软·件·工·程·师·典·藏

ASP.NET 开发

典型模块大全

（修订版）

■ 明日科技 房大伟 庞娅娟 编著

27个实战模块，43个编程完美解决方案
5项核心技术，涵盖ASP.NET开发的方方面面
每周6天，每天13小时答疑服务

超 值 光 盘

书中所有模块的源代码
348分钟多媒体语音视频教学录像
《ASP.NET 编程词典体验版》

人民邮电出版社
POSTS & TELECOM PRESS

软·件·工·程·师·典·藏

本书提供的模块源代码可直接运用于二次开发，提高开发效率

基础功能开发	会员注册登录模块	密码找回模块	上传与下载模块	万能打印模块
数据库	博客模块	在线考试模块	数据备份与数据恢复模块	网络硬盘模块
	系统权限分配模块			
图形图像与多媒体	在线音乐模块	电子相册模块	播客模块	图片资源管理模块
网络通信	电子邮件发送与接收模块	工作记录管理模块	论坛模块	网站备忘录
	手机短消息管理模块	聊天室模块	基于XML技术的留言本模块	Rss在线订阅模块
电子商务	购物车模块	在线银行支付模块	网上客户管理模块	电子邮件模块
查询统计与分析	投票系统模块	网上问卷调查模块	搜索引擎模块	网站统计分析模块
高级开发	Ajax技术	LINQ数据访问技术	服务类技术	网站统计分析技术

超值光盘

书中所有模块的源代码

348分钟多媒体语音视频教学录像

《ASP.NET 编程词典体验版》（相当于1000页电子文档，即学即用，多种方式查询，免费升级）

超值配套服务

服务网站：www.mingribook.com

QQ：100310286 100310063

服务信箱：mingrisoft@mingrisoft.com

或 mingrisoft@vip.sina.com

服务电话：0431-84978981/84978982

装帧设计：董志桢

分类建议：计算机/程序设计/ASP.NET

人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-22587-0

ISBN 978-7-115-22587-0

定价：89.80 元（附光盘）

ASP.NET

开发

典型模块大全

（修订版）

■ 明日科技 房大伟 庞娅娟 编著

人 民 邮 电 出 版 社

北 京

图书在版编目（CIP）数据

ASP.NET开发典型模块大全 / 房大伟，庞娅娟编著
. -- 2版（修订本）. -- 北京 : 人民邮电出版社，
2010.5
（软件工程师典藏）
ISBN 978-7-115-22587-0

Ⅰ. ①A… Ⅱ. ①房… ②庞… Ⅲ. ①主页制作－程序
设计 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2010)第046129号

内 容 提 要

本书以关键技术和热点技术为核心，通过 27 个典型模块和 5 章热点技术，全面地介绍了如何使用 ASP.NET 进行各领域的 Web 项目开发。全书共 3 篇分为 32 章，第 1 篇关键模块篇，覆盖网站开发的关键领域，内容涉及论坛、博客、播客、网络硬盘、电子邮件、在线考试、网站备忘录、在线短消息、网站访问量统计与分析、系统后台管理权限分配等网站关键模块；第 2 篇常见模块篇，覆盖网站开发的各个领域，内容涉及网站会员注册及登录、会员密码找回、留言本、上传与下载、图片资源管理、搜索引擎、网上问卷调查、RSS 在线订阅、聊天室、购物车、在线银行支付、手机短消息管理、在线音乐、投票系统、万能打印、数据自动备份与恢复等常见模块；第 3 篇热点技术应用篇，解决网站开发在某个领域遇到的技术难题，内容涉及 LINQ 数据访问技术、安全技术、服务技术、Ajax、高级应用技术等。

本书附有配套光盘。光盘提供了书中所有案例的全部源代码，并经过精心调试，在 Windows XP 和 Windows 2000 下全部通过，保证能够正常运行。此外，光盘中还提供有编程词典试用版软件。

本书案例涉及领域广泛，实用性非常强。学习本书读者可以了解各个领域的特点，能够针对某一行业进行软件开发，也可以通过光盘中提供的模块源代码进行二次开发，以减少开发系统所需要的时间。本书适合各级软件开发人员学习使用，也可供大、中专院校师生学习参考。

软件工程师典藏

ASP.NET 开发典型模块大全（修订版）

◆ 编　　著　明日科技　房大伟　庞娅娟
责任编辑　黄　焱

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京隆昌伟业印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：45.5
字数：1 230 千字　　2010 年 5 月第 2 版
印数：5 001 - 9 000 册　　2010 年 5 月北京第 1 次印刷

ISBN 978-7-115-22587-0

定价：89.80 元（附光盘）

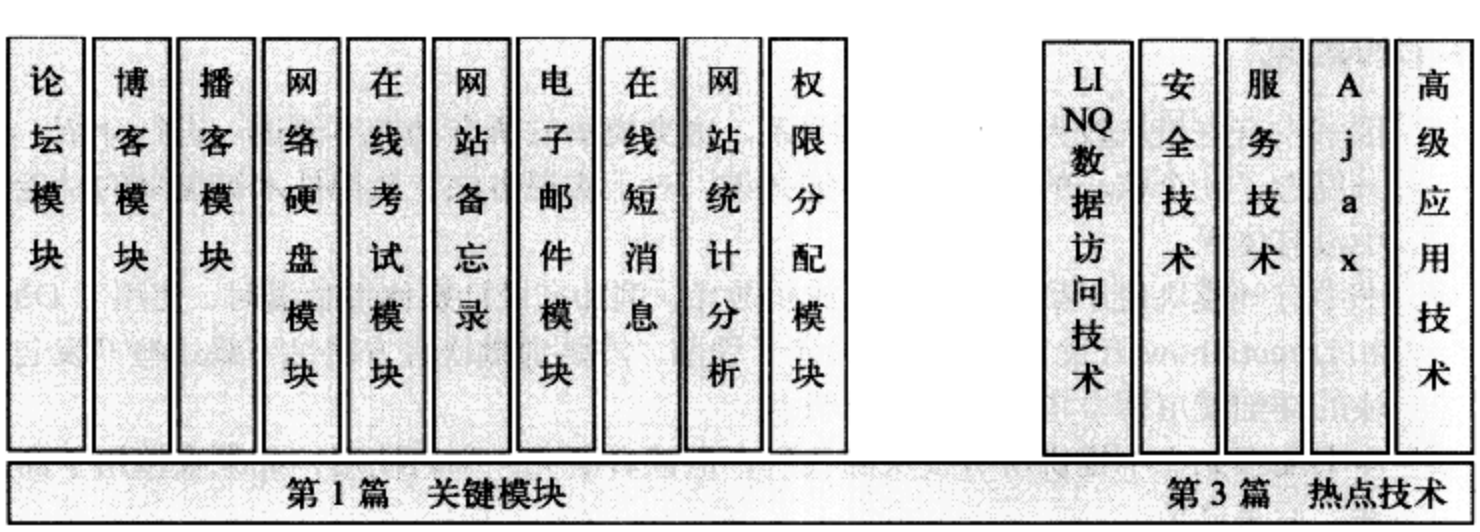
读者服务热线：(010)67132692　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前 言

作为项目开发人员，相信许多人都有过由于项目紧迫而加班加点的经历。如果有现成的模块作为参考，项目的开发进度会大幅度提高。本书提供了 27 个典型模块和 5 个方向的热点技术，几乎涵盖了 Web 项目开发的各个方面。本书以模块的核心技术为导向，介绍模块的设计思路和实现过程，特别适合作为项目开发人员的参考书。

本书内容

本书共 3 篇，分为 32 章，涉及 27 个典型模块和 5 章热点技术，第 1 篇关键模块篇，在本书的前 10 章分别覆盖网站开发的关键领域，内容涉及论坛、博客、播客、网络硬盘、电子邮件、在线考试、网站备忘录、在线短消息、网站访问量统计与分析、系统后台管理权限分配等网站关键模块；第 2 篇常见模块篇，通过 17 个常见模块覆盖网站开发的各个领域，内容涉及网站会员注册及登录、会员密码找回、留言本、上传与下载、图片资源管理、搜索引擎、网上问卷调查、RSS 在线订阅、聊天室、购物车、在线银行支付、手机短消息管理、在线音乐、投票系统、万能打印、数据自动备份与恢复等常见模块；第 3 篇热点技术应用篇，解决网站开发在某个领域遇到的技术难题，内容涉及 LINQ 数据访问技术、安全技术、服务技术、Ajax、高级应用技术等。每项专题对应模块如下图所示。



本书特点

- 突出核心技术，注重解决方案

书中的模块以设计思路开始，介绍模块的各种设计方法，然后介绍实现模块需要使用的各项技术。使读者掌握实现模块的多项技术和方法。

- 模块典型，涉及范围广泛

书中的模块均是作者经过反复斟酌、精挑细选的，范围涉及程序设计和网络开发的各个方面，尽量做到读者每设计一个模块，都能从书中获得实现技术和方法。

- 代码规范，注释全面

书中模块代码在注重代码执行效率的同时，是笔者严格按照统一代码缩进、统一命名规范的原则精心编写的。代码注释规范而且非常全面，关键代码和函数几乎每行语句均有注释。

- 注意\说明\技巧特色标识

在介绍模块的过程中，为了扩展知识面、提示读者容易出错的地方、提供开发中的经验、技巧等，书中随处可以见到注意、说明、技巧等提示性信息。例如：

说 明

内容页是用户通过浏览器调用的资源，它是一种特殊的网页类型（即绑定到一个母版页的网页），它只能包含 Content 控件。并且在<asp:Content>标签外部是不允许有服务器控件或文本文字的。同时在一个内容页中可以有一个或多个 Content 控件，但它们必须指向母版中的不同占位符。

- 光盘内容超值、赠品丰富

本书附带光盘中不但提供了书中所有模块的源代码，还提供了编程词典试用版学习软件。

本书约定

- 因书中有些模块较大，限于篇幅关系，这类模块的所有功能不能逐一进行介绍，作者只筛选了几个核心的功能进行详细介绍，对于实现方法、使用技术等重复的功能，文中没有体现。
- 书中有些模块使用第 3 方的工具或函数库。例如在设计媒体播放器时，使用了 DirectX 和 Direct Show 开发包，用户在编译工程前，需要到微软官方网站下载这些开发包。模块的详细使用请参考光盘使用说明书。
- 本书配套光盘中提供所有模块源代码，供读者研究、学习使用，如果直接用于商业用途，后果自负。

读者人群

- ☑ ASP.NET 初学者
- ☑ 编程爱好者
- ☑ 大、中专院校的教师和学生
- ☑ 一线软件开发人员
- ☑ 培训机构的老师和学员
- ☑ 初、中级程序开发人员

技术支持

本书由明日科技组织编写，参加编写的有张跃廷、苏宇、房大伟、贾伟红、刘玲玲、刘欣、梁晓岚、梁冰、顾彦玲、黄锐、杨丽、刘春芬、刘书娟、刘爽、寇长梅、董大永、李明霞、张艳、乔敏、李贺、高春艳、邹天思、潘凯华、刘中华、孙明丽、庞娅娟、吕继迪、孙鹏、王小科、吕双、梁水、刘彬彬、安剑、孙秀梅、赛奎春、宋坤、王国辉、李钟尉、马文强、王殊宇、陈丹丹、王毅、

尹相群等。由于作者水平有限，错漏之处在所难免，请广大读者批评指正。如果读者在使用本书时遇到问题，可以通过明日科技网站进行咨询，我们承诺在 5 个工作日以内给您及时回复。明日科技图书服务网站是 www.mingribook.com，服务信箱是 mingrisoft@mingrisoft.com，服务电话是 0431-84978981/84978982。

书山有路勤为径，学海无涯苦作舟。希望读者通过本书的学习，能够掌握网络开发各个方面的技能，将其融会贯通，应用到实际的工作中去，成为网络开发领域的精英。

明日科技

2008 年 11 月

目 录

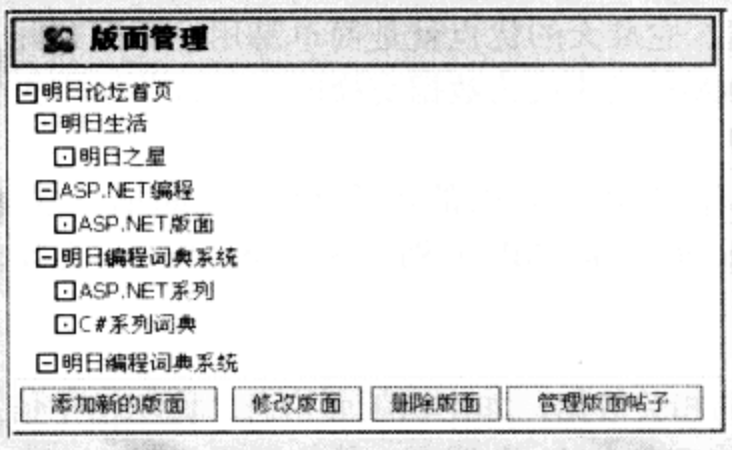
论坛模块

第1章

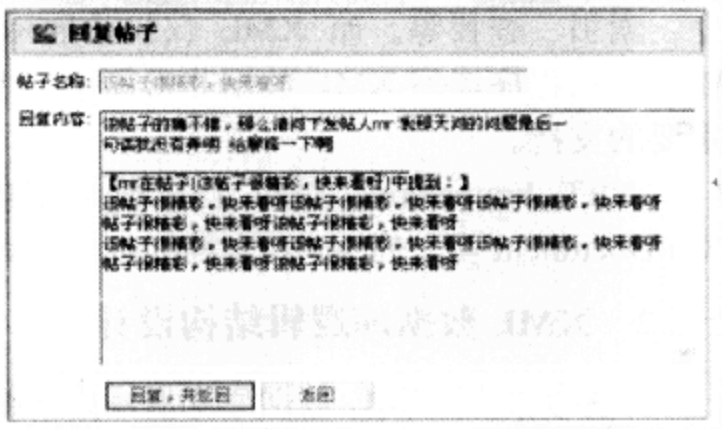
实例位置：光盘\mr\01\

随着网络的普及，论坛的功能越来越强大，大部分的商业网站、技术网站以及个人网站纷纷在自己的网站上开辟论坛，作为网站与访问者、访问者与访问者之间交流的平台，同时还提供在线技术支持和在线服务等功能。在信息交流的过程中使信息能够共享，使访问者获取更多、更新的信息，网站管理者通过论坛能够快速发现问题并解决问题，在不断积累经验的过程中又可以发布新的信息反馈给网站访问者，由此可见，论坛在人们生活中的重要性，论坛模块是一种值得掌握的程序，本论坛模块由 XML 技术实现，使得论坛开发更加方便快捷。通过本章的学习，读者能够学到以下内容。

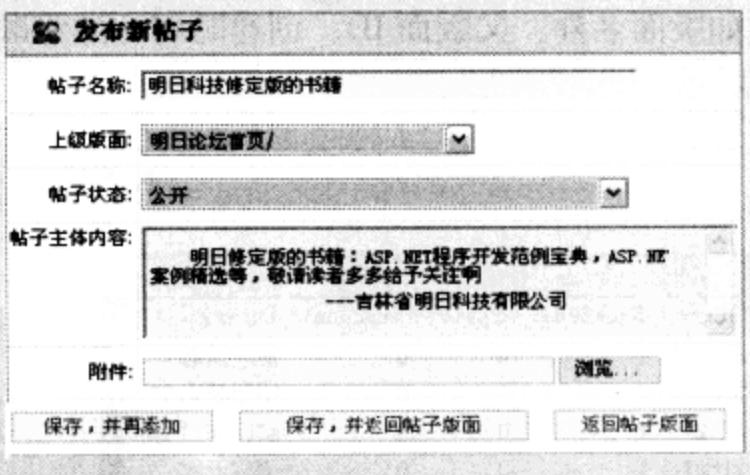
论坛版面管理



论坛帖子回复

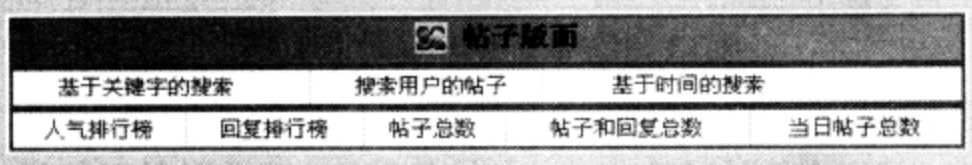


发布论坛新帖



基于时间搜索帖子

帖子版面相关功能



1.1 概　　述

在网络技术发展迅速、信息交流频繁的今天，论坛的出现使信息交流更为方便快捷。论坛也称为 BBS（Bulletin Board System），即电子公告板。论坛是用于发布公告信息和访问者之间讨论问题的在线交流平台。

本论坛模块将介绍如何创建一个基于 XML 技术的 BBS 系统（网站或应用程序），用户可以在这个论坛模块中发布自己的帖子并进行讨论等，管理员进入后台后可以对论坛版面、用户信息进行管理等操作，用户帖子及管理员版面等信息都存储在 XML 文件中。

1.2 XML 数据库设计

1.2.1 XML 数据库概述

XML（Extensible Markup Language），是一种类似于 HTML 语言的标记语言。XML 以简易而标准的方式保存各种信息（如文字和数字等信息），它适用于不同应用程序间的数据交换，而这种交换不以预先定义的一组数据结构为前提，增强了可扩展性。XML 与 Access、Oracle、SQL Server 等数据库不同。数据库一般提供了更强有力的数据存储和分析能力，如数据的排序、索引、查找等。而 XML 仅仅是展示数据，它最大的优点就是简单易用。正因如此，才使得应用程序读写 XML 数据非常简单，这为 XML 很快成为数据交换的惟一公共语言提供了重要的支持。

.NET Framework 提供了处理丰富的操作 XML 文档的类和组件，如描述文档的 XmlDocument 类，写 XML 文档的 XmlTextWrite 类，读 XML 文档的 XmlTextReader 类等。

