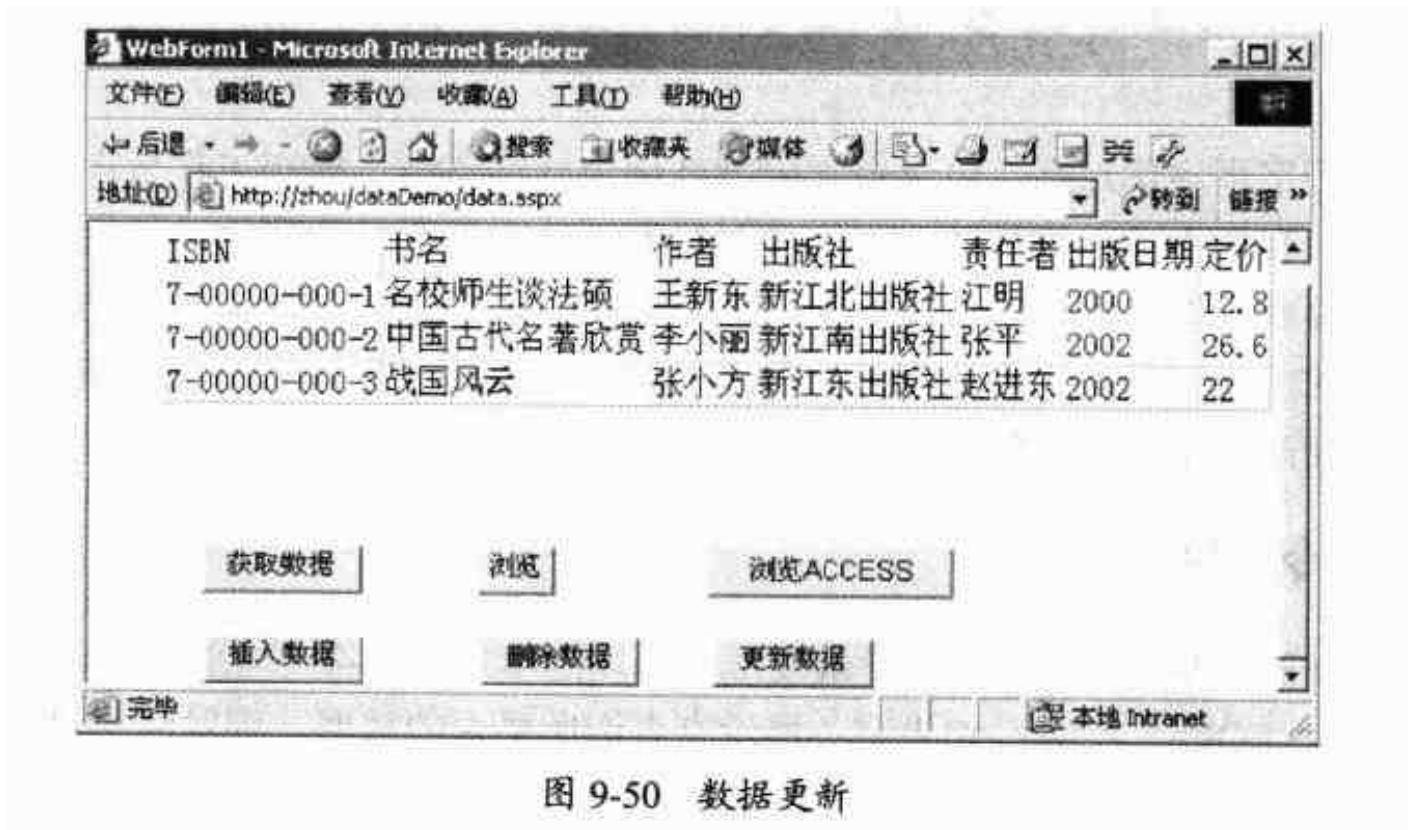


图 9-50 数据更新

本章小结

本章利用 ASP.NET 应用程序，简单介绍了 Web 数据库基础知识。通过本章的学习，不但可以掌握数据库的基础知识，而且有助于 Web 应用程序的开发。有关数据库的 Web 服务在本书第 11 章讲解。

第 10 章 数据库的异步套接字网络应用

在 Win32 开发中，数据库也是常用内容之一，在很多情况下，Win32 的数据库网络应用比 Web 数据库应用更方便，而 Web 服务速度较慢，灵活性差。本章主要讲解如何用异步套接字开发网络数据库程序。本章的程序文本编码使用 UTF8 码。

10.1 异步套接字的数据库服务器开发

异步套接字程序优点是，可以最大限度地使用网络资源，提高工作效率；其缺点是，在网络忙时，数据不能同步到达，在许多时候要等待后续数据的到达，这给程序带来了不便。在具体商业开发中，要根据实际情况，选择究竟是使用同步套接字还是异步套接字。一般而言，对于经常处理大量数据的服务器来说，使用异步服务较经济一些；但如果要降低出错率，提高程序的安全性，则可选择同步操作。

10.1.1 命令识别

本章开发的异步服务系统，客户端发送命令以回车加换行符（<CRLF>）结束。服务器发送数据以“<EOF>”结束。如果服务器没有查询所要求的数据，则发回“<ER>”。本章使用“图书馆”数据库，该数据库的定义如图 10-1 所示。

列名	数据类型	长度	允许空
ISBN	char	30	
书名	char	100	✓
作者	char	30	✓
出版社	char	50	✓
责任者	char	30	✓
出版日期	char	30	✓
定价	float	8	✓

图 10-1 基础数据库数据表

本程序适合图书馆使用，支持查找子字符串。读者可以通过“书名”、“作者”、“出版社”、“出版日期”查询数据库的全部数据。客户端发送的命令有以下几个部分组成：

<判断码><SP><参数><SP><参数>...

其中“判断码”是四位，比如“1000”，意思是通过“书名”查询数据。如果客户端发送如下命令：

1000 中国

该命令意思是：SELECT * FROM <特定数据库> WHERE 书名中包含“中国”二字。

再看如下命令：

1100 中国 张

该命令意思是：SELECT * FROM <特定数据库> WHERE 书名中包含“中国”二字并且作者的姓名中包含“张”字。

判断码的一位代表“书名”，第二位代表“作者”，第三位代表“出版社”，第四位代表“出版日期”。

10.1.2 检查命令是否发送完毕

客户端发送命令以“\r\n”（CRLF）结束。由于服务器使用的是异步套接字，在网络忙时，可能会出现不能一次把全部命令接受完毕，而需要在网络空闲时继续接收命令。下列代码首先判断命令是否接收完毕：如果没有接收完毕，则将接受的数据存放在字符串 getCommand 并返回空字符串；如果接收完毕，则返回存放全部命令的字符串 getCommand。字符串 getCommand 的初始状态是空字符串，每次建立连接时，自动赋值为空。

```
private string chenEnd(string aimString)
{
    int x=aimString.IndexOf("\r\n");
    if(x!=-1)
    {
        string subComm=aimString.Substring(0,x);
        getCommand=getCommand+subComm;
        return getCommand;
    }
    else{
        getCommand=getCommand+aimString;
        return "";
    }
}
```

10.1.3 接收并执行命令

下列代码演示如何接收命令、查询数据库以及发送数据。

```
private void ReadCallback(IAsyncResult ar)
{
    StateObject state = (StateObject) ar.AsyncState;
    Socket tt = state.workSocket;
    // 结束读取并获取读取字节数
```

```
int bytesRead = handler.EndReceive(ar);
state.sb.Append(System.Text.Encoding.UTF8.GetString(
    state.buffer,0,bytesRead));
string content = state.sb.ToString();
state.sb.Remove(0,content.Length);
string command=chenEnd(content);
getCommand="";
if(command!="")
{
    richTextBox1.AppendText(command+"\r\n");
    spliCommand(command);
    string sele1="SELECT * FROM 表1";
    sqlConnection1.Open();
    sqlSelectCommand1=new SqlCommand (sele1,sqlConnection1 );
    SqlDataReader reader1=sqlSelectCommand1.ExecuteReader();
    Exists=false;
    while(reader1.Read())
    {
        if(spliComm[0]=="1000")
        {
            int x=reader1.GetString(1).ToString().IndexOf
                (spliComm[1]);
            if(x!=-1)
            {
                sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
                    +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
                    +reader1.GetString(2)+"\r\n"+ "出版社: "
                    +reader1.GetString(3)+"\r\n"+ "责任编辑: "
                    +reader1.GetString(4)+"\r\n"+ "出版日期: "
                    +reader1.GetString(5)+"\r\n"+ "定价: "
                    +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF> ");
                Exists=true;
            }
        }
        //*****
        if(spliComm[0]=="0100")
        {
            int x=reader1.GetString(2).ToString().IndexOf
                (spliComm[1]);
            if(x!=-1)
            {
                sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
                    +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
                    +reader1.GetString(2)+"\r\n"+ "出版社: "
                    +reader1.GetString(3)+"\r\n"+ "责任编辑: "
                    +reader1.GetString(4)+"\r\n"+ "出版日期: "
                    );
```

```
+reader1.GetString(5)+"\r\n"+"定价: "
+reader1.GetDouble(6).ToString()+"\r\n"+"<EOF>");

Exists=true;
}

//*****
if(spliComm[0]=="0010")
{
    int x=reader1.GetString(3).ToString().IndexOf
    (spliComm[1]);
    if(x!=-1)
    {
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
        +"书名: "+reader1.GetString(1)+"\r\n"+"作者: "
        +reader1.GetString(2)+"\r\n"+"出版社: "
        +reader1.GetString(3)+"\r\n"+"责任者: "
        +reader1.GetString(4)+"\r\n"+"出版日期: "
        +reader1.GetString(5)+"\r\n"+"定价: "
        +reader1.GetDouble(6).ToString()+"\r\n"+"<EOF>");

        Exists=true;
    }
}

//*****
if(spliComm[0]=="0001")
{
    int x=reader1.GetString(5).ToString().IndexOf
    (spliComm[1]);
    if(x!=-1)
    {
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
        +"书名: "+reader1.GetString(1)+"\r\n"+"作者: "
        +reader1.GetString(2)+"\r\n"+"出版社: "
        +reader1.GetString(3)+"\r\n"+"责任者: "
        +reader1.GetString(4)+"\r\n"+"出版日期: "
        +reader1.GetString(5)+"\r\n"+"定价: "
        +reader1.GetDouble(6).ToString()+"\r\n"+"<EOF>");

        Exists=true;
    }
}

//*****
if(spliComm[0]=="1100")
{
    int x=reader1.GetString(1).ToString().IndexOf
    (spliComm[1]);
    int y=reader1.GetString(2).ToString().IndexOf
    (spliComm[2]);
```

```
if((x!=-1)&&(y!=-1))
{
    sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
        +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
        +reader1.GetString(2)+"\r\n"+ "出版社: "
        +reader1.GetString(3)+"\r\n"+ "责任编辑: "
        +reader1.GetString(4)+"\r\n"+ "出版日期: "
        +reader1.GetString(5)+"\r\n"+ "定价: "
        -reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF>");

    Exists=true;
}
//*****
if(spliComm[0]=="1010")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(3).ToString().IndexOf
        (spliComm[2]);
    if((x!=-1)&&(y!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
            +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
            +reader1.GetString(2)+"\r\n"+ "出版社: "
            +reader1.GetString(3)+"\r\n"+ "责任编辑: "
            +reader1.GetString(4)+"\r\n"+ "出版日期: "
            +reader1.GetString(5)+"\r\n"+ "定价: "
            +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF>");

        Exists=true;
    }
}
//*****
if(spliComm[0]=="1001")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(5).ToString().IndexOf
        (spliComm[2]);
    if((x!=-1)&&(y!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
            +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
            +reader1.GetString(2)+"\r\n"+ "出版社: "
            +reader1.GetString(3)+"\r\n"+ "责任编辑: "
            +reader1.GetString(4)+"\r\n"+ "出版日期: "
            +reader1.GetString(5)+"\r\n"+ "定价: "
        );
    }
}
```

```
        +reader1.GetDouble(6).ToString() +"\r\n"+<EOF>");  
    Exists=true;  
}  
}  
//*****  
if(spliComm[0]=="0110")  
{  
    int x=reader1.GetString(2).ToString().IndexOf  
        (spliComm[1]);  
    int y=reader1.GetString(3).ToString().IndexOf  
        (spliComm[2]);  
    if((x!=-1)&&(y!=-1))  
    {  
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"  
            +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "  
            +reader1.GetString(2)+"\r\n"+ "出版社: "  
            +reader1.GetString(3)+"\r\n"+ "责任编辑: "  
            +reader1.GetString(4)+"\r\n"+ "出版日期: "  
            +reader1.GetString(5)+"\r\n"+ "定价: "  
            +reader1.GetDouble(6).ToString() +"\r\n"+<EOF>");  
        Exists=true;  
    }  
}  
//*****  
if(spliComm[0]=="0101")  
{  
    int x=reader1.GetString(2).ToString().IndexOf  
        (spliComm[1]);  
    int y=reader1.GetString(5).ToString().IndexOf  
        (spliComm[2]);  
    if((x!=-1)&&(y!=-1))  
    {  
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"  
            +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "  
            +reader1.GetString(2)+"\r\n"+ "出版社: "  
            +reader1.GetString(3)+"\r\n"+ "责任编辑: "  
            +reader1.GetString(4)+"\r\n"+ "出版日期: "  
            +reader1.GetString(5)+"\r\n"+ "定价: "  
            +reader1.GetDouble(6).ToString() +"\r\n"+<EOF>");  
        Exists=true;  
    }  
}  
//*****  
if(spliComm[0]=="0011")  
{  
    int x=reader1.GetString(3).ToString().IndexOf
```

```
        (spliComm[1]);
        int y=reader1.GetString(5).ToString().IndexOf
            (spliComm[2]);
        if((x!=-1)&&(y!=-1))
        {
            sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
                +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
                +reader1.GetString(2)+"\r\n"+ "出版社: "
                +reader1.GetString(3)+"\r\n"+ "责任编辑: "
                +reader1.GetString(4)+"\r\n"+ "出版日期: "
                +reader1.GetString(5)+"\r\n"+ "定价: "
                +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF>");
            Exists=true;
        }
    }
    //*****if(spliComm[0]=="1110")
    {
        int x=reader1.GetString(1).ToString().IndexOf
            (spliComm[1]);
        int y=reader1.GetString(2).ToString().IndexOf
            (spliComm[2]);
        int z=reader1.GetString(3).ToString().IndexOf
            (spliComm[3]);
        if((x!=-1)&&(y!=-1)&&(z!=-1))
        {
            sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
                +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
                +reader1.GetString(2)+"\r\n"+ "出版社: "
                +reader1.GetString(3)+"\r\n"+ "责任编辑: "
                +reader1.GetString(4)+"\r\n"+ "出版日期: "
                +reader1.GetString(5)+"\r\n"+ "定价: "
                +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF>");
            Exists=true;
        }
    }
    //*****if(spliComm[0]=="1101")
    {
        int x=reader1.GetString(1).ToString().IndexOf
            (spliComm[1]);
        int y=reader1.GetString(2).ToString().IndexOf
            (spliComm[2]);
        int z=reader1.GetString(5).ToString().IndexOf
            (spliComm[3]);
        if((x!=-1)&&(y!=-1)&&(z!=-1))
```

```
{  
    sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"  
        +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "  
        +reader1.GetString(2)+"\r\n"+ "出版社: "  
        +reader1.GetString(3)+"\r\n"+ "责任编辑: "  
        +reader1.GetString(4)+"\r\n"+ "出版日期: "  
        +reader1.GetString(5)+"\r\n"+ "定价: "  
        +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF>");  
    Exists=true;  
}  
//*****  
if(spliComm[0]=="1011")  
{  
    int x=reader1.GetString(1).ToString().IndexOf  
        (spliComm[1]);  
    int y=reader1.GetString(3).ToString().IndexOf  
        (spliComm[2]);  
    int z=reader1.GetString(5).ToString().IndexOf  
        (spliComm[3]);  
    if((x!=-1)&&(y!=-1)&&(z!=-1))  
{  
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"  
            +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "  
            +reader1.GetString(2)+"\r\n"+ "出版社: "  
            +reader1.GetString(3)+"\r\n"+ "责任编辑: "  
            +reader1.GetString(4)+"\r\n"+ "出版日期: "  
            +reader1.GetString(5)+"\r\n"+ "定价: "  
            +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF>");  
        Exists=true;  
    }  
}  
//*****  
if(spliComm[0]=="0111")  
{  
    int x=reader1.GetString(2).ToString().IndexOf  
        (spliComm[1]);  
    int y=reader1.GetString(3).ToString().IndexOf  
        (spliComm[2]);  
    int z=reader1.GetString(5).ToString().IndexOf  
        (spliComm[3]);  
    if((x!=-1)&&(y!=-1)&&(z!=-1))  
{  
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"  
            +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "  
            +reader1.GetString(2)+"\r\n"+ "出版社: "  
            +reader1.GetString(3)+"\r\n"+ "责任编辑: "  
            +reader1.GetString(4)+"\r\n"+ "出版日期: "  
            +reader1.GetString(5)+"\r\n"+ "定价: "  
            +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF>");  
        Exists=true;  
    }  
}
```

```
        +reader1.GetString(3)+"\r\n"+ "责任者: "
        +reader1.GetString(4)+"\r\n"+ "出版日期: "
        +reader1.GetString(5)+"\r\n"+ "定价: "
        +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF> ");
    Exists=true;
}
}
//*****
if(spliComm[0]=="1111")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(2).ToString().IndexOf
        (spliComm[2]);
    int z=reader1.GetString(3).ToString().IndexOf
        (spliComm[3]);
    int m=reader1.GetString(5).ToString().IndexOf
        (spliComm[4]);
    if((x!=-1)&&(y!=-1)&&(z!=-1)&&(m!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)+"\r\n"
            +"书名: "+reader1.GetString(1)+"\r\n"+ "作者: "
            +reader1.GetString(2)+"\r\n"+ "出版社: "
            +reader1.GetString(3)+"\r\n"+ "责任者: "
            +reader1.GetString(4)+"\r\n"+ "出版日期: "
            +reader1.GetString(5)+"\r\n"+ "定价: "
            +reader1.GetDouble(6).ToString()+"\r\n"+ "<EOF> ");
        Exists=true;
    }
}
if(Exists==false)
{
    sendMessage("<ER> not exists!"+ "<EOF> ");
}
reader1.Close();
sqlConnection1.Close();
}
// 重新开始读取数据
tt.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
    new AsyncCallback(ReadCallback), state);
}
```

10.1.4 服务器开发

下面要新加一个应用程序项目，图 10-2 是本例的界面。其中“服务器名称”文本框为 textBox1，“监听端口”文本框为 textBox2，“服务器状态”文本框为 textBox3，“接收信息”文本框为 richTextBox1；“开始监听”按钮为 button1，“停止监听”按钮为 button2。

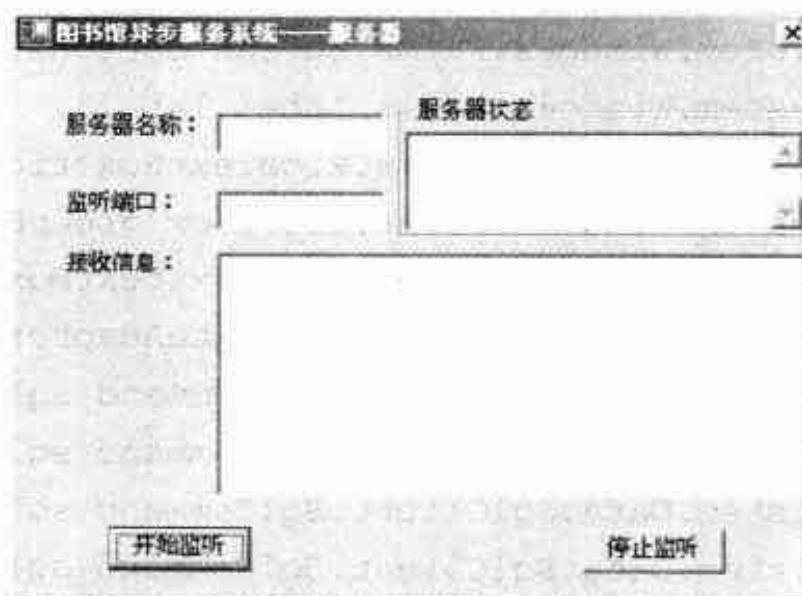


图 10-2 服务器界面

首先建立与数据库——“图书馆”的连接，然后参考下列代码开发异步服务器。

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;//新加的
using System.Net.Sockets;//新加的
using System.Threading;//新加的
using System.Data.SqlClient;
namespace Socket_Bin
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    ///
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox textBox2;
        private IPAddress myIP=IPAddress.Parse("127.0.0.1");//新加的
        private IPEndPoint MyServer;//新加的
```

```
private Socket mySocket;
private Socket handler;
private string getCommand="";
private string[] spliComm;
private bool Exists=false;
private static ManualResetEvent myReset =
    new ManualResetEvent(false);
private System.Windows.Forms.Button button2;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.RichTextBox richTextBox1;
private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.TextBox textBox3;
private System.Data.SqlClient.SqlDataAdapter sqlDataAdapter1;
private System.Data.SqlClient.SqlCommand sqlSelectCommand1;
private System.Data.SqlClient.SqlCommand sqlInsertCommand1;
private System.Data.SqlClient.SqlCommand sqlUpdateCommand1;
private System.Data.SqlClient.SqlCommand sqlDeleteCommand1;
private System.Data.SqlClient.SqlConnection sqlConnection1;
private Socket_Bind__.books books1;

/// <summary>
/// Required designer variable.
/// </summary>
private System.ComponentModel.Container components = null;
public Form1()
{
    //
    // Required for Windows Form Designer support
    //
    InitializeComponent();
    //
    // TODO: Add any constructor code after InitializeComponent call
    //
}
/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
}
```

```
base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        IPHostEntry myHost=new IPHostEntry();
        myHost=Dns.GetHostByName(textBox1.Text);
        string IPstring=myHost.AddressList[0].ToString();
        myIP = IPAddress.Parse(IPstring);

    }
    catch{MessageBox.Show("您输入的 IP 地址格式不正确, 请重新输入!");}
    try
    {
        MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
        mySocket =new Socket(AddressFamily.InterNetwork,
            SocketType.Stream,ProtocolType.Tcp);
        mySocket.Bind(MyServer);
        mySocket.Listen(50);
        textBox3.AppendText("主机"+textBox1.Text+"端口"
            +textBox2.Text+"开始监听.....\r\n");
        Thread thread=new Thread(new ThreadStart(target));
        thread.Start();
    }
    catch(Exception ee){textBox3.AppendText(ee.Message+"\r\n");}
}

private void target()
{
    while(true)
    { //设为非终止
        myReset.Reset();
        mySocket.BeginAccept( new AsyncCallback(AcceptCallback),
            null);
    }
}
```

```
        mySocket );
    //阻塞当前线程，直到收到请求信号
    myReset.WaitOne();
}
}

private void AcceptCallback(IAsyncResult ar)
{
    // 将事件设为终止
    myReset.Set();
    Socket listener = (Socket) ar.AsyncState;
    handler = listener.EndAccept(ar);
    //初始化
    getCommand="";
    // 获取状态
    StateObject state = new StateObject();
    state.workSocket = handler;
    textBox3.AppendText("与客户建立连接。 \r\n");
    try
    {
        byte[] byteData = System.Text.Encoding.UTF8.GetBytes
            ("已经准备好，请通话！"+ "\r\n<EOF>");
        handler.BeginSend(byteData, 0, byteData.Length, 0,
            new AsyncCallback(SendCallback), handler);
    }
    catch(Exception ee){MessageBox.Show(ee.Message);}
    Thread thread=new Thread(new ThreadStart(begReceive));
    thread.Start();
}
private void begReceive()
{
    StateObject state = new StateObject();
    state.workSocket = handler;
    handler.BeginReceive(state.buffer,0,StateObject.BufferSize,0,
        new AsyncCallback(ReadCallback), state);
}
private void ReadCallback(IAsyncResult ar)
{
    StateObject state = (StateObject) ar.AsyncState;
    Socket tt = state.workSocket;
    // 结束读取并获取读取字节数
    int bytesRead = handler.EndReceive(ar);
    state.sb.Append(System.Text.Encoding.UTF8.GetString(
        state.buffer,0,bytesRead));
    string content = state.sb.ToString();
    state.sb.Remove(0,content.Length);
    string command=chenEnd(content);
}
```

```
getCommand="";
if(command!="")
{
    richTextBox1.AppendText(command+"\r\n");
    spliCommand(command);
    string sele1="SELECT * FROM 表1";
    sqlConnection1.Open();
    sqlSelectCommand1=new SqlCommand (sele1,sqlConnection1 );
    SqlDataReader reader1=sqlSelectCommand1.ExecuteReader();
    Exists=false;
    while(reader1.Read())
    {
        if(spliComm[0]=="1000")
        {
            int x=reader1.GetString(1).ToString().IndexOf
                (spliComm[1]);
            if(x!=-1)
            {
                sendMessage("ISBN: "+reader1.GetString(0)
                    +"\r\n"+书名: "+reader1.GetString(1)+"\r\n"
                    +"作者: "+reader1.GetString(2)+"\r\n"+出版社: "
                    +reader1.GetString(3)+"\r\n"+责任者: "
                    +reader1.GetString(4)+"\r\n"+出版日期: "
                    +reader1.GetString(5)+"\r\n"+定价: "
                    +reader1.GetDouble(6).ToString()+"\r\n"
                    +"<EOF>");

                Exists=true;
            }
        }
        //***** *****
        if(spliComm[0]=="0100")
        {
            int x=reader1.GetString(2).ToString().IndexOf
                (spliComm[1]);
            if(x!=-1)
            {
                sendMessage("ISBN: "+reader1.GetString(0)
                    +"\r\n"+书名: "+reader1.GetString(1)
                    +"\r\n"+作者: "+reader1.GetString(2)
                    +"\r\n"+出版社: "+reader1.GetString(3)
                    +"\r\n"+责任者: "+reader1.GetString(4)
                    +"\r\n"+出版日期: "+reader1.GetString(5)
                    +"\r\n"+定价: "
                    +reader1.GetDouble(6).ToString()
                    +"\r\n"+<EOF>");

                Exists=true;
            }
        }
    }
}
```

```
        ]
    }
//***** *****
if(spliComm[0]=="0010")
{
    int x=reader1.GetString(3).ToString().IndexOf
        (spliComm[1]);
    if(x!=-1)
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\r\n"+ "书名: "+reader1.GetString(1)
            +"\r\n"+ "作者: "+reader1.GetString(2)
            +"\r\n"+ "出版社: "+reader1.GetString(3)
            +"\r\n"+ "责任者: "+reader1.GetString(4)
            +"\r\n"+ "出版日期: "+reader1.GetString(5)
            +"\r\n"+ "定价: "
            +reader1.GetDouble(6).ToString()
            +"\r\n"+ "<EOF>");
        Exists=true;
    }
}
//*****
if(spliComm[0]=="0001")
{
    int x=reader1.GetString(5).ToString().IndexOf
        (spliComm[1]);
    if(x!=-1)
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\r\n"+ "书名: "+reader1.GetString(1)
            +"\r\n"+ "作者: "+reader1.GetString(2)
            +"\r\n"+ "出版社: "+reader1.GetString(3)
            +"\r\n"+ "责任者: "+reader1.GetString(4)
            +"\r\n"+ "出版日期: "+reader1.GetString(5)
            +"\r\n"+ "定价: "
            +reader1.GetDouble(6).ToString()
            +"\r\n"+ "<EOF>");
        Exists=true;
    }
}
//*****
if(spliComm[0]=="1100")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(2).ToString().IndexOf
```

```
(spliComm[2]);
if((x!=-1)&&(y!=-1))
{
    sendMessage("ISBN: "+reader1.GetString(0)
        +"\r\n"+ "书名: "+reader1.GetString(1)+
        "\r\n"+ "作者: "+reader1.GetString(2)
        +"\r\n"+ "出版社: "+reader1.GetString(3)
        +"\r\n"+ "责任者: "
        +reader1.GetString(4)+"\r\n"+ "出版日期: "
        +reader1.GetString(5)+"\r\n"+ "定价: "
        +reader1.GetDouble(6).ToString()
        +"\r\n"+ "<EOF>");
    Exists=true;
}
//*****
if(spliComm[0]=="1010")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(3).ToString().IndexOf
        (spliComm[2]);
    if((x!=-1)&&(y!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0) +
            "\r\n"+ "书名: "+reader1.GetString(1)
            +"\r\n"+ "作者: "+reader1.GetString(2)
            +"\r\n"+ "出版社: "+reader1.GetString(3)
            +"\r\n"+ "责任者: "+reader1.GetString(4)
            +"\r\n"+ "出版日期: "+reader1.GetString(5)
            +"\r\n"+ "定价: "
            +reader1.GetDouble(6).ToString()
            +"\r\n"+ "<EOF>");
        Exists=true;
    }
}
//*****
if(spliComm[0]=="1001")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(5).ToString().IndexOf
        (spliComm[2]);
    if((x!=-1)&&(y!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)
```

```
        +"\r\n"+ "书名: "+reader1.GetString(1)
        +"\r\n"+ "作者: "+reader1.GetString(2)
        +"\r\n"+ "出版社: "+reader1.GetString(3)
        +"\r\n"+ "责任者: "+reader1.GetString(4)
        +"\r\n"+ "出版日期: "+reader1.GetString(5)
        +"\r\n"+ "定价: "
        +reader1.GetDouble(6).ToString()
        +"\r\n"+ "<EOF>");

    Exists=true;
}

//*****
if(spliComm[0]=="0110")
{
    int x=reader1.GetString(2).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(3).ToString().IndexOf
        (spliComm[2]);
    if((x!=-1)&&(y!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\r\n"+ "书名: "+reader1.GetString(1)
            +"\r\n"+ "作者: "+reader1.GetString(2)
            +"\r\n"+ "出版社: "+reader1.GetString(3)
            +"\r\n"+ "责任者: "+reader1.GetString(4)
            +"\r\n"+ "出版日期: "+reader1.GetString(5)
            +"\r\n"+ "定价: "
            +reader1.GetDouble(6).ToString()
            +"\r\n"+ "<EOF>");

        Exists=true;
    }
}

//*****
if(spliComm[0]=="0101")
{
    int x=reader1.GetString(2).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(5).ToString().IndexOf
        (spliComm[2]);
    if((x!=-1)&&(y!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\r\n"+ "书名: "+reader1.GetString(1)
            +"\r\n"+ "作者: "+reader1.GetString(2)
            +"\r\n"+ "出版社: "+reader1.GetString(3)
            +"\r\n"+ "责任者: "+reader1.GetString(4)
```

```
        +"\\r\\n"+ "出版日期: "+reader1.GetString(5)
        +"\\r\\n"+ "定价: "
        +reader1.GetDouble(6).ToString()
        +"\\r\\n"+ "<EOF>"));
    Exists=true;
}
}
//*****
if(spliComm[0]=="0011")
{
    int x=reader1.GetString(3).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(5).ToString().IndexOf
        (spliComm[2]);
    if((x!=-1)&&(y!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\\r\\n"+ "书名: "+reader1.GetString(1)
            +"\\r\\n"+ "作者: "+reader1.GetString(2)
            +"\\r\\n"+ "出版社: "+reader1.GetString(3)
            +"\\r\\n"+ "责任者: "+reader1.GetString(4)
            +"\\r\\n"+ "出版日期: "+reader1.GetString(5)
            +"\\r\\n"+ "定价: "
            +reader1.GetDouble(6).ToString()
            +"\\r\\n"+ "<EOF>"));
        Exists=true;
    }
}
//*****
if(spliComm[0]=="1110")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(2).ToString().IndexOf
        (spliComm[2]);
    int z=reader1.GetString(3).ToString().IndexOf
        (spliComm[3]);
    if((x!=-1)&&(y!=-1)&&(z!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\\r\\n"+ "书名: "+reader1.GetString(1)
            +"\\r\\n"+ "作者: "+reader1.GetString(2)
            +"\\r\\n"+ "出版社: "+reader1.GetString(3)
            +"\\r\\n"+ "责任者: "+reader1.GetString(4)
            +"\\r\\n"+ "出版日期: "+reader1.GetString(5)
            +"\\r\\n"+ "定价: "
```

```
        +reader1.GetDouble(6).ToString()
        +"\\r\\n"+<EOF>");

Exists=true;
}

}

//*****if(spliComm[0]=="1101")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(2).ToString().IndexOf
        (spliComm[2]);
    int z=reader1.GetString(5).ToString().IndexOf
        (spliComm[3]);
    if((x!=-1)&&(y!=-1)&&(z!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\\r\\n"+书名: "+reader1.GetString(1)
            +"\\r\\n"+作者: "+reader1.GetString(2)
            +"\\r\\n"+出版社: "+reader1.GetString(3)
            +"\\r\\n"+责任者: "+reader1.GetString(4)
            +"\\r\\n"+出版日期: "+reader1.GetString(5)
            +"\\r\\n"+定价: "
            +reader1.GetDouble(6).ToString()
            +"\\r\\n"+<EOF>");

Exists=true;
    }
}

//*****if(spliComm[0]=="1011")
{
    int x=reader1.GetString(1).ToString().IndexOf
        (spliComm[1]);
    int y=reader1.GetString(3).ToString().IndexOf
        (spliComm[2]);
    int z=reader1.GetString(5).ToString().IndexOf
        (spliComm[3]);
    if((x!=-1)&&(y!=-1)&&(z!=-1))
    {
        sendMessage("ISBN: "+reader1.GetString(0)
            +"\\r\\n"+书名: "+reader1.GetString(1)
            +"\\r\\n"+作者: "+reader1.GetString(2)
            +"\\r\\n"+出版社: "+reader1.GetString(3)
            +"\\r\\n"+责任者: "+reader1.GetString(4)
            +"\\r\\n"+出版日期: "+reader1.GetString(5)
            +"\\r\\n"+定价: "
            +reader1.GetDouble(6).ToString()
```

```
        +"\\r\\n"+<EOF>");  
    Exists=true;  
}  
}  
//*****  
if(spliComm[0]=="0111")  
{  
    int x=reader1.GetString(2).ToString().IndexOf  
        (spliComm[1]);  
    int y=reader1.GetString(3).ToString().IndexOf  
        (spliComm[2]);  
    int z=reader1.GetString(5).ToString().IndexOf  
        (spliComm[3]);  
    if((x!=-1)&&(y!=-1)&&(z!=-1))  
    {  
        sendMessage("ISBN: "+reader1.GetString(0)  
            +"\\r\\n"+书名: "+reader1.GetString(1)  
            +"\\r\\n"+作者: "+reader1.GetString(2)  
            +"\\r\\n"+出版社: "+reader1.GetString(3)  
            +"\\r\\n"+责任者: "+reader1.GetString(4)  
            +"\\r\\n"+出版日期: "+reader1.GetString(5)  
            +"\\r\\n"+定价: "  
            +reader1.GetDouble(6).ToString()  
            +"\\r\\n"+<EOF>);  
        Exists=true;  
    }  
}  
//*****  
if(spliComm[0]=="1111")  
{  
    int x=reader1.GetString(1).ToString().IndexOf  
        (spliComm[1]);  
    int y=reader1.GetString(2).ToString().IndexOf  
        (spliComm[2]);  
    int z=reader1.GetString(3).ToString().IndexOf  
        (spliComm[3]);  
    int m=reader1.GetString(5).ToString().IndexOf  
        (spliComm[4]);  
    if((x!=-1)&&(y!=-1)&&(z!=-1)&&(m!=-1))  
    {  
        sendMessage("ISBN: "+reader1.GetString(0)  
            +"\\r\\n"+书名: "+reader1.GetString(1)  
            +"\\r\\n"+作者: "+reader1.GetString(2)  
            +"\\r\\n"+出版社: "+reader1.GetString(3)  
            +"\\r\\n"+责任者: "+reader1.GetString(4)  
            +"\\r\\n"+出版日期: "+reader1.GetString(5)  
            +"\\r\\n"+定价: "
```

```
        +reader1.GetDouble(6).ToString()
        +"\r\n"+<EOF>});
    Exists=true;
}
}
if(Exists==false)
{
    sendMessage("<ER> not exists!"+<EOF>");
}
reader1.Close();
sqlConnection1.Close();
}
// 重新开始读取数据
tt.BeginReceive(state.buffer, 0, StateObject.BufferSize, 0,
    new AsyncCallback(ReadCallback), state);
}
private void button2_Click(object sender, System.EventArgs e)
{
    try
    {
        mySocket.Close();
        textBox3.AppendText("主机"+textBox1.Text+"端口"
            +textBox2.Text+"监听停止! \r\n");
    }
    catch{MessageBox.Show("监听尚未开始，关闭无效！");}
}
private void sendMessage(string message)
{
    byte[] byteData = System.Text.Encoding.UTF8.GetBytes
        (message+"\r\n");
    //开始发送数据
    handler.BeginSend(byteData, 0, byteData.Length, 0,
        new AsyncCallback(SendCallback), handler);
}
private void SendCallback(IAsyncResult ar)
{
    try
    {
        handler = (Socket) ar.AsyncState;
        int bytesSent = handler.EndSend(ar);
    }
    catch(Exception eee)
    {
        MessageBox.Show(eee.Message);
    }
}
```

```
private string chenEnd(string aimString){  
    int x=aimString.IndexOf("\r\n");  
    if(x!=-1){  
        string subComm=aimString.Substring(0,x);  
        getCommand=getCommand+subComm;  
        return getCommand;  
    }  
    else{  
        getCommand=getCommand+aimString;  
        return "";  
    }  
}  
private void spliCommand(string aimCommand)  
{  
    char[] a=new char[]{' '};  
    spliComm=new string[5];  
    spliComm=aimCommand.Split(a);  
}  
}  
}
```

下面是自定义类 StateObject 的代码:

```
using System;  
using System.Net;  
using System.Net.Sockets;  
using System.Threading;  
using System.Text;  
namespace Socket_Bin  
{  
    /// <summary>  
    ///  
    /// </summary>  
    public class StateObject  
    {  
  
        public Socket workSocket = null;  
        public const int BufferSize = 1024;  
        public byte[] buffer = new byte[BufferSize];  
        public StringBuilder sb = new StringBuilder();  
        public StateObject()  
        {  
            //  
            // TODO: 在此处添加构造函数逻辑  
            //  
        }  
    }  
}
```

10.2 客户端开发

本节使用异步套接字技术，开发图书馆服务系统的客户端。和服务器一样，异步客户端也要检查数据库数据是否接收完毕，服务器数据以“<EOF>”结束。只要是用 String 类的 IndexOf 方法，就可以检查数据里是否存在“<EOF>”，由此可以判断数据是否接收完毕。

10.2.1 检查数据是否接收完毕

下列代码首先判断数据是否接收完毕：如果没有接收完毕，则将接受的数据存放在字符串 getData 并返回空字符串；如果接收完毕，则返回存放全部命令的字符串 getData。字符串 getData 的初始状态是空字符串，每次建立连接时，自动赋值为空。

```
private string chenEnd(string aimString)
{
    int x=aimString.IndexOf("<EOF>");
    if(x!=-1)
    {
        string subComm=aimString.Substring(0,x);
        getData=getData+subComm;
        return getData;
    }
    else
    {
        getData=getData+aimString;
        return "";
    }
}
```

10.2.2 发送命令

下列代码演示如何向服务器发送命令。

```
private void button2_Click(object sender, System.EventArgs e)
{
    string sendSele=null;
    if(checkBox1.Checked)
    {
        sendSele="1000"+ textBox4.Text+"\r\n";
    }
    if(checkBox2.Checked)
    {
        sendSele="0100"+ textBox5.Text+"\r\n";
    }
    if(checkBox3.Checked)
```

```
{  
    sendSele="0010"+ " "+textBox6.Text+"\r\n";  
}  
if(checkBox4.Checked)  
{  
    sendSele="0001"+ " "+textBox7.Text+"\r\n";  
}  
//*****  
if(checkBox1.Checked&&checkBox2.Checked)  
{  
    sendSele="1100"+ " "+textBox4.Text+" "+textBox5.Text+"\r\n";  
}  
if(checkBox1.Checked&&checkBox3.Checked)  
{  
    sendSele="1010"+ " "+textBox4.Text+" "+textBox6.Text+"\r\n";  
}  
if(checkBox1.Checked&&checkBox4.Checked)  
{  
    sendSele="1001"+ " "+textBox4.Text+" "+textBox7.Text+"\r\n";  
}  
//*****  
if(checkBox2.Checked&&checkBox3.Checked)  
{  
    sendSele="0110"+ " "+textBox5.Text+" "+textBox6.Text+"\r\n";  
}  
if(checkBox2.Checked&&checkBox4.Checked)  
{  
    sendSele="0101"+ " "+textBox5.Text+" "+textBox7.Text+"\r\n";  
}  
//*****  
if(checkBox3.Checked&&checkBox4.Checked)  
{  
    sendSele="0011"+ " "+textBox6.Text+" "+textBox7.Text+"\r\n";  
}  
//*****  
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked)  
{  
    sendSele="1110"+ " "+textBox4.Text+" "+textBox5.Text  
        +" "+textBox6.Text+"\r\n";  
}  
if(checkBox1.Checked&&checkBox2.Checked&&checkBox4.Checked)  
{  
    sendSele="1101"+ " "+textBox4.Text+" "+textBox5.Text  
        +" "+textBox7.Text+"\r\n";  
}  
//*****
```

```

if(checkBox1.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    sendSele="1011"+ " "+textBox4.Text+" "+textBox6.Text
        +" "+textBox7.Text+"\r\n";
}
//*****
if(checkBox2.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    sendSele="0111"+ " "+textBox5.Text+" "+textBox6.Text
        +" "+textBox7.Text+"\r\n";
}
//*****
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked
    &&checkBox4.Checked)
{
    sendSele="1111"+ " "+textBox4.Text+" "+textBox5.Text
        +" "+textBox6.Text+" "+textBox7.Text+"\r\n";
}
byte[] byteData = System.Text.Encoding.UTF8.GetBytes(sendSele);
mySocket.BeginSend(byteData, 0, byteData.Length, 0,
new AsyncCallback(SendCallback), mySocket);
}

```

10.2.3 接收数据

下列代码演示如何从数据库接收返回的数据。

```

private void ReceiveCallback( IAsyncResult ar )
{
    try
    {
        StateObject state = (StateObject) ar.AsyncState;
        Socket client = state.workSocket;
        int bytesRead = client.EndReceive(ar);
        state.sb.Append(System.Text.Encoding.UTF8.GetString(state.buffer,
            0,bytesRead));
        string backString=state.sb.ToString();
        state.sb.Remove(0,backString.Length);
        string feedback=chenEnd(backString);
        if(feedback!="")
        {
            richTextBox1.AppendText(feedback);
            getData="";
        }
        client.BeginReceive(state.buffer,0,StateObject.BufferSize,0,
            new AsyncCallback(ReceiveCallback), state);
    }
}

```

```
        }
    catch {}
}
```

10.2.4 客户端开发

下面要新建一个 Windows 应用程序项目，图 10-3 是该程序的界面。其中“服务器名称”文本框是 textBox1，“请求端口”文本框是 textBox2，“程序状态”文本框是 textBox3；“书名”文本框是 textBox4，“作者”文本框是 textBox5，“出版社”文本框是 textBox6，“出版日期”文本框是 textBox7；“请求连接”按钮是 button1，“查询数据”按钮是 button2，“关闭连接”按钮是 button3。

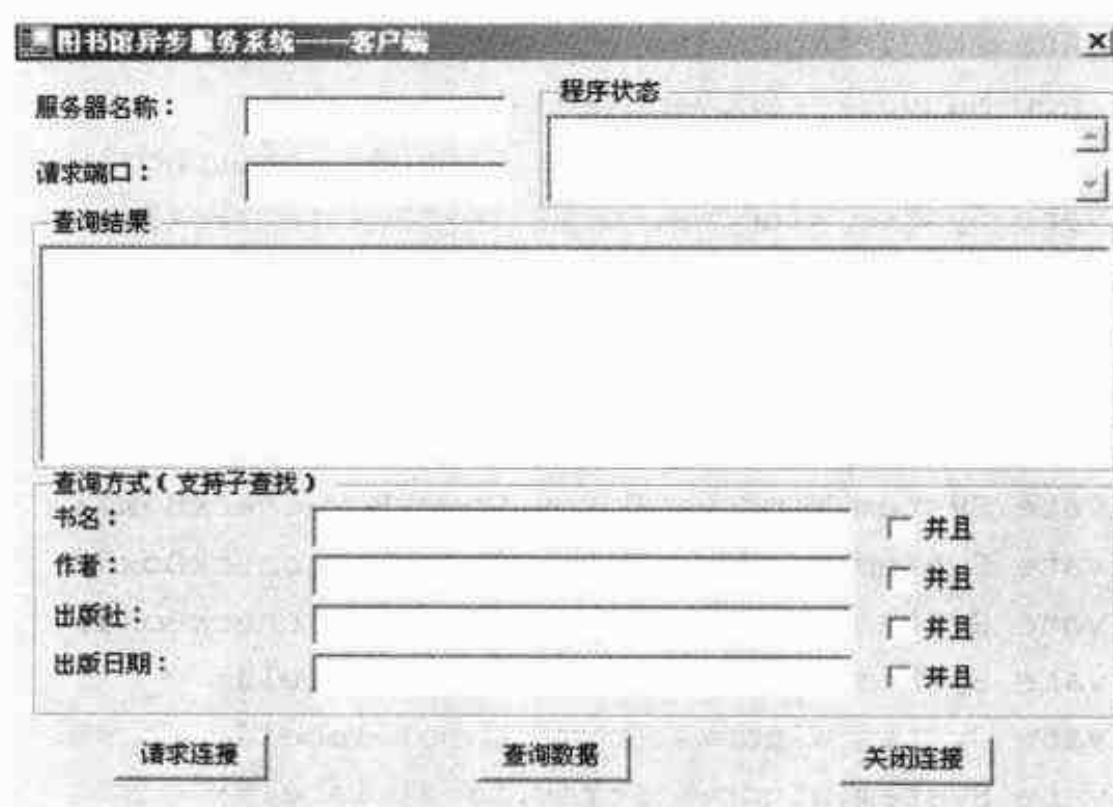


图 10-3 客户端界面

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace Sock_Conn
{
    /// <summary>
    /// Summary description for Form1.
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
```

```
{  
    private System.Windows.Forms.Label label1;  
    private System.Windows.Forms.TextBox textBox1;  
    private System.Windows.Forms.Label label2;  
    private System.Windows.Forms.Button button1;  
    private System.Windows.Forms.TextBox textBox2;  
    private IPAddress myIP=IPAddress.Parse("127.0.0.1");  
    private string getData="";  
    private IPEndPoint MyServer;  
    private Socket mySocket;  
    private static ManualResetEvent connectReset =  
        new ManualResetEvent(false);  
    private static ManualResetEvent sendReset =  
        new ManualResetEvent(false);  
    private System.Windows.Forms.GroupBox groupBox1;  
    private System.Windows.Forms.TextBox textBox3;  
    private System.Windows.Forms.GroupBox groupBox2;  
    private System.Windows.Forms.Button button3;  
    private System.Windows.Forms.Button button2;  
    private System.Windows.Forms.CheckBox checkBox1;  
    private System.Windows.Forms.CheckBox checkBox2;  
    private System.Windows.Forms.CheckBox checkBox3;  
    private System.Windows.Forms.CheckBox checkBox4;  
    private System.Windows.Forms.Label label3;  
    private System.Windows.Forms.Label label4;  
    private System.Windows.Forms.Label label5;  
    private System.Windows.Forms.Label label6;  
    private System.Windows.Forms.TextBox textBox4;  
    private System.Windows.Forms.TextBox textBox5;  
    private System.Windows.Forms.TextBox textBox6;  
    private System.Windows.Forms.TextBox textBox7;  
    private System.Windows.Forms.GroupBox groupBox3;  
    private System.Windows.Forms.RichTextBox richTextBox1;  
    /// <summary>  
    /// Required designer variable.  
    /// </summary>  
    private System.ComponentModel.Container components = null;  
  
    public Form1()  
    {  
        //  
        // Required for Windows Form Designer support  
        //  
        InitializeComponent();  
  
        //
```

```
// TODO: Add any constructor code after InitializeComponent call
//
}

/// <summary>
/// Clean up any resources being used.
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// The main entry point for the application.
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}

private void button1_Click(object sender, System.EventArgs e)
{
    try
    {
        IPHostEntry myHost=new IPHostEntry();
        myHost=Dns.GetHostByName(textBox1.Text);
        string IPstring=myHost.AddressList[0].ToString();
        myIP =IPAddress.Parse(IPstring);
    }
    catch{MessageBox.Show("您输入的 IP 地址格式不正确, 请重新输入!");}
    try
    {
        MyServer=new IPEndPoint(myIP,Int32.Parse(textBox2.Text));
        mySocket =new Socket(AddressFamily.InterNetwork,
            SocketType.Stream,ProtocolType.Tcp);
    }
}
```

```
        mySocket.BeginConnect( MyServer,
            new AsyncCallback(ConnectCallback), mySocket);
        connectReset.WaitOne();
    }
    catch(Exception ee){MessageBox.Show(ee.Message);}
}
private void ConnectCallback(IAsyncResult ar)
{
    try
    {
        Socket client = (Socket) ar.AsyncState;
        client.EndConnect(ar);
        //初始化
        getData="";
        try
        {
            byte[] byteData = System.Text.Encoding.UTF8.GetBytes
                ("准备完毕，可以通话！"+'\n\r');
            mySocket.BeginSend(byteData, 0, byteData.Length, 0,
                new AsyncCallback(SendCallback), mySocket);
        }
        catch(Exception ee){MessageBox.Show(ee.Message);}
        textBox3.AppendText("与主机"+textBox1.Text+"端口"
            +textBox2.Text+"建立连接！\r\n");
        Thread thread=new Thread(new ThreadStart(target));
        thread.Start();
        //设置事件终止
        connectReset.Set();
    }
    catch
    {
    }
}
private void button2_Click(object sender, System.EventArgs e)
{
    string sendSele=null;
    if(checkBox1.Checked)
    {
        sendSele="1000"+" "+textBox4.Text+"\r\n";
    }
    if(checkBox2.Checked)
    {
        sendSele="0100"+" "+textBox5.Text+"\r\n";
    }
    if(checkBox3.Checked)
    {
```

```
sendSele="0010"+ " "+textBox6.Text+"\r\n";
}
if(checkBox4.Checked)
{
    sendSele="0001"+ " "+textBox7.Text+"\r\n";
}
//*****
if(checkBox1.Checked&&checkBox2.Checked)
{
    sendSele="1100"+ " "+textBox4.Text
        +" "+textBox5.Text+"\r\n";
}
if(checkBox1.Checked&&checkBox3.Checked)
{
    sendSele="1010"+ " "+textBox4.Text
        +" "+textBox6.Text+"\r\n";
}
if(checkBox1.Checked&&checkBox4.Checked)
{
    sendSele="1001"+ " "+textBox4.Text+
        +textBox7.Text+"\r\n";
}
//*****
if(checkBox2.Checked&&checkBox3.Checked)
{
    sendSele="0110"+ " "+textBox5.Text+
        +textBox6.Text+"\r\n";
}
if(checkBox2.Checked&&checkBox4.Checked)
{
    sendSele="0101"+ " "+textBox5.Text+
        +textBox7.Text+"\r\n";
}
//*****
if(checkBox3.Checked&&checkBox4.Checked)
{
    sendSele="0011"+ " "+textBox6.Text+
        +textBox7.Text+"\r\n";
}
//*****
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked)
{
    sendSele="1110"+ " "+textBox4.Text+
        "+textBox5.Text
        +" "+textBox6.Text+"\r\n";
}
if(checkBox1.Checked&&checkBox2.Checked&&checkBox4.Checked)
```

```
{  
    sendSele="1101"+ " "+textBox4.Text+" "+textBox5.Text  
        +" "+textBox7.Text+"\r\n";  
}  
//*****  
if(checkBox1.Checked&&checkBox3.Checked&&checkBox4.Checked)  
{  
    sendSele="1011"+ " "+textBox4.Text+" "+textBox6.Text  
        +" "+textBox7.Text+"\r\n";  
}  
//*****  
if(checkBox2.Checked&&checkBox3.Checked&&checkBox4.Checked)  
{  
    sendSele="0111"+ " "+textBox5.Text+" "+textBox6.Text  
        +" "+textBox7.Text+"\r\n";  
}  
//*****  
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked  
    &&checkBox4.Checked)  
{  
    sendSele="1111"+ " "+textBox4.Text+" "+textBox5.Text  
        +" "+textBox6.Text+" "+textBox7.Text+"\r\n";  
}  
byte[] byteData = System.Text.Encoding.UTF8.GetBytes(sendSele);  
mySocket.BeginSend(byteData, 0, byteData.Length, 0,  
    new AsyncCallback(SendCallback), mySocket);  
}  
private void SendCallback(IAsyncResult ar)  
{  
    try  
{  
        Socket client = (Socket) ar.AsyncState;  
        //设为终止  
        sendReset.Set();  
    }  
    catch (Exception e)  
    {  
        Console.WriteLine(e.ToString());  
    }  
}  
private void target()  
{  
    try  
{  
        StateObject state = new StateObject();  
        state.workSocket = mySocket;
```

```
mySocket.BeginReceive( state.buffer, 0,
    StateObject.BufferSize, 0,
    new AsyncCallback(ReceiveCallback), state);
}
catch(Exception ee)
{
    MessageBox.Show(ee.Message);
}
}
private void ReceiveCallback( IAsyncResult ar )
{
    try
    {
        StateObject state = (StateObject) ar.AsyncState;
        Socket client = state.workSocket;
        int bytesRead = client.EndReceive(ar);
        state.sb.Append(System.Text.Encoding.UTF8.GetString
            (state.buffer,0,bytesRead));
        string backString=state.sb.ToString();
        state.sb.Remove(0,backString.Length);
        string feedback=chenEnd(backString);
        if(feedback!="")
        {
            richTextBox1.AppendText(feedback);
            getData="";
        }
        client.BeginReceive(state.buffer,
            0,StateObject.BufferSize,0,
            new AsyncCallback(ReceiveCallback), state);
    }
    catch {}
}
private string chenEnd(string aimString)
{
    int x=aimString.IndexOf("<EOF>");
    if(x!=-1)
    {
        string subComm=aimString.Substring(0,x);
        getData=getData+subComm;
        return getData;
    }
    else
    {
        getData=getData+aimString;
        return "";
    }
}
```

```
        }
        private void button3_Click(object sender, System.EventArgs e)
        {
            mySocket.Close();
        }
    }
}
```