1.2.2 XML 数据库逻辑结构设计

本系统采用 XML 数据库来展示论坛帖子的相关数据，如论坛版面信息、论坛帖子信息和帖子回复信息等。这里将主要介绍本应用程序的 3 个 XML 数据表，具体说明如下。

1. 论坛版面表 Board

论坛版面表 Board 用来存储版面的信息，如版面名称、父版面 ID、创建时间、显示顺序、状态、子版面数量等，表的字段说明如图 1.1 所示。

XmlDatabase/Board.xml

数据表: 数据:

Board

Board 的数据

	ID	Name	ParentID	CreateDate	ShowOrder	State	SubCount	Remark
▶	0	明日论坛首	-1	2008-8-25 0	1	0	0	备注信息
	1	明日生活	0	2008-9-26 1	1	0	0	明日精彩生
	2	ASP.NET编程	0	2008-10-27	1	0	0	ASP.NET网站
	3	明日科技宝	0	2008-11-27	1	0	0	.NET开发
*								

图 1.1　论坛版面表 Board 的字段说明

2. 论坛帖子表 Title

论坛帖子表 Title 用于存储帖子的信息，如帖子名称、发布该帖子的用户 ID、该帖子所属版面的 ID、帖子的具体内容等。该表的字段说明如图 1.2 所示。

XmlDatabase/Title.xml

数据表: 数据:

Title

Title 的数据

	ID	Name	Body	UserID	BoardID	CreateDate	VisitNum	ReplyNum	State
▶	0	默认	默认	1	0	2008-01-25	5	0	0
	1	ASP.NET程序	明日科技修	1	1	2008-4-25	3	0	0
	2	.NET编程	public void	1	0	2008-5-25	9	0	0
	3	ASP.NET编程	ASP.NET编程	1	0	2008-6-25	21	0	0
	4	mrguanweiho	mrfangdawei	1	0	2008-7-25	5	0	0
	5	2008.08.08	08年北京奥	1	0	2008-8-25	36	1	0
	6	明日理念	用今日的辛	1	1	2008-6-25	9	2	0
	7	明日编程词	明日编程词	1	3	2008-5-25	16	1	0
	8	明日好朋友	明日科技ASP	1	2	2008-11-25	9	3	0
	9	我爱明日科	我爱明日科	39	0	2008-4-17	4	1	0
	10	该帖子很精	该帖子很精	1	0	2008-4-18	5	0	0
	11	顶	我顶，我顶	1	0	2008-4-18	0	0	0
	12	超好看的图	超好看的图	1	0	2008-4-18	1	0	0
	13	手机	手机手机	1	3	2008-4-18	0	0	0
	14	888888	我的手机号	1	0	2008-4-18	6	1	0
*									

图 1.2　论坛帖子表 Title 的字段说明

3. 帖子回复表 Reply

帖子回复表 Reply 用来存储帖子回复的信息，如帖子的主体内容、发布该帖回复人的 ID、所回复的帖子 ID、帖子创建时间等。表的字段说明如图 1.3 所示。

XmlDatabase/Reply.xml

数据表: 数据:

Reply

Reply 的数据

	ID	Body	UserID	TitleID	CreateDate
▶	8	明日科技	1	7	2008-1-28
	9	感谢你对明	1	9	2008-4-23
	10		1	10	2008-4-23
	11	明日科技，	1	9	2008-4-23
	12	感觉明日科	1	3	2008-4-23
	13	代码就这一	1	2	2008-4-23
*					

图 1.3　帖子回复表 Reply 的字段说明

1.3　关键技术详解

本论坛模块主要是基于 XML 技术的在线论坛，其主要关键技术就是如何操作 XML 数据库中的数据，如定义操作 XML 数据库的参数、读取 XML 文件中的数据、更新 XML 文件中的数据等。应用 XML 文件存储数据可有效地提高论坛帖子发布、回复等显示速度。

1.3.1　定义操作 XML 数据库的参数

在介绍操作 XML 数据库方法之前，首先介绍这些方法所使用的参数（Paramter）。在此，笔者把这些方法操作 XML 数据库时所使用的参数称为 XML 参数，每一个参数都被定义为 XmlParamter 类型。该类主要包括的属性有 Name、Value 和 Direction，分别表示参数的名称、

值和方向。定义的 XmlParamter 类的程序代码如下。

例程 1 代码位置：光盘\mr\01\XMLBBS\ App_Code\ XmlDatabase.cs

```
//定义该类为密封类，阻止该类被继承
public sealed class XmlParamter
{
    //实例构造函数
  public XmlParamter(){}
  //定义属性Name
    private string name;
    public string Name
    {
        get { return this.name; }
        set { this.name = value; }
    }
  //定义属性Value
    private string value;
    public string Value
        {
            get { return this.value; }
        set { this.value = value; }
    }
  //定义属性Direction
    private ParameterDirection direction;
    public ParameterDirection Direction
    {
        get { return this.direction; }
        set { this.direction = value; }
    }
}
```

其中，参数方向定义为枚举类型 ParameterDirection。该枚举类型包含 7 个枚举值，具体说明如下。

- Insert：表示操作此参数的方法为“插入”。
- Update：表示操作此参数的方法为“更新”。
- Equal：表示操作此参数的方法为“相等”。
- NotEqual：表示操作此参数的方法为“不相等”。
- Little：表示操作此参数的方法为“小于”。
- Great：表示操作此参数的方法为“大于”。
- Like ：表示操作此参数的方法为“模糊匹配”。

定义枚举类型 ParameterDirection 的程序代码如下。

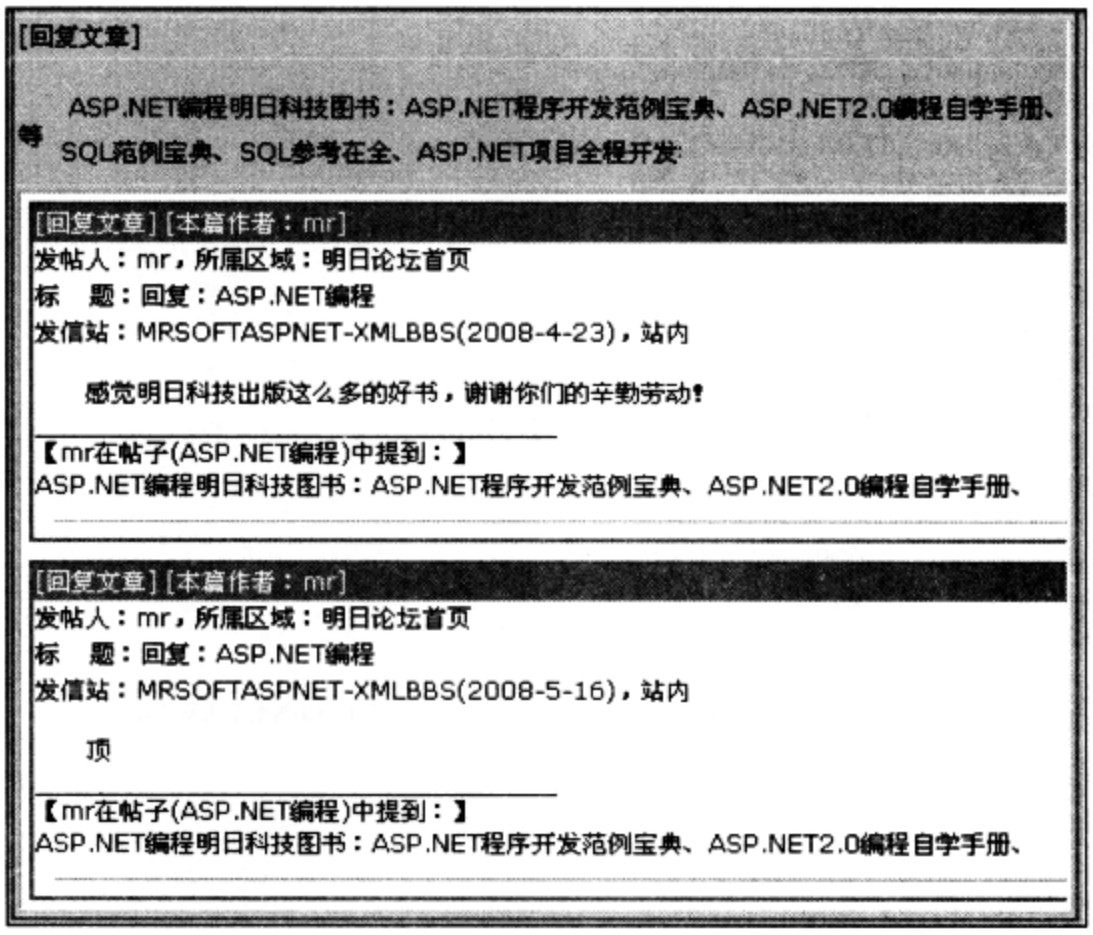
例程 2 代码位置：光盘\mr\01\XMLBBS\ App_Code\ XmlDatabase.cs

```
public enum ParameterDirection
{
    Insert,         //插入
    Update,         //更新
    Equal,          //相等
    NotEqual,       //不相等
    Little,         //小于
    Great,          //大于
    Like            //模糊匹配
}
```

1.3.2 读取 XML 中的数据

以论坛回复模块为例来看应用 GetData 方法读取 XML 中数据的实际运行效果，如图 1.4 所示。

上述运行效果图为网站根目录下 ProjectBBS 文件夹中的 ViewTitle.aspx 页运行效果图，在该页的后台代码中调用 GetData 方法，读取 XML 文件中的数据代码如下。


[回复文章]

等 ASP.NET编程明日科技图书：ASP.NET程序开发范例宝典、ASP.NET2.0编程自学手册、SQL范例宝典、SQL参考在全、ASP.NET项目全程开发

[回复文章] [本篇作者：mr]
发帖人：mr，所属区域：明日论坛首页
标　题：回复：ASP.NET编程
发信站：MRSOFTASPNET-XMLBBS(2008-4-23)，站内

感觉明日科技出版这么多的好书，谢谢你们的辛勤劳动！

【mr在帖子(ASP.NET编程)中提到：】
ASP.NET编程明日科技图书：ASP.NET程序开发范例宝典、ASP.NET2.0编程自学手册、

[回复文章] [本篇作者：mr]
发帖人：mr，所属区域：明日论坛首页
标　题：回复：ASP.NET编程
发信站：MRSOFTASPNET-XMLBBS(2008-5-16)，站内

顶

【mr在帖子(ASP.NET编程)中提到：】
ASP.NET编程明日科技图书：ASP.NET程序开发范例宝典、ASP.NET2.0编程自学手册、

图 1.4　读取 XML 中的数据示例图

例程 3　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS\ ViewTitle.aspxcs

```
//调用公共类中的GetData方法，读取XML文件中的数据
private object GetData(DataTable dt,string sColumnName,string dColumnName,string sValue)
{
    DataRow[] rows = dt.Select(sColumnName + "='" + sValue + "'");
    if(rows.Length <= 0) return null;
    return rows[0][dColumnName];
}
```

读取 XML 文件中数据的方法封装在公共类 XmlDatabase 中，主要由方法 GetData（string path,string tableName）实现，在该方法中参数 path 表示被检索数据的 XML 文件的链接地址；参数 tableName 表示被检索数据的表的名称，即 XML 文件根节点名称。


说 明

获取 XML 文件中的数据也可以通过创建一个 DataSet 对象，然后使用数据集中的 ReadXml()方法读取 XML 文件中的数据。

方法 GetData（string path,string tableName）用来检索 XML 文件，并返回一个 DataTable 对象（DataTable 是内存中的数据表，既可以独立存在，也可以以成员的形式存在于 DataSet 中），具体实现的代码如下。

例程 4　代码位置：光盘\mr\01\XMLBBS\ App_Code\ XmlDatabase.cs

```
public static DataTable GetData(string path,string tableName)
{
    //创建XmlDocument类的实例
    XmlDocument xmldoc = new XmlDocument();
    //调用XmlDocument类中的Load()方法加载XML文件
    xmldoc.Load(path);
    //创建DataTable类型的变量dt
    DataTable dt = new DataTable();
    //获取根节点
    XmlNode rootNode = xmldoc.SelectSingleNode("/" + tableName + "s");
     //判断节点及其子节点是否为空，为空将返空值
```

```
        if(rootNode == null) return null;
        if(rootNode.ChildNodes.Count <= 0) return null;
        //创建保存记录的数据列
        foreach(XmlAttribute attr in rootNode.ChildNodes[0].Attributes)
        {
            dt.Columns.Add(new DataColumn(attr.Name,typeof(string)));
        }
        //创新获取数据节点的XPath
        string xmlPath = "/" + tableName + "s/" + tableName;
         //获取XML节点下的所有节点
        XmlNodeList nodeList = xmldoc.SelectNodes(xmlPath);
         //遍历所有节点
        foreach(XmlNode node in nodeList)
        {
            //创建数据表行，并在其中添加数据
             DataRow row = dt.NewRow();
            foreach(DataColumn column in dt.Columns)
            {    ///读取每一个属性
                row[column.ColumnName] = node.Attributes[column.ColumnName].Value;
            }
            //将数据表行添加到数据表中
            dt.Rows.Add(row);
        }
        //返回DataTable对象dt
        return dt;
    }
```

1.3.3 向 XML 文件中插入数据

使用 AddXmlData 方法向 XML 文件中插入数据，以注册会员发帖为例，用户的发帖信息全部存储在 XML 数据库中，实际运行效果如图 1.5 所示。

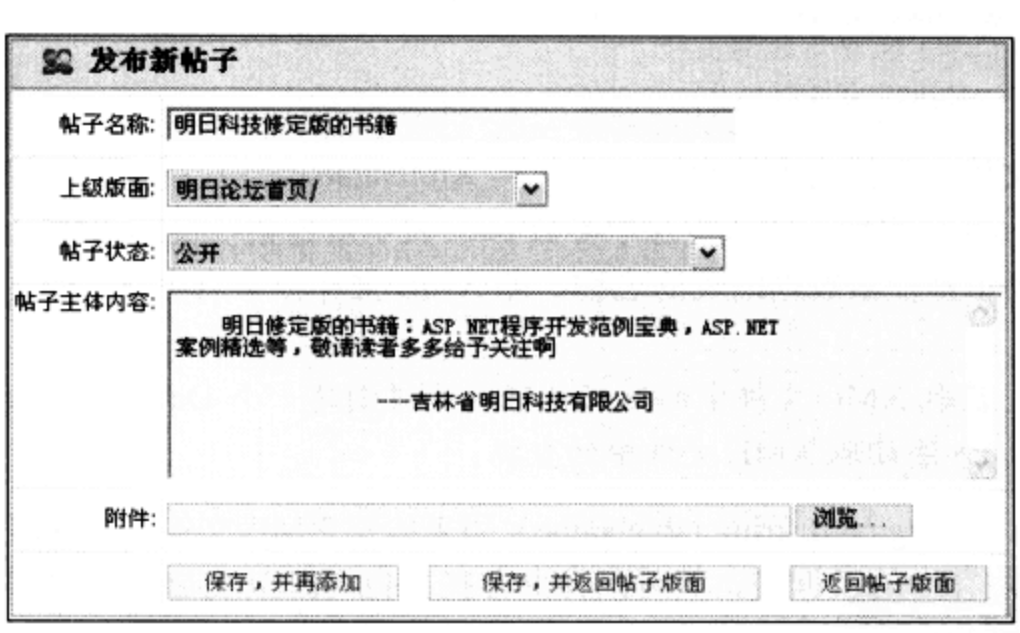


图 1.5　使用 AddXmlData 方法向 XML 文件中插入数据示例

图 1.5 所示为网站根目录下 ProjectBBS 文件夹中的 AddTitle.aspx 页运行效果图，该页后台代码中在向 XML 文件中插入数据时，首先调用公共类 BBS 中添加论坛帖子的 AddTitle 方法，在此方法中调用了向 XML 文件中插入数据的基本方法 AddXmlData，实现最终要求。关键代码如下。

例程 5　代码位置：光盘\mr\01\XMLBBS\ App_Code\ BBS.cs

```
//添加论坛帖子的AddTitle方法
public int AddTitle(string name,string body,int userID,int boardID,byte state)
{
    //获取帖子Title列表中的字段，以实现添加操作
```

```
        XmlParamter[] param = {
            XmlDatabase.CreateInsertParameter("Name",name),
            XmlDatabase.CreateInsertParameter("Body",body),
            XmlDatabase.CreateInsertParameter("UserID",userID.ToString()),
            XmlDatabase.CreateInsertParameter("BoardID",boardID.ToString()),
            XmlDatabase.CreateInsertParameter("CreateDate",DateTime.Now.ToShortDateString()),
            XmlDatabase.CreateInsertParameter("VisitNum","0"),
            XmlDatabase.CreateInsertParameter("ReplyNum","0"),
            XmlDatabase.CreateInsertParameter("State",state.ToString())
        };
        //调用公共类中的AddXmlData向XML数据库中添加数据
        return (XmlDatabase.AddXmlData(XmlBBS.TitleFilePath,TitleTableName,param));
    }
```

向 XML 文件中添加数据的方法封装在公共类 XmlDatabase 中，由该类中的 AddXmlData 方法实现，首先创建 XmlDocument 类的实例，并调用 XmlDocument 类中的 Load()方法加载保存数据的 XML 文件。然后，应用“XmlNode node = xmldoc.SelectSingleNode("/" + tableName + "s");”语句查找 XML 文件中节点，并在其中创建要添加数据的节点。当数据全部添加完成之后，再调用 XmlDocument 类的 Save()方法保存所添加的数据，最后返回一个整数。该方法具体实现的代码如下。