下面是自定义类 StateObject 的代码：

```
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using System.Text;

namespace Sock_Conn
{
    /// <summary>
    ///
    /// </summary>
    public class StateObject
    {
        public Socket workSocket = null;
        public const int BufferSize = 256;
        public byte[] buffer = new byte[BufferSize];
        public StringBuilder sb = new StringBuilder();
        public StateObject()
        {
            //
            // TODO: 在此处添加构造函数逻辑
            //
        }
    }
}
```

10.3 演示

首先启动服务器，输入适当的服务器名称和端口，开始监听。然后启动客户端，并建立与服务器的连接。图 10-4 是客户端的运行情况。



图 10-4 客户端运行情况

图 10-5 是客户端的运行情况。

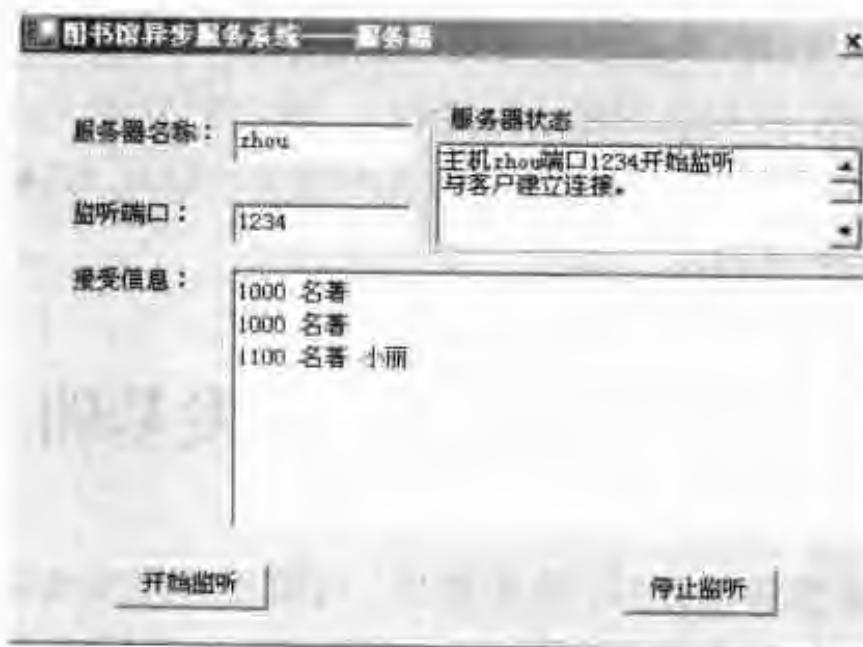


图 10-5 服务器运行情况

第 11 章 XML Web services 开发

XML Web services 是提供特定功能元素（如应用程序逻辑）的可编程实体，该实体可供任何数量的、可能是完全不同的系统用 Internet 标准（如 XML 和 HTTP）来访问它。XML Web services 既可以在内部由单个应用程序使用，也可通过 Internet 公开，以供任何数量的应用程序使用。由于可以通过标准接口访问，因此 XML Web services 使异类系统能作为单个计算网络协同运行。XML Web services 为实现数据和系统的互操作性提供了一种可行的解决方案。XML Web services 使用基于 XML 的消息处理作为基本的数据通信方式，以帮助消除使用不同服务模型、操作系统和编程语言的系统之间存在的差异。开发人员可以用像过去在创建分布式应用程序时使用服务一样的方式创建将来自各种源的 XML Web services 组合在一起的应用程序。XML Web services 的核心特征之一是服务的实现与使用之间的高度抽象化。通过将基于 XML 的消息处理用作创建和访问服务的机制，XML Web services 客户端和 XML Web services 提供程序之间除输入、输出和位置之外无需互相了解其他信息。微软曾大胆预言，作为 Internet 的下一个革命性的进步，XML Web services 将成为把所有计算设备链接到一起的基本结构。

11.1 Web 服务开发基础

本节主要学习 Web 服务的开发、使用基础，包括如何开发 Web 服务，如何在 ASP.NET 程序中使用 Web 服务，如何在 Win32 程序中使用 Web 服务等内容。

11.1.1 关于特性

11.1.1.1 特性的概念和特点

特性是提供有关编程元素（如类型、字段、方法和属性）的附加信息的描述性标记。特性主要具有以下几个特点：

- 可以将一个或多个特性应用到整个程序集、模块或较小的程序元素（如类和属性）。
- 特性可以与方法和属性相同的方式接受参数。
- 特性可以自定义。
- 自定义特性在基于 System.Attribute 类的特性类中定义。特性类自身使用 AttributeUsage 来提供有关可以如何使用特性的附加信息。

11.1.1.2 特性的用途

特性主要有以下几个用途：

- 在 XML Web services 中, WebMethod 特性标记方法, 以指示该方法应可通过 SOAP 协议进行调用。使用 WebServiceAttribute 类的 Namespace 特性标记 XML Web services 的命名空间。
- 描述当与本机代码进行交互操作时如何封送方法参数。
- 描述类、方法和接口的 COM 属性。
- 将组件标记为 COM, 以便 Visual Basic 编译器生成创建 COM 组件所需的附加代码。
- 使用 DllImportAttribute 类调用非托管代码。
- 在标题、版本、说明或商标方面描述您的程序集。
- 描述要持久性序列化类的哪些成员。
- 描述如何映射类成员和 XML 节点以便进行 XML 序列化。
- 描述方法的安全要求。
- 指定用于强制安全性的特性。
- 由实时 (JIT) 编译器控制优化, 以便调试代码。

11.1.1.3 标记 XML Web services 方法

在 Web 服务开发中, 特性非常重要。尤其是在开发服务方法时, 必须使用 WebMethod 特性进行标记, 否则该方法就不能提供服务。在 XML Web services 中, 使用 WebMethod 特性标记方法, 以指示该方法应可通过 SOAP 协议进行调用。下列代码演示如何声明标记服务方法。

```
[WebMethod]
public string UpString(string getString){
    string backString = getString.ToUpper();
    return backString;
}
```

其中 “[WebMethod]” 声明了 “UpString ()” 为 Web 服务方法。

11.1.1.4 标记 XML Web services 命名空间

另外, 对于用 ASP.NET 创建的 XML Web services, 可以使用 WebServiceAttribute 类 WebService 特性的 Namespace 属性更改默认命名空间。XML Web service 命名空间为 URI, 每个 XML Web service 都需要一个惟一的命名空间, 以便客户端应用程序能够将它与 Web 上的其他服务区分开。开发阶段的 Web 服务放在 <http://tempuri.org/> 下, 而已发布的 XML Web services 则应使用更为永久的命名空间。这时, 应使用特定的的命名空间来标识 XML Web service。下面的代码实例将命名空间设置 “<http://zhou/webservices>”。

```
[WebService(Namespace="http://zhou/webservices/")]
public class MyWebService {
    ...
    ...
}
```

WebServiceAttribute 类的成员特性及其用途如表 11-1 所示。

表 11-1 Web ServiceAttribute 类的成员特性及其用途

特 性	用 途
Description	XML Web services 的描述性消息
Name	获取或设置 XML Web services 的名称
Namespace	获取或设置用于 XML Web services 的默认 XML 命名空间
TypeId (从 Attribute 继承)	当在派生类中实现时, 获取该 Attribute 的惟一标识符

11.1.2 第一个 Web 服务开发

如图 11-1 所示, 新建一个“ASP.NET Web 服务”项目, 本例项目名称为 FirstWebService。

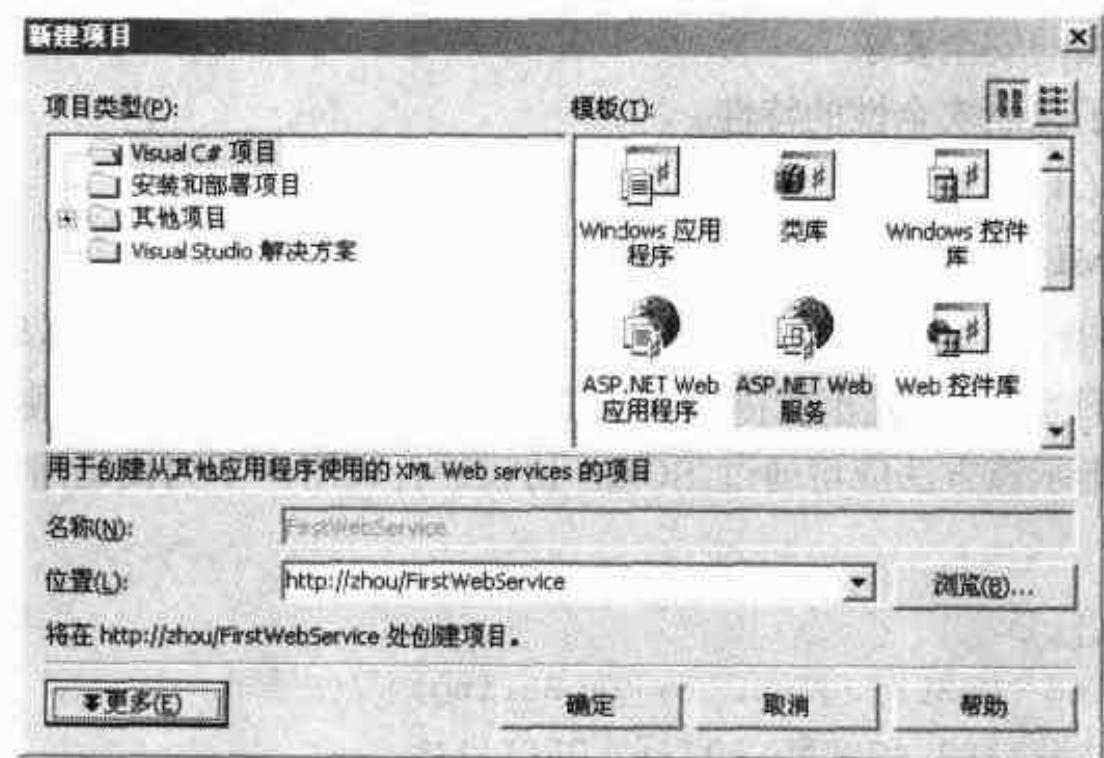


图 11-1 新建 Web 服务对话框

如图 11-2 所示, 在“解决方案资源管理器”把 Service1.asmx 修改成 FirstService.asmx。



图 11-2 修改文件名

然后进入代码编辑窗口, 系统自动产生的代码如下:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
```

```
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace FirstWebService
{
    /// <summary>
    /// Service1 的摘要说明
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
        {
            //CODEGEN: 该调用是 ASP.NET Web 服务设计器所必需的
            InitializeComponent();
        }

        #region Component Designer generated code

        //Web 服务设计器所必需的
        private IContainer components = null;

        /// <summary>
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改
        /// 此方法的内容
        /// </summary>
        private void InitializeComponent()
        {
        }

        /// <summary>
        /// 清理所有正在使用的资源
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #endregion

        // WEB 服务示例
        // HelloWorld() 示例服务返回字符串 Hello World
    }
}
```

```
// 若要生成，请取消注释下列行，然后保存并生成项目  
// 若要测试此 Web 服务，请按 F5 键  
  
// [WebMethod]  
public string HelloWorld()  
{  
    return "Hello World";  
}  
}
```

下面开发一个最简单的 Web 服务，该服务只返回一个字符串“Welcome to you!”。然后上述代码中添加如下方法：

```
[WebMethod]  
public string welcome()  
{  
    return "Welcome to you!";  
}
```

全部代码变为：

```
using System;  
using System.Collections;  
using System.ComponentModel;  
using System.Data;  
using System.Diagnostics;  
using System.Web;  
using System.Web.Services;  
  
namespace FirstWebService  
{  
    /// <summary>  
    /// Service1 的摘要说明  
    /// </summary>  
    public class Service1 : System.Web.Services.WebService  
    {  
        public Service1()  
        {  
            //CODEGEN: 该调用是 ASP.NET Web 服务设计器所必需的  
            InitializeComponent();  
        }  
        #region Component Designer generated code  
  
        //Web 服务设计器所必需的  
        private IContainer components = null;  
  
        /// <summary>  
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改  
    }
```

```
/// 此方法的内容
/// </summary>
private void InitializeComponent()
{
}

/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if(disposing && components != null)
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#endregion

// WEB 服务示例
// HelloWorld() 示例服务返回字符串 Hello World
// 若要生成，请取消注释下列行，然后保存并生成项目
// 若要测试此 Web 服务，请按 F5 键

// [WebMethod]
// public string HelloWorld()
// {
//     return "Hello World";
// }

[WebMethod]
```



图 11-3 服务页在浏览器的浏览情况

```
public string welcome()
{
    return "Welcome to you!";
}
```

单击菜单“调试”→“开始执行”，即可生成 Web 服务（服务）。如果 Web 服务生成没有错误，会出现如图 11-3 所示的页面。

为了测试 Web 服务是否正确，单击页面上的 welcome（就是上面刚开发的 Web 服务方法），会出现如图 11-4 所示的界面。

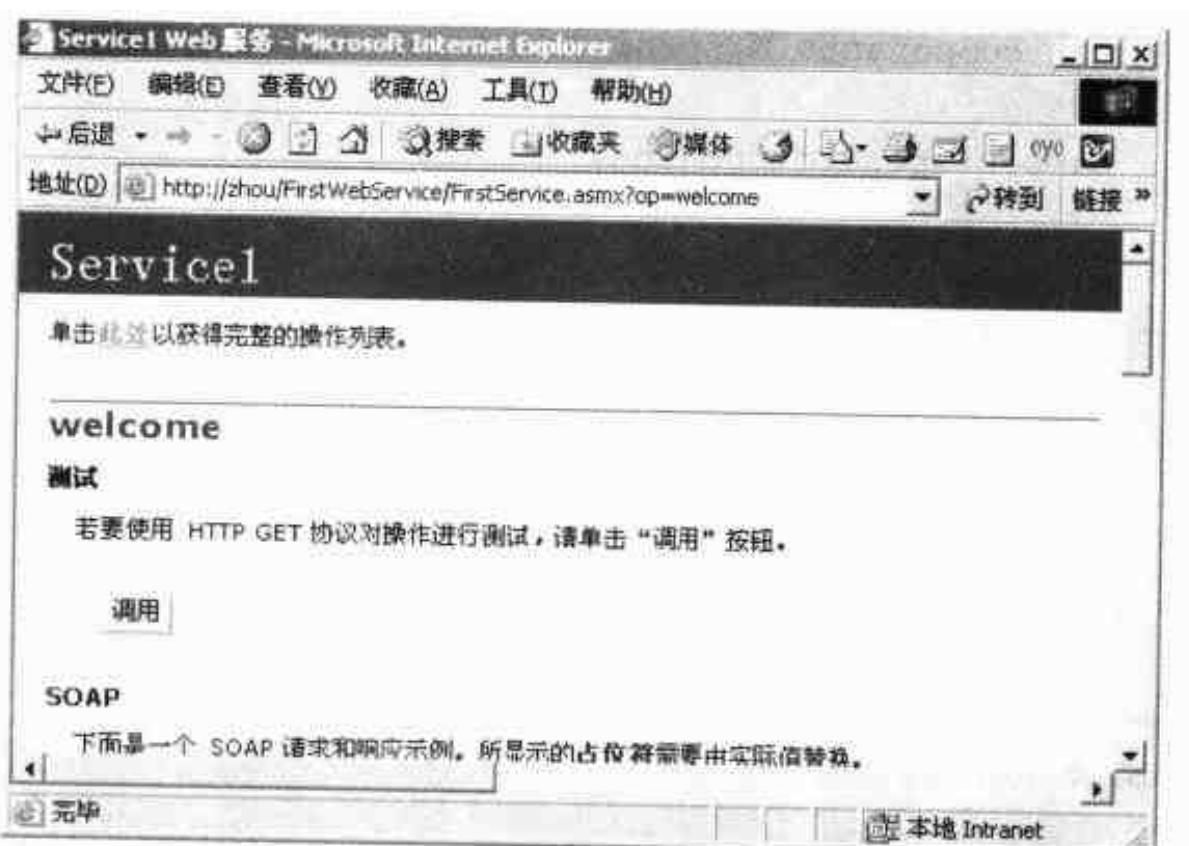
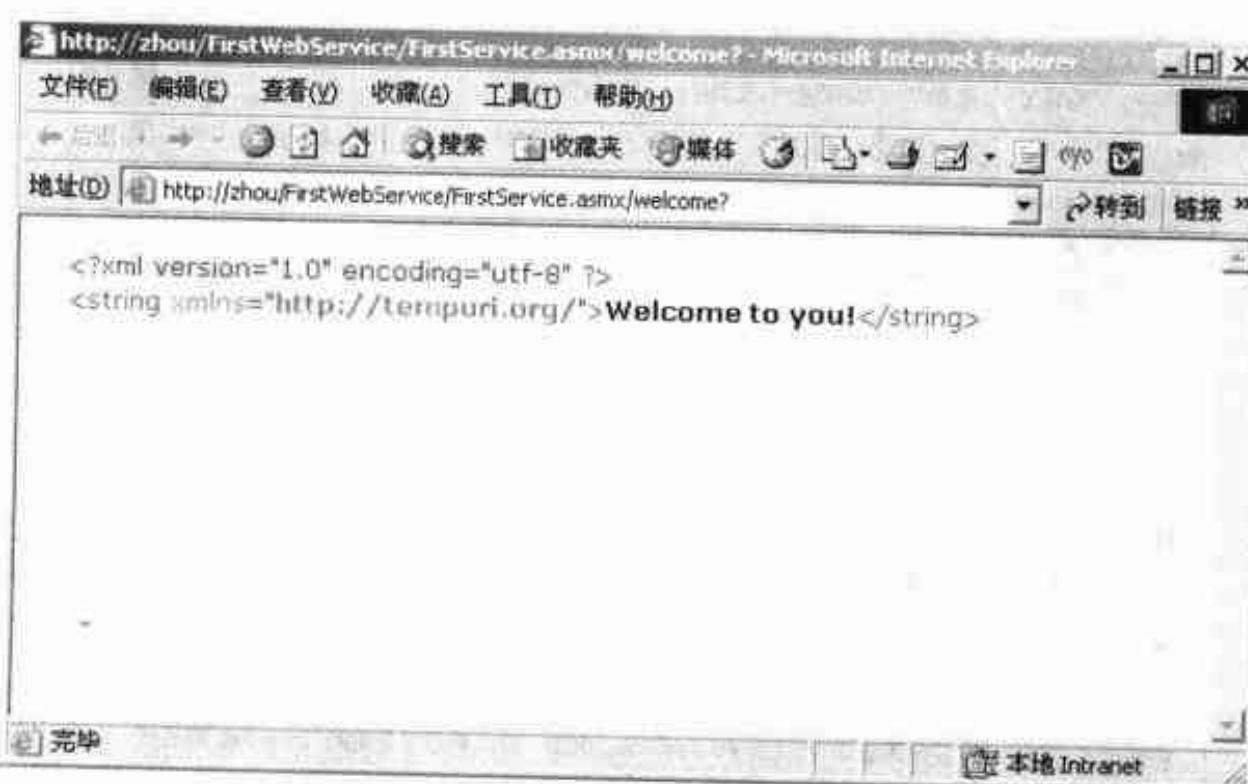


图 11-4 “调用”服务方法页面

单击“调试”按钮，测试结果如图 11-5 所示。



在上面页面中，出现了<string xmlns="http://tempuri.org/">Welcome to you!</string>，和预想值完全相同，证明 Web 服务开发正确无误。

11.1.3 Web 服务的使用

(1) 新建项目。

新建一个 ASP.NET Web 应用程序，本例的项目名称为 UseWebService。

(2) 添加 Web 引用。

单击菜单“项目”→“添加 Web 引用”，出现如图 11-6 所示的界面。

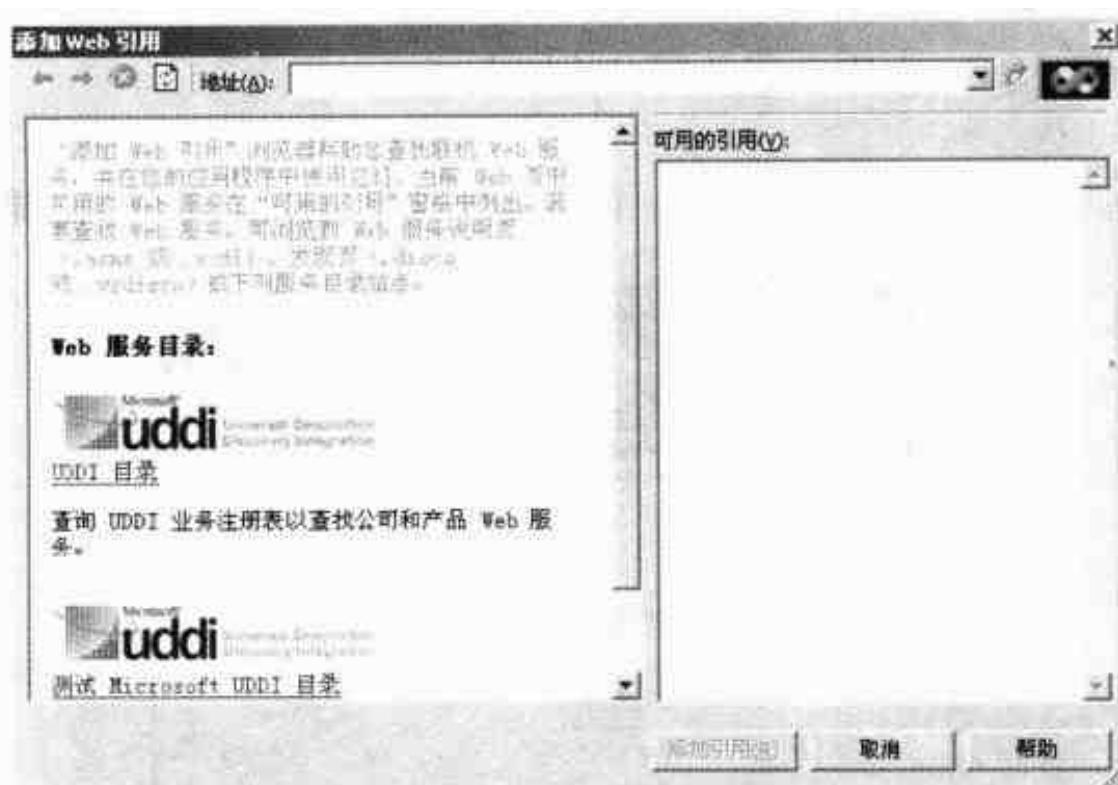


图 11-6 添加 Web 引用对话框

然后在地址栏里填写正确的 Web 服务所在的站点。如果要把上面开发的服务加进去，只要填写 http://zhou/FirstWebService/FirstService.asmx 即可，然后单击 ➔ 箭头，进入该页面。结果如图 11-7 所示。

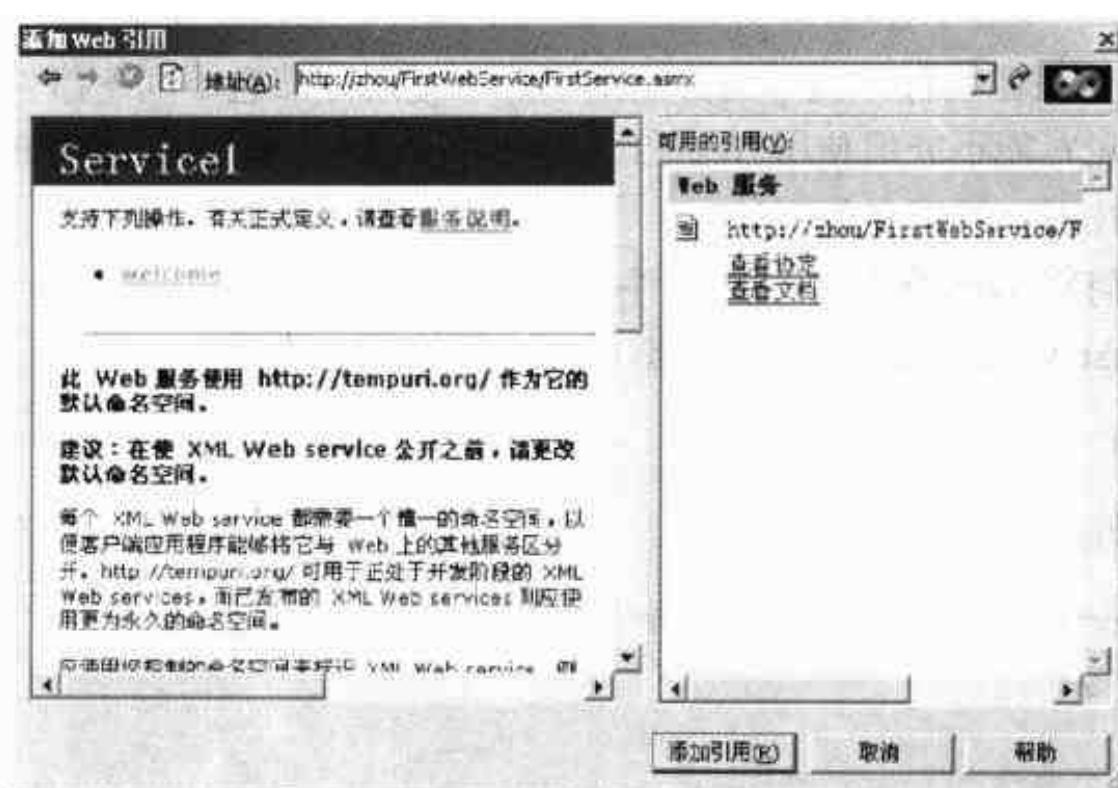


图 11-7 加入 Web 服务

然后单击“添加引用”，即可将该 Web 服务加入进来。

(3) 设计页面。

在页面上拖放一个 TextBox 控件和一个 Button 控件。

(4) Button 控件的 Click 事件。

```
private void Button1_Click(object sender, System.EventArgs e)
{
    zhou.Service1 service=new zhou.Service1();
    TextBox1.Text=service.welcome();
}
```

在上述代码中，zhou 是 Web 服务所在的站点。

(5) 演示。

单击菜单“调试”→“开始执行”，单击页面上的 Button 按钮，结果如图 11-8 所示。



图 11-8 Web 服务使用结果

11.1.4 将 Web 服务修改成组件

Web 服务虽然很像组件，但本身并不是组件，使用方法也不同于组件。能不能把 Web 服务修改成组件呢？能不能像使用 Win32 组件一样，不但可以调用组件的方法，而且还可以使用组件的属性呢？答案是肯定的。

下面演示如何将 Web 服务修改成组件，然后再使用组件。

重新回到 FirstWebService 的代码编辑窗口，添加私有成员：

```
private int x=0;
```

然后添加属性：

```
[Description("将属性 y 的值赋给 x 并返回 x")]
public int y{
    get{
        return x;
    }
    set{
```

```
    x=value;
}
}
```

上述代码的 “[Description(“将属性 y 的值赋给 x 并返回 x”)]” 是为了描述属性 y。
最后再将 welcome 方法修改成下列代码：

```
public string welcome(string aimString)
{
    if(x==0)
    {
        return aimString;
    }
    else
    {
        aimString=aimString.ToUpper();
        return aimString;
    }
}
```

该组件的全部代码如下：

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;

namespace FirstWebService
{
    /// <summary>
    /// Service1 的摘要说明
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        private int x=0;
        public Service1()
        {
            //CODEGEN: 该调用是 ASP.NET Web 服务设计器所必需的
            InitializeComponent();
        }
        #region Component Designer generated code
        //Web 服务设计器所必需的
        private IContainer components = null;
        /// <summary>
```

```
/// 设计器支持所需的方法 - 不要使用代码编辑器修改
/// 此方法的内容
/// </summary>
private void InitializeComponent()
{
}

/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if(disposing && components != null)
    {
        components.Dispose();
    }
    base.Dispose(disposing);
}

#endregion
// WEB 服务示例
// HelloWorld() 示例服务返回字符串 Hello World
// 若要生成，请取消注释下列行，然后保存并生成项目
// 若要测试此 Web 服务，请按 F5 键
//[WebMethod]
//public string HelloWorld()
//{
//    return "Hello World";
//}

[Description("将属性 y 的值赋给 x 并返回 x")]
public int y{
    get{
        return x;
    }
    set{
        x=value;
    }
}

//[WebMethod]
public string welcome(string aimString){
    if(x==0)
    {
        return aimString;
    }
    else{
        aimString=aimString.ToUpper();
        return aimString;
    }
}
```

```

    }
}

}

```

单击菜单“生成”→“生成”，即可生成组件 FirstWebService.dll。下面演示该组件的使用。

(1) 回到 UseWebService 项目页面设计窗口，按图 11-9 所示设计页面。

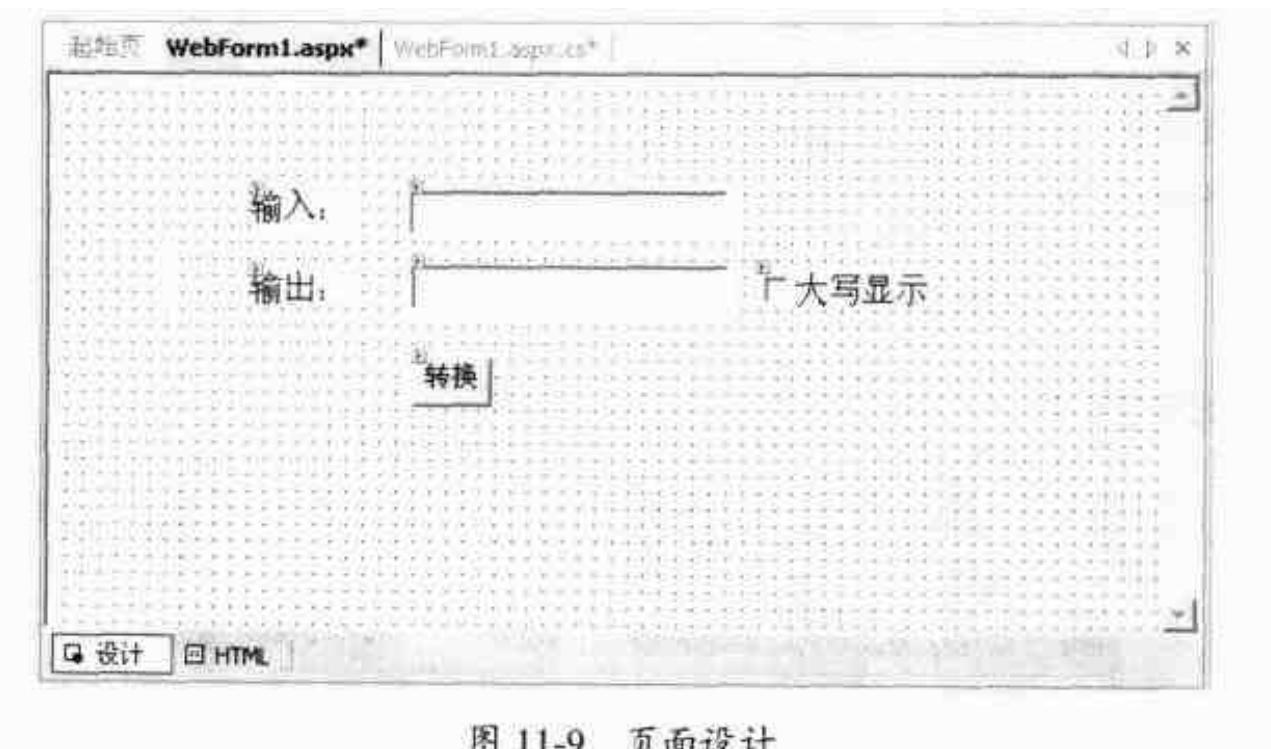


图 11-9 页面设计

(2) 如图 11-10 所示，单击菜单“项目”→“添加引用”，将 FirstWebService.dll 加入进去。

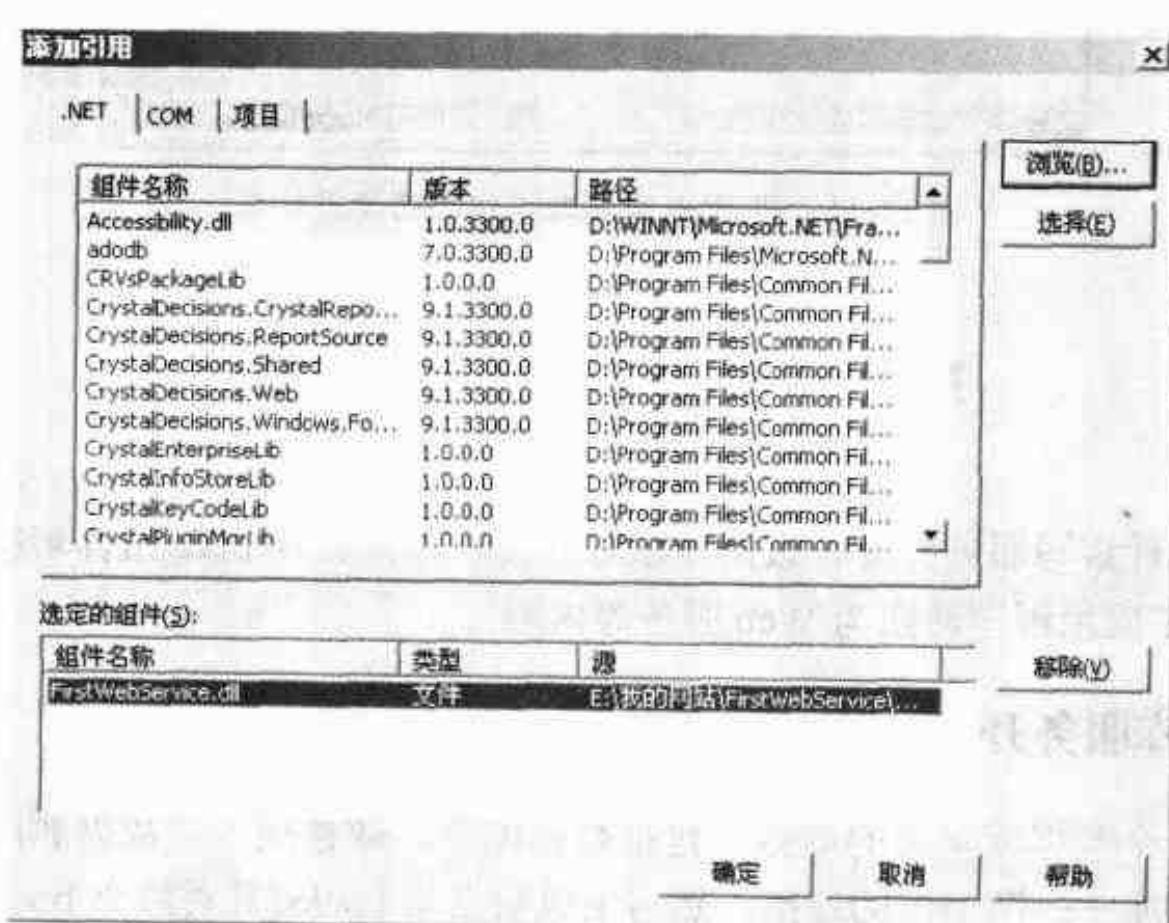


图 11-10 添加引用对话框

然后添加下列代码：

```
using FirstWebService;
```

(3) “转换”按钮的 Click 事件的代码。

```

private void Button1_Click(object sender, System.EventArgs e)
{
    FirstWebService.Service1 newService=new FirstWebService.Service1();
    int x=0;
    if(CheckBox1.Checked)
    {
        x=1;
    }
    newService.y=x;
    zhou.Service1 service=new zhou.Service1();
    TextBox2.Text=newService.welcome(TextBox1.Text);
}

```

(4) 演示。

图 11-11 是执行结果。



图 11-11 将 Web 服务改成组件的使用结果

11.2 Web 服务高级开发

本节讲解文件读写服务开发、数据库服务开发、如何将 Win32 组件转换为 Web 服务、如何将 ASP.NET 应用程序转换为 Web 服务等内容。

11.2.1 数据库服务开发

下面开发一个数据库常用的服务，包括数据浏览、数据插入、数据删除、数据更新。首先建立与数据库——图书馆的连接，然后生成数据集 books(其对象为 book1)。

1. 数据浏览服务

```

[WebMethod]
public DataSet getDataSet()
{
    sqLSelectCommand1.CommandText="SELECT * FROM 表 1";
}

```

```
    sqlDataAdapter1.Fill(books1.表1);
    return books1;
}
```

上述方法返回 DataSet 类型值，即数据集 books1。

2. 数据插入服务

```
[WebMethod]
public void insert(string command)
{
    books1.AcceptChanges();
    sqlConnection1.Open();
    sqlInsertCommand1=new SqlCommand(command, sqlConnection1);
    sqlInsertCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}
```

参数是完成数据插入的 SQL 语句。

3. 数据删除服务

```
[WebMethod]
public void delete(string command)
{
    books1.AcceptChanges();
    sqlConnection1.Open();
    sqlDeleteCommand1=new SqlCommand(command, sqlConnection1);
    sqlDeleteCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}
```

4. 数据更新服务

```
[WebMethod]
public void update(string command)
{
    books1.AcceptChanges();
    sqlConnection1.Open();
    sqlUpdateCommand1=new SqlCommand(command, sqlConnection1);
    sqlUpdateCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}
```

下面是该服务的关键代码：

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
```

```
using System.Web.Services;
using System.IO;
using System.Data.SqlClient;

namespace WebServiceImprove
{
    /// <summary>
    /// Service1 的摘要说明
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        private int x=0;
        public Service1()
        {
            //CODEGEN: 该调用是 ASP.NET Web 服务设计器所必需的
            InitializeComponent();
        }

        private System.Data.SqlClient.SqlDataAdapter sqlDataAdapter1;
        private System.Data.SqlClient.SqlCommand sqlSelectCommand1;
        private System.Data.SqlClient.SqlCommand sqlInsertCommand1;
        private System.Data.SqlClient.SqlCommand sqlUpdateCommand1;
        private System.Data.SqlClient.SqlCommand sqlDeleteCommand1;
        private System.Data.SqlClient.SqlConnection sqlConnection1;
        private WebServiceImprove.books books1;
        #region Component Designer generated code

        //Web 服务设计器所必需的
        private IContainer components = null;

        /// <summary>
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改
        /// 此方法的内容。
        /// </summary>
        /// <summary>
        /// 清理所有正在使用的资源
        /// </summary>
        protected override void Dispose( bool disposing )
        {
            if(disposing && components != null)
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #endregion
        // WEB 服务示例
```

```
// HelloWorld() 示例服务返回字符串 Hello World
// 若要生成，请取消注释下列行，然后保存并生成项目
// 若要测试此 Web 服务，请按 F5 键
//[WebMethod]
//public string HelloWorld()
//{
//    return "Hello World";
//}
[WebMethod]
public DataSet getDataSet(){
    sqlSelectCommand1.CommandText="SELECT * FROM 表 1";
    sqlDataAdapter1.Fill(books1.表 1);
    return books1;
}
[WebMethod]
public void delete(string command){
    books1.AcceptChanges();
    sqlConnection1.Open();
    sqlDeleteCommand1=new SqlCommand(command, sqlConnection1);
    sqlDeleteCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}
[WebMethod]
public void insert(string command)
{books1.AcceptChanges();
    sqlConnection1.Open();
    sqlInsertCommand1=new SqlCommand(command, sqlConnection1);
    sqlInsertCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}
[WebMethod]
public void update(string command)
{books1.AcceptChanges();
    sqlConnection1.Open();
    sqlUpdateCommand1=new SqlCommand(command, sqlConnection1);
    sqlUpdateCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}
}
```

该服务使用比较方便，在数据插入、删除、更新时，客户端只要输入特定的 SQL 语句作为参数即可。

11.2.2 如何将 Win32 组件替换为 Web 服务

一般说，将 Win32 组件转换为 Web 服务是一件很方便的事，但并不是所有的 Win32 组件均可转换为 Web 服务。根据 Web 服务的工作原理，如果特定的类型对象不能够序列化，就无法提供 Web 服务。但相当多的 Win32 组件经过处理可以转换成 Web 服务，这样可节省大量的劳动。下面是一个简单的 Win32 组件，该组件具有一般组件的基本特征。

```
using System;
using System.IO;
namespace OpenWriteText
{
    /// <summary>
    /// Class1 的摘要说明
    /// </summary>
    public class Class1
    {
        private string encoding;
        public Class1()
        {
            // 
            // TODO：在此处添加构造函数逻辑
            //
        }
        public string getencoding{
            get{
                return encoding;
            }
            set{
                encoding=value;
            }
        }
        public string openText(string path){
            FileStream readFile=File.OpenRead(path);
            StreamReader readText=null;
            if(encoding=="UTF8")
            {
                readText=new StreamReader(readFile,
                    System.Text.Encoding.UTF8);
            }
            else{
                readText=new StreamReader(readFile,
                    System.Text.Encoding.ASCII);
            }
            string getString=readText.ReadToEnd();
            readText.Close();
        }
    }
}
```

```
    readFile.Close();
    return getString;
}
public void writeText(string path, string text){
    FileStream writeFile=File.OpenWrite(path);
    StreamWriter writeText=null;
    if(encoding=="UTF8")
    {
        writeText=new StreamWriter(writeFile,
            System.Text.Encoding.UTF8);
    }
    else{
        writeText=new StreamWriter(writeFile,
            System.Text.Encoding.ASCII);
    }
    writeText.Write(text);
    writeText.Close();
    writeFile.Close();
}
}
```

上述组件有一个私有字符串全局变量：

```
private string encoding;
```

有一个公开属性，该属性用于设置和获取字符串 encoding 的值。

```
public string getencoding
{
    get
    {
        return encoding;
    }
    set
    {
        encoding=value;
    }
}
```

有两个公开方法，第一个用于读其文件文本，第二个方法用于向文件写文本。如果 encoding 的值是“UTF8”，就用 UTF8 码对文本进行编码；否则用 ASCII 码对文本进行编码。

```
public string openText(string path)
{
    FileStream readFile=File.OpenRead(path);
    StreamReader readText=null;
```

```
if(encoding=="UTF8")
{
    readText=new StreamReader(readFile,System.Text.Encoding.UTF8);
}
else{
    readText=new StreamReader(readFile,System.Text.Encoding.ASCII);
}
string getString=readText.ReadToEnd();
readText.Close();
readFile.Close();
return getString;
}

public void writeText(string path,string text)
{
    FileStream writeFile=File.OpenWrite(path);
    StreamWriter writeText=null;
    if(encoding=="UTF8")
    {
        writeText=new StreamWriter(writeFile,System.Text.Encoding.UTF8);
    }
    else{
        writeText=new StreamWriter(writeFile,System.Text.Encoding.ASCII);
    }
    writeText.Write(text);
    writeText.Close();
    writeFile.Close();
}
```

由于在 Web 服务中，字定义属性在客户端是不可见的，声明的全局变量也是不可见的（无论是公开的还是私有的），所以在转换 Win32 组件时，应尽量将全局变量和属性删除，将其修改为 Web 服务方法的参数。另外，Web 服务的公开方法一定要用特性声明。上述 Win32 组件如果修改成以下代码，就可以是一个标准的 Web 服务。

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.IO;
namespace OpWrTextService
{
    /// <summary>
    /// Service1 的摘要说明
    /// </summary>
    public class Service1 : System.Web.Services.WebService
```

```
{  
    public Service1()  
    {  
        //CODEGEN: 该调用是 ASP.NET Web 服务设计器所必需的  
        InitializeComponent();  
    }  
    #region Component Designer generated code  
    //Web 服务设计器所必需的  
    private IContainer components = null;  
    /// <summary>  
    /// 设计器支持所需的方法 - 不要使用代码编辑器修改  
    /// 此方法的内容。  
    /// </summary>  
    private void InitializeComponent()  
    {  
    }  
    /// <summary>  
    /// 清理所有正在使用的资源  
    /// </summary>  
    protected override void Dispose( bool disposing )  
    {  
        if(disposing && components != null)  
        {  
            components.Dispose();  
        }  
        base.Dispose(disposing);  
    }  
    #endregion  
    // WEB 服务示例  
    // HelloWorld() 示例服务返回字符串 Hello World  
    // 若要生成，请取消注释下列行，然后保存并生成项目  
    // 若要测试此 Web 服务，请按 F5 键  
    // [WebMethod]  
    //public string HelloWorld()  
    //{
/>    //    return "Hello World";  
    //}  
    [WebMethod]  
    public string openText(string path,string encoding)  
    {  
        FileStream readFile=File.OpenRead(path);  
        StreamReader readText=null;  
        if(encoding=="UTF8")  
        {  
            readText=new StreamReader(readFile,  
                System.Text.Encoding.UTF8);  
        }  
    }
```

```
else
{
    readText=new StreamReader(readFile,
        System.Text.Encoding.ASCII);
}
string getString=readText.ReadToEnd();
readText.Close();
readFile.Close();
return getString;
}
[WebMethod]
public void writeText(string path,string text,string encoding)
{
    FileStream writeFile=File.OpenWrite(path);
    StreamWriter writeText=null;
    if(encoding=="UTF8")
    {
        writeText=new StreamWriter(writeFile,
            System.Text.Encoding.UTF8);
    }
    else
    {
        writeText=new StreamWriter(writeFile,
            System.Text.Encoding.ASCII);
    }
    writeText.Write(text);
    writeText.Close();
    writeFile.Close();
}
}
```

下面测试一下该服务，看是否能正确工作。单击菜单“调试”→“开始执行”，结果如图 11-12 所示。

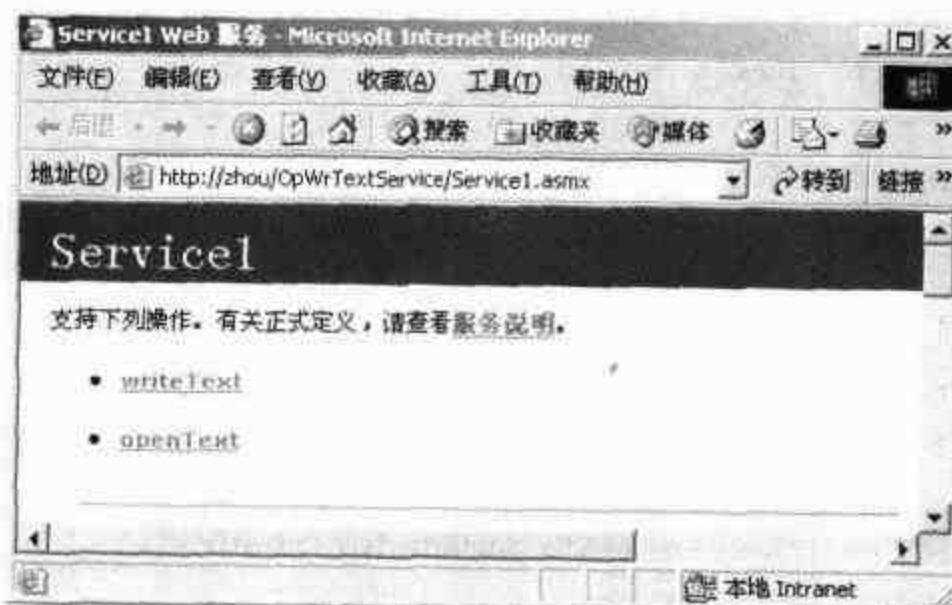


图 11-12 将 Win32 组件转化成 Web 服务

单击“页面”上的“openText”方法，进入下一个页面，如图 11-13 所示，在页面上输入文本文件按路径的编码方式。在笔者的 e:\aa.txt 里，存放着一段用 UTF8 编码方式存放的文本：“你好，现在测试从 Win32 转换过来的 Web 服务，如果代码可以识别，证明正确无误。”，单击“调用”按钮，如果下一个页面显示该文本，证明从 Win32 组件转换到 Web 服务正确无误。