例程 6 代码位置：光盘\mr\01\XMLBBS\ App_Code\ XmlDatabase.cs

```
public static int AddXmlData(string path,string tableName,params XmlParamter[] param)
{
    //创建XmlDocument类的实例
    XmlDocument xmldoc = new XmlDocument();
    //调用XmlDocument类中的Load()方法加载XML文件
     xmldoc.Load(path);
     //选择根节点
     XmlNode node = xmldoc.SelectSingleNode("/" + tableName + "s");
          if(node == null) return -1;
          //创建新记录的ID值
          int newID = DataTypeConvert.ConvertToInt(node.LastChild.Attributes["ID"].Value) + 1;
          if(newID < 1) return -1;
          //创建一个新节点
          XmlNode newNode = xmldoc.CreateNode(XmlNodeType.Element,tableName,null);
          if(newNode == null)return -1;
          //添加ID的值
          newNode.Attributes.Append(CreateNodeAttribute(xmldoc,"ID",newID.ToString()));
          //添加新节点的属性
          foreach(XmlParamter p in param)
          {
               newNode.Attributes.Append(CreateNodeAttribute(xmldoc,p.Name,p.Value));
          }
          ///将新节点追加到根节点中
          node.AppendChild(newNode);
    //保存XML文档
    xmldoc.Save(path);
          return newID;
    }
}
```

1.3.4 更新 XML 文件中的数据

使用 UpdateData 方法更新 XML 中的数据，以管理员更新论坛版面为例，实际运行效果如图 1.6 所示。

图 1.6 所示为网站根目录下 Admin 文件夹中的 UpdateBoard.aspx 页运行效果图，该页后台代码中在更新 XML 文件中的数据时，首先调用论坛版面类 Board 中更新论坛帖子的 UpdateBoard 方法，在此方法中调用了更新 XML 文件中数据的基本方法 UpdateData，实现最终

要求。关键代码如下。

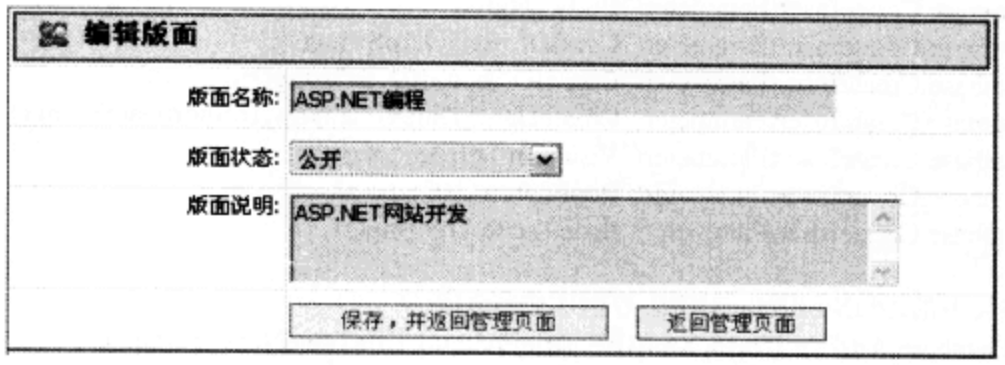


图 1.6　使用 UpdateData 方法更新 XML 中的数据示例

例程 7　代码位置：光盘\mr\01\XMLBBS\ App_Code\ Board.cs

```
public int UpdateBoard(int boardID,string name,byte state,string remark)
{
  //在XML文件中获取论坛版面Board列表中字段，以便实现更新操作
  XmlParamter[] param = {
          XmlDatabase.CreateEqualParameter("ID",boardID.ToString()),
          XmlDatabase.CreateUpdateParameter("Name",name),
          XmlDatabase.CreateUpdateParameter("State",state.ToString()),
          XmlDatabase.CreateUpdateParameter("Remark",remark)
    };
   //获取数据
    return (XmlDatabase.UpdateData(XmlBBS.BoardFilePath,TableName,param));
}
```

更新 XML 文件中数据的方法封装在公共类 XmlDatabase 中，由该类中的 UpdateData 方法实现，首先创建 XmlDocument 类的实例，并调用 XmlDocument 类中的 Load()方法加载保存数据的 XML 文件；然后创建 XML 中的参数列表并检索满足 Xpath 条件的节点，更新该节点，最后调用 XmlDocument 类的 Save()方法保存所更新的数据。该方法具体实现的代码如下。

例程 8　代码位置：光盘\mr\01\XMLBBS\ App_Code\ XmlDatabase.cs

```
public static int UpdateData(string path,string tableName,params XmlParamter[] param)
{
  //创建XmlDocument类的实例
  XmlDocument xmldoc = new XmlDocument();
  //调用XmlDocument类中的Load()方法加载XML文件
  xmldoc.Load(path);
//创建选择被修改节点的XPath
string xmlPath = "/" + tableName + "s/" + tableName;
int operationCount = 0;
StringBuilder operation = new StringBuilder();
foreach(XmlParamter p in param)
{
    if(p.Direction == ParameterDirection.Insert
        || p.Direction == ParameterDirection.Update)
    {
        continue;
    }
      //创建条件表达式
    switch(p.Direction)
    {
        case ParameterDirection.Equal:
            operation.Append("@" + p.Name + "='" + p.Value + "'");
            break;
        case ParameterDirection.NotEqual:
            operation.Append("@" + p.Name + "<>'" + p.Value + "'");
            break;
        case ParameterDirection.Little:
```

```
                operation.Append("@" + p.Name + "<'" + p.Value + "'");
                break;
            case ParameterDirection.Great:
                operation.Append("@" + p.Name + ">'" + p.Value + "'");
                break;
            case ParameterDirection.Like:
                operation.Append("contains(@" + p.Name + ",'" + p.Value + "')");
                break;
            default: break;
        }
        operationCount++;
          //应用StringBuilder类中的Append方法追加“and”字符串
        operation.Append(" and ");
    }
    if(operationCount > 0)
    {
        operation.Remove(operation.Length - 5,5);
        xmlPath += "[" + operation.ToString() + "]";
    }
   //获取XML文件中的所有节点
    XmlNodeList nodeList = xmldoc.SelectNodes(xmlPath);
    if(nodeList == null) return -1;
    //遍历XML文件中节点中的所有子节点
    foreach(XmlNode node in nodeList)
    {     //修改单个节点的属性
        foreach(XmlParamter p in param)
        {
            if(p.Direction == ParameterDirection.Update)
            {
                node.Attributes[p.Name].Value = p.Value;
            }
        }
    }
   //保存XML文档
    xmldoc.Save(path);
     return nodeList.Count;
}
```

1.3.5 删除 XML 文件中的数据

使用 DeleteXmlData 方法删除 XML 中的数据，以删除论坛版面信息为例，实际运行效果如图 1.7 所示。

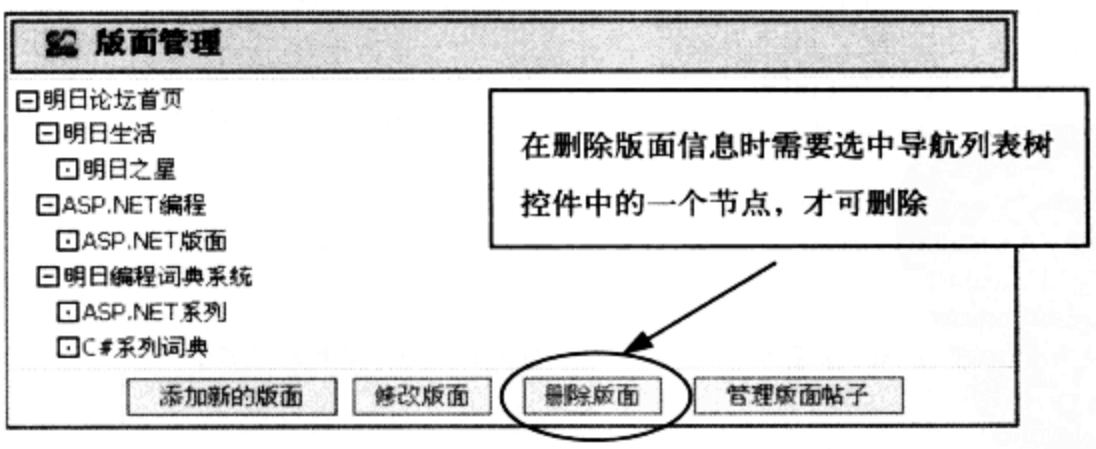


图 1.7　使用 DeleteXmlData 方法删除 XML 中的数据示例

图 1.7 所示为网站根目录下 Admin 文件夹中的 BoardManage.aspx 页运行效果图，该页后台代码中在删除存储在 XML 文件中的版面数据时，首先调用论坛版面类 Board 中删除论坛版面的 DeleteBoard 方法，在此方法中调用了删除 XML 文件中数据的基本方法 DeleteXmlData，实

现最终要求。关键代码如下。

例程 9 代码位置：光盘\mr\01\XMLBBS\ App_Code\ XmlDatabase.cs

```
public static int DeleteDataIDParam(string path,string tableName,int idValue)
{     //创建ID参数
      XmlParamter[] param = {
            XmlDatabase.CreateEqualParameter(DataCommon.IDPARAM,idValue.ToString())
      };
      //返回执行结果
     return (XmlDatabase.DeleteXmlData(path, tableName, param));
}
```

删除 XML 文件中数据的方法封装在公共类 XmlDatabase 中，在该类中定义了 1 个 DeleteData 方法，该方法首先创建 XmlDocument 类的实例，并调用 XmlDocument 类中的 Load() 方法加载保存数据的 XML 文件。然后创建 XML 中的参数列表并检索满足 Xpath 条件的节点，并删除该节点，最后调用 XmlDocument 类的 Save()方法保存删除后的数据。该方法具体实现的代码如下。

例程 10 代码位置：光盘\mr\01\XMLBBS\ App_Code\ XmlDatabase.cs

```
public static int DeleteXmlData(string path, string tableName, params XmlParamter[] param)
{
       //创建XmlDocument类的实例
       XmlDocument xmldoc = new XmlDocument();
       //调用XmlDocument类中的Load()方法加载XML文件
       xmldoc.Load(path);
      //创新选择被删除节点的XPath
      string xmlPath = "/" + tableName + "s/" + tableName;
      int operationCount = 0;
      StringBuilder operation = new StringBuilder();
      foreach(XmlParamter p in param)
      {
            if(p.Direction == ParameterDirection.Insert
                  || p.Direction == ParameterDirection.Update)
            {
                  continue;
            }
            switch(p.Direction)
            {
                  case ParameterDirection.Equal:
                        operation.Append("@" + p.Name + "='" + p.Value + "'");
                        break;
            case ParameterDirection.NotEqual:
                  operation.Append("@" + p.Name + "<>'" + p.Value + "'");
                  break;
            case ParameterDirection.Little:
                  operation.Append("@" + p.Name + "<'" + p.Value + "'");
                        break;
                  case ParameterDirection.Great:
                        operation.Append("@" + p.Name + ">'" + p.Value + "'");
                        break;
                  case ParameterDirection.Like:
                        operation.Append("contains(@" + p.Name + ",'" + p.Value + "')");
                        break;
                  default: break;
            }
            operationCount++;
            operation.Append(" and ");
      }
      if(operationCount > 0)
      {
            operation.Remove(operation.Length - 5,5);
```

```
            xmlPath += "[" + operation.ToString() + "]";
        }
        //获取XML文件中的所有节点
        XmlNodeList nodeList = xmldoc.SelectNodes(xmlPath);
        if(nodeList == null) return -1;
        //删除被选择的节点
        foreach(XmlNode node in nodeList)
        {       //删除单个节点
            XmlNode parentNode = node.ParentNode;
            parentNode.RemoveChild(node);
        }
        //保存删除后的XML文档
        xmldoc.Save(path);
        return nodeList.Count;
    }
```

1.4 公共类的封装与设计

本节介绍应用程序中使用的公共类，如操作 XML 数据库的类、操作 SQL Server 2005 数据库的类等，类的应用可以减少重复代码的编写，有利于代码维护。公共类编写提供系统共用的方法，如本系统建立的 XmlDatabase 类，即 XmlDatabase 文件，如图 1.8 所示。

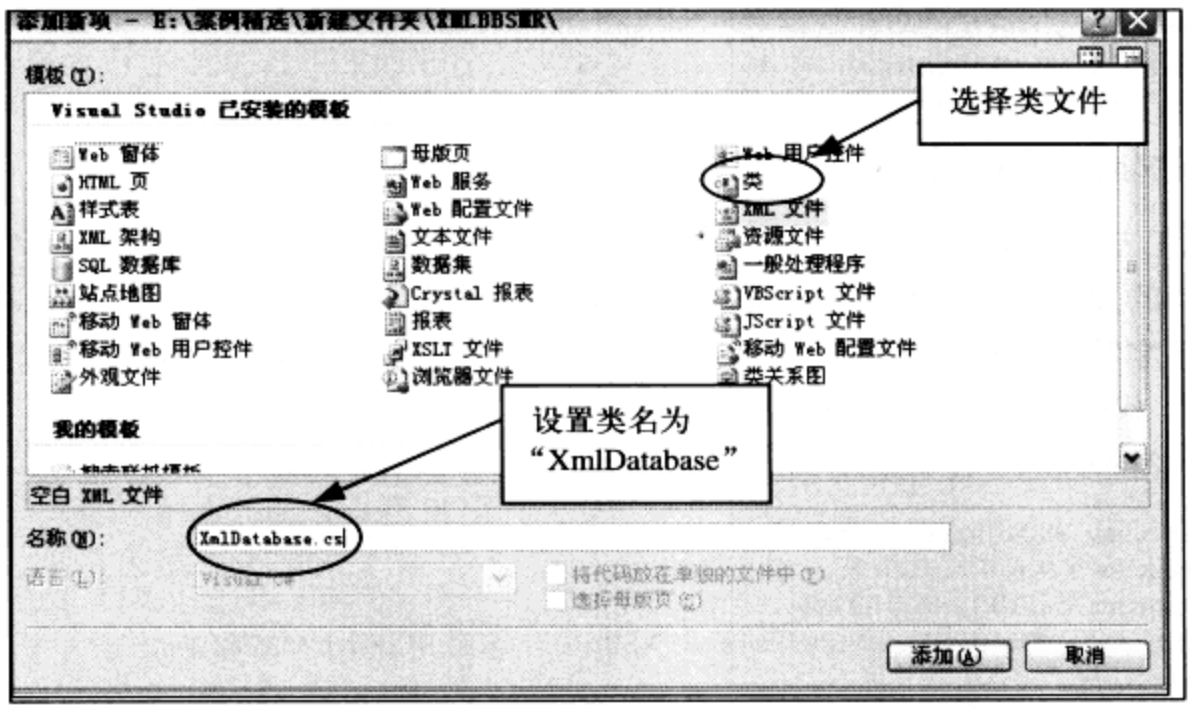


图 1.8 创建 XmlDatabase 类

1.4.1 Web.Config 文件设计

设计公共类前，首先需要配置 Web.Config 文件。在本系统中，Web.Config 文件配置系统的总体信息，如与 XML 数据文件连接的字符串。该文件的具体配置过程如下。

例程 11 代码位置：光盘\mr\01\XMLBBS\ Web.Config

```
<?xml version="1.0"?>
<configuration>
  <appSettings/>
  <connectionStrings>
    <add name="SQLSERVERCONNECTIONSTRING" connectionString="data source=MRFDW\MRFDW;user id=sa;pwd=";
database=Web2ASPNET2DB" providerName="System.Data.SqlClient"/>
    <add name="BOARDFILEPATH" connectionString="~/XmlDatabase/Board.xml"/>
    <add name="TITLEFILEPATH" connectionString="~/XmlDatabase/Title.xml"/>
    <add name="REPLYFILEPATH" connectionString="~/XmlDatabase/Reply.xml"/>
    <add name="ATTACHMENTFILEPATH" connectionString="~/XmlDatabase/Attachment.xml"/>
```

```
        <add name="MESSAGEFILEPATH" connectionString="~/XmlDatabase/Message.xml"/>
        <add name="USERSTATFILEPATH" connectionString="~/XmlDatabase/UserStat.xml"/>
      </connectionStrings>
      <system.web>
        <compilation debug="true">
          <assemblies>
            <add assembly="System.Web.Extensions, Version=1.0.61025.0, Culture=neutral, PublicKeyToken=
31BF3856AD364E35"/>
          </assemblies>
        </compilation>
        <authentication mode="Windows"/>
      </system.web>
    </configuration>
```

上述配置文件中的关键代码主要为连接 XML 数据库的字符串，如连接版面信息的 XML 列表、帖子信息的 XML 列表等，具体连接的 XML 在以下代码注释中标记出：

```
//定义连接论坛版面列表Board.xml的字符串
<add name="BOARDFILEPATH" connectionString="~/XmlDatabase/Board.xml"/>
//定义连接帖子列表Title.xml的字符串
<add name="TITLEFILEPATH" connectionString="~/XmlDatabase/Title.xml"/>
//定义边接帖子回复列表Reply.xml的字符串
<add name="REPLYFILEPATH" connectionString="~/XmlDatabase/Reply.xml"/>
//定义连接帖子附件列表Attachment.xml的字符串
<add name="ATTACHMENTFILEPATH" connectionString="~/XmlDatabase/Attachment.xml"/>
//定义连接帖子属性列表Message.xml的字符串
<add name="MESSAGEFILEPATH" connectionString="~/XmlDatabase/Message.xml"/>
//定义连接用户信息列表UserStat.xml的字符串
<add name="USERSTATFILEPATH" connectionString="~/XmlDatabase/UserStat.xml"/>
```

1.4.2 操作 XML 连接路径类

操作 XML 连接路径的公共类为在 App_Code 文件夹下的 XMLBBS，在该类中定义了论坛版面连接路径的属性、论坛帖子及回复信息连接路径的属性等。另外，该公共类还设置了与配置文件的连接字符串（这些字符串用来建立与 XML 数据库的连接），具体程序代码如下。

例程 12 代码位置：光盘\mr\01\XMLBBS\ App_Code\ XMLBBS.cs

```
public class XmlBBS
{
   public static int TextStringLength = 2147483647;
   public static int NormalRoleID = 2;
   public static int XmlBBSProjectID = 4;
   public static string XmlBBSTotalVisitNumber = "XMLBBSTOTALVISITNUMBER";
   public XmlBBS()
{
}
      private static string boardFilePath;
      public static string BoardFilePath
{
      get { return boardFilePath; }
      set { boardFilePath = value; }
}
      private static string titleFilePath;
      public static string TitleFilePath
{
      get { return titleFilePath; }
      set { titleFilePath = value; }
}
private static string replyFilePath;
public static string ReplyFilePath
{
      get { return replyFilePath; }
      set { replyFilePath = value; }
```

```
    }
    private static string attachmentFilePath;
    public static string AttachmentFilePath
    {
        get { return attachmentFilePath; }
        set { attachmentFilePath = value; }
    }
    private static string userStatFilePath;
    public static string UserStatFilePath
    {
        get { return userStatFilePath; }
        set { userStatFilePath = value; }
    }
    public static void SystemInit(HttpServerUtility server)
    {
    //连接配置文件中的BOARDFILEPATH字符串，进而操作存储在Board.xml文件中的版面信息
    boardFilePath = server.MapPath(
    ConfigurationManager.ConnectionStrings["BOARDFILEPATH"].ConnectionString);
    //连接配置文件中的TITLEFILEPATH字符串，进而操作存储在Title.xml文件中的帖子信息
    titleFilePath = server.MapPath(
    ConfigurationManager.ConnectionStrings["TITLEFILEPATH"].ConnectionString);
    //连接配置文件中的REPLYFILEPATH字符串，进而操作存储在Reply.xml文件中的回帖信息
    replyFilePath = server.MapPath(
    ConfigurationManager.ConnectionStrings["REPLYFILEPATH"].ConnectionString);
    //连接配置文件中的ATTACHMENTFILEPATH字符串，进而操作存储在Attachment.xml文件中的帖子附件信息
    attachmentFilePath = server.MapPath(
    ConfigurationManager.ConnectionStrings["ATTACHMENTFILEPATH"].ConnectionString);
    //连接配置文件中的MESSAGESHIELDFILEPATH字符串，进而操作存储在UserStat.xml文件中的用户信息
    userStatFilePath = server.MapPath(
    ConfigurationManager.ConnectionStrings["USERSTATFILEPATH"].ConnectionString);
        }
    }
}
```

1.5 论坛版面设计与管理

论坛版面管理模块主要包括版面管理、新开版面、编辑版面等功能，这些功能只能由管理员进行操作管理。

1.5.1 论坛版面管理

论坛版面管理由网站根目录下 Admin 文件夹下的 BoardManage.aspx 页实现，主要以树型结构显示论坛的版面信息，并提供了添加新版面、修改版面、删除版面等功能。页面运行的结果如图 1.9 所示。

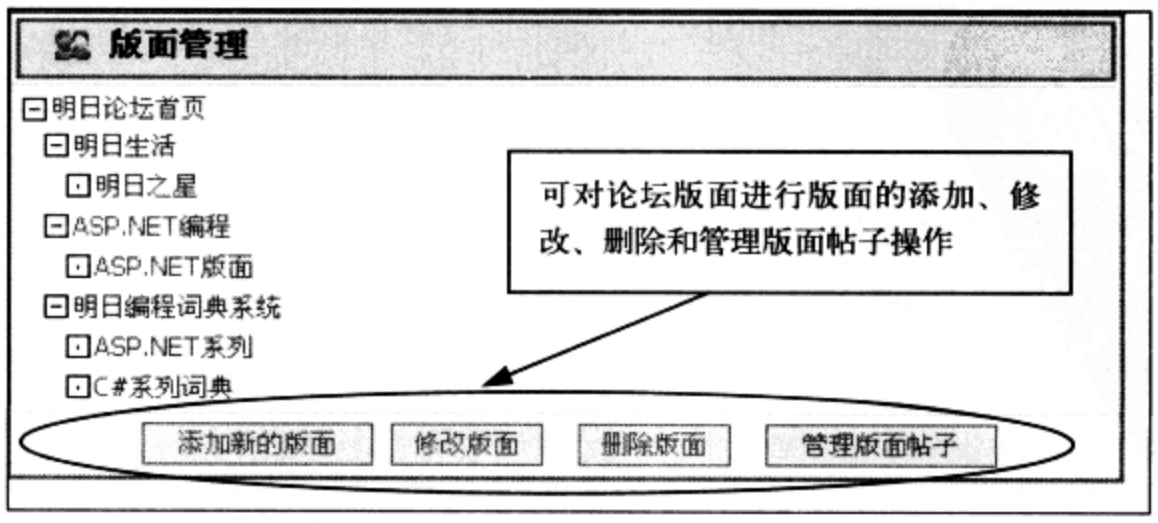


图 1.9 论坛版面管理页

1. 页面设计

在该页面（BoardManage.aspx）的设计中，主要添加了 1 个 TreeView 控件和 4 个 TextBox 控件，具体功能描述如表 1.1 所示。

表 1.1　　论坛版面管理应用的控件

控件类型	控件名称	主要属性设置	用　途
标准/TextBox 控件	btnAdd	CommandName 属性设置为 add SkinID 属性设置为 btnSkin CssClass 属性设置为 Button Text 属性设置为“添加新的版面”	页面导航，显示论坛版面信息
	btnUpdate	CommandName 属性设置为 update SkinID 属性设置为 btnSkin Text 属性设置为“修改版面”	提供修改新版面的链接
	btnTitle	CausesValidation 属性设置为 False CommandName 属性设置为 title SkinID 属性设置为 btnSkin	提供管理版面帖子的链接
	btnDelete	CausesValidation 属性设置为 False CommandName 属性设置为 delete SkinID 属性设置为 btnSkin	删除版面
标准/ TreeView 控件	tvBoard	EnableClientScript 设置为 Flase，不展开和折叠 TreeView 控件 NodeIndent 属性设置为 10，即 TreeView 控件的子节点的缩进量为 10 SkinID 属性设置为 tvSkin	页面导航，显示论坛版面信息

2. 代码编写

在该页面的前台页面设计中应用了 ASP.NET 中的主题，即应用主题只需在 HTML 页面头声明如下代码（粗体加黑处为主题中的样式表主题）：

例程 13　代码位置：光盘\mr\01\XMLBBS\ Admin \ BoardManage.aspx

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="BoardManage.aspx.cs" Inherits="ProjectBBS_
BoardManage" StylesheetTheme="MRSOFTASPNET"%>
```

首选通过页面 BoardManage.aspx 的 Page_Load 事件判断是否是管理员登录，然后调用自定义方法 BindPageData()以显示论坛的版面信息，同时为 btnDelete 按钮添加删除确认对话框，当用户单击此按钮时，该页面弹出一个确认对话框。事件 Page_Load 和自定义方法 BindPageData() 代码如下。

例程 14　代码位置：光盘\mr\01\XMLBBS\ Admin \ BoardManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //判断用户是否是管理员登录
    if (Session["AdminUserName"] == null)
    {
        //返回到上一个页面
        Response.Write("<script>history.back()</script>");
        //跳转到登录页面
        Server.Transfer("~/Admin/AdminLogin/BoardLogin.aspx");
    }
    if (!Page.IsPostBack)
    {
        //调用自定义方法绑定版面信息
        BindPageData();
    }
    //为删除按钮添加确认对话框
```

```
btnDelete.Attributes.Add("onclick", "return confirm('你确定要删除所选择的版面吗?');");
}
//自定义方法BindPageData获取论坛版面信息
private void BindPageData()
{
  //实例化公共类Board
  Board board = new Board();
  //调用该类中的CreateHiberarchyTree方法获取TreeView控件中的数据
  board.CreateHiberarchyTree(tvBoard, false, false);
}
```

该页面中的“添加新的版面”、“修改版面”、“删除版面”和“管理版面帖子”按钮都定义了 1 个 Command 事件，该事件的名称为 OperationBtn_Command(object sender, CommandEventArgs e)，当用户单击上述按钮时都会触发该事件，实现不同的操作。该事件主要的程序代码如下。

例程 15 代码位置：光盘\mr\01\XMLBBS\ Admin \ BoardManage.aspx.cs

```
protected void OperationBtn_Command(object sender, CommandEventArgs e)
{
  if (string.IsNullOrEmpty(e.CommandName) == true) return;
  //判断TreeView控件节点是否被选择
  if (tvBoard.SelectedNode == null)
  {
      Dialog.OpenDialog(Response, "请选择操作的节点。");
      return;
  }
  switch (e.CommandName.ToLower())
  {
    case "add"://添加版面
  {
      Server.Transfer("~/Admin/AddBoard.aspx?BoardID=" + tvBoard.SelectedNode.Value);
      break;
    }
case "update":// 修改版面
   {
     Server.Transfer("~/Admin/UpdateBoard.aspx?BoardID=" + tvBoard.SelectedNode.Value);
    break;
     }
case "title": //管理版面帖子
{
     Server.Transfer("~/Admin/AdminTitleManage.aspx?BoardID=" + tvBoard.SelectedNode.Value);
      break;
     }
case "delete":// 删除版面
     {
         DeleteBoard(DataTypeConvert.ConvertToInt(tvBoard.SelectedNode.Value));
         break;
     }
default: break;
 }
}
```

说 明 上述事件根据调用其按钮的 CommandName 属性的值执行不同的操作。如果该属性的值为“add”，则执行重定向到添加新版面的页面 AddBoard.aspx；如果该属性的值为“update”，则执行重定向到修改版面的页面 UpdateBoard.aspx，并把被修改版面的 ID 值传递到该页面；如果该属性的值为“title”，则执行重定向到管理版面帖子的页面 TitleManage.aspx，并把选择版面的 ID 值传递到该页面；如果该属性的值为“delete”，则执行删除版面的操作。

1.5.2 创建论坛版面

新开论坛版面由网站根目录下 Admin 文件夹下的 AddBoard.aspx 页实现，在该页面中通过设置新版面的名称、上级版面、版面说明信息等来创建一个新的论坛版面，页面运行效果如图 1.10 所示。

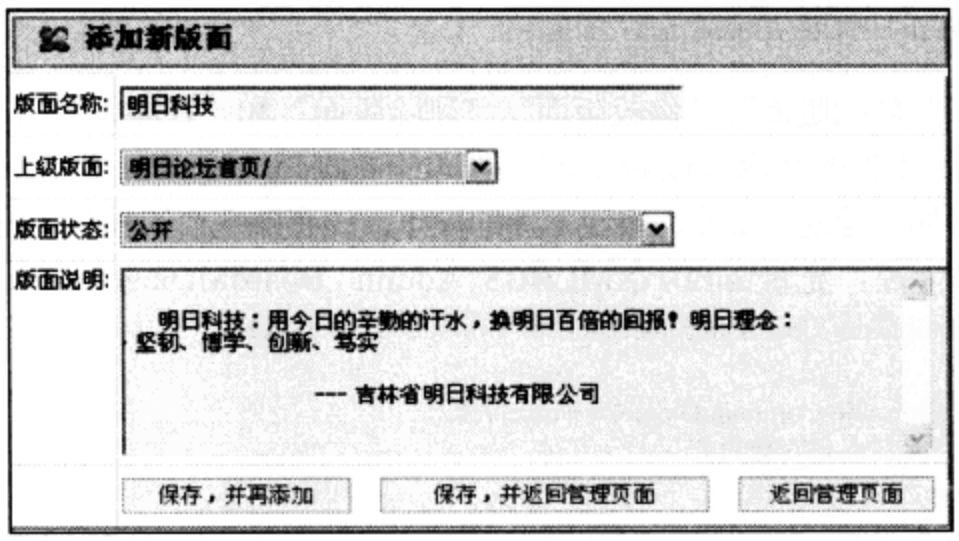


图 1.10 新开版面运行结果图

1. 页面设计

页面 AddBoard.aspx 中添加了 2 个 TextBox 控件、3 个 Button 控件、1 个非空验证控件和 2 个下拉列表控件，具体功能描述如表 1.2 所示。

表 1.2 主要控件及用途

控件类型	控件名称	主要属性设置	用途
标准/ DropDownList	ddlBoard	SkinID 设置为 ddlSkin	显示系统中的版面层次结构树
	ddlState	SkinID 设置为 ddlSkin	显示版面所处的状态
标准/ TetxBox	tbName	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin	用来输入新版面的名称
	tbRemark	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin	用来输入版面说明信息
标准/ Button	btnAdd	CssClass 属性设置为 Button SkinID 属性设置为 btnSkin Enabled 属性设置为 false	添加新版面到数据库中，并再次添加
	btnAddAndReturn	CssClass 属性设置为 Button SkinID 属性设置为 btnSkin Enabled 属性设置为 false	保存新版面到数据库中，并重定向版面管理员
	btnReturn	CssClass 属性设置为 Button SkinID 属性设置为 btnSkin	重定向帖子管理页
验证/ RequiredFieldValidator	rfName	Disply 属性设置 Dynamic ControlToValidate 属性设置为 tbName ErrorMessage 属性设置为"不能为空。"	对输入的版面名称时进行非空验证

2. 代码编写

在页面装载 Page_Load 事件中主要用来初始化该页面中的数据，并调用自定义方法 BindPageData()绑定 GridView 控件中的数据，代码如下。

例程 16　代码位置：光盘\mr\01\XMLBBS\ Admin \ AddBoard.aspx.cs

```
int boardID = -1;
protected void Page_Load(object sender, EventArgs e)
{
  //判断用户是否是管理员登录
  if (Session["AdminUserName"] == null)
  {
      //返回到上一个页面
      Response.Write("<script>history.back()</script>");
      //跳转到登录页面
      Response.Redirect("~/Admin/AdminLogin.aspx");
  }
  //获取被修改信息的ID
  if (Request.Params["BoardID"] != null)
  {
   boardID = DataTypeConvert.ConvertToInt(Request.Params["BoardID"].ToString());
  }
  if (!IsPostBack)
  {
      //调用自定义方法BindPageData
      BindPageData();
  }
  //设置按钮可用性
 ListControl[] list ={
      ddlBoard,
      ddlState
  };
  //设置按钮的可见性
  ButtonEnable.ControlButtonEnable(btnAdd,list);
  ButtonEnable.ControlButtonEnable(btnAddAndReturn,list);
}
```

在事件 Page_Load 事件调用了 BindPageData 自定义方法，用来显示系统中的版面层次结构，代码如下。

例程 17　代码位置：光盘\mr\01\XMLBBS\ Admin \ AddBoard.aspx.cs

```
private void BindPageData()
{
   //实例化公共类Board
   Board board = new Board();
   //调用该公共类中的CreateHiberarchyBoard方法，获取绑定到下拉列表中的数据
   board.CreateHiberarchyBoard(ddlBoard);
  //调用该公共类中的ListSelectedItem方法，设置列表控件的选择项
   ListSelectedItem.ListSelectedItemByValue(ddlBoard, boardID.ToString());
}
```

双击 AddBoard.aspx 页面中的“保存，并再添加”和“保存，并返回管理页面”按钮都会实现添加新版面到数据库中的功能。其中，第 1 个按钮添加新版面到数据库中，并准备再次添加新的版面时，将触发该按钮的 btnAdd_Click 事件；第 2 个按钮添加新版面到数据库中后，可重定向到管理页面（BoardManage.aspx），将触发该按钮的 btnAddAndReturn_Click 事件。上述两个事件的程序代码如下。

例程 18　代码位置：光盘\mr\01\XMLBBS\ Admin \ AddBoard.aspx.cs

```
protected void btnAdd_Click(object sender, EventArgs e)
{
    Board board = new Board();//实例化
  //执行添加操作
  if (board.AddBoard(tbName.Text, DataTypeConvert.ConvertToInt(ddlBoard.SelectedValue),
      byte.Parse(ddlState.SelectedValue), tbRemark.Text) > 0)
  {
      //调用公共类中的Dialog方法弹出对话框
      {
          Dialog.OpenDialog(Response, "恭喜您，添加新版面成功……");
```

```
        }
    }
}
protected void btnAddAndReturn_Click(object sender, EventArgs e)
    {
    Board board = new Board();
    //执行添加操作
    if (board.AddBoard(tbName.Text,
          DataTypeConvert.ConvertToInt(ddlBoard.SelectedValue),
          byte.Parse(ddlState.SelectedValue),
        tbRemark.Text) > 0)
 {
      Dialog.OpenDialog(Response, "恭喜您，添加新版面成功……");
      //返回管理页面
      Server.Transfer("~/Admin/BoardManage.aspx");
 }
}
```

1.5.3 编辑论坛版面

编辑论坛版面由页面 UpdateBoard.aspx 实现，在该页面上可以编辑版面的名称、上级版面、版面说明信息等，页面运行结果如图 1.11 所示。

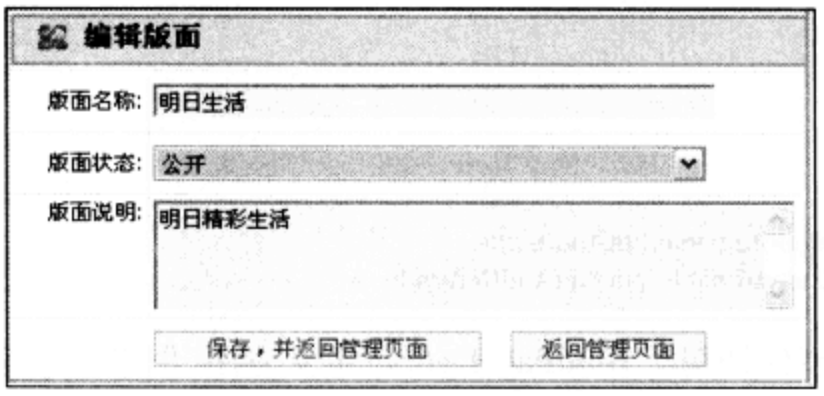


图 1.11　编辑论坛版面运行结果图

1. 页面设计

在该页面中添加了 2 个 TextBox 控件、2 个 Button 控件、1 个非空验证控件和 1 个下拉列表控件，这些控件的具体功能描述如表 1.3 所示。

表 1.3　　主要控件及用途

控 件 类 型	控 件 名 称	主要属性设置	用　途
标准/ DropDownList	ddlState	SkinID 属性设置为 ddlSkin	选择版面所处的状态
标准/ TetxBox	tbName	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin	编辑版面的名称
	tbRemark	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin TextMode 属性设置为 MultiLine	编辑版面的说明信息
标准/ Button	btnUpdateAndReturn	CssClass 属性设置为 Button" Text 属性设置为"保存，并返回管理页面" Enabled 属性设置为 False SkinID 属性设置为 btnSkin	将版面的修改的信息保存到数据库中，并重定向管理页
	btnReturn	CssClass 属性设置为 Button Text 属性设置为“返回管理页面” CausesValidation 属性设置为 False SkinID 属性设置为 btnSkin	重定向管理页面
验证/ RequiredFieldValidator	rfName	ControlToValidate 属性设置为 tbName ErrorMessage 属性设置为"不能为空。"	对输入的版面名称进行非空验证