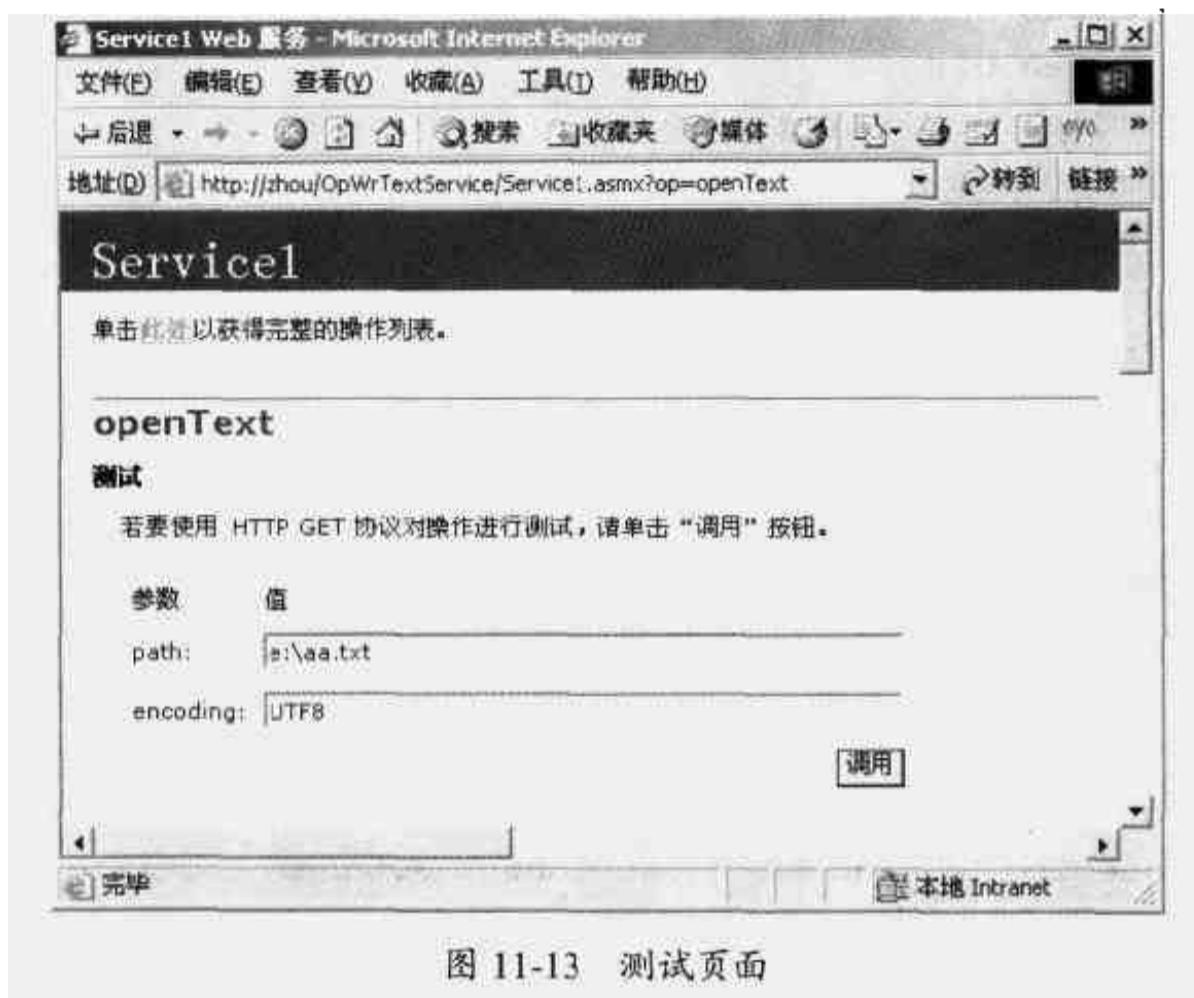


图 11-13 测试页面

测试的结果如图 11-14 所示，文本可以识别，证明 Web 服务正确无误。

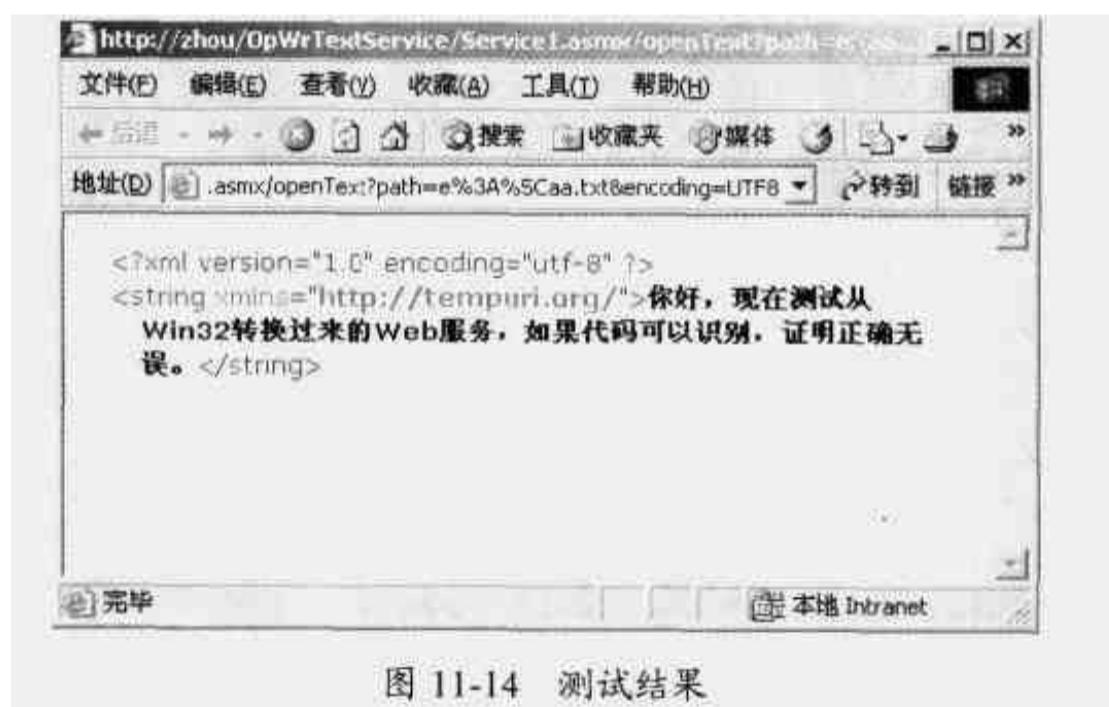


图 11-14 测试结果

11.2.3 将 Web 应用程序括换为 Web 服务

一般而言，从 ASP.NET Web 应用程序转换为 ASP.NET Web 服务，不会存在系统不支持的异常。凡可以正常运行的 ASP.NET Web 应用程序，都可以转换为 ASP.NET Web 服务。下面以一个十分简单的例子演示如何将 ASP.NET Web 应用程序转换为 ASP.NET Web 服务。下面是一个 ASP.NET 应用程序。

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Web;
using System.Web.SessionState;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.HtmlControls;
using System.IO;

namespace ApplicationToService
{
    /// <summary>
    /// WebForm1 的摘要说明
    /// </summary>
    public class WebForm1 : System.Web.UI.Page
    {
        protected System.Web.UI.WebControls.TextBox TextBox1;
        protected System.Web.UI.WebControls.Button Button1;
        protected System.Web.UI.WebControls.Label Label1;
        protected System.Web.UI.WebControls.TextBox TextBox2;
        protected System.Web.UI.WebControls.Label Label2;
        private void Page_Load(object sender, System.EventArgs e)
        {
            // 在此处放置用户代码以初始化页面
        }
        #region Web Form Designer generated code
        override protected void OnInit(EventArgs e)
        {
            //
            // CODEGEN: 该调用是 ASP.NET Web 窗体设计器所必需的
            //
            InitializeComponent();
            base.OnInit(e);
        }
        /// <summary>
        /// 设计器支持所需的方法 - 不要使用代码编辑器修改
        /// 此方法的内容
        /// </summary>
        private void InitializeComponent()
        {
            this.Button1.Click += new System.EventHandler(this.Button1_Click);
            this.Load += new System.EventHandler(this.Page_Load);
        }
    }
}
```

```
    }
    #endregion
    private void Button1_Click(object sender, System.EventArgs e)
    {
        int i=Directory.GetFiles(TextBox1.Text).Length;
        TextBox2.Text=i.ToString();
    }
}
```

该程序功能非常简单，仅仅是获取 TextBox1.Text 所指定的路径下文件数量。图 11-15 是该程序的界面。

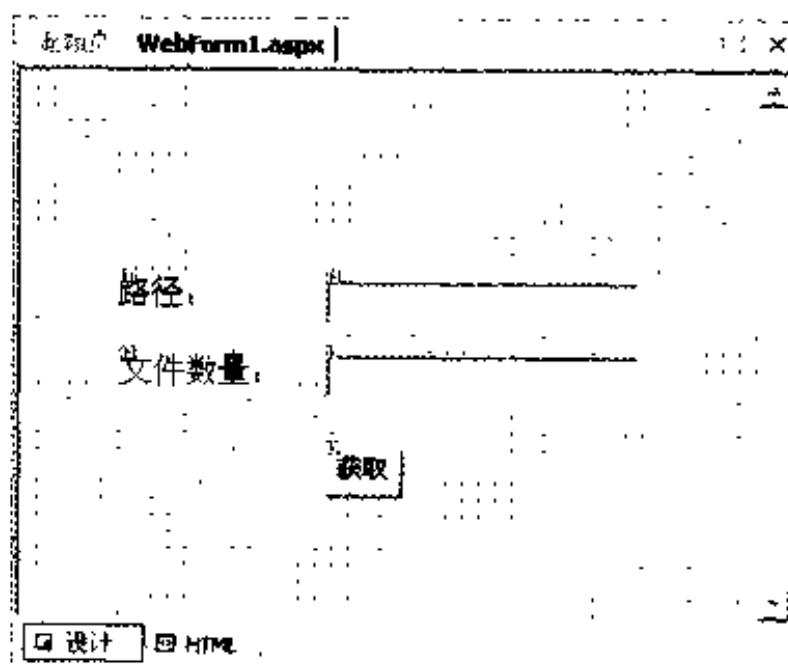


图 11-15 Web 应用程序界面

由于 ASP.NET Web 服务的属性和全局变量在客户端是不可见的，所以要尽量删除这些不必要的属性和变量的定义，将其转换成公开方法的参数。另外，Web 服务的公开方法一定要用特性声明。下面是转换后的 Web 服务的代码。

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.IO;
namespace AppToSer
{
    /// <summary>
    /// Service1 的摘要说明
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
    }
```

```
{  
    //CODEGEN: 该调用是 ASP.NET Web 服务设计器所必需的  
    InitializeComponent();  
}  
  
#region Component Designer generated code  
  
//Web 服务设计器所必需的  
private IContainer components = null;  
  
/// <summary>  
/// 设计器支持所需的方法 - 不要使用代码编辑器修改  
/// 此方法的内容  
/// </summary>  
private void InitializeComponent()  
{  
}  
  
/// <summary>  
/// 清理所有正在使用的资源  
/// </summary>  
protected override void Dispose( bool disposing )  
{  
    if(disposing && components != null)  
    {  
        components.Dispose();  
    }  
    base.Dispose(disposing);  
}  
  
#endregion  
  
// WEB 服务示例  
// HelloWorld() 示例服务返回字符串 Hello World  
// 若要生成，请取消注释下列行，然后保存并生成项目  
// 若要测试此 Web 服务，请按 F5 键  
  
//[WebMethod]  
//public string HelloWorld()  
//{  
//    return "Hello World";  
//}  
[WebMethod]  
public string getFileNumber(string path)  
{  
    int i=Directory.GetFiles(path).Length;
```

```
        string FileNumber=i.ToString();
        return FileNumber;
    }
}
```

图 11-16 是测试笔者“E:\TEMP\”的结果。

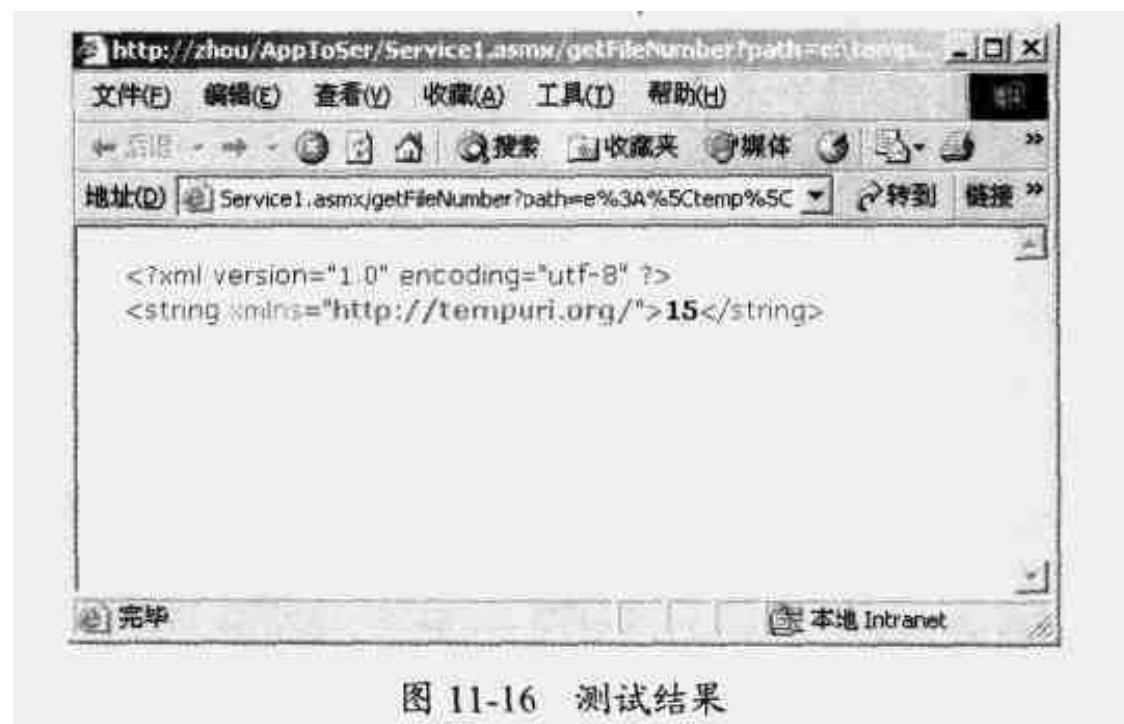


图 11-16 测试结果

11.3 XML Web 服务使用实例

在 11.2 开发了一个数据库服务，包括数据浏览、插入、删除、更新。本节将给出使用该服务的完整实例。下面是开发过程。

(1) 新建一个 Win32 项目，本例项目名称为 useWebService。

(2) 添加 Web 引用。

单击菜单“项目”→“添加 Web 引用”，将 11.2 节的数据库服务添加进去。

(3) 界面设计。

图 11-17 是该程序的界面。其中 Button 控件自上而下分别为 button1\button2, button3, button4, button5。TextBox 控件自上而下、自左至右分别为：textBox5, textBox6, textBox7, textBox8, textBox9, textBox10, textBox11, textBox1, textBox2, textBox3, textBox4。

(4) 编码。

① 查询功能的实现。

```
private void button1_Click(object sender, System.EventArgs e)
{
    DataSet mySet=dataService.getDataSet();
    dataGrid1.DataSource=mySet.Tables["表1"];
}
```

② 插入功能的实现。

```
private void button2_Click(object sender, System.EventArgs e)
```

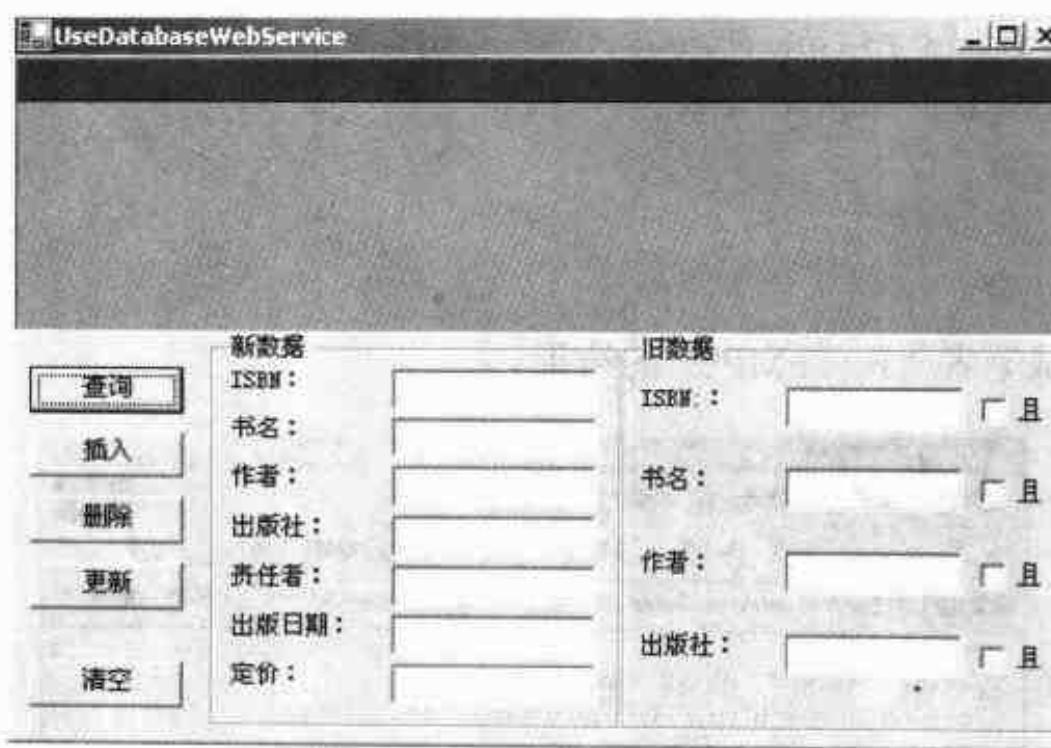


图 11-17 程序界面

```

{
    string insert = "INSERT INTO 表1(ISBN,书名,作者,出版社,责任者,出版日期,
    定价)VALUES (" + " " + textBox5.Text + " " + ", " +
    + textBox6.Text + " " + ", " + " " + textBox7.Text + " " + ", "
    + " " + textBox8.Text + " " + ", " + " " + textBox9.Text + " " + ", " +
    + textBox10.Text + " " + ", " + " " + Double.Parse(textBox11.Text) + " " + ")";
    dataService.insert(insert);
}

```

③ 删除功能的实现。

```

private void button3_Click(object sender, System.EventArgs e)
{
    string delete=null;
    if(checkBox1.Checked)
    {
        delete="ISBN=" + " " + textBox5.Text + " ";
    }
    if(checkBox2.Checked)
    {
        delete="书名=" + " " + textBox6.Text + " ";
    }
    if(checkBox3.Checked)
    {
        delete="作者=" + " " + textBox7.Text + " ";
    }
    if(checkBox4.Checked)
    {
        delete="出版社=" + " " + textBox8.Text + " ";
    }
}

//*****

```

```
if(checkBox1.Checked&&checkBox2.Checked)
{
    delete= " (" + "ISBN=" + " " + textBox5.Text + " " + " ) " + "AND " + " ( " + "书名=" +
        " " + textBox6.Text + " " + " ) ";
}
if(checkBox1.Checked&&checkBox3.Checked)
{
    delete= " (" + "ISBN=" + " " + textBox5.Text + " " + " ) " + "AND " + " ( " + "作者=" +
        " " + textBox7.Text + " " + " ) ";
}
if(checkBox1.Checked&&checkBox4.Checked)
{
    delete= " (" + "ISBN=" + " " + textBox5.Text + " " + " ) " + "AND ( 出版社=" +
        " " + textBox8.Text + " ) ";
}
//*****
if(checkBox2.Checked&&checkBox3.Checked)
{
    delete= " ( 书名=" + " " + textBox6.Text + " " + " ) " + "AND ( 作者=" +
        " " + textBox7.Text + " " + " ) ";
}
if(checkBox2.Checked&&checkBox4.Checked)
{
    delete= " ( 书名=" + " " + textBox6.Text + " " + " ) " + "AND ( 出版社=" +
        " " + textBox8.Text + " " + " ) ";
}
//*****
if(checkBox3.Checked&&checkBox4.Checked)
{
    delete= " ( 作者=" + " " + textBox6.Text + " " + " ) " + "AND ( 出版社=" +
        " " + textBox8.Text + " " + " ) ";
}
//*****
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked)
{
    delete= " (" + "ISBN=" + " " + textBox5.Text + " " + " ) " + "AND " + " ( " + "书名=" +
        " " + textBox6.Text + " " + " ) " + "AND ( 作者=" +
        " " + textBox7.Text + " " + " ) ";
}
if(checkBox1.Checked&&checkBox2.Checked&&checkBox4.Checked)
{
    delete= " (" + "ISBN=" + " " + textBox5.Text + " " + " ) " + "AND " + " ( " + "书名=" +
        " " + textBox6.Text + " " + " ) " + "AND ( 出版社=" + " " +
        " " + textBox8.Text + " " + " ) ";
}
//*****
```

```

if(checkBox1.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    delete= "(ISBN=" + " " + textBox5.Text + " " + " ) " + "AND (ISBN=" + " "
        + textBox5.Text + " " + " ) " + "AND (作者=" + " "
        + textBox7.Text + " " + " ) " + "AND (出版社=" + " "
        + textBox8.Text + " " + " ) ";
}
//*****
if(checkBox2.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    delete= ("+" + "书名=" + " " + textBox6.Text + " " + " ) " + "AND (作者=" +
        + " " + textBox7.Text + " " + " ) " + "AND (出版社=" +
        + " " + textBox8.Text + " " + " ) ";
}
//*****
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked
    &&checkBox4.Checked)
{
    delete= "(ISBN=" + " " + textBox5.Text + " " + " ) " + "AND (书名=" +
        + " " + textBox6.Text + " " + " ) " + "AND (作者=" + " "
        + textBox7.Text + " " + " ) " + "AND (出版社=" + " "
        + textBox8.Text + " " + " ) ";
}
string deleCommand="DELETE FROM 表1 WHERE "+delete;
dataService.delete(deleCommand);
}

```

④ 更新功能的实现。

```

private void button4_Click(object sender, System.EventArgs e)
{
    string update=null;
    if(checkBox1.Checked)
    {
        update="ISBN=" + " " + textBox5.Text + " " ;
    }
    if(checkBox2.Checked)
    {
        update="书名=" + " " + textBox6.Text + " " ;
    }
    if(checkBox3.Checked)
    {
        update="作者=" + " " + textBox7.Text + " " ;
    }
    if(checkBox4.Checked)
    {
        update="出版社=" + " " + textBox8.Text + " " ;
    }
}

```

```
}

//*****
if(checkBox1.Checked&&checkBox2.Checked)
{
    update= ("+"+ISBN+" "+textBox5.Text+" "+"+" "+") "+"AND "+"("+"书名="
    +" "+textBox6.Text+" "+"+");
}

if(checkBox1.Checked&&checkBox3.Checked)
{
    update= ("+"+ISBN+" "+textBox5.Text+" "+"+" "+") "+"AND "+"("+"作者="
    +" "+textBox7.Text+" "+"+");
}

if(checkBox1.Checked&&checkBox4.Checked)
{
    update= ("+"+ISBN+" "+textBox5.Text+" "+"+" "+") "+"AND ("出版社="
    +" "+textBox8.Text+" ")");
}

//*****
if(checkBox2.Checked&&checkBox3.Checked)
{
    update= ("书名="+" "+textBox6.Text+" "+"+" "+") "+"AND ("作者="
    +" "+textBox7.Text+" "+"+");
}

if(checkBox2.Checked&&checkBox4.Checked)
{
    update= ("书名="+" "+textBox6.Text+" "+"+" "+") "+"AND ("出版社="
    +" "+textBox8.Text+" "+"+");
}

//*****
if(checkBox3.Checked&&checkBox4.Checked)
{
    update= ("作者="+" "+textBox6.Text+" "+"+" "+") "+"AND ("出版社="
    +" "+textBox8.Text+" "+"+");
}

//*****
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked)
{
    update= ("+"+ISBN+" "+textBox5.Text+" "+"+" "+") "+"AND "+"("+"书名="*
    +" "+textBox6.Text+" "+"+" "+") "+"AND ("作者="+" "
    +textBox7.Text+" "+"+" "+");
}

if(checkBox1.Checked&&checkBox2.Checked&&checkBox4.Checked)
{
    update= ("+"+ISBN+" "+textBox5.Text+" "+"+" "+") "+"AND "+"("+"书名="*
    +" "+textBox6.Text+" "+"+" "+") "+"AND ("出版社="+" "
    +textBox8.Text+" "+"+" "+");
}
```

```

}

//*****
if(checkBox1.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    update=(ISBN+" "+textBox5.Text+" "+") "+"AND (ISBN="+
        +textBox5.Text+" "+") "+"AND (作者="+
        +textBox7.Text+" "+") "+"AND (出版社="+
        +textBox8.Text+" "+") ";
}

//*****
if(checkBox2.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    update="("+"书名="+" "+textBox6.Text+" "+") "+"AND (作者="+
        +textBox7.Text+" "+") "+"AND (出版社="+
        +textBox8.Text+" "+") ";
}

//*****
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked
    &&checkBox4.Checked)
{
    update=(ISBN+" "+textBox5.Text+" "+") "+"AND (书名="+
        +textBox6.Text+" "+") "+"AND (作者="+
        +textBox7.Text+" "+") "+"AND (出版社="+
        +textBox8.Text+" "+") ";
}

string updateCommand="UPDATE 表 1 SET "+ISBN+" "+textBox5.Text
    +" "+", "+书名+" "+textBox6.Text+" "+", "+作者+" "+
    +textBox7.Text+" "+", "+出版社+" "+textBox8.Text+" "+", "
    +" 责任者="+" "+textBox9.Text+" "+", "+出版日期="+
    +textBox10.Text+" "+", "+定价="+
    +Double.Parse(textBox11.Text)+" "+ WHERE "+update;
dataService.update(updateCommand);
}

```

⑤ 清空 DataGridView 控件内容。

清空 DataGridView 控件的内容，要靠窗体上的“清空”按钮来实现。该按钮的作用只有一个，就是将 DataGridView 控件中的内容清空，以防影响下一次浏览的结果。该按钮的 Click 事件的代码为：

```

private void button5_Click(object sender, System.EventArgs e)
{
    dataGridView1.DataSource=null;
}

```

下面是该程序的代码：

```
using System;
```

```
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace useWebService
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.DataGrid dataGridView1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.Button button2;
        private System.Windows.Forms.Button button3;
        private System.Windows.Forms.Button button4;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
        private System.Windows.Forms.Label label10;
        private System.Windows.Forms.Label label11;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.TextBox textBox6;
        private System.Windows.Forms.TextBox textBox7;
        private System.Windows.Forms.TextBox textBox8;
        private System.Windows.Forms.TextBox textBox9;
        private System.Windows.Forms.TextBox textBox10;
        private System.Windows.Forms.TextBox textBox11;
        private System.Windows.Forms.CheckBox checkBox1;
        private System.Windows.Forms.CheckBox checkBox2;
        private System.Windows.Forms.CheckBox checkBox3;
        private System.Windows.Forms.CheckBox checkBox4;
        private System.Windows.Forms.Button button5;
```

```
/// <summary>
/// 必需的设计器变量
/// </summary>
private System.ComponentModel.Container components = null;
private zhou.Service1 dataService;
public Form1()
{
    //
    // Windows 窗体设计器支持所必需的
    //
    InitializeComponent();
    dataService=new zhou.Service1();

    //
    // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
    //
}

/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
.....
.....
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}
```

```
private void button1_Click(object sender, System.EventArgs e)
{
    DataSet mySet=dataService.getDataSet();
    dataGrid1.DataSource=mySet.Tables["表 1"];
}

private void button2_Click(object sender, System.EventArgs e)
{
    string insert="INSERT INTO 表 1(ISBN,书名,作者,出版社,责任者,
    出版日期,定价)VALUES "+("+""
    +textBox5.Text+"','"+"+textBox6.Text+"','"+"+"
    +textBox7.Text+"','"+"+textBox8.Text+"','"+"+"
    +textBox9.Text+"','"+"+textBox10.Text+"','"+"+"
    +Double.Parse(textBox11.Text)+"') ";
    dataService.insert(insert);
}

private void button3_Click(object sender, System.EventArgs e)
{
    string delete=null;
    if(checkBox1.Checked)
    {
        delete="ISBN="+textBox5.Text+"'";
    }
    if(checkBox2.Checked)
    {
        delete="书名='"+textBox6.Text+"'";
    }
    if(checkBox3.Checked)
    {
        delete="作者='"+textBox7.Text+"'";
    }
    if(checkBox4.Checked)
    {
        delete="出版社='"+textBox8.Text+"'";
    }
    //*****if(checkBox1.Checked&&checkBox2.Checked)
    {
        delete="("+"ISBN='"+textBox5.Text+"') "+"AND"+("
            +"书名='"+textBox6.Text+"') ";
    }
    if(checkBox1.Checked&&checkBox3.Checked)
    {
        delete="("+"ISBN='"+textBox5.Text+"') "+"AND"+("
            +"作者='"+textBox7.Text+"') ";
    }
}
```

```
if(checkBox1.Checked&&checkBox4.Checked)
{
    delete= ("+"ISBN)+"'"+textBox5.Text+"'"+") "+"AND (出版社="
    +"'" +textBox8.Text+"') ";
}
//*****
if(checkBox2.Checked&&checkBox3.Checked)
{
    delete= "(书名)+"'"+textBox6.Text+"'"+") "+"AND (作者="
    +"'" +textBox7.Text+"') ";
}
if(checkBox2.Checked&&checkBox4.Checked)
{
    delete= "(书名)+"'"+textBox6.Text+"'"+") "+"AND (出版社="
    +"'" +textBox8.Text+"') ";
}
//*****
if(checkBox3.Checked&&checkBox4.Checked)
{
    delete= "(作者)+"'"+textBox6.Text+"'"+") "+"AND (出版社="
    +"'" +textBox8.Text+"') ";
}
//*****
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked)
{
    delete= ("+"ISBN)+"'"+textBox5.Text+"'"+") "+"AND "+("
    +"书名)+"'"+textBox6.Text+"'"+") "+"AND (作者="
    +"'" +textBox7.Text+"') ";
}
if(checkBox1.Checked&&checkBox2.Checked&&checkBox4.Checked)
{
    delete= ("+"ISBN)+"'"+textBox5.Text+"'"+") "+"AND "+("
    +"书名)+"'"+textBox6.Text+"'"+") "+"AND (出版社="
    +"'" +textBox8.Text+"') ";
}
//*****
if(checkBox1.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    delete= "(ISBN)+"'"+textBox5.Text+"'"+") "+"AND (ISBN="
    +"'" +textBox5.Text+"') "+"AND (作者="
    +"'" +textBox7.Text+"') "+"AND (出版社="
    +"'" +textBox8.Text+"') ";
}
//*****
if(checkBox2.Checked&&checkBox3.Checked&&checkBox4.Checked)
{
    delete= ("+"书名)+"'"+textBox6.Text+"'"+") "+"AND (作者="
}
```

```
+""+textBox7.Text+" "+") "+"AND (出版社=" +"
+textBox8.Text+" "+") ";
}
//*****
if (checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked
&&checkBox4.Checked)
{
    delete=(ISBN)+" "+textBox5.Text+" "+") "+"AND (书名="
+""+textBox6.Text+" "+") "+"AND (作者=" +
+textBox7.Text+" "+") "+"AND (出版社=" +
+textBox8.Text+" "+") ";
}
string deleCommand="DELETE FROM 表 1 WHERE "+delete;
dataService.delete(deleCommand);
}

private void button5_Click(object sender, System.EventArgs e)
{
    dataGridView1.DataSource=null;
}
private void button4_Click(object sender, System.EventArgs e)
{
    string update=null;
    if (checkBox1.Checked)
    {
        update=ISBN+" "+textBox5.Text+" ";
    }
    if (checkBox2.Checked)
    {
        update=书名+" "+textBox6.Text+" ";
    }
    if (checkBox3.Checked)
    {
        update=作者+" "+textBox7.Text+" ";
    }
    if (checkBox4.Checked)
    {
        update=出版社+" "+textBox8.Text+" ";
    }
}
//*****
if (checkBox1.Checked&&checkBox2.Checked)
{
    update=" ("+ISBN+" "+textBox5.Text+" "+") "+"AND "+("
+书名+" "+textBox6.Text+" "+") ";
}
if (checkBox1.Checked&&checkBox3.Checked)
{
    update=" ("+ISBN+" "+textBox5.Text+" "+") "+"AND "+("
```

```
    +"作者="+"'"+textBox7.Text+"'"+"')";  
}  
if(checkBox1.Checked&&checkBox4.Checked)  
{  
    update="("+"ISBN="+"'"+textBox5.Text+"'"+"')"+"AND (出版社="  
    +"'"+"+textBox8.Text+"'"");  
}  
//*****  
if(checkBox2.Checked&&checkBox3.Checked)  
{  
    update="(书名="+"'"+textBox6.Text+"'"+"')"+"AND (作者="  
    +"'"+"+textBox7.Text+"'"");  
}  
if(checkBox2.Checked&&checkBox4.Checked)  
{  
    update="(书名="+"'"+textBox6.Text+"'"+"')"+"AND (出版社="  
    +"'"+"+textBox8.Text+"'"");  
}  
//*****  
if(checkBox3.Checked&&checkBox4.Checked)  
{  
    update="(作者="+"'"+textBox6.Text+"'"+"')"+"AND (出版社="  
    +"'"+"+textBox3.Text+"'"");  
}  
//*****  
if(checkBox1.Checked&&checkBox2.Checked&&checkBox3.Checked)  
{  
    update="("+"ISBN="+"'"+textBox5.Text+"'"+"')"+"AND "+(""  
    +"书名="+"'"+textBox6.Text+"'"+"')"+"AND (作者="  
    +"'"+"+textBox7.Text+"'"");  
}  
if(checkBox1.Checked&&checkBox2.Checked&&checkBox4.Checked)  
{  
    update="("+"ISBN="+"'"+textBox5.Text+"'"+"')"+"AND "+(""  
    +"书名="+"'"+textBox6.Text+"'"+"')"+"AND (出版社="  
    +"'"+"+textBox8.Text+"'"");  
}  
//*****  
if(checkBox1.Checked&&checkBox3.Checked&&checkBox4.Checked)  
{  
    update="(ISBN="+"'"+textBox5.Text+"'"+"')"+"AND (ISBN="  
    +"'"+"+textBox5.Text+"'"+"')"+"AND (作者="  
    +"'"+"+textBox7.Text+"'"+"')"+"AND (出版社="  
    +"'"+"+textBox8.Text+"'"");  
}  
//*****  
if(checkBox2.Checked&&checkBox3.Checked&&checkBox4.Checked)
```

```

{
    update= " (" + "书名=" + " '" + textBox6.Text + "' " + " ) " + "AND (作者="
    + " '" + textBox7.Text + "' " + " ) " + "AND (出版社="
    + " '" + textBox8.Text + "' " + " ) ";
}

//*****+
if (checkBox1.Checked && checkBox2.Checked && checkBox3.Checked
    && checkBox4.Checked)
{
    update= " (ISBN=" + " '" + textBox5.Text + "' " + " ) " + "AND (书名="
    + " '" + textBox6.Text + "' " + " ) " + "AND (作者="
    + " '" + textBox7.Text + "' " + " ) " + "AND (出版社="
    + " '" + textBox8.Text + "' " + " ) ";
}

string updateCommand="UPDATE 表 1 SET "+ "ISBN="
    +" '" + textBox5.Text + "' " + " , " + "书名=" + " '" + textBox6.Text
    + " ' " + " , " + "作者=" + " '" + textBox7.Text + "' " + " , " + "出版社="
    + " '" + textBox8.Text + "' " + " , " + "责任者=" + " '" + textBox9.Text
    + " ' " + " , " + "出版日期=" + " '" + textBox10.Text + "' " + " , " + "定价=" + " ' "
    + Double.Parse(textBox11.Text) + " ' " + " WHERE " + update;
dataService.update(updateCommand);
}
}
}

```

(5) 演示。

图 11-18 是浏览原始数据的结果。

The screenshot shows a Windows application window titled "UseDatabaseWebService". Inside the window, there is a table with columns: ISBN, 书名 (Book Name), 作者 (Author), 出版社 (Publisher), 责任者 (Editor), 出版日期 (Publication Date), and 定价 (Price). The table contains three rows of data:

ISBN	书名	作者	出版社	责任者	出版日期	定价
3-00000-000	名校师生谈	王新东	新江北出版	江明	2000	12.0
7-00000-000	中国古代名	李小丽	新江南出版	吴平	2002	26.0
7-00000-000	战国风云	张小方	新江东出版	赵进东	2002	22.0

Below the table is a form with two panes: "新数据" (New Data) and "旧数据" (Old Data). The "新数据" pane contains fields for ISBN, 书名, 作者, 出版社, 责任者, 出版日期, and 定价, each with an associated text input field. The "旧数据" pane contains fields for ISBN, 书名, 作者, 出版社, 责任者, 出版日期, and 定价, each with an associated text input field and a radio button labeled "且" (And).

图 11-18 原始数据

图 11-19 是插入一条 ISBN 为“7-00000-000-4”、书名为“计算机原理”、作者为“王立平”、出版社为“南方电子出版社”、责任者为“李立”、出版日期为“2002”、定价为“12”的数据之后的显示结果。

图 11-20 是将上述刚插入的数据更新后的结果。

图 11-21 是删除出版社为“南方电子出版”的数据之后的结果。

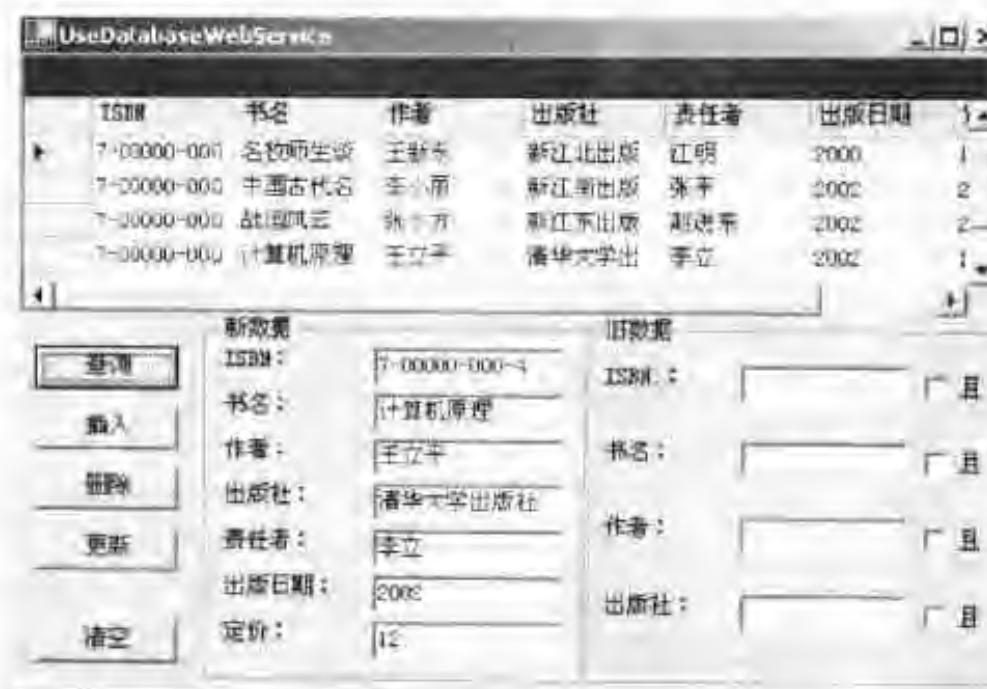


图 11-19 数据插入后的结果

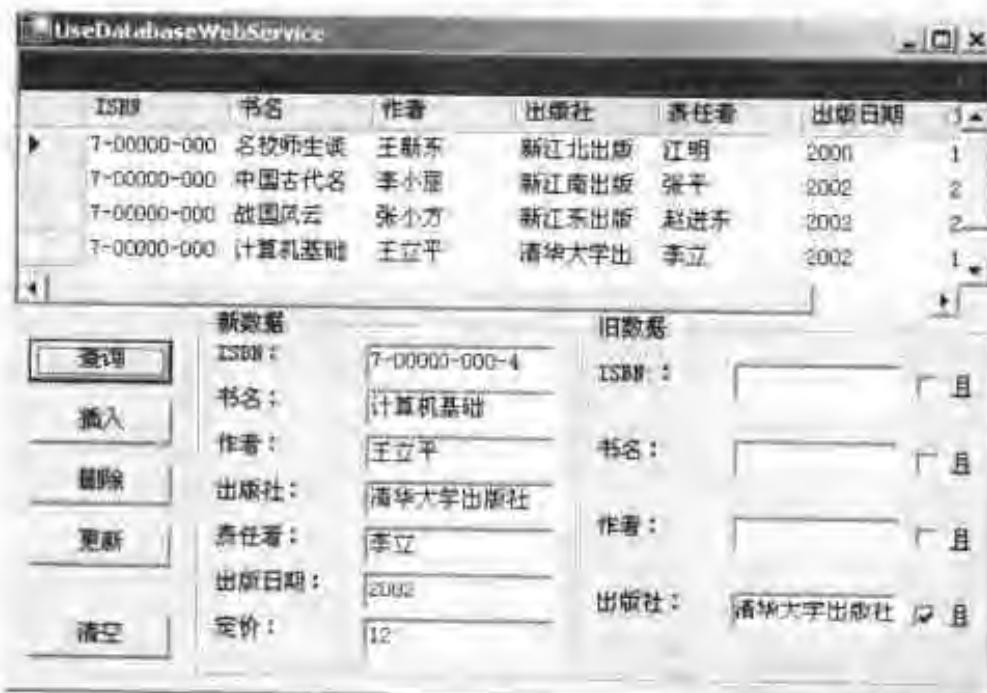


图 11-20 数据更新后的结果



图 11-21 删除数据后的结果

本章小结

本章使用异步套接字技术并利用 TCP 协议开发了一个 Win32 网络数据库。通过本章的学习，有助于读者迅速开发企业级网络应用程序。

第 12 章 分布式商贸财务系统开发实例

本章开发一个分布式商贸财务系统。该系统分为三个部分，第一部分是基础数据库，第二部分是商业逻辑，第三部分是客户界面。基础数据库使用 SQL Server 的数据库，商业逻辑用 XML ASP.NET Services 完成，用户界面用 Win32 界面完成。

12.1 解决方案简介

12.1.1 程序的主要功能

与商业软件相比，该程序是一个十分简单的财务管理软件，主要具有以下几个方面的功能：

- 进货部门可以插入、删除、修改有关进货的数据。
- 销售部门插入、删除、修改有关销售的数据。
- 财务部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的全部流水账。
- 财务部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的全部分类账。
- 财务部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的全部盈亏。
- 财务部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的某种商品的盈亏。
- 管理（经理）部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的全部流水账。
- 管理（经理）部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的全部分类账。
- 管理（经理）部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的全部盈亏。

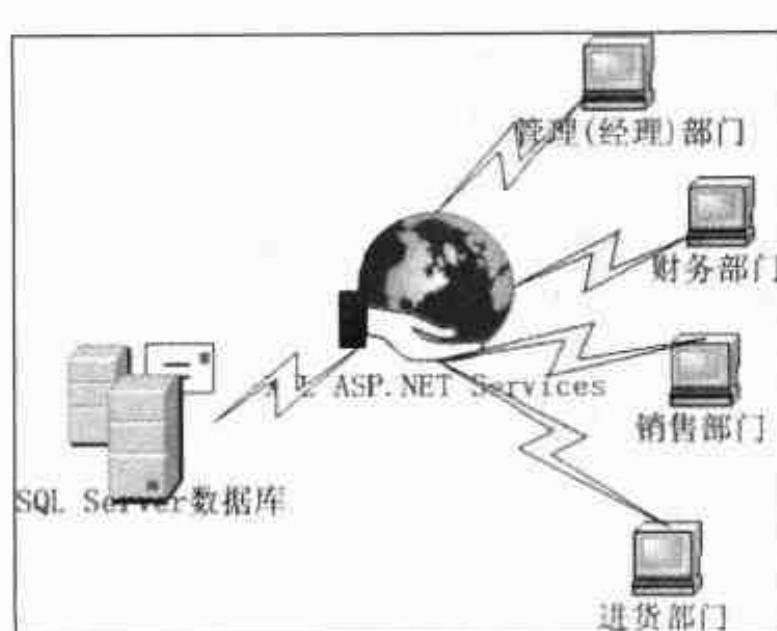


图 12-1 解决方案

- 管理(经理)部门可以查询从 XX 年 XX 月 XX 日到从 XX 年 XX 月 XX 日的某种商品的盈亏。

该程序的解决方案如图 12-1 所示。

12.1.2 基础数据库

本程序的基础数据库是一个 SQL Server 数据库，该数据库定义如图 12-2 所示。

	列名	数据类型	长度	允许空
1	自动编号	bigint	8	
2	发票编号	char	20	✓
3	日期	datetime	8	✓
4	商品编号	char	10	✓
5	商品名称	char	30	✓
6	买进数量	float	8	✓
7	买进单价	float	8	✓
8	卖出数量	float	8	✓
9	卖出单价	float	8	✓

图 12-2 基础数据库数据表

下列代码可以建立上述数据表(下列代码将数据表建在了 msdb 数据库)：

```
private void button1_Click(object sender, System.EventArgs e)
{
    //与本地主机的用户名为 sa, 密码为空的 msdb 数据库建立连接
    SqlConnection myConnection = new SqlConnection();
    myConnection.ConnectionString =
        "data source=localhost;
         initial catalog=msdb;
         persist security info=False;
         user id=sa;
         workstation id=ZHOU;packet size=4096 ";
    //建立数据库数据表
    SqlCommand createDatatable=new SqlCommand
        ("CREATE TABLE 表1 (自动编号 bigint IDENTITY PRIMARY KEY,
        发票编号 char(20), 日期 datetime, 商品编号 char(10), 商品名称 char(30),
        买进数量 float, 买进单价 float, 卖出数量 float,
        卖出单价 float)",myConnection);
    myConnection.Open();
    createDatatable.ExecuteNonQuery();
    myConnection.Close();
}
```

12.2 XML ASP.NET Services 开发

本节用 XML ASP.NET 服务开发商贸财务管理系统的商业逻辑，主要包括数据浏览服务、数据编辑服务和盈亏统计服务等。

12.2.1 特定时间段内全部商品流水账服务

首先新建一个 ASP.NET Web 服务项目，本例项目名称为“business”。然后添加如下引用：

```
using System.Data.SqlClient;
```

然后使用数据适配器或直接使用代码建立与 msdb 数据库表 1 的连接。再建立数据集 getData，其对象实例为 getData1。

下列代码提供全部商品流水账的服务：

```
[WebMethod]
public DataSet allLiuShuiZhang(string year1, string month1, string day1,
    string year2, string month2, string day2)
{
    string date1=year1+"-"+month1+"-"+day1;
    string date2=year2+"-"+month2+"-"+day2;
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表 1
        WHERE (日期>='"+date1+"') "+" AND (日期<='"+date2+"')"
        +" ORDER BY 日期 ASC";
    sqlDataAdapter1.Fill(getData1.表 1);
    return getData1;
}
```

上述方法提供从 date1 到 date2 这一段时间的全部流水账服务。

12.2.2 特定时间段内特定商品流水账服务

下列代码提供特定商品的流水账服务：

```
[WebMethod]
public DataSet singleLiuShuiZhang(string year1, string month1, string day1,
    string year2, string month2, string day2, string shangpinbianhao)
{
    string date1=year1+"-"+month1+"-"+day1;
    string date2=year2+"-"+month2+"-"+day2;
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表 1
        WHERE (日期>='"+date1+"') "+" AND (日期<='"+date2+"')"
        +" AND (商品编号='"+shangpinbianhao+"') "+" ORDER BY 日期 ASC";
    sqlDataAdapter1.Fill(getData1.表 1);
```

```

        return getData1;
    //
}

```

在上面代码中，首先要指定特定商品的商品编号，然后提供起始年月日和终止年月日。

12.2.3 特定时间段内所有商品的经营盈亏服务

下列代码提供从 XX 年 XX 月 XX 日到 XX 年 XX 月 XX 日的所有商品的盈亏服务：

```

[WebMethod]
public string allInOut(string year1,string month1,string day1,
    string year2,string month2,string day2)
{
    string sele="SELECT * FROM 表 1";
    DateTime time1=new DateTime(Int32.Parse(year1),Int32.Parse(month1),
        Int32.Parse (day1));
    DateTime time2=new DateTime(Int32.Parse(year2),Int32.Parse(month2),
        Int32.Parse (day2));
    double x=0;
    double y=0;
    sqlConnection1.Open();
    sqlSelectCommand1=new SqlCommand (sele,sqlConnection1 );
    SqlDataReader reader=sqlSelectCommand1.ExecuteReader();
    while(reader.Read())
    {
        DateTime time=reader.GetDateTime(2);
        if((time>=time1)&&(time<=time2))
        {
            double in1=reader.GetDouble(7);
            double in2=reader.GetDouble(8);
            x=x+in1*in2;
            double out1=reader.GetDouble(5);
            double out2=reader.GetDouble(6);
            y=y+out1*out2;
        }
    }
    reader.Close();
    sqlConnection1.Close();
    double end=x-y;
    return end.ToString();
}

```

12.2.4 特定时间段内特定商品的经营盈亏服务

下列代码提供特定时间段内特定商品的经营盈亏服务：

```

[WebMethod]
public string singleInOut(string shangpinbianhao,string year1,

```

```

    string month1, string day1, string year2, string month2, string day2)
{
    string sele = "SELECT * FROM 表1 WHERE " + "(商品编号="
        + "''"+shangpinbianhao+"''"+")";
    DateTime time1 = new DateTime(Int32.Parse(year1), Int32.Parse(month1),
        Int32.Parse(day1));
    DateTime time2 = new DateTime(Int32.Parse(year2), Int32.Parse(month2),
        Int32.Parse(day2));
    double x = 0;
    double y = 0;
    sqlConnection1.Open();
    sqlSelectCommand1 = new SqlCommand(sele, sqlConnection1);
    SqlDataReader reader = sqlSelectCommand1.ExecuteReader();
    while (reader.Read())
    {
        DateTime time = reader.GetDateTime(2);
        if ((time >= time1) && (time <= time2))
        {
            double in1 = reader.GetDouble(7);
            double in2 = reader.GetDouble(8);
            x = x + in1 * in2;
            double out1 = reader.GetDouble(5);
            double out2 = reader.GetDouble(6);
            y = y + out1 * out2;
        }
        // if((time>=time1)&&(time<=time2)){
    }
    reader.Close();
    sqlConnection1.Close();
    double end = x - y;
    return end.ToString();
}

```

12.2.5 进货数据编辑服务

下列代码提供进货数据的编辑服务，主要包括数据的插入、删除和修改。

```

[WebMethod]
public void buyEdit(string mission, string fapiaobianhao, string riqi,
    string shangpinbianhao, string shangpinmingcheng, string maijinshuliang,
    string maijindanjia, string oldfapiaobianhao)
{
    if (mission == "insert")
    {
        string command = "INSERT INTO 表1(发票编号,日期,商品编号,商品名称,
            买进数量,买进单价,卖出数量,卖出单价)VALUES "
            + "(" + "''"+fapiaobianhao+"''"+", "+ "''"+riqi+"''"+", "+ "''"
            + shangpinbianhao+"''"+", "+ "''"+shangpinmingcheng+"''"+", "+ "''"
            + maijinshuliang+"''"+", "+ "''"+maijindanjia+"''"+", "+ "''"+0+"''"

```

```

        +" , "+" "+0+" "+") ";
getdata1.AcceptChanges();
sqlconnection1.Open();
sqlInsertCommand1=new SqlCommand(command, sqlconnection1);
sqlInsertCommand1.ExecuteNonQuery();
sqlconnection1.Close();
}
if(mission=="delete")
{
    string command="DELETE FROM 表1 WHERE 发票编号="
        +" "+oldfapiao bianhao+" ";
getdata1.AcceptChanges();
sqlconnection1.Open();
sqlDeleteCommand1=new SqlCommand(command, sqlconnection1);
sqlDeleteCommand1.ExecuteNonQuery();
sqlconnection1.Close();
}
if(mission=="update")
{
    string command="UPDATE 表1 SET 发票编号="+
        +fapiao bianhao+" "+", "+日期="+" "+riqi+" "+", "+商品编号="+
        +" "+shangpinbianhao+" "+", "+商品名称="+" "+
        +shangpinmingcheng+" "+", "+买进数量="+" "+
        +mai jin shuliang+" "+", "+买进单价="+" "+mai jin dan jia+" "+
        +" "+卖出数量="+" "+C+" "+", "+卖出单价="+" "+0+" "+
        +" WHERE 发票编号="+" "+oldfapiao bianhao+" ";
getdata1.AcceptChanges();
sqlconnection1.Open();
sqlUpdateCommand1=new SqlCommand(command, sqlconnection1);
sqlUpdateCommand1.ExecuteNonQuery();
sqlconnection1.Close();
}
}

```

12.2.6 售货数据编辑服务

下列代码提供进货数据的编辑服务，主要包括数据的插入、删除和修改。

```

[WebMethod]
public void sellEdit(string mission, string fapiao bianhao, string riqi,
    string shangpinbianhao, string shangpinmingcheng, string maichushuliang,
    string maichudanjia, string oldfapiao bianhao)
{
    if(mission=="insert")
    {
        string command="INSERT INTO 表1(发票编号, 日期, 商品编号, 商品名称,

```

```
        买进数量,买进单价,卖出数量,卖出单价)VALUES "
        +" (" + "" +fapiaobianhao+ " " + ", " + " " +riqi+ " " + ", " + " "
        +shangpinbianhao+ " " + ", " + " " +shangpinmingcheng+ " " + ", " + " " +0
        + " " + ", " + " " +0+ " " + ", " + " " +maichushuliang+ " " + ", " + " "
        +maichudanjia+ " " + ")";
        getData1.AcceptChanges();
        sqlConnection1.Open();
        sqlInsertCommand1=new SqlCommand(command, sqlConnection1);
        sqlInsertCommand1.ExecuteNonQuery();
        sqlConnection1.Close();
    }
    if(mission=="delete")
    {
        string command="DELETE FROM 表1 WHERE 发票编号="
        +" "+oldfapiaobianhao+ "";
        getData1.AcceptChanges();
        sqlConnection1.Open();
        sqlDeleteCommand1=new SqlCommand(command, sqlConnection1);
        sqlDeleteCommand1.ExecuteNonQuery();
        sqlConnection1.Close();
    }
    if(mission=="update")
    {
        string command="UPDATE 表1 SET 发票编号="
        +" "+fapiaobianhao+ " " + ", " + " 日期=" + " " +riqi+ " " + ", " + " 商品编号="
        + " " +shangpinbianhao+ " " + ", " + " 商品名称=" + " "
        +shangpinmingcheng+ " " + ", " + " 买进数量=" + " " +0+ " " + ", " + " 买进单价="
        + " " +0+ " " + ", " + " 卖出数量=" + " " +maichushuliang+ " " + ", " +
        + " 卖出单价=" + " " +maichudanjia+ " " + " WHERE 发票编号="
        + " " +oldfapiaobianhao+ "";
        getData1.AcceptChanges();
        sqlConnection1.Open();
        sqlUpdateCommand1=new SqlCommand(command, sqlConnection1);
        sqlUpdateCommand1.ExecuteNonQuery();
        sqlConnection1.Close();
    }
}
```