2. 代码编写

在页面 Page_Load 事件中首先判断是否是管理员登录，如果是管理员登录则将获取的被修改版面信息的 BoardID 值存储在整型变量 boardID 中，然后判断页面是否首次加载，并判断 boardID 值是否大于–1，大于–1 则调用自定义方法 BindPageData(int boardID)显示被编辑版面的信息，如版面的名称等。代码如下。

例程 19 代码位置：光盘\mr\01\XMLBBS\ Admin \ UpdateBoard.aspx.cs

```
int boardID = -1;
protected void Page_Load(object sender, EventArgs e)
{
 //判断用户是否是管理员登录
 if (Session["AdminUserName"] == null)
 {
     //返回到上一个页面
     Response.Write("<script>history.back()</script>");
     //跳转到管理员登录页面
     Response.Redirect("~/Admin/AdminLogin.aspx");
 }
   //将获取的被修改的版面信息的BoardID值存储在整型变量boardID中
   if (Request.Params["BoardID"] != null)
   {
    boardID = DataTypeConvert.ConvertToInt(Request.Params["BoardID"].ToString());
   }
   if (!Page.IsPostBack)
   {    //显示被修改的数据
        if (boardID > -1)
        {
            //调用自定义方法绑定版面信息
         BindPageData(boardID);
        }
   }
}
```

自定义一个 BindPageData(int boardID)方法，绑定论坛版面信息，这里显示版面信息主要是通过创建内存表 DataTable 来实现，代码如下。

例程 20 代码位置：光盘\mr\01\XMLBBS\ Admin \ UpdateBoard.aspx.cs

```
private void BindPageData(int boardID)
{
 //实例化公共类
 Board board = new Board();
 //定义一个DataTable类型的变量dt，并将获得版面ID保存在该变量中
 DataTable dt = board.GetSingleBoard(boardID);
 if (dt == null) return;
 if (dt.Rows.Count <= 0) return;
 //通过定义的内存表DataTalbe编辑版面的名称
 tbName.Text = dt.Rows[0]["Name"].ToString();
//通过定义的内存表DataTalbe编辑版面的说明信息
tbRemark.Text = dt.Rows[0]["Remark"].ToString();
//把列表控件的选择项设置为指定项
ListSelectedItem.ListSelectedItemByValue(ddlState, dt.Rows[0]["State"].ToString());
}
```

双击 UpdateBoard.aspx 页面中的“保存，并返回管理页面”按钮，将触发该按钮的 Click 事件，并调用 Board 类中的 UpdateBoard 方法把修改后的版面信息保存到数据库中，并重定向到管理页 BoardManage.aspx，代码如下。

例程 21 代码位置：光盘\mr\01\XMLBBS\ Admin \ UpdateBoard.aspx.cs

```
protected void btnUpdateAndReturn_Click(object sender, EventArgs e)
{
```

```
    //实例化公共类Board
    Board board = new Board();
    if (board.UpdateBoard(boardID, tbName.Text,
        byte.Parse(ddlState.SelectedValue), tbRemark.Text) > 0)
    {
        //调用Dialog中的OpenDialog方法弹出对话框
        Dialog.OpenDialog(Response, "恭喜您，修改版面信息成功……");
        Server.Transfer("~/Admin/BoardManage.aspx");
    }
}
```

1.6 论坛帖子设计与管理

论坛帖子模块包括发布论坛新帖、查看论坛帖子和论坛帖子回复。

1.6.1 发布论坛新帖

发布论坛新帖由页面 AddTitle.aspx 实现，在该页面上可以设置新帖子的名称、选择所属版面、状态和帖子主体内容，并可以上传帖子附件。页面运行结果如图 1.12 所示。

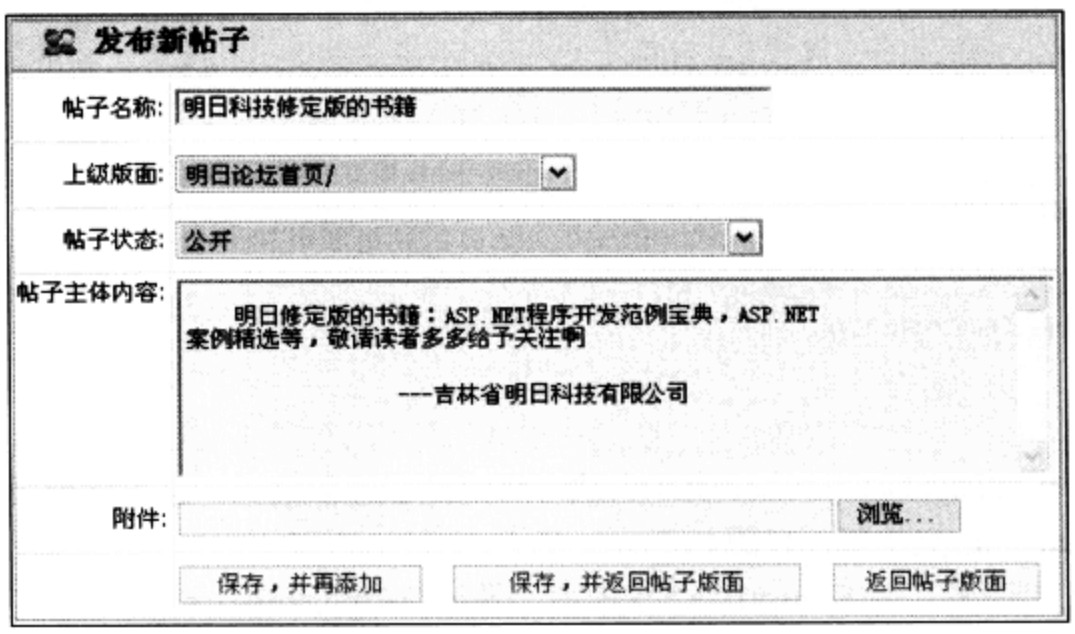


图 1.12 发布论坛新帖页面运行结果图

1. 页面设计

页面 AddTitle.aspx 中添加了 2 个 TextBox 控件、3 个 Button 控件、2 个非空验证控件、2 个下拉列表控件和 1 个上传文件控件，具体功能描述如表 1.4 所示。

表 1.4　　主要控件及用途

控件类型	控件名称	主要属性设置	用途
标准/ DropDownList	ddlBoard	SkinID 属性设置为 ddlSkin	设置新帖发布到的版面
	ddlState	SkinID 属性设置为 ddlSkin	设置新帖子状态
标准/ TetxBox	tbName	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin MaxLength 属性设置为 50	输入帖子的名称
	tbBody	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin MaxLength 属性设置为 8000 TextMode 属性设置为 MultiLine	输入新帖的主体内容

续表

控 件 类 型	控 件 名 称	主要属性设置	用　　途
标准/ Button	btnAdd	CssClass 属性设置为 Button SkinID 属性设置为 btnSkin Enabled 属性设置为 False	发布新帖，并准备再发布
	btnAddAndReturn	SkinID 属性设置为 btnSkin Enabled 属性设置为 False	发布新帖，并重定向到帖子版面
	btnReturn	CssClass 属性设置为 Button CausesValidation 属性设置为 False SkinID 属性设置为 btnSkin	返回帖子版面
验证/ RequiredFieldValidator	rfName	ControlToValidate 属性设置为 tbName ErrorMessage 属性设置为"帖子名称不能为空!"	对输入的帖子名称进行非空验证
	rfBody	ControlToValidate 属性设置为 tbBody Display 属性设置为 Dynamic ErrorMessage 属性设置为"不能为空。"	对输入的帖子内容进行非空验证

2. 代码编写

在页面 Page_Load 事件中首先判断用户是否登录，如果用户登录则将获取的论坛版面 BoardID 值并存储在整型变量 boardID 中，然后判断页面是否是首次加载，最后调用自定义方法 BindPageData 显示系统中的当前版面的层次信息。主要序代码如下。

例程 22 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddTitle.aspx.cs

```
int boardID = -1;
protected void Page_Load(object sender, EventArgs e)
{
  //判断用户是否登录
  UserInfo info = (UserInfo)UserCommonOperation.GetUserInfo(Session);
  if (info == null)
  {    //返回到上一个页面
       Response.Write("<script>history.back()</script>");
       //跳转到登录页面
       Response.Redirect("~/ProjectBBS/UserLogin.aspx");
       return;
  }
  //获取新帖所属的页面ID值，并保存在变量boardID中
  if (Request.Params["BoardID"] != null)
  {
       boardID = DataTypeConvert.ConvertToInt(Request.Params["BoardID"].ToString());
  }
  if (!Page.IsPostBack)
  {
       //调用自定义方法BindPageData显示系统中的当前版面的层次信息
       BindPageData();
  }
}
```

自定义方法 BindPageData()，用来显示论坛版面的层次信息，并判断存储版面的 BoardID 值的 boardID 变量是否大于 0，如果大于 0 则显示选择帖子的版面信息，程序代码如下。

例程 23 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddTitle.aspx.cs

```
public void BindPageData()
{
     ///显示版面的层次信息
     Board board = new Board();
      board.CreateHiberarchyBoard(ddlBoard);
```

```
        if (boardID > 0)
        {    ///选择帖子的版面
         ListSelectedItem.ListSelectedItemByValue(ddlBoard, boardID.ToString());
        }
}
```

双击 AddTitle.aspx 页面中的“保存，并再添加”和“保存，并返回管理页面”按钮都会实现发布论坛新帖的功能。其中，第 1 个按钮发布新帖之后，准备再次发布新帖子，该按钮将触发 btnAdd_Click 事件；第 2 个按钮发布新帖之后，重定向到帖子版面 TitleManage.aspx，该按钮触发 btnAddAndReturn_Click 事件。上述两个事件的程序代码如下。

例程 24 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddTitle.aspx.cs

```
protected void btnAdd_Click(object sender, EventArgs e)
{
    //添加新的帖子
    string url = string.Empty;
    //调用自定义方法AddTitle方法将帖子保存到数据库中
    int titleID = AddTitle(out url);
    if (titleID > 0 && string.IsNullOrEmpty(url) == true)
    {
        //调用Dialog类中的OpenDialog方法弹出对话框
        Dialog.OpenDialog(Response, "恭喜您，添加新帖子成功，但是你没有上传附件……");
        return;
    }
   if (titleID > 0)
   {
   //添加附件信息到数据库中
   BBS bbs = new BBS();
   if (bbs.AddAttachment(fileUpLoad.FileName,url, fileUpLoad.PostedFile.ContentType, titleID) > 0)
    {
        //弹出添加新帖成功对话框
        Dialog.OpenDialog(Response, "恭喜您，添加新帖子成功……");
    }
    else
    {
        //弹出添加新帖成功，上传附件失败对话框
        Dialog.OpenDialog(Response, "恭喜您，添加新帖子成功，但是上传附件失败……");
    }
   }
  }
}
protected void btnAddAndReturn_Click(object sender, EventArgs e)
{
     //添加新的帖子
     string url = string.Empty;
     int titleID = AddTitle(out url);
 if (titleID > 0 && string.IsNullOrEmpty(url) == true)
 {
     Dialog.OpenDialog(Response, "恭喜您，添加新帖子成功，但是你没有上载附件……");
     //返回管理页面
     Server.Transfer("~/ProjectBBS/TitleManage.aspx");
     return;
 }
 if (titleID > 0)
 {    //添加附件信息到数据库中
     ……//省略和上一事件相同位置处的代码
     Server.Transfer("~/ProjectBBS/TitleManage.aspx");
 }
}
```

在以上两个事件代码中，应用了一个自定义方法 AddAttachment(FileUpload fu)，该方法主要用来实现上传帖子附件功能，具体代码如下。

例程 25　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddTitle.aspx.cs

```
private string AddAttachment(FileUpload fu)
{
    //判断上传文件控件是否存在文件
    if(fu.HasFile==false) return null;
    //获取上传文件名称
    string tfName=fu.PostedFile.FileName;
    //创建基于时间的文件名称
    string fileName=DealwithString.CreatedStringByTime()+tfName.Substring(tfName.LastIndexOf("."));
    fileName ="../XmlDatabase/Files/"+fileName;
    //获取服务器端的文件名称
    string allfilePath=Server.MapPath(fileName);
    //判断基于服务器端的文件名是否存在，如果存在则不能上传
    if(File.Exists(allfilePath) == true)
    {
        //弹出对话框
        Dialog.OpenDialog(Response,
            "你上传的文件" + fileName + "已经存在，不能上传所选择的文件");
    }
    try
    {
        //保存上传文件，并返回基于服务器端的文件名称
        fu.SaveAs(allfilePath); return (fileName);
    }
    catch(Exception ex)
    {
        //导向到错误捕捉页面，并获取当前请求的原始URL及错误信息
        Server.Transfer("~/ProjectBBS/ErrorPage.aspx?Url=" +Request.RawUrl + "& ErrorMsg = "+ex.Message,false);
      }
  return null;
}
```

另外，在 btnAdd_Click 和 btnAddAndReturn_Click 事件中还调用了一个自定义方法 AddTitle，该自定义方法根据是否上传附件设置帖子的状态，然后将帖子保存到数据库中，并返回该帖子的附件的链接地址，具体代码如下。

例程 26　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddTitle.aspx.cs

```
private int AddTitle(out string url)
{
    //获取用户登录信息
    UserInfo info = (UserInfo)UserCommonOperation.GetUserInfo(Session);
    if (info == null)
    {
        url = string.Empty;
        return -1;
    }
    //实例化BBS
    BBS bbs = new BBS();
    //显示帖子的状态
    byte state = (byte)TitleState.Reply;
    //调用自定义方法AddAttachmenth上传附件，获取用户上传文件的链接地址
    url = AddAttachment(fileUpLoad);
    //根据是否上传附件设置帖子的状态
    if (string.IsNullOrEmpty(url) == true)
    {
        //显示用户未上传附件状态
        state = byte.Parse(ddlState.SelectedValue);
    }
    else
    {
        //显示用户上传附件状态
        state = (byte)((int)(TitleState.ReplyAttachment) + int.Parse(ddlState.SelectedValue));
```

```
        }
        //添加帖子到数据库中，同时返回新添加帖子的ID值
        return (bbs.AddTitle(tbName.Text,tbBody.Text, info.UserID,
            DataTypeConvert.ConvertToInt(ddlBoard.SelectedValue), state));
    }
```

1.6.2 查看论坛帖子

查看论坛帖子由页面 ViewTitle.aspx 以列表的形式显示帖子内容及帖子回复的内容，页面运行结果如图 1.13 所示。

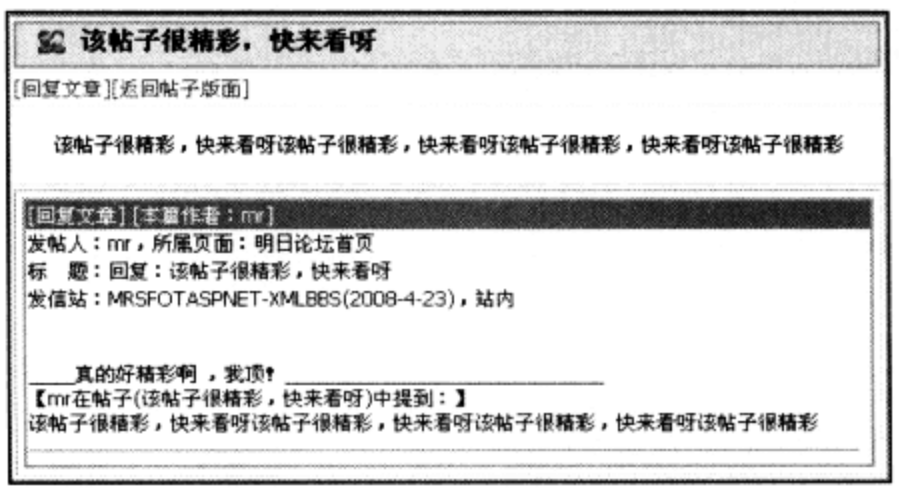


图 1.13　查看论坛帖子页面运行结果图

1. 页面设计

在页面 ViewTitle.aspx 中添加了 1 个 GridView 控件，ID 的属性值为 gvTitle，以列表的形式显示帖子的内容及帖子的回复内容，该页面的 HTML 主要设计代码如下。

例程 27　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ ViewTitle.aspx

```
    <a href="AddReply.aspx?TitleID=<%= titleID.ToString() %>">[回复文章]</a><a href="TitleManage.aspx?BoardID=0">[返回帖子版面]</a>
    <%= Body.Replace("\n","<br>") %> <br /><br />
    <asp:GridView ID="gvTitle" runat="server" SkinID="gvSkin"  Font-Names="Tahoma" Width="100%" DataKeyNames="ID" ShowHeader="False">
    <Columns>
    <asp:TemplateField>
    <ItemTemplate>
    <table class="Table" width="100%" border="0" id="TABLE1">
<td bgcolor="#5a7dd1"><font color="white"><a href="AddReply.aspx?TitleID=<%# Eval("TitleID") %>"><font color="white">[回复文章]</font></a>
    <font color="white">[本篇作者：<%# Eval("UserName") %>]</font>
    <td>发帖人：<%# Eval("UserName") %>，所属页面：<%# Eval("BoardName")%>
    </table>
    </ItemTemplate>
    <ItemStyle Width="100%" />
    </asp:TemplateField>
    </Columns>
    </asp:GridView>
```