下面是 business 项目的主要代码:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
```

```
using System.Data.SqlClient;

namespace business
{
    /// <summary>
    /// Service1 的摘要说明
    /// </summary>
    public class Service1 : System.Web.Services.WebService
    {
        public Service1()
        {
            //CODEGEN: 该调用是 ASP.NET Web 服务设计器所必需的
            InitializeComponent();
        }

        private System.Data.SqlClient.SqlDataAdapter sqlDataAdapter1;
        private System.Data.SqlClient.SqlCommand sqlSelectCommand1;
        private System.Data.SqlClient.SqlCommand sqlInsertCommand1;
        private System.Data.SqlClient.SqlCommand sqlUpdateCommand1;
        private System.Data.SqlClient.SqlCommand sqlDeleteCommand1;
        private System.Data.SqlClient.SqlConnection sqlConnection1;
        private business.getData getData1;

        //下面系统自动产生的代码已经删除
        #region Component Designer generated code
        .....
        .....
        #endregion

        //上面系统自动产生的代码已经删除
        // WEB 服务示例
        // HelloWorld() 示例服务返回字符串 Hello World
        // 若要生成，请取消注释下列行，然后保存并生成项目
        // 若要测试此 Web 服务，请按 F5 键

        // [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }

        [WebMethod]
        public DataSet allLiuShuiZhang(string year1, string month1,
            string day1, string year2, string month2, string day2)
        {
            string date1=year1+"-"+month1+"-"+day1;
            string date2=year2+"-"+month2+"-"+day2;
            sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表 1
                WHERE (日期>='"+date1+"') "+" AND (日期<="
        }
    }
}
```

```
+"""+date2+""")+" ORDER BY 日期 ASC";
sqlDataAdapter1.Fill(getData1.表1);
return getData1;
}

[WebMethod]
public DataSet singleLiuShuiZhang(string year1,string month1,
string day1,string year2,string month2,string day2,
string shangpinbianhao)
{
    string date1=year1+"-"+month1+"-"+day1;
    string date2=year2+"-"+month2+"-"+day2;
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表1
        WHERE (日期>='"+""+date1+"')+" AND (日期<="
        +"""+date2+""")+" AND (商品编号='"+""+shangpinbianhao+"')"
        +" ORDER BY 日期 ASC";
    sqlDataAdapter1.Fill(getData1.表1);
    return getData1;
    //
}
[WebMethod]
public string allInOut(string year1,string month1,string day1,
string year2,string month2,string day2)
{
    string sele="SELECT * FROM 表1";
    DateTime time1=new DateTime(Int32.Parse(year1),
        Int32.Parse(month1),Int32.Parse(day1));
    DateTime time2=new DateTime(Int32.Parse(year2),
        Int32.Parse(month2),Int32.Parse(day2));
    double x=0;
    double y=0;
    sqlConnection1.Open();
    sqlSelectCommand1=new SqlCommand (sele,sqlConnection1 );
    SqlDataReader reader=sqlSelectCommand1.ExecuteReader();
    while(reader.Read())
    {
        DateTime time=reader.GetDateTime(2);
        if((time>=time1)&&(time<=time2))
        {
            double in1=reader.GetDouble(7);
            double in2=reader.GetDouble(8);
            x=x+in1*in2;
            double out1=reader.GetDouble(5);
            double out2=reader.GetDouble(6);
            y=y+out1*out2;
        }
    }
}
```

```
        reader.Close();
        sqlConnection1.Close();
        double end=x-y;
        return end.ToString();
    }
    [WebMethod]
    public string singleInOut(string shangpinbianhao,
        string year1,string month1,string day1,string year2,
        string month2,string day2)
    {
        string sele="SELECT * FROM 表1 WHERE "+"(商品编号="
        +"'" +shangpinbianhao+"'" +")";
        DateTime time1=new DateTime(Int32.Parse(year1),
            Int32.Parse(month1),Int32.Parse(day1));
        DateTime time2=new DateTime(Int32.Parse(year2),
            Int32.Parse(month2),Int32.Parse(day2));
        double x=0;
        double y=0;
        sqlConnection1.Open();
        sqlSelectCommand1=new SqlCommand (sele,sqlConnection1 );
        SqlDataReader reader=sqlSelectCommand1.ExecuteReader();
        while(reader.Read())
        {
            DateTime time=reader.GetDateTime(2);
            if((time>=time1)&&(time<=time2))
            {
                double in1=reader.GetDouble(7);
                double in2=reader.GetDouble(8);
                x=x+in1*in2;
                double out1=reader.GetDouble(5);
                double out2=reader.GetDouble(6);
                y=y+out1*out2;
                // if((time>=time1)&&(time<=time2)){
            }
            reader.Close();
            sqlConnection1.Close();
            double end=x-y;
            return end.ToString();
        }
    }
    [WebMethod]
    public void buyEdit(string mission,string fapiaobianhao,
        string riqi,string shangpinbianhao,string shangpinmingcheng,
        string maijinshuliang,string maijindanjia,
        string oldfapiaobianhao)
    {
        if(mission=="insert")
```

```
{  
    string command="INSERT INTO 表1(发票编号,日期,商品编号,  
    商品名称,买进数量,买进单价,卖出数量,卖出单价)VALUES "  
    +"("+"'"+fapiaobianhao+"'"', "+'"'+riqi+"'"', "+'"'+  
    +shangpinbianhao+"'"', "+'"'+shangpinmingcheng+"'"'  
    +'", "+'"'+maijinshuliang+"'"', "+'"'+maijindanjia+"'"'  
    +'", "+'"'+0+"'"', "+'"'+0+"'"')";  
    getDatal.AcceptChanges();  
    sqlConnection1.Open();  
    sqlInsertCommand1=new SqlCommand(command,sqlConnection1);  
    sqlInsertCommand1.ExecuteNonQuery();  
    sqlConnection1.Close();  
}  
if(mission=="delete")  
{  
    string command="DELETE FROM 表1 WHERE 发票编号="  
    +"'"'+oldfapiaobianhao+"'"';  
    getDatal.AcceptChanges();  
    sqlConnection1.Open();  
    sqlDeleteCommand1=new SqlCommand(command,sqlConnection1);  
    sqlDeleteCommand1.ExecuteNonQuery();  
    sqlConnection1.Close();  
}  
if(mission=="update")  
{  
    string command="UPDATE 表1 SET 发票编号="  
    +"'"'+fapiaobianhao+"'"', "+"日期="'"'+riqi+"'"', "  
    +"商品编号="'"'+shangpinbianhao+"'"', "+"商品名称="'"'  
    +'", "+"'"'+shangpinmingcheng+"'"', "+"'"'+maijinshuliang+"'"', "+"'"'+  
    +maijindanjia+"'"', "+"'"'+卖出数量="'"'+0+"'"', "+"'"'+  
    +"'"'+卖出单价="'"'+0+"'"', "+"'"'+卖出单价="'"'+0+"'"', "+"'"'+  
    +"'"'+0+"'"', "+"'"'+0+"'"' WHERE 发票编号="  
    +"'"'+oldfapiaobianhao+"'"';  
    getDatal.AcceptChanges();  
    sqlConnection1.Open();  
    sqlUpdateCommand1=new SqlCommand(command,sqlConnection1);  
    sqlUpdateCommand1.ExecuteNonQuery();  
    sqlConnection1.Close();  
}  
}  
[WebMethod]  
public void sellEdit(string mission,string fapiaobianhao,  
    string riqi,string shangpinbianhao,string shangpinmingcheng,  
    string maichushuliang,string maichudanjia,  
    string oldfapiaobianhao)  
{
```

```
if (mission=="insert")
{
    string command="INSERT INTO 表1(发票编号,日期,商品编号,
        商品名称,买进数量,买进单价,卖出数量,卖出单价)VALUES "
        +"(" +""+fapiaobianhao+" "+", "+""+riqi+" "+", "+""+
        +shangpinbianhao+" "+", "+""+shangpinmingcheng+" "+
        +"", "+""+0+" "+", "+""+0+" "+", "+""+
        +maichushuliang+" "+", "+""+maichudanjia+" "+")";
    getData1.AcceptChanges();
    sqlConnection1.Open();
    sqlInsertCommand1=new SqlCommand(command,sqlConnection1);
    sqlInsertCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}

if (mission=="delete")
{
    string command="DELETE FROM 表1 WHERE 发票编号="
        +" "+oldfapiaobianhao+" ";
    getData1.AcceptChanges();
    sqlConnection1.Open();
    sqlDeleteCommand1=new SqlCommand(command,sqlConnection1);
    sqlDeleteCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}

if (mission=="update")
{
    string command="UPDATE 表1 SET 发票编号="
        +" "+fapiaobianhao+" ", "+日期="+" "+riqi+" ", "
        +商品编号="+" "+shangpinbianhao+" ", "+商品名称="*
        +" "+shangpinmingcheng+" ", "+买进数量="+" "+0
        +" "+", "+买进单价="+" "+0+" "+", "+卖出数量="*
        +" "+maichushuliang+" ", "+卖出单价="+" "+
        +maichudanjia+" " WHERE 发票编号="
        +" "+oldfapiaobianhao+" ";
    getData1.AcceptChanges();
    sqlConnection1.Open();
    sqlUpdateCommand1=new SqlCommand(command,sqlConnection1);
    sqlUpdateCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
}
}
```

12.3 客户端开发

客户端主要包括进货部门、售货部门、财务部门和管理（经理）部门，其界面是 Win32 界面。

12.3.1 进货部门客户端开发

(1) 新建项目。

新建一个 Windows 应用程序项目，本例项目名称为 buyEdition。

(2) 界面设计。

图 12-3 是该程序的界面，其中所有 TextBox 控件的 ReadOnly 控件为 true。在图中，文本框自上而下分定为 textBox1, textBox2, textBox3, textBox4, textBox5, textBox6, textBox7, textBox8。单选按钮自上而下分定为 radioButton1, radioButton2, radioButton3。



图 12-3 进货部门客户端界面

(3) 编写代码。

① “插入新数据” 单选按钮的 CheckedChanged 事件的代码。

```
private void radioButton1_CheckedChanged(object sender, System.EventArgs e)
{
    if(radioButton1.Checked)
    {
        textBox1.Text="insert";
        textBox8.Text="";
        textBox2.ReadOnly=false;
        textBox3.ReadOnly=false;
        textBox4.ReadOnly=false;
        textBox5.ReadOnly=false;
        textBox6.ReadOnly=false;
        textBox7.ReadOnly=false;
        textBox8.ReadOnly=true;
    }
}
```

```

}
}
```

② “删除旧数据” 单选按钮的 CheckedChanged 的代码。

```

private void radioButton2_CheckedChanged(object sender, System.EventArgs e)
{
    if(radioButton2.Checked)
    {
        textBox1.Text="delete";
        textBox8.ReadOnly=false;
        textBox2.ReadOnly=true;
        textBox3.ReadOnly=true;
        textBox4.ReadOnly=true;
        textBox5.ReadOnly=true;
        textBox6.ReadOnly=true;
        textBox7.ReadOnly=true;
    }
}
```

③ “修改旧数据”的 CheckedChanged 事件的代码。

```

private void radioButton3_CheckedChanged(object sender, System.EventArgs e)
{
    if(radioButton3.Checked)
    {
        textBox1.Text="update";
        textBox2.ReadOnly=false;
        textBox3.ReadOnly=false;
        textBox4.ReadOnly=false;
        textBox5.ReadOnly=false;
        textBox6.ReadOnly=false;
        textBox7.ReadOnly=false;
        textBox8.ReadOnly=false;
    }
}
```

④ “确定” 按钮的 Click 事件的代码。

```

private void button1_Click(object sender, System.EventArgs e)
{
    zhou.Service1 myService=new zhou.Service1();
    if(radioButton1.Checked)
    {
        myService.buyEdit(textBox1.Text, textBox2.Text, textBox3.Text,
                        textBox4.Text, textBox5.Text, textBox6.Text, textBox7.Text,
                        textBox8.Text);
    }
    if(radioButton2.Checked)
```

```
{  
    myService.buyEdit(textBox1.Text, textBox2.Text, textBox3.Text,  
                      textBox4.Text, textBox5.Text, textBox6.Text, textBox7.Text,  
                      textBox8.Text);  
}  
if (radioButton3.Checked)  
{  
    myService.buyEdit(textBox1.Text, textBox2.Text, textBox3.Text,  
                      textBox4.Text, textBox5.Text, textBox6.Text, textBox7.Text,  
                      textBox8.Text);  
}  
}  
}
```

下面是该程序的主要代码：

```
using System;  
using System.Drawing;  
using System.Collections;  
using System.ComponentModel;  
using System.Windows.Forms;  
using System.Data;  
  
namespace buyEdition  
{  
    /// <summary>  
    /// Form1 的摘要说明  
    /// </summary>  
    public class Form1 : System.Windows.Forms.Form  
    {  
        private System.Windows.Forms.GroupBox groupBox1;  
        private System.Windows.Forms.RadioButton radioButton1;  
        private System.Windows.Forms.RadioButton radioButton2;  
        private System.Windows.Forms.RadioButton radioButton3;  
        private System.Windows.Forms.GroupBox groupBox2;  
        private System.Windows.Forms.TextBox textBox1;  
        private System.Windows.Forms.TextBox textBox2;  
        private System.Windows.Forms.TextBox textBox3;  
        private System.Windows.Forms.TextBox textBox4;  
        private System.Windows.Forms.TextBox textBox5;  
        private System.Windows.Forms.TextBox textBox6;  
        private System.Windows.Forms.TextBox textBox7;  
        private System.Windows.Forms.Button button1;  
        private System.Windows.Forms.TextBox textBox8;  
        private System.Windows.Forms.Label label1;  
        private System.Windows.Forms.Label label2;  
        private System.Windows.Forms.Label label3;  
        private System.Windows.Forms.Label label4;
```

```
private System.Windows.Forms.Label label5;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label8;
/// <summary>
/// 必需的设计器变量
/// </summary>
private System.ComponentModel.Container components = null;

public Form1()
{
    //
    // Windows 窗体设计器支持所必需的
    //
    InitializeComponent();
    //
    // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
    //
}

/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
...
...
#endregion
//上面系统自动产生的代码已经删除

/// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
static void Main()
{
    Application.Run(new Form1());
}
```

```
}

private void button1_Click(object sender, System.EventArgs e)
{
    zhou.Service1 myService=new zhou.Service1();
    if(radioButton1.Checked)
    {
        myService.buyEdit(textBox1.Text, textBox2.Text,
                           textBox3.Text, textBox4.Text, textBox5.Text,
                           textBox6.Text, textBox7.Text, textBox8.Text);
    }
    if(radioButton2.Checked)
    {
        myService.buyEdit(textBox1.Text, textBox2.Text,
                           textBox3.Text, textBox4.Text, textBox5.Text, textBox6.Text,
                           textBox7.Text, textBox8.Text);
    }
    if(radioButton3.Checked)
    {
        myService.buyEdit(textBox1.Text, textBox2.Text,
                           textBox3.Text, textBox4.Text, textBox5.Text,
                           textBox6.Text, textBox7.Text, textBox8.Text);
    }
}
private void radioButton1_CheckedChanged(object sender,
                                       System.EventArgs e)
{
    if(radioButton1.Checked)
    {
        textBox1.Text="insert";
        textBox8.Text="";
        textBox2.ReadOnly=false;
        textBox3.ReadOnly=false;
        textBox4.ReadOnly=false;
        textBox5.ReadOnly=false;
        textBox6.ReadOnly=false;
        textBox7.ReadOnly=false;
        textBox8.ReadOnly=true;
    }
}
private void radioButton2_CheckedChanged(object sender,
                                       System.EventArgs e)
{
    if(radioButton2.Checked)
    {
        textBox1.Text="delete";
        textBox8.ReadOnly=false;
        textBox2.ReadOnly=true;
    }
}
```

```

        textBox3.ReadOnly=true;
        textBox4.ReadOnly=true;
        textBox5.ReadOnly=true;
        textBox6.ReadOnly=true;
        textBox7.ReadOnly=true;
    }
}

private void radioButton3_CheckedChanged(object sender,
    System.EventArgs e)
{
    if(radioButton3.Checked)
    {
        textBox1.Text="update";
        textBox2.ReadOnly=false;
        textBox3.ReadOnly=false;
        textBox4.ReadOnly=false;
        textBox5.ReadOnly=false;
        textBox6.ReadOnly=false;
        textBox7.ReadOnly=false;
        textBox8.ReadOnly=false;
    }
}
}
}

```

12.3.2 售货部门客户端开发

(1) 新建项目。

新建一个 Windows 应用程序项目，本例项目名称为 useSell。

(2) 界面设计。

图 12-4 是该程序的界面，其中所有 TextBox 控件的 ReadOnly 控件为 true。在图中，文本框自上而下分定为 textBox1, textBox2, textBox3, textBox4, textBox5, textBox6, textBox7, textBox8。单选按钮自上而下分定为 radioButton1, radioButton2, radioButton3。



图 12-4 售货部门客户端界面

下面是该程序的代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace useSell
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox8;
        private System.Windows.Forms.TextBox textBox7;
        private System.Windows.Forms.TextBox textBox6;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Button button1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.RadioButton radioButton3;
        private System.Windows.Forms.RadioButton radioButton2;
        private System.Windows.Forms.RadioButton radioButton1;
        /// <summary>
        /// 必需的设计器变量
        /// </summary>
        private System.ComponentModel.Container components = null;
        public Form1()
        {
            //
            // Windows 窗体设计器支持所必需的
            //
            InitializeComponent();
        }

        [STAThread]
        static void Main()
        {
            Application.Run(new Form1());
        }

        private void InitializeComponent()
        {
            this.groupBox2 = new System.Windows.Forms.GroupBox();
            this.label8 = new System.Windows.Forms.Label();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.label4 = new System.Windows.Forms.Label();
            this.label3 = new System.Windows.Forms.Label();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.textBox8 = new System.Windows.Forms.TextBox();
            this.textBox7 = new System.Windows.Forms.TextBox();
            this.textBox6 = new System.Windows.Forms.TextBox();
            this.textBox5 = new System.Windows.Forms.TextBox();
            this.textBox4 = new System.Windows.Forms.TextBox();
            this.textBox3 = new System.Windows.Forms.TextBox();
            this.textBox2 = new System.Windows.Forms.TextBox();
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.button1 = new System.Windows.Forms.Button();
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.radioButton3 = new System.Windows.Forms.RadioButton();
            this.radioButton2 = new System.Windows.Forms.RadioButton();
            this.radioButton1 = new System.Windows.Forms.RadioButton();
            this.SuspendLayout();
            // 
            // groupBox2
            //
            this.groupBox2.Location = new System.Drawing.Point(12, 12);
            this.groupBox2.Name = "groupBox2";
            this.groupBox2.Size = new System.Drawing.Size(250, 100);
            this.groupBox2.TabIndex = 1;
            this.groupBox2.TabStop = false;
            this.groupBox2.Text = "groupBox2";
            // 
            // label8
            //
            this.label8.Location = new System.Drawing.Point(12, 12);
            this.label8.Name = "label8";
            this.label8.Size = new System.Drawing.Size(100, 25);
            this.label8.TabIndex = 0;
            this.label8.Text = "label8";
            // 
            // label7
            //
            this.label7.Location = new System.Drawing.Point(12, 12);
            this.label7.Name = "label7";
            this.label7.Size = new System.Drawing.Size(100, 25);
            this.label7.TabIndex = 0;
            this.label7.Text = "label7";
            // 
            // label6
            //
            this.label6.Location = new System.Drawing.Point(12, 12);
            this.label6.Name = "label6";
            this.label6.Size = new System.Drawing.Size(100, 25);
            this.label6.TabIndex = 0;
            this.label6.Text = "label6";
            // 
            // label5
            //
            this.label5.Location = new System.Drawing.Point(12, 12);
            this.label5.Name = "label5";
            this.label5.Size = new System.Drawing.Size(100, 25);
            this.label5.TabIndex = 0;
            this.label5.Text = "label5";
            // 
            // label4
            //
            this.label4.Location = new System.Drawing.Point(12, 12);
            this.label4.Name = "label4";
            this.label4.Size = new System.Drawing.Size(100, 25);
            this.label4.TabIndex = 0;
            this.label4.Text = "label4";
            // 
            // label3
            //
            this.label3.Location = new System.Drawing.Point(12, 12);
            this.label3.Name = "label3";
            this.label3.Size = new System.Drawing.Size(100, 25);
            this.label3.TabIndex = 0;
            this.label3.Text = "label3";
            // 
            // label2
            //
            this.label2.Location = new System.Drawing.Point(12, 12);
            this.label2.Name = "label2";
            this.label2.Size = new System.Drawing.Size(100, 25);
            this.label2.TabIndex = 0;
            this.label2.Text = "label2";
            // 
            // label1
            //
            this.label1.Location = new System.Drawing.Point(12, 12);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(100, 25);
            this.label1.TabIndex = 0;
            this.label1.Text = "label1";
            // 
            // textBox8
            //
            this.textBox8.Location = new System.Drawing.Point(12, 12);
            this.textBox8.Name = "textBox8";
            this.textBox8.Size = new System.Drawing.Size(100, 25);
            this.textBox8.TabIndex = 0;
            this.textBox8.Text = "textBox8";
            // 
            // textBox7
            //
            this.textBox7.Location = new System.Drawing.Point(12, 12);
            this.textBox7.Name = "textBox7";
            this.textBox7.Size = new System.Drawing.Size(100, 25);
            this.textBox7.TabIndex = 0;
            this.textBox7.Text = "textBox7";
            // 
            // textBox6
            //
            this.textBox6.Location = new System.Drawing.Point(12, 12);
            this.textBox6.Name = "textBox6";
            this.textBox6.Size = new System.Drawing.Size(100, 25);
            this.textBox6.TabIndex = 0;
            this.textBox6.Text = "textBox6";
            // 
            // textBox5
            //
            this.textBox5.Location = new System.Drawing.Point(12, 12);
            this.textBox5.Name = "textBox5";
            this.textBox5.Size = new System.Drawing.Size(100, 25);
            this.textBox5.TabIndex = 0;
            this.textBox5.Text = "textBox5";
            // 
            // textBox4
            //
            this.textBox4.Location = new System.Drawing.Point(12, 12);
            this.textBox4.Name = "textBox4";
            this.textBox4.Size = new System.Drawing.Size(100, 25);
            this.textBox4.TabIndex = 0;
            this.textBox4.Text = "textBox4";
            // 
            // textBox3
            //
            this.textBox3.Location = new System.Drawing.Point(12, 12);
            this.textBox3.Name = "textBox3";
            this.textBox3.Size = new System.Drawing.Size(100, 25);
            this.textBox3.TabIndex = 0;
            this.textBox3.Text = "textBox3";
            // 
            // textBox2
            //
            this.textBox2.Location = new System.Drawing.Point(12, 12);
            this.textBox2.Name = "textBox2";
            this.textBox2.Size = new System.Drawing.Size(100, 25);
            this.textBox2.TabIndex = 0;
            this.textBox2.Text = "textBox2";
            // 
            // textBox1
            //
            this.textBox1.Location = new System.Drawing.Point(12, 12);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(100, 25);
            this.textBox1.TabIndex = 0;
            this.textBox1.Text = "textBox1";
            // 
            // button1
            //
            this.button1.Location = new System.Drawing.Point(12, 12);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(100, 25);
            this.button1.TabIndex = 0;
            this.button1.Text = "button1";
            // 
            // groupBox1
            //
            this.groupBox1.Location = new System.Drawing.Point(12, 12);
            this.groupBox1.Name = "groupBox1";
            this.groupBox1.Size = new System.Drawing.Size(250, 100);
            this.groupBox1.TabIndex = 1;
            this.groupBox1.TabStop = false;
            this.groupBox1.Text = "groupBox1";
            // 
            // radioButton3
            //
            this.radioButton3.Location = new System.Drawing.Point(12, 12);
            this.radioButton3.Name = "radioButton3";
            this.radioButton3.Size = new System.Drawing.Size(100, 25);
            this.radioButton3.TabIndex = 0;
            this.radioButton3.Text = "radioButton3";
            // 
            // radioButton2
            //
            this.radioButton2.Location = new System.Drawing.Point(12, 12);
            this.radioButton2.Name = "radioButton2";
            this.radioButton2.Size = new System.Drawing.Size(100, 25);
            this.radioButton2.TabIndex = 0;
            this.radioButton2.Text = "radioButton2";
            // 
            // radioButton1
            //
            this.radioButton1.Location = new System.Drawing.Point(12, 12);
            this.radioButton1.Name = "radioButton1";
            this.radioButton1.Size = new System.Drawing.Size(100, 25);
            this.radioButton1.TabIndex = 0;
            this.radioButton1.Text = "radioButton1";
            // 
            this.ResumeLayout(false);

        }
    }
}
```

```
//  
InitializeComponent();  
//  
// TODO: 在 InitializeComponent 调用后添加任何构造函数代码  
//  
}  
/// <summary>  
/// 清理所有正在使用的资源  
/// </summary>  
protected override void Dispose( bool disposing )  
{  
    if( disposing )  
    {  
        if (components != null)  
        {  
            components.Dispose();  
        }  
    }  
    base.Dispose( disposing );  
}  
  
//下面系统自动产生的代码已经删除  
#region Windows Form Designer generated code  
...  
...  
#endregion  
//上面系统自动产生的代码已经删除  
  
/// <summary>  
/// 应用程序的主入口点  
/// </summary>  
[STAThread]  
static void Main()  
{  
    Application.Run(new Form1());  
}  
private void radioButton1_CheckedChanged(object sender,  
System.EventArgs e)  
{  
    if(radioButton1.Checked)  
    {  
        textBox1.Text="insert";  
        textBox8.Text="";  
        textBox2.ReadOnly=false;  
        textBox3.ReadOnly=false;  
        textBox4.ReadOnly=false;
```

```
        textBox5.ReadOnly=false;
        textBox6.ReadOnly=false;
        textBox7.ReadOnly=false;
        textBox8.ReadOnly=true;
    }
}

private void radioButton2_CheckedChanged(object sender,
    System.EventArgs e)
{
    if(radioButton2.Checked)
    {
        textBox1.Text="delete";
        textBox8.ReadOnly=false;
        textBox2.ReadOnly=true;
        textBox3.ReadOnly=true;
        textBox4.ReadOnly=true;
        textBox5.ReadOnly=true;
        textBox6.ReadOnly=true;
        textBox7.ReadOnly=true;
    }
}

private void radioButton3_CheckedChanged(object sender,
    System.EventArgs e)
{
    if(radioButton3.Checked)
    {
        textBox1.Text="update";
        textBox2.ReadOnly=false;
        textBox3.ReadOnly=false;
        textBox4.ReadOnly=false;
        textBox5.ReadOnly=false;
        textBox6.ReadOnly=false;
        textBox7.ReadOnly=false;
        textBox8.ReadOnly=false;
    }
}

private void button1_Click(object sender, System.EventArgs e)
{
    zhou.Service1 myService=new zhou.Service1();
    if(radioButton1.Checked)
    {
        myService.sellEdit(textBox1.Text,(textBox2.Text,
            textBox3.Text,textBox4.Text,textBox5.Text,
            textBox6.Text,textBox7.Text,textBox8.Text);
    }
    if(radioButton2.Checked)
```

```

        {
            myService.sellEdit(textBox1.Text, textBox2.Text,
                textBox3.Text, textBox4.Text, textBox5.Text,
                textBox6.Text, textBox7.Text, textBox8.Text);
        }
        if (radioButton3.Checked)
        {
            myService.sellEdit(textBox1.Text, textBox2.Text,
                textBox3.Text, textBox4.Text, textBox5.Text,
                textBox6.Text, textBox7.Text, textBox8.Text);
        }
    }
}