2. 代码编写

在页面 Page_Load 事件中首先应用 ASP.NET 内置对象 Request 获取被显示帖子的 titleID 值并保存在定义的字符型变量 titleID 中，然后调用自定义方法 BindPageData，绑定 GridView 控件数据。具体实现代码如下。

例程 28　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ ViewTitle.aspx.cs

```
    protected void Page_Load(object sender, EventArgs e)
    {
```

```
        //获取被显示帖子的ID值，并保存在变量titleID中
         if(Request.Params["TitleID"]!=null)
         {
              titleID=DataTypeConvert.ConvertToInt(Request.Params["TitleID"].ToString());
         }
        if(!IsPostBack)
        {
            if(titleID>-1)
            {
                //调用自定义方法BindPageData，绑定GridView控件数据
                BindPageData(titleID);
            }
        }
    }
```

定义一个自定义方法 BindPageData(int titleID)，在该方法中定义了用户可根据分类的版面来查看相关帖子信息的方法，帖子的内容及其回复内容都将以列表形式显示在 GridView 控件中，具体实现的代码如下。

例程 29 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ ViewTitle.aspx.cs

```
private void BindPageData(int titleID)
{
    //实例化用户信息类User
    User user = new User();
    //定义一个DataSet数据集类型的变量ds，获取用户信息
    DataSet ds = user.GetUsersByDS();
    if (ds == null) return;
    if (ds.Tables.Count <= 0) return;
    //实例化论坛版面信息类Board
    Board board = new Board();
    //定义一个DataTable类型的变量，获取论坛版面信息
    DataTable dtBoard = board.GetBoards();
    if (dtBoard == null) return;
    //实例化公共类BBS
    BBS bbs = new BBS();
    //定义一个DataTable类型的变量，获取存储在XML文件中的帖子信息
    DataTable dtTitle = bbs.GetTitles();
    if (dtTitle == null) return;
    //获取帖子的ID号
    DataRow[] titleRows = dtTitle.Select("ID='" + titleID.ToString() + "'");
    if (titleRows.Length <= 0) return;
    //设置帖子的名称和内容
    Name = ModuleTitle1.Title = titleRows[0]["Name"].ToString();
    Body = titleRows[0]["Body"].ToString();
    //更新帖子访问数
    bbs.UpdateTitleVisitNum(titleID, DataTypeConvert.ConvertToInt(titleRows[0]["VisitNum"].ToString()) + 1);
    //获取帖子的回复数据
    DataTable dtReply = bbs.GetReplyByTitle(titleID);
    if (dtReply == null) return;
    //定义DataTable的框架，即UserName和BoardName两列
    dtReply.Columns.Add(new DataColumn("UserName", typeof(string)));
    dtReply.Columns.Add(new DataColumn("BoardName", typeof(string)));
    //定义DataTable的数据行，并填充数据
    foreach (DataRow row in dtReply.Rows)
{
     foreach (DataColumn column in dtReply.Columns)
     {
         if (column.ColumnName == "UserName")
         {
             row[column.ColumnName] = GetData(ds.Tables[0],
                 "ID",
                 column.ColumnName,
                 row["UserID"].ToString());
         }
```

```
            if (column.ColumnName == "BoardName")
            {
                row[column.ColumnName] = GetData(dtBoard,
                    "ID",
                    "Name",
                    GetData(dtTitle, "ID", "BoardID", row["TitleID"].ToString()).ToString());
            }
        }
    }
    //绑定GridView控件的数据
    MRSOFTASPNET.CommonOperation.DataBinder.BindGridViewData(gvTitle, dtReply);
    }
```

1.6.3 论坛帖子回复

论坛帖子回复由页面 AddReply.aspx 实现，该页首先显示被回复帖子的名称和内容（这里帖子名称不可编辑），用户可以根据帖子的具体内容进行回复。页面运行结果如图 1.14 所示。

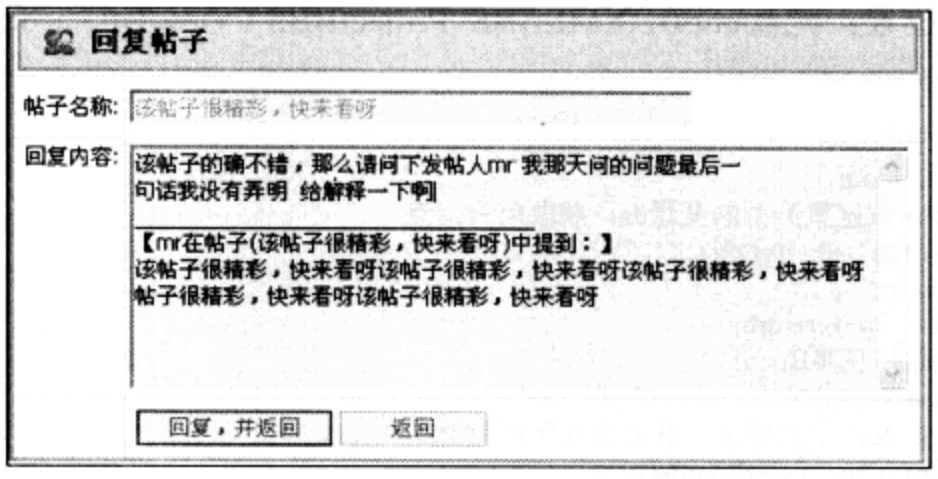


图 1.14 论坛帖子回复页面运行结果图

1. 页面设计

在该页面中添加了 2 个 TextBox 控件、2 个 Button 控件和 1 个非空验证控件，这些控件的具体功能描述如表 1.5 所示。

表 1.5 主要控件及用途

控件类型	控件名称	主要属性设置	用途
标准/ TetxBox	tbName	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin MaxLength 属性设置为 50	显示帖子的名称
	tbBody	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin MaxLength 属性设置为 8000 TextMode 属性设置为 MultiLine	显示帖子的主体内容
标准/ Button	btnUpdateAndReturn	CssClass 属性设置为 Button SkinID 属性设置为 btnSkin Enabled 属性设置为 False	执行帖子回复操作
	btnReturn	CssClass 属性设置为 Button CausesValidation 属性设置为 False SkinID 属性设置为 btnSkin	返回查看帖子页面
验证/ RequiredFieldValidator	rfName	ControlToValidate 属性设置为 tbBody Display 属性设置为 Dynamic ErrorMessage 属性设置为"回复内容不能为空!"	对用户输入帖子的回复内容进行非空验证

2. 代码编写

在页面 Page_Load 事件中首先判断用户是否登录，然后应用 ASP.NET 内置对象 Request 对象获取被回复帖子的 TitleID 值，最后调用自定义方法 BindPageData(int titleID)显示被回复帖子的名称和内容，主要程序代码如下。

例程 30 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddReply.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
  //判断用户是否登录
  UserInfo info = (UserInfo)UserCommonOperation.GetUserInfo(Session);
  if (info == null)
  {
      //返回上一个页面
      Response.Write("<script>history.back()</script>");
      //跳转到登录页面
      Server.Transfer("~/ProjectBBS/UserLogin.aspx");
      return;
  }
  //获取回复帖子的ID
  if(Request.Params["TitleID"]!=null)
  {
      titleID=DataTypeConvert.ConvertToInt(Request.Params["TitleID"].ToString());
  }
  if(!IsPostBack)
  {
    //调用自定义方法BindPageData方法，显示被回复帖子信息
      BindPageData(titleID);
  }
}
```

自定义 1 个 BindPageData 方法，在该方法中首先实例化公共类 BBS，然后调用该类中的 GetSingleTitle()方法将获取被回复帖子的 titleID 值，并将其保存在定义的 DataTable 类型的变量 dt 中，进而通过创建的数据表行显示被回复帖子的信息，被回复帖子信息的输出样式由 StringBuilder 类（一个可变字符字符串类）实现。具体实例代码如下。

例程 31 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddReply.aspx.cs

```
private void BindPageData(int titleID)
{
    //实例化公共类BBS
    BBS bbs = new BBS();
    //定义一个DataTable类型的变量dt，并将获取的titleID保存在该变量中
    DataTable dt = bbs.GetSingleTitle(titleID);
    if (dt == null) return;
    if (dt.Rows.Count <= 0) return;
    tbName.Text = dt.Rows[0]["Name"].ToString();
    //实例化一个可变字符字符串类StringBuilder，用于字符串的追加
    StringBuilder sb = new StringBuilder();
   sb.Append("\n\n_____\n");
    //获取用户信息
    MRSOFTASPNET.XmlBBS.User user = new User();
    //实例化一个SqlDataReader对象获取用户的ID
    SqlDataReader dr = user.GetSingleUser(DataTypeConvert.ConvertToInt(dt.Rows[0]["UserID"].ToString()));
  if (dr == null) return;
  //设置帖子内容
  if (dr.Read())
  {
      //应用StringBuilder类的Append方法进行字符串串联
      sb.Append("【" + dr["UserName"].ToString() + "在帖子(" + dt.Rows[0]["Name"].ToString() + ")中提到：】" + "\n");
   }
   dr.Close();
```

```
            //设置帖子内容
            sb.Append(dt.Rows[0]["Body"].ToString().Length > 100 ? dt.Rows[0]["Body"].ToString().Substring(0, 100) + "..." :
dt.Rows[0]["Body"].ToString());
            tbBody.Text = sb.ToString();
            ViewState["ReplyNum"] = dt.Rows[0]["ReplyNum"];
        }
    }
```

双击 AddReply.aspx 页面中的“回复，并返回”按钮，触发该按钮的 btnUpdateAndReturn_Click 事件，首先获取用户登录信息，然后实例化公共类 BBS 执行帖子回复操作，同时更新帖子回复数，回帖成功后将给予提示信息并返回查看帖子页面。代码如下。

例程 32 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ AddReply.aspx.cs

```
protected void btnUpdateAndReturn_Click(object sender, EventArgs e)
{
    //获取用户登录信息
    UserInfo info = (UserInfo)UserCommonOperation.GetUserInfo(Session);
    if (info == null) return;
    //实例化公共类BBS
    BBS bbs = new BBS();
    if(bbs.AddReply(tbBody.Text,info.UserID,titleID)>0)
    {
     if(ViewState["ReplyNum"] !=null)
      {
         int replyNum=DataTypeConvert.ConvertToInt(ViewState["ReplyNum"].ToString())+1;
         //更新回帖数
         bbs.UpdateTitleReplyNum(titleID,replyNum);
      }
    }
    //弹出回帖成功信息
    Dialog.OpenDialog(Response,"恭喜您，回复帖子信息成功.....");
    //返回查看帖子页面
    Server.Transfer("~/ProjectBBS/ViewTitle.aspx?TitleID="+titleID.ToString());
}
```

1.7 论坛帖子搜索、统计及排行

本节主要介绍应用程序中与搜索、统计及排行的相关功能，如基于关键字的搜索、帖子统计和热门帖子排行等。

1.7.1 基于关键字的搜索

基于关键字的搜索由页面 SearchByKey.aspx 实现，通过输入相关的关键字执行搜索操作，并以列表形式显示搜索结果。页面运行结果如图 1.15 所示。

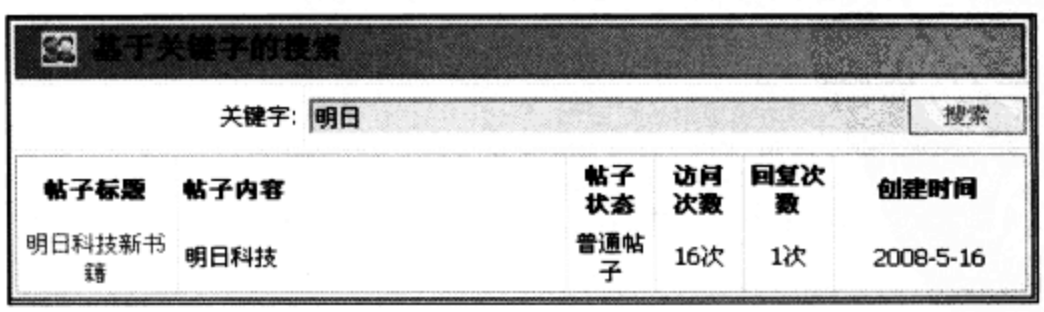


图 1.15 基于关键字的搜索页面运行结果图

1. 页面设计

在该页面中添加了 1 个 TextBox 控件、1 个 Button 控件、1 个非空验证控件和 1 个 GridView 控件，这些控件的具体功能描述如表 1.6 所示。

表 1.6　　主要控件及用途

控件类型	控件名称	主要属性设置	用　途
标准/ TetxBox	TbKey	CssClass 属性设置为 TextBox SkinID 属性设置为 tbSkin MaxLength 属性设置为 50	输入搜索的关键字
标准/ Button	btnSearch	CssClass 属性设置为 Button SkinID 属性设置为 btnSkin	执行搜索操作
标准/ GridView	gvTitle	AutoGenerateColumns 属性设置为 False SkinID 属性设置为 gvSkin DataKeyNames 属性设置为 ID	显示搜索结果
验证/ RequiredFieldValidator	rfKey	ErrorMessage 属性设置为"不能为空。" Display 属性设置为 Dynamic	进行非空验证

该页的前台页面设计效果如图 1.16 所示。

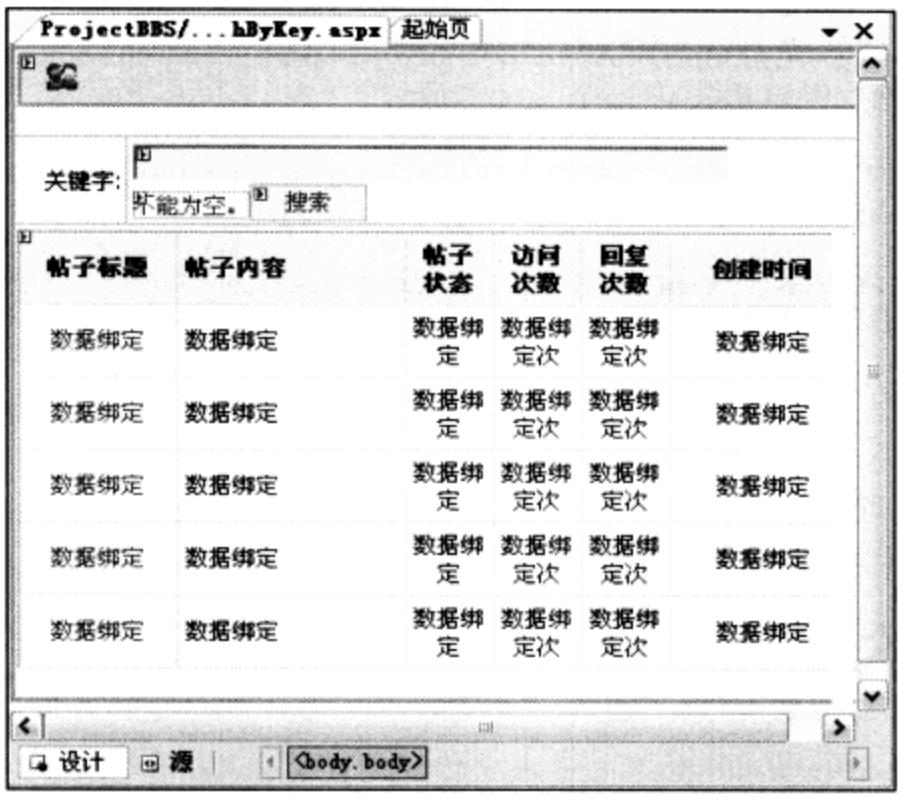


图 1.16　基于关键字的搜索页面设计效果图

该页面的 HTML 主要设计代码如下。

例程 33　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Search /SearchByKey.aspx

```
关键字:<asp:TextBox ID="tbKey" CssClass="TextBox" runat="server" SkinID="tbSkin" MaxLength="50" Width="300px">
</asp:TextBox><asp:RequiredFieldValidator ID="rfKey" runat="server" ControlToValidate="tbKey" ErrorMessage="不能为空。"
Display="Dynamic"></asp:RequiredFieldValidator><asp:Button ID="btnSearch" runat="server" Text="搜索" CssClass="Button"
OnClick="btnSearch_Click" SkinID="btnSkin" Width="60px" />
    <asp:GridView ID="gvTitle" runat="server" AutoGenerateColumns="False" Font-Names="Tahoma" Width="100%"
SkinID="gvSkin" DataKeyNames="ID">
    <HeaderStyle Font-Names="Tahoma" HorizontalAlign="Center" />
    <EmptyDataTemplate>
    // 帖子列表为空
    </EmptyDataTemplate><EmptyDataRowStyle ForeColor="Blue" />
    <RowStyle BorderColor="#add8e6" BorderStyle="Ridge" BorderWidth="1px" HorizontalAlign="Center" />
    <Columns>
    <asp:TemplateField ItemStyle-Width="15%" HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign= "Center"
HeaderText="帖子标题">
    <ItemTemplate><a href='../ViewTitle.aspx?TitleID=<%# Eval("ID") %>'><%# Eval("Name") %></a>
    </ItemTemplate></asp:TemplateField>
    <asp:TemplateField ItemStyle-Width="37%" HeaderStyle-HorizontalAlign="Left" ItemStyle-HorizontalAlign="left"
HeaderText="帖子内容">
```

```
        <ItemTemplate><%# Eval("Body").ToString().Length > 60 ? Eval("Body").ToString().Substring(0,60) + "..." : Eval("Body") %>
        </ItemTemplate></asp:TemplateField>
        <asp:TemplateField ItemStyle-Width="8%" HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign=
"Center" HeaderText="帖子状态">
        <ItemTemplate><%# FormatState(byte.Parse(Eval("State").ToString())) %></ItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField ItemStyle-Width="8%" HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign=
"Center" HeaderText="访问次数">
        <ItemTemplate><%# Eval("VisitNum") %>次</ItemTemplate></asp:TemplateField>
        <asp:TemplateField ItemStyle-Width="8%" HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign=
"Center" HeaderText="回复次数">
        <ItemTemplate><%# Eval("ReplyNum") %>次</ItemTemplate>
        </asp:TemplateField><asp:BoundField DataField="CreateDate"   ItemStyle-Width="18%" DataFormatString="{0:d}"
HeaderText="创建时间" HtmlEncode="false"/>
        </Columns></asp:GridView>
```

2. 代码编写

双击 SearchByKey.aspx 页面设计效果图中的“搜索”按钮，触发该按钮的 bynSearch_Click 事件，执行搜索操作，代码如下。

例程 34 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Search /SearchByKey.aspx.cs

```
protected void btnSearch_Click(object sender, EventArgs e)
{
    //实例化BBS公共类
    BBS bbs = new BBS();
    //执行删除操作
    MRSOFTASPNET.CommonOperation.DataBinder.BindGridViewData(
        gvTitle, bbs.GetTitleByNameKey(tbKey.Text.Trim()));
}
```

在上述事件代码中应用了公共类 BBS 中的 GetTitleByNameKey，该方法用来获取 XML 文件中的数据，代码如下。

例程 35 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Search \ SearchByKey.aspx.cs

```
public DataTable GetTitleByNameKey(string key)
{
    XmlParamter[] param = {
    XmlDatabase.CreateLikeParameter("Name",key)
};
// 获取数据
return (XmlDatabase.GetData(XmlBBS.TitleFilePath,TitleTableName,param));
}
```

1.7.2 基于时间的搜索

基于时间的搜索由页面 SearchByDate.aspx 实现，通过选择搜索的日期或时间来搜索相关帖子，并以列表形式显示搜索结果。页面运行结果如图 1.17 所示。

图 1.17 基于时间的搜索页面运行结果图

1. 页面设计

在该页面中添加了 1 个 Calendar 控件、1 个 Button 控件和 1 个 GridView 控件，这些控件的具体功能描述如表 1.7 所示。

表 1.7　　主要控件及用途

控件类型	控件名称	控件主要属性设置	用　途
标准/ Calendar	cDate	DayNameFormat 属性设置为 Shortest	选择搜索帖子的日期
标准/ Button	btnSearch	CssClass 属性设置为 Button SkinID 属性设置为 btnSkin	执行搜索操作
标准/ GridView	gvTitle	AutoGenerateColumns 属性设置为 False SkinID 属性设置为 gvSkin DataKeyNames 属性设置为 ID	显示搜索结果

该页的前台页面设计效果如图 1.18 所示。

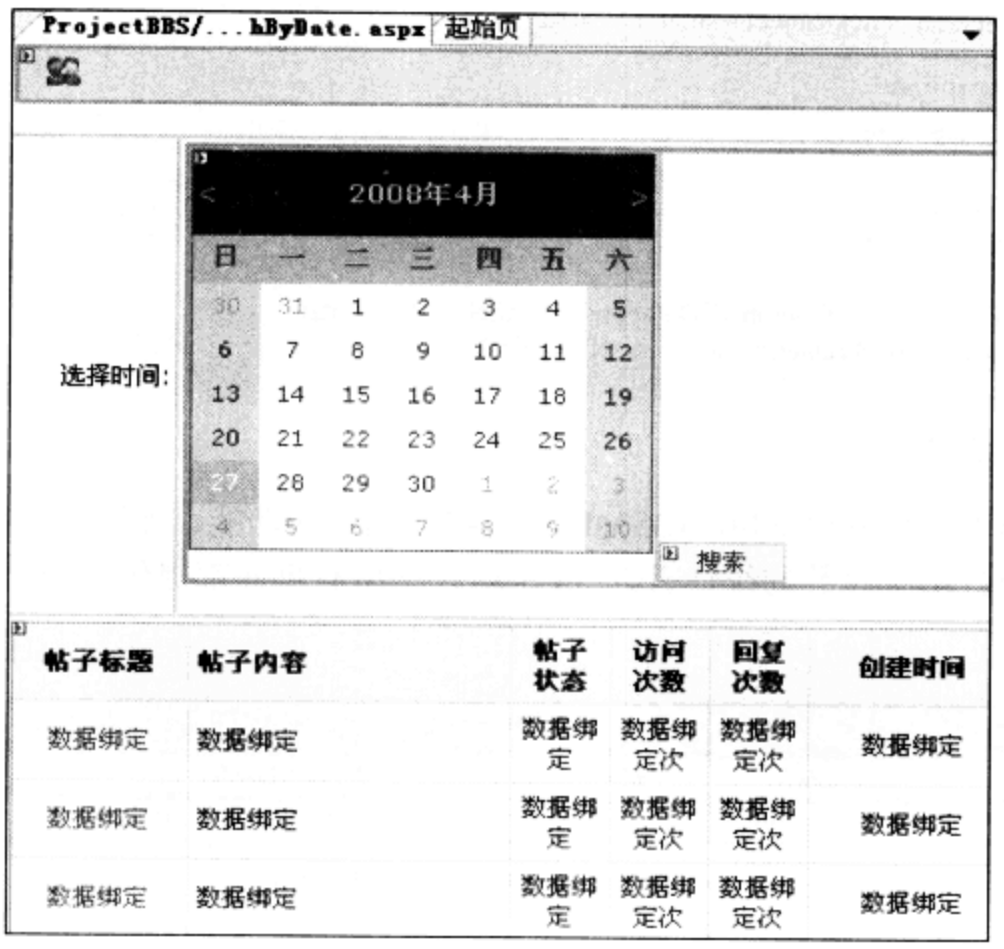


图 1.18　基于时间的搜索页面设计效果图

该页面的 HTML 主要设计代码如下：

例程 36　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Search \SearchByDate.aspx

```
选择时间<asp:Calendar ID="cDate" runat="server" BackColor="White" BorderColor="#3366CC" BorderWidth="1px"
CellPadding="1" DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"
ForeColor="#003399" Height="200px" Width="220px">
</asp:Calendar>
<asp:TemplateField ItemStyle-Width="15%" HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign=
"Center" HeaderText="帖子标题">
<ItemTemplate><a href='../ViewTitle.aspx?TitleID=<%# Eval("ID") %>'><%# Eval("Name") %></a>
</ItemTemplate></asp:TemplateField>
...//该处省略的代码与第2.10.1小节中相同位置处的代码相同
</Columns></asp:GridView>
```

2. 代码编写

页面装载 Page_Load 事件中设置日历控件显示的时间为系统当前时间，代码如下。

例程 37　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Search \SearchByDate.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
  if (!IsPostBack)
  {
      //设置日历控件为当前系统时间
      cDate.SelectedDate = DateTime.Now;
  }
}
```

DateTime.New 属性：获取 1 个 DateTime 对象，该对象设置为此计算机上的当前日期和时间，表示为本地时间。

双击 SearchByDate.aspx 页面设计效果图中的“搜索”按钮，触发该按钮的 bynSearch_Click 事件，执行搜索操作，代码如下。

例程 38　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Search \SearchByDate.aspx.cs

```
protected void btnSearch_Click(object sender, EventArgs e)
{
 //判断选择日期是否为空
 if (cDate.SelectedDate != null)
 {
     //实例化公共类BSS
     BBS bbs = new BBS();
  //执行搜索操作
  MRSOFTASPNET.CommonOperation.DataBinder.BindGridViewData(
      gvTitle, bbs.GetTitleByDate(cDate.SelectedDate));
 }
```

1.7.3　论坛帖子统计

本节主要介绍本应用程序 XMLBBSMR 中与统计相关的功能，如帖子总数的统计、帖子回复总数的统计和当日帖子总数的统计。该在线论坛几乎所有页面都提供了上述帖子统计的链接，如图 1.19 所示。

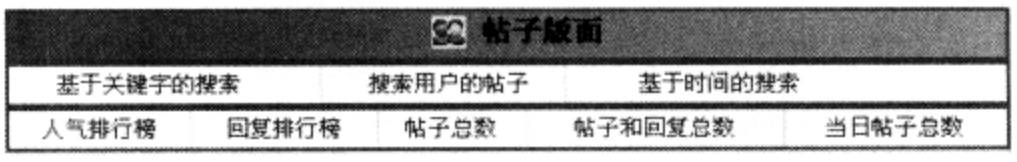


图 1.19　统计网站数据的链接按钮

该应用程序将此模块设计成通用模块，即做成了用户自定义控件。

1. 统计帖子总数

统计帖子总数由页面 StatTitleTotal.aspx 实现，在该页面的 Page_Load 事件中编写如下代码从数据库中获取统计到的帖子总数，并显示在指定页面上。

例程 39　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Stat /StatTitleTotal.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
     //实例化BBS
     BBS bbs = new BBS();
     //获取帖子数据
     DataTable dt = bbs.GetTitles();
     if (dt != null)
     {
         Response.Write("帖子总数为：  "+ dt.Rows.Count.ToString() + "<br>");
```

```
        }
    }
```

2. 统计帖子回复总数

统计帖子回复总数由页面 StatTitleReplyTotal.aspx 实现，在该页面的 Page_Load 事件中编写如下代码从数据库中获取统计到的帖子回复总数，并显示在指定页面上。

例程 40 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Stat /StatTitleTotal.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
//实例化BBS
    BBS bbs = new BBS();
//获取帖子信息
DataTable dtTitle = bbs.GetTitles();
//获取帖子回复信息
DataTable dtReply = bbs.GetReplys();
if (dtTitle != null && dtReply != null)
{
    Response.Write("帖子回复的总数为：  "+ (dtTitle.Rows.Count + dtReply.Rows.Count).ToString() + "<br>");
}
}
```

3. 统计当日帖子总数

统计帖子总数由页面 StatTitleDay.aspx 实现，在该页面的 Page_Load 事件中编写如下代码从数据库中获取统计到的当日帖子总数，并显示在指定页面上。

例程 41 代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Stat \ StatTitleReplyTotal.aspx .cs

```
protected void Page_Load(object sender, EventArgs e)
{
 //实例化公共类BBS
 BBS bbs = new BBS();
 //获取帖子总数
 DataTable dtTitle = bbs.GetTitleByDate(DateTime.Now);
 if (dtTitle != null)
  {
     Response.Write("今天的帖子总数为："
             + dtTitle.Rows.Count.ToString() + "<br>");
  }
}
```

1.7.4 热门帖子排行

热门帖子排行由页面 ViewHotTitle.aspx 实现，该页面按照帖子被访问的次数，以倒序方式显示系统中所有的帖子。页面运行结果如图 1.20 所示。

热门帖子排行

帖子标题	帖子内容	帖子状态	访问次数	回复次数	创建时间
2008.08.08日	08年北京奥运会开幕，我们支持北京！支持奥运会！One Word,One Dream!	普通帖子	36次	1次	2008-8-25
ASP.NET编程	ASP.NET编程明日科技图书：ASP.NET程序开发范例宝典、ASP.NET2.0编程自学手册、SQL范例宝典、SQL...	普通帖子	26次	1次	2008-6-25
明日编程词典	明日编程词典,www.mrbccd.com全程服务，编程无忧！	普通帖子	16次	1次	2008-5-25
.NET编程	public void class fangawei	普通帖子	12次	1次	2008-5-25
mrfangdawei	明日科技是我的家，我家它，我会用心呵护它！	携带附件的帖子	0次	0次	2008-4-23

图 1.20 热门帖子排行页面运行结果图

1. 页面设计

在页面 ViewHotTitle.aspx 添加了 1 个 GridView 控件，以帖子访问次数来显示帖子的排行，其 ID 属性值为 gvTitle。

该页的前台页面设计效果如图 1.21 所示。

图 1.21　热门帖子排行页面设计效果图

该页面的 HTML 主要设计代码如下。

例程 42　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Stat \ ViewHotTitle.aspx

```
    <asp:GridView ID="gvTitle" runat="server" AutoGenerateColumns="False" Font-Names="Tahoma" Width="100%"
SkinID="gvSkin" DataKeyNames="ID">
    <Columns>
    <asp:TemplateField ItemStyle-Width="15%" HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign=
"Center" HeaderText="帖子标题">
    <ItemTemplate>
    <a href='../ViewTitle.aspx?TitleID=<%# Eval("ID") %>'><%# Eval("Name") %></a>
    </ItemTemplate></asp:TemplateField>
        ...//该处省略的代码与第2.10.1小节中相同位置处的代码相同
    </Columns></asp:GridView>
```

2. 代码编写

在页面 Page_Load 事件中首先判断页面是否首次加载，然后调用自定义方法 BindPageData() 按照帖子被访问的次数以倒序的方式显示系统中所有的帖子，这些帖子都显示在 gvTitle 控件中。

页面装载 Page_Load 事件和自定义方法 BindPageData()的程序代码如下。

例程 43　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Stat \ ViewHotTitle.aspx.cs

```
    protected void Page_Load(object sender, EventArgs e)
     {
        if (!Page.IsPostBack)
          {
```

```
        //调用自定义方法BindPageData绑定控件数据
                BindPageData();
            }
        }
        private void BindPageData()
        {
            //实例化公共类BBS
            BBS bbs = new BBS();
            //定义一个DataTalbe内存表
            DataTable dtTitle = bbs.GetTitles();
            if (dtTitle == null) return;
            //添加访问次数列
            dtTitle.Columns.Add("VisitNumber", typeof(int));
            //设置每一行的值
            foreach (DataRow row in dtTitle.Rows)
            {
                row["VisitNumber"] = row["VisitNum"];
            }
            //定义一个该内存表的视图，获取排序后的数据
            DataView dvTitle = dtTitle.DefaultView;
            dvTitle.Sort = "VisitNumber DESC";
            //绑定gvTitle控件的数据
            MRSOFTASPNET.CommonOperation.DataBinder.BindGridViewData(
                gvTitle, dvTitle.ToTable());
        }
```

1.7.5 热门回复帖子排行

热门回复帖子排行由页面 AnswerHotTitle.aspx 实现，该页面按照帖子被回复的次数，以倒序方式显示系统中所有的帖子。页面运行结果如图 1.22 所示。

帖子标题	帖子内容	帖子状态	访问次数	回复次数	创建时间
明日好朋友	明日科技ASPNET部门是可好的朋友了，他们几个从来不打架	普通帖子	9次	3次	2008-11-25
我爱明日科技	我爱明日科技这个家庭，它给我了温暖和一颗爱心！	普通帖子	9次	3次	2008-4-17
明日理念	用今日的辛勤汗水，换明日的百倍回报，明日科技是可好的朋友了	普通帖子	10次	2次	2008-6-25
.NET编程	public void class fangawei	普通帖子	12次	1次	2008-5-25
ASP.NET编程	ASP.NET编程明日科技图书：ASP.NET程序开发范例宝典、ASP.NET2.0编程自学手册、SQL范例宝典、SQL...	普通帖子	26次	1次	2008-6-25

图 1.22 热门回复帖子排行运行结果图

1. 页面设计

页面 AnswerHotTitle.aspx 添加了 1 个 GridView 控件，以帖子被回复的次数来显示帖子的排行，其 ID 属性值为 gvTitle。

该页的前台页面设计效果如图 1.23 所示。

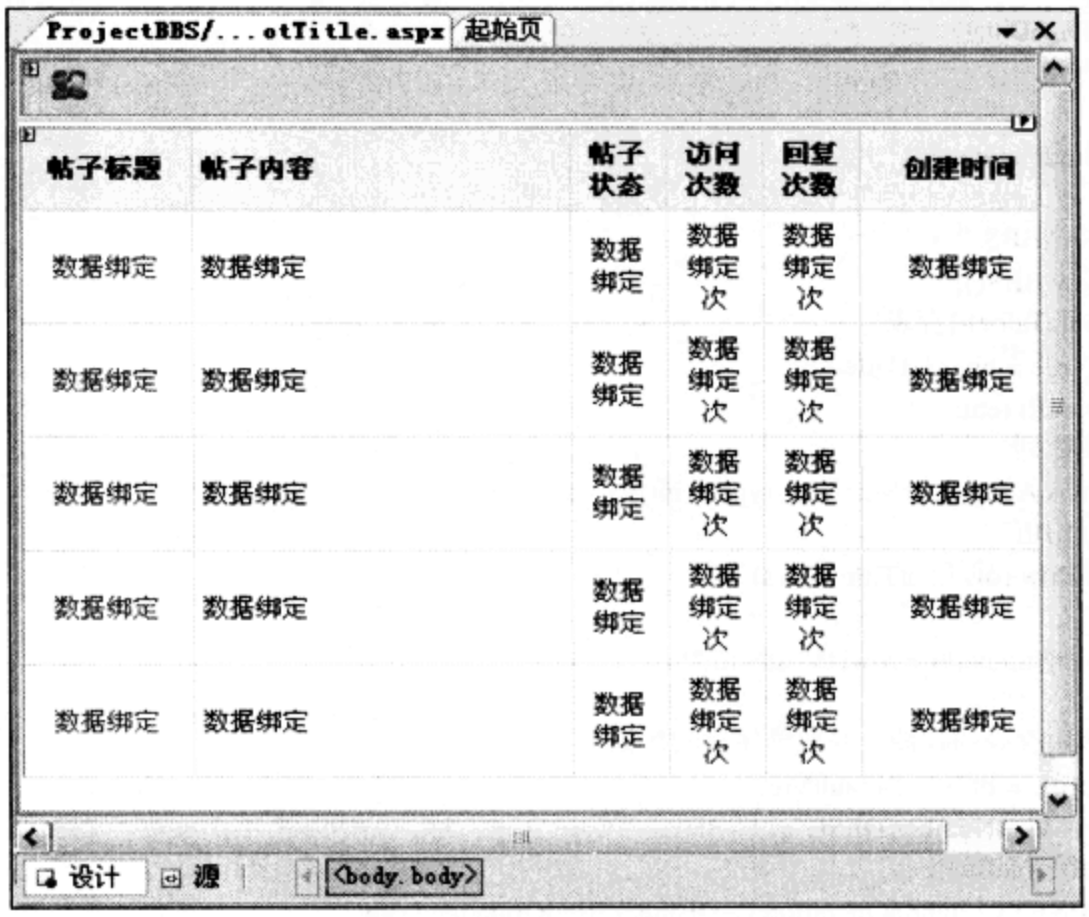


图 1.23　热门帖子排行页面设计效果图

该页面的 HTML 主要设计代码如下。

例程 44　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Stat \ AnswerHotTitle.aspx

```
    <asp:GridView ID="gvTitle" runat="server" AutoGenerateColumns="False" Font-Names="Tahoma" Width="100%"
SkinID="gvSkin" DataKeyNames="ID">
    <Columns>
    <asp:TemplateField ItemStyle-Width="15%" HeaderStyle-HorizontalAlign="Center" ItemStyle-HorizontalAlign=
"Center" HeaderText="帖子标题">
    <ItemTemplate>
    <a href='../ViewTitle.aspx?TitleID=<%# Eval("ID") %>'><%# Eval("Name") %></a>
    </ItemTemplate></asp:TemplateField>
    ...//该处省略的代码与第2.10.1小节中相同位置处的代码相同
    </Columns></asp:GridView>
```

2. 代码编写

在页面 Page_Load 事件中首先判断页面是否首次加载，然后调用自定义方法 BindPageData()，按照帖子被回复的次数以倒序的方式显示系统中所有的帖子，这些帖子都显示在 gvTitle 控件中。

页面装载 Page_Load 事件和自定义方法 BindPageData()的程序代码如下。

例程 45　代码位置：光盘\mr\01\XMLBBS\ ProjectBBS \ Stat \ AnswerHotTitle.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
//调用自定义方法BindPageData绑定控件数据
        BindPageData();
    }
}
private void BindPageData()
{
  //实例化公共类BBS
  BBS bbs = new BBS();
  //定义DataTable内存表，获取帖子信息
```

```
    DataTable dtTitle = bbs.GetTitles();
    if (dtTitle == null) return;
    //添加回复次数列
    dtTitle.Columns.Add("ReplyNumber", typeof(int));
    //设置每一行的值
    foreach (DataRow row in dtTitle.Rows)
    {
        row["ReplyNumber"] = row["ReplyNum"];
    }
    //定义该表的一个默认视图
    DataView dvTitle = dtTitle.DefaultView;
    dvTitle.Sort = "ReplyNumber DESC";
    //绑定gvTitle控件的数据
    MRSOFTASPNET.CommonOperation.DataBinder.BindGridViewData(
    gvTitle, dvTitle.ToTable());
}
```

1.8 程序打包与发布

网站开发完成后最终的目地是将其发布到 Internet 上，以提供用户浏览访问。实现的网站发布可以使用两种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍下使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

（1）首先打开要发布的网站，在网站的项目名称上，单击鼠标右键后弹出的快捷方式菜单中选择“发布网站”选项，如图 1.24 所示。

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮，网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择“...”路径按钮。如图 1.25 所示。

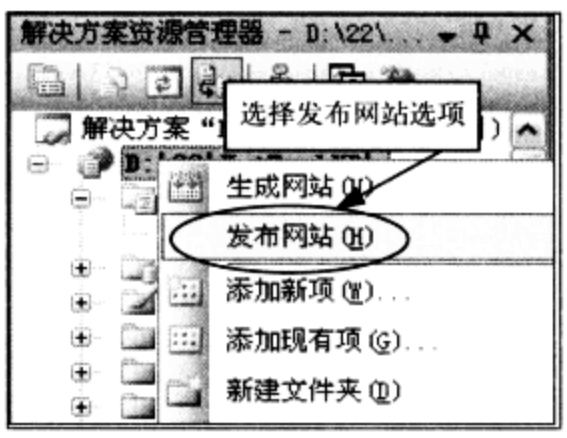


图 1.24 选择发布网站

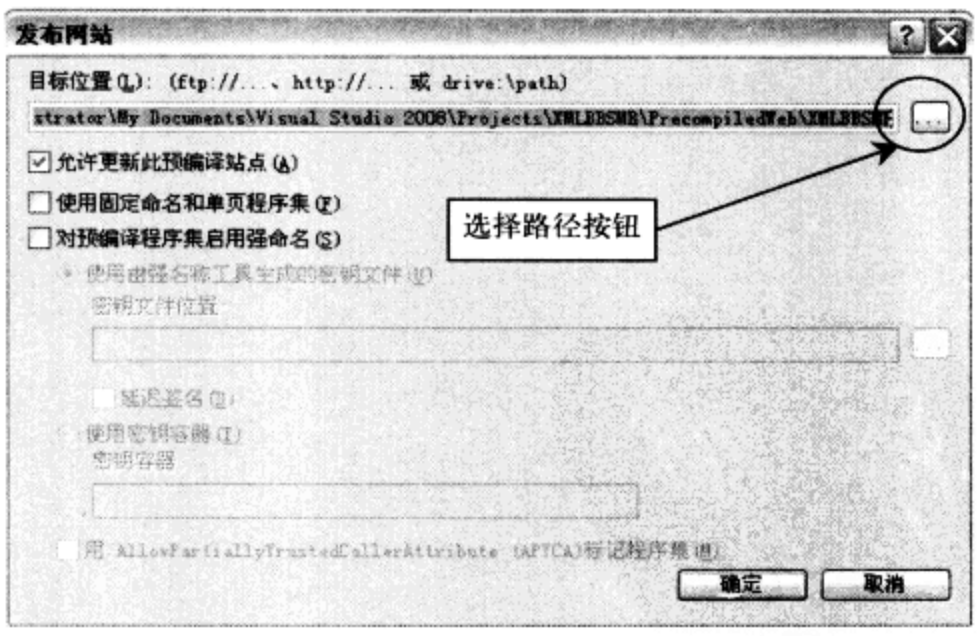


图 1.25 选择路径按钮

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选择中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码如图 1.26 所示。填写完毕后选择“打开”按钮将会返回到“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

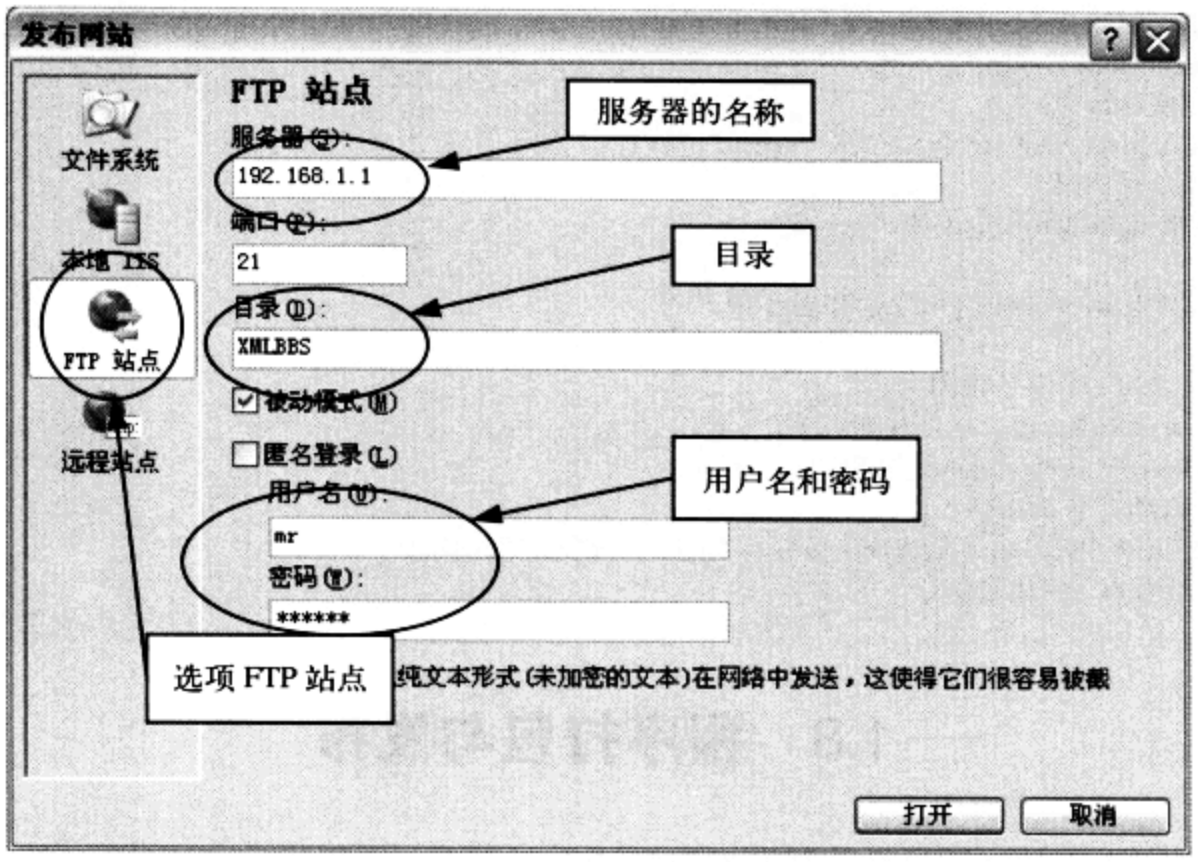


图 1.26　FTP 站点选项

博客模块

第 2 章

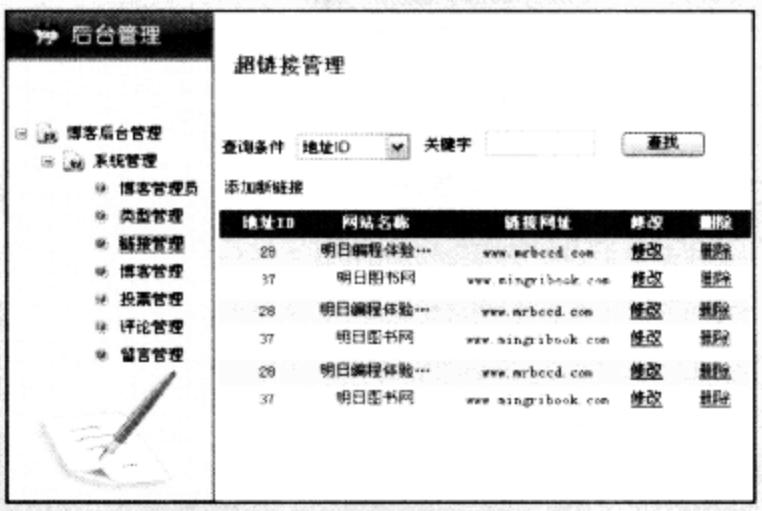
实例位置：光盘\mr\02\

博客的个性化和平民视角使得它提供的消息更贴近人们的生活，所以很多人都想建立自己的网络空间。过去，传统的网络交往方式主要是留言本、BBS（论坛）、聊天室及 IM（即时通信）等，但它们或多或少都存在着不足。留言本主要用来留言，不能进行留言回复；BBS 主要用来探讨问题；IM 要想发挥作用，必须要求交流的双方同时在线；而聊天室更是闲人的乐园。博客的存在，可以说是一种网络的虚拟社区。在这里用户可以通过网络日志的形式方便快捷地发表自己的心得体会，及时有效并轻松地与他人交流。本模块将介绍如何创建一个博客系统。通过本章的学习，读者能够学到以下内容。

博客文章评论管理

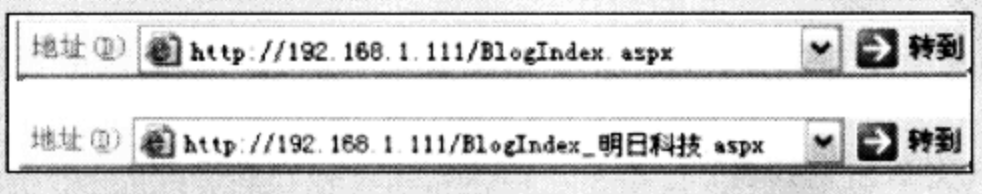
博客文章管理

博客链接管理



热点文章排行

通过 IE 地址栏进入用户 Blog



2.1 模块功能概述

随着博客（Blog）的快速发展，博客已经成为朋友、家庭之间越来越盛行的沟通工具，是当今网络不可缺少的一部分。博客（Blog）作为网络的一种表现形式正在越来越受到大家的欢迎。博客最主要的应用有两个方面：一是新的人际交流方式；二是以个人为中心的传播出版。其中以具有鲜明个人特色的传播出版引人瞩目。以个人为中心的博客，以独特的视角、敏锐的观察力，逐渐冲击着传统媒体，尤其是新闻界多年来形成的传统观念和道德规范。

本模块介绍如何设计与开发一个博客系统。用户可以在该博客模块中创建属于自己的博客空间，以发表自己的文章，并对其他博客好友文章进行浏览、评论等操作，管理员可对博客发表的文章进行分类管理、网站的链接管理、用户信息管理等。另外，在博客首页中设置了如热点文章排行、博客最新发表文章等功能。

2.2 数据库设计

在程序开发过程中，数据库设计是非常重要的一个环节。一个设计良好的数据库结构，可以提高效率，方便维护，并且为以后进行功能的扩充留有余地。这就好比高楼大厦一样，有稳固的基础，才能有优秀的成果。

2.2.1 数据库概要说明

本网站采用 SQL Server 2000 数据库，名称为 db_Blog，其中包含 9 张数据表。

从读者角度出发，使读者对本网站数据库中数据表有一个更清晰的认识，笔者在此设计了数据表树型结构图，如图 2.1 所示。其中包含了对系统中所有数据表的相关描述。

图 2.1 系统数据表结构图

2.2.2 数据库逻辑设计

由于本书的篇幅所限，笔者在此只给出较重要的数据表，其他数据表请参见本书附带的光盘。

- ST_news（文章信息列表）

文章信息列表用来记录文章信息，如表 2.1 所示。

表 2.1 **ST_news（文章信息列表）**

字　段	类　型	长　度	是否为空	说　明
ST_n_id	int	4	否	文章惟一 ID 号
ST_n_author	nvarchar	50	是	文章作者
ST_n_title	nvarchar	200	是	文章标题
ST_n_key	nvarchar	200	是	文章摘要
ST_n_content	ntextr	50	是	文章内容
ST_n_date	datetime	8	是	文章发表日期
ST_n_hit	bigint	8	是	文章人气
ST_n_re	bigint	8	是	文章回复
ST_c_id	bigint	8	是	文章类型编号
ST_c_name	nvarchar	50	是	文章类型名称
ST_n_iscmd	int	4	是	文章回复数

- ST_replay（评论信息列表）

评论信息列表用来保存评论信息，如表 2.2 所示。

表 2.2 **ST_replay（评论信息列表）**

字　段	类　型	长　度	是否为空	说　明
ST_r_id	bigint	8	否	评论惟一 ID 号
ST_r_nick	nvarchar	50	是	昵称
ST_r_title	nvarchar	250	是	评论标题
ST_r_content	nvarchar	250	是	评论内容
ST_r_date	datetime	8	是	评论发表时间
ST_n_id	int	4	是	评论文章编号

- ST_message（留言信息列表）

留言信息列表用来保存留言信息，如表 2.3 所示。

表 2.3 **ST_message（留言信息列表）**

字　段	类　型	长　度	是否为空	说　明
ST_id	bigint	8	否	留言惟一 ID 号
ST_nickname	nvarchar	50	是	网友昵称
ST_title	nvarchar	50	是	留言标题
ST_homepage	nvarchar	50	是	个人主页
ST_content	ntext	16	是	留言内容
ST_mdate	datetime	8	是	留言时间
ST_hf	ntext	16	是	留言回复

2.3 关键技术详解

2.3.1 通过 IE 地址栏进入用户 Blog

当注册为该博客的会员并发表了自己的博客文章后，如图 2.2 和图 2.3 所示（在配置完的 IIS 中运行的结果），在主界面的 IE 地址栏内指定位置输入一个下划线加上用户注册的用户名

（例如，“_明日科技，_编程词典”等）后即可快速查找用户所发表的文章。同样，可以以这种方式来快速查找该博客中自己所发表的文章。

图 2.2　通过 IE 地址栏快速查找用户文章 1

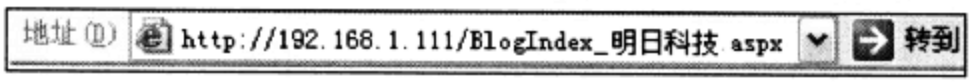


图 2.3　通过 IE 地址栏快速查找用户文章 2

实现以上功能，主要是应用了 URL 重写。URL 重写是截取传入 Web 请求并自动将请求重定向到其他 URL 的过程，比如浏览器发来请求 hostname/188.aspx，服务器则自动将这个请求中定向为 http://hostname/Default.aspx?id=188。该重写功能的实现主要是在 ASP.NET 中引入了一个名为 URLRewriter 的 dll 文件，该 dll 文件可构建一个使用简单 URL 重写的 ASP.NET Web 应用程序。应用 URL 重写可缩短 url，隐藏实际路径提高安全性；易于用户记忆和输入；易于被搜索引擎收录。

URLRewriter.dll 文件可以从微软官方网站上下载。

在项目的 Bin 文件夹下添加引用了 URLRewriter.dll 文件后，在 web.config 里设置如下代码，除了实现主界面的 IE 地址栏指定位置内输入一个下划线加上用户名后可快速查找到所发表的全部文章这个功能外，还实现了在博客主界面中当用户单击“点击阅读全文”超级链接和单击“博客文章类型”列表中任一个类别时进行 URL 重写。

例程 1　代码位置：光盘\mr\02\myBlog\ Web.config

```
<?xml version="1.0"?>
<configuration>
    <configSections>
        <section name="RewriterConfig" type="URLRewriter.Config.RewriterConfigSerializerSectionHandler, URLRewriter"/>
    <connectionStrings>
<RewriterConfig>
    <Rules>
        <!-- Rules for Blog Content Displayer -->
        <RewriterRule>
            <LookFor>~/(\d+)\.aspx</LookFor