```

12.3.3 财务部门客户端开发

(1) 新建项目。

新建一个 Windows 应用程序项目，本例项目名称为 buyEdition。

(2) 界面设计。

图 12-5 是该程序的界面。窗体最上面的是 dataGridView1。“商品经营盈亏输出”栏目里的 TextBox 控件为 textBox8。输入时间栏目里的文本框自左至右、自上而下分定为 textBox1, textBox2, textBox3, textBox4, textBox5, textBox6, textBox7。“操作选项”栏目里的单选按钮自上而下分定为 radioButton1, radioButton2, radioButton3, radioButton4。

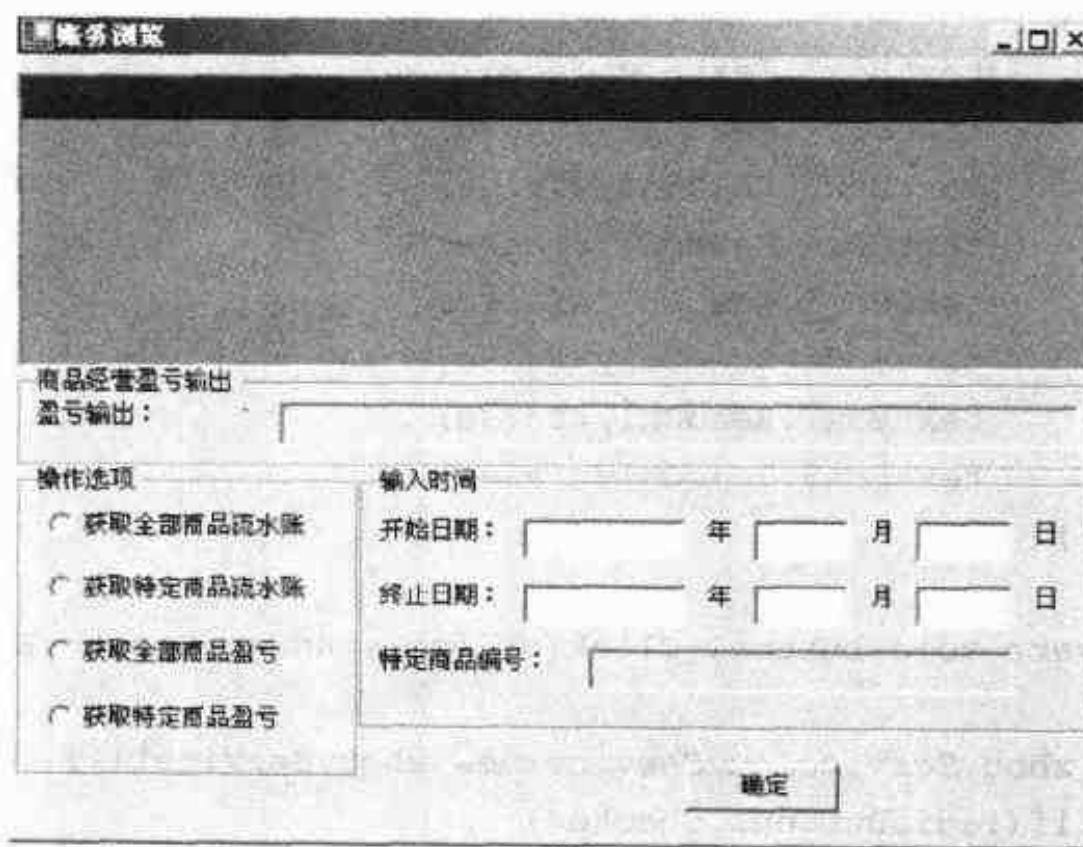


图 12-5 财务部门客户端界面

(3) 编写代码。

① “获取全部商品流水账”单选按钮的 CheckedChanged 事件的代码。

```
private void radioButton1_CheckedChanged(object sender, System.EventArgs e)
```

```
{  
    textBox7.ReadOnly=true;  
}
```

- ② “获取特定商品流水账”单选按钮的 CheckedChanged 事件的代码。

```
private void radioButton2_CheckedChanged(object sender, System.EventArgs e)  
{  
    textBox7.ReadOnly=false;  
}
```

- ③ “获取全部商品盈亏”单选按钮的 CheckedChanged 事件的代码。

```
private void radioButton3_CheckedChanged(object sender, System.EventArgs e)  
{  
    textBox7.ReadOnly=true;  
}
```

- ④ “获取特定商品盈亏”单选按钮的 CheckedChanged 事件的代码。

```
private void radioButton4_CheckedChanged(object sender, System.EventArgs e)  
{  
    textBox7.ReadOnly=false;  
}
```

- ⑤“确定”按钮的 Click 事件的代码。

```
private void button1_Click(object sender, System.EventArgs e)  
{  
    zhou.Service1 myService=new zhou.Service1();  
    if(radioButton1.Checked)  
    {  
        DataSet dataset=myService.allLiuShuiZhang(textBox1.Text,  
            textBox2.Text, textBox3.Text, textBox4.Text,  
            textBox5.Text, textBox6.Text);  
        dataGridView1.DataSource=dataset.Tables["表1"];  
    }  
    if(radioButton2.Checked)  
    {  
        DataSet dataset=myService.singleLiuShuiZhang(textBox1.Text,  
            textBox2.Text, textBox3.Text, textBox4.Text,  
            textBox5.Text, textBox6.Text, textBox7.Text);  
        dataGridView1.DataSource=dataset.Tables["表1"];  
    }  
    if(radioButton3.Checked)  
    {  
        textBox8.Text=myService.allInOut(textBox1.Text,  
            textBox2.Text, textBox3.Text, textBox4.Text,  
            textBox5.Text, textBox6.Text);  
    }  
}
```

```
        }

        if(radioButton4.Checked)
        {

            textBox8.Text=myService.singleInOut(textBox7.Text,
                textBox1.Text, textBox2.Text, textBox3.Text,
                textBox4.Text, textBox5.Text, textBox6.Text);
        }
    }
```

下面是该程序的主要代码：

```
using System;
using System.Drawing;
using System.Collections;
using System.ComponentModel;
using System.Windows.Forms;
using System.Data;

namespace accountant
{
    /// <summary>
    /// Form1 的摘要说明
    /// </summary>
    public class Form1 : System.Windows.Forms.Form
    {
        private System.Windows.Forms.DataGridView dataGridView1;
        private System.Windows.Forms.GroupBox groupBox1;
        private System.Windows.Forms.GroupBox groupBox2;
        private System.Windows.Forms.RadioButton radioButton1;
        private System.Windows.Forms.RadioButton radioButton2;
        private System.Windows.Forms.RadioButton radioButton3;
        private System.Windows.Forms.RadioButton radioButton4;
        private System.Windows.Forms.Label label1;
        private System.Windows.Forms.TextBox textBox1;
        private System.Windows.Forms.Label label2;
        private System.Windows.Forms.TextBox textBox2;
        private System.Windows.Forms.Label label3;
        private System.Windows.Forms.TextBox textBox3;
        private System.Windows.Forms.Label label4;
        private System.Windows.Forms.Label label5;
        private System.Windows.Forms.TextBox textBox4;
        private System.Windows.Forms.Label label6;
        private System.Windows.Forms.TextBox textBox5;
        private System.Windows.Forms.Label label7;
        private System.Windows.Forms.TextBox textBox6;
        private System.Windows.Forms.Label label8;
        private System.Windows.Forms.Label label9;
```

```
private System.Windows.Forms.TextBox textBox7;
private System.Windows.Forms.Button button1;
private System.Windows.Forms.GroupBox groupBox3;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.TextBox textBox8;
/// <summary>
/// 必需的设计器变量
/// </summary>
private System.ComponentModel.Container components = null;

public Form1()
{
    //
    // Windows 窗体设计器支持所必需的
    //
    InitializeComponent();

    //
    // TODO: 在 InitializeComponent 调用后添加任何构造函数代码
    //
}

/// <summary>
/// 清理所有正在使用的资源
/// </summary>
protected override void Dispose( bool disposing )
{
    if( disposing )
    {
        if (components != null)
        {
            components.Dispose();
        }
    }
    base.Dispose( disposing );
}

//下面系统自动产生的代码已经删除
#region Windows Form Designer generated code
.....
.....
#endregion
//上面系统自动产生的代码已经删除
/// <summary>
/// 应用程序的主入口点
/// </summary>
[STAThread]
```

```
static void Main()
{
    Application.Run(new Form1());
}

private void radioButton1_CheckedChanged(object sender,
    System.EventArgs e)
{
    textBox7.ReadOnly=true;
}

private void radioButton2_CheckedChanged(object sender,
    System.EventArgs e)
{
    textBox7.ReadOnly=false;
}

private void radioButton3_CheckedChanged(object sender,
    System.EventArgs e)
{
    textBox7.ReadOnly=true;
}

private void radioButton4_CheckedChanged(object sender,
    System.EventArgs e)
{
    textBox7.ReadOnly=false;
}

private void button1_Click(object sender, System.EventArgs e)
{
    zhou.Service1 myService=new zhou.Service1();
    if(radioButton1.Checked)
    {
        DataSet dataset=myService.allLiuShuiZhang(textBox1.Text,
            textBox2.Text, textBox3.Text, textBox4.Text,
            textBox5.Text, textBox6.Text);
        dataGridView1.DataSource=dataset.Tables["表 1"];
    }
    if(radioButton2.Checked)
    {
        DataSet dataset=myService.singleLiuShuiZhang
            (textBox1.Text, textBox2.Text, textBox3.Text,
            textBox4.Text, textBox5.Text,
            textBox6.Text, textBox7.Text);
        dataGridView1.DataSource=dataset.Tables["表 1"];
    }
    if(radioButton3.Checked)
    {
```

```
        textBox8.Text=myService.allInOut(textBox1.Text,
            textBox2.Text,(textBox3.Text, textBox4.Text,
            textBox5.Text, textBox6.Text);
    }
    if (radioButton4.Checked)
    {
        textBox8.Text=myService.singleInOut(textBox7.Text,
            textBox1.Text, textBox2.Text, textBox3.Text,
            textBox4.Text, textBox5.Text, textBox6.Text);
    }
}
}
```

12.3.4 管理（经理）部门客户端开发

同 12.3.3 小节，这里不再赘述。

12.3.5 演示

下面简单演示一下商贸财务系统。为了方便，有关数据的插入、删除和修改方面，只在进货部门演示，不再在售货部门演示。售货部门的数据操作可参照进货部门进行。流水账、经营盈亏只在财务部门演示，不再在管理（经理）部门演示。

1. 进货数据插入

如图 12-6 所示，在“操作选项”栏里选中“插入新数据”单选按钮，然后在文本框里输入相应的商品品息，最后单击“确定”按钮即可。



图 12-6 数据插入

2. 数据修改

如图 12-7 所示，在“操作选项”栏里选中“修改旧数据”单选按钮，然后在文本框里输入相应的商品品息，最后单击“确定”按钮即可。



图 12-7 数据修改

3. 删除数据

如图 12-8 所示，在“操作选项”栏里选中“删除旧数据”单选按钮，然后在文本框里输入相应的商品品息，最后单击“确定”按钮即可。



图 12-8 数据删除

发票编号	日期	商品编号	商品名称	买进数量	买进单价	卖出
0000001	2002-3-11	A0001	打印机	100	350	0
0000002	2002-3-12	A0002	扫描仪	120	420	0
0000003	2002-3-15	A0001	打印机	0	0	50
0000004	2002-3-16	A0002	扫描仪	0	0	60
0000005	2002-3-22	A0001	打印机	0	0	36

商品经营盈亏输出
盈亏输出： []

操作选项

- 获取全部商品流水账
- 获取特定商品流水账
- 获取全部商品盈亏
- 获取特定商品盈亏

输入时间

开始日期： 2002 年 3 月 11 日

终止日期： 2002 年 5 月 26 日

特定商品编号： []

确定

图 12-9 获取全部商品流水账

4. 获取全部商品流水账

如图 12-9 所示，在“操作选项”栏里选中“获取全部商品流水账”单选按钮，然后输入起止日期，最后单击“确定”按钮即可。

5. 获取特定商品流水账

如图 12-10 所示，在“操作选项”栏里选中“获取特定商品流水账”单选按钮，然后输入起止日期和特定商品的商品编号，最后单击“确定”按钮即可。

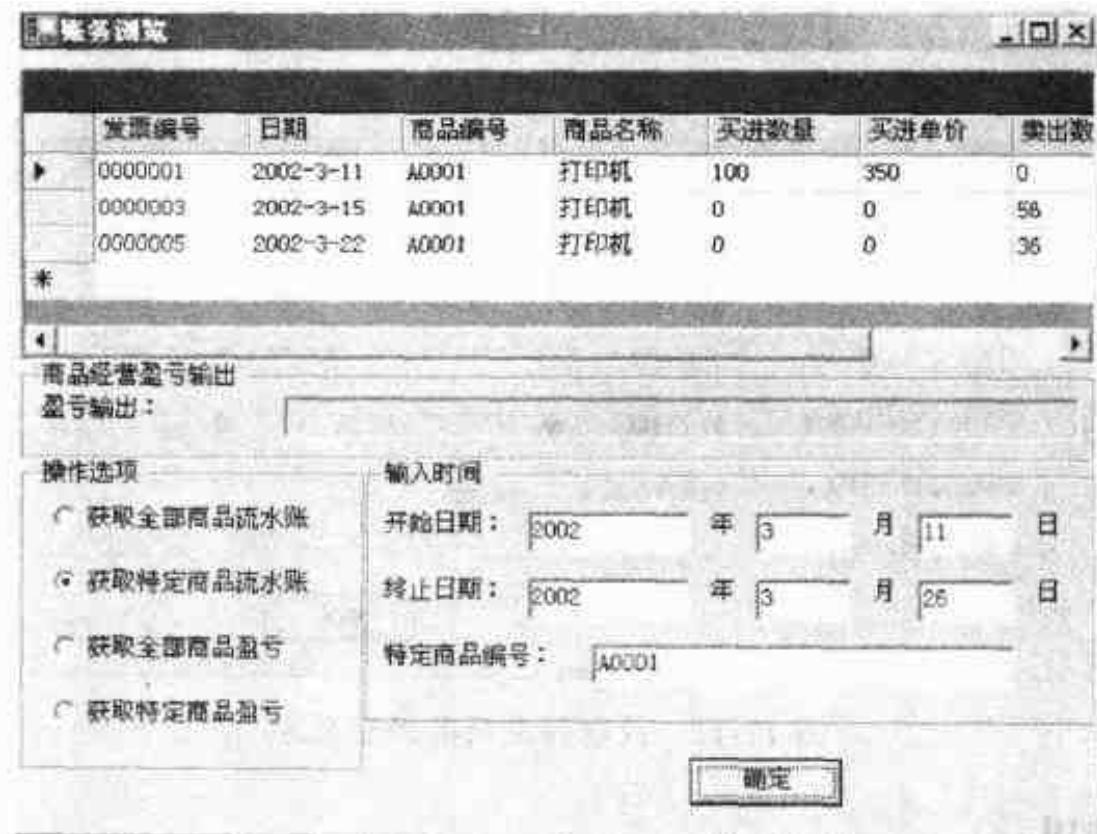


图 12-10 获取特定商品流水账

6. 获取全部商品经营的盈亏

如图 12-11 所示，在“操作选项”栏里选中“获取全部商品盈亏”单选按钮，然后输入起止日期，最后单击“确定”按钮即可。在“商品经营盈亏输出”栏里输出的查询结果是“46035.1”。

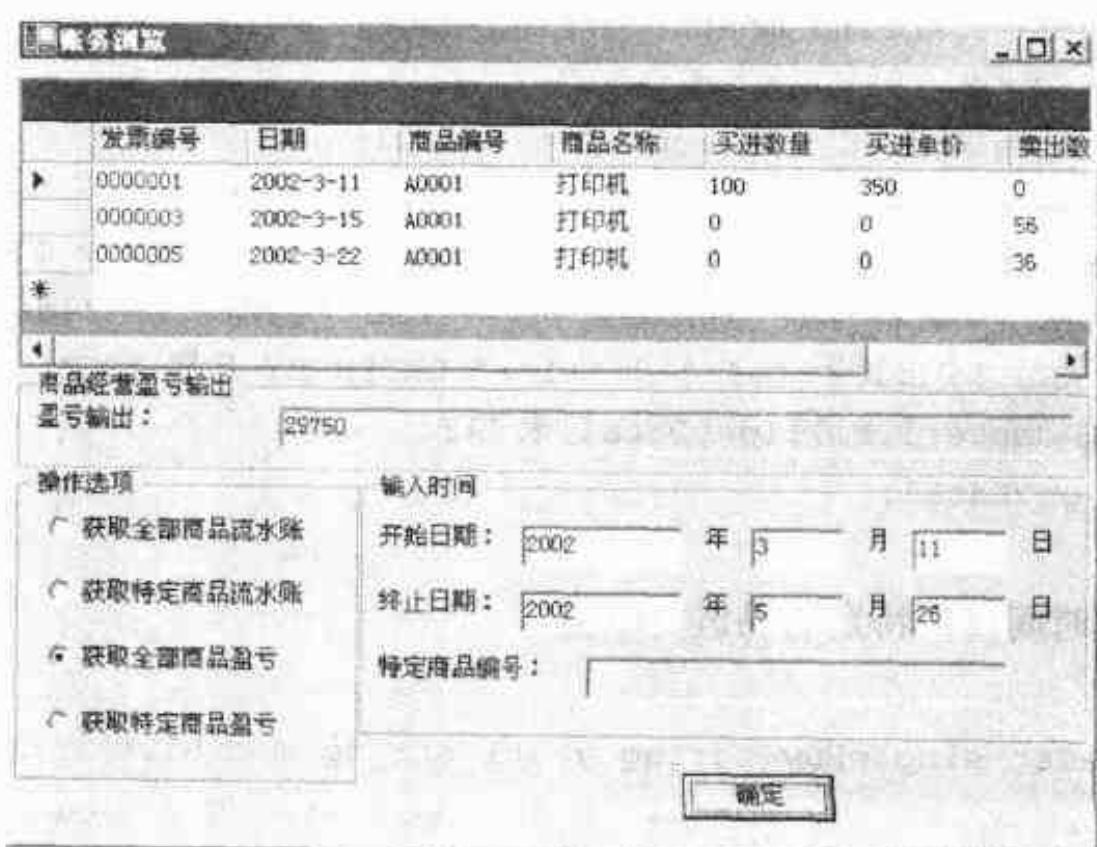


图 12-11 获取全部商品经营的盈亏

7. 获取特定商品经营盈亏

如图 12-12 所示，在“操作选项”栏里选中“获取特定商品盈亏”单选按钮，然后输入起止日期和特定商品的编号，最后单击“确定”按钮即可。在“商品经营盈亏输出”栏里输出的查询结果是“33594.8”。

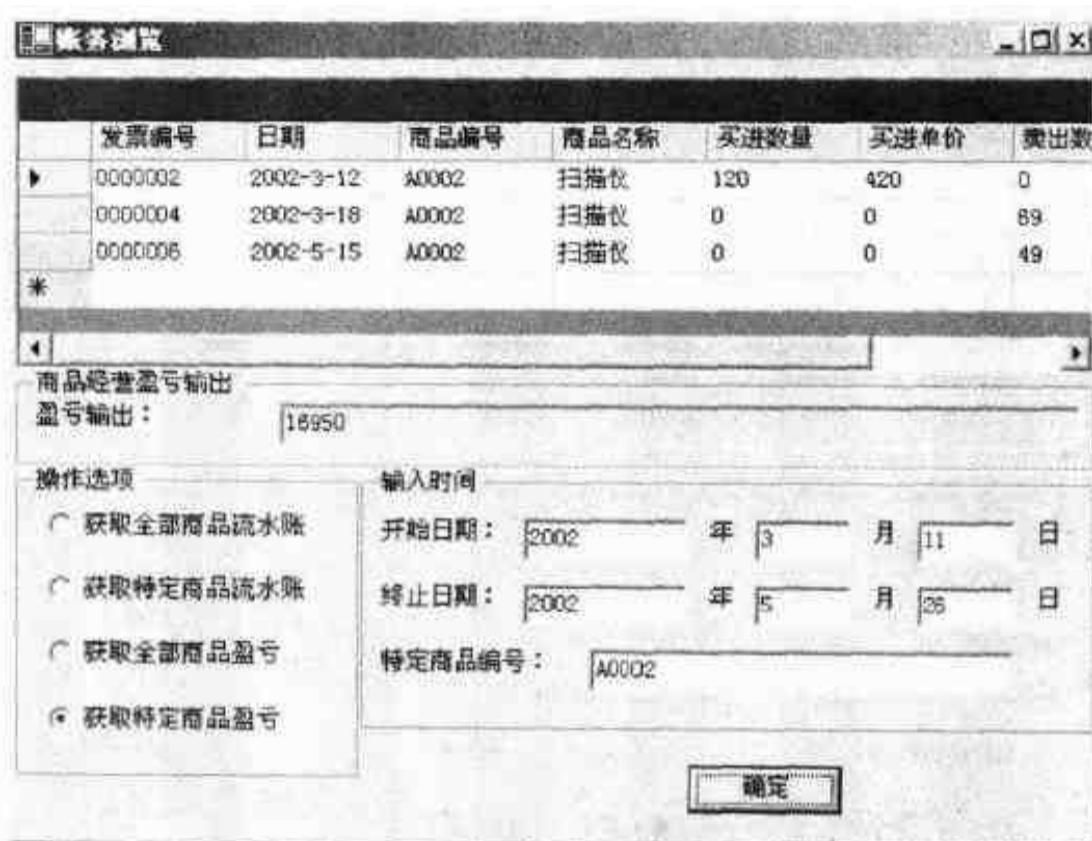


图 12-12 获取特定商品经营盈亏

12.3.6 改进意见

除了上述基本功能外，财务管理系统还应具有查询全部进货、全部销售、特定商品进货、特定商品售货的数据。下面是参考代码：

1. 获取特定时间段内全部进货账目

```
[WebMethod]
public DataSet allBuy(string year1, string month1, string day1,
    string year2, string month2, string day2)
{
    string date1=year1+"-"+month1+"-"+day1;
    string date2=year2+"-"+month2+"-"+day2;
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表1
        WHERE (日期>='"+date1+"') "+" AND (日期<='"+date2+"') "
        +" AND (卖出数量='0+') "+" ORDER BY 日期 ASC";
    sqlDataAdapter1.Fill(getData1.表1);
    return getData1;
}
```

2. 获取特定时间段内特定商品进货账目

```
[WebMethod]
public DataSet singleBuy(string year1, string month1, string day1,
    string year2, string month2, string day2, string shangpinbianhao)
{
```

```
string date1=year1+"-"+month1+"-"+day1;
string date2=year2+"-"+month2+"-"+day2;
sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表1
WHERE (日期>='"+date1+"') AND (日期<='"+date2+"')
AND (商品编号='"+shangpinbianhao+"') AND (卖出数量="
+"0+) ORDER BY 日期 ASC";
sqlDataAdapter1.Fill(getData1.表1);
return getData1;
//
```

3. 获取特定时间段内全部售货账目

```
[WebMethod]
public DataSet allSell(string year1,string month1,string day1,
string year2,string month2,string day2)
{
    string date1=year1+"-"+month1+"-"+day1;
    string date2=year2+"-"+month2+"-"+day2;
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表1
WHERE (日期>='"+date1+"') AND (日期<='"+date2+"')
AND (买进数量='0+') ORDER BY 日期 ASC";
    sqlDataAdapter1.Fill(getData1.表1);
    return getData1;
}
```

4. 获取特定时间段内特定商品售货账目

```
[WebMethod]
public DataSet singleSell(string year1,string month1,string day1,
string year2,string month2,string day2,string shangpinbianhao)
{
    string date1=year1+"-"+month1+"-"+day1;
    string date2=year2+"-"+month2+"-"+day2;
    sqlDataAdapter1.SelectCommand.CommandText="SELECT * FROM 表1
WHERE (日期>='"+date1+"') AND (日期<='"+date2+"')
AND (商品编号='"+shangpinbianhao+"') AND (买进数量="
+"0+) ORDER BY 日期 ASC";
    sqlDataAdapter1.Fill(getData1.表1);
    return getData1;
}
```

本章小结

本章使用 XML Web Services 技术和 Win32 技术，开发了一个完整的分布式财务管理系
统。通过该系统的开发，读者可系统掌握 Web 开发和数据库综合开发技术，为构建企业级
应用软件打下坚实的基础。

[G e n e r a l I n f o r m a t i o n]

书名 = Visual C# .NET 网络核心编程

作者 =

页数 = 1000

SS号 = 0

出版日期 =

封面
书名
版权
前言
目录
正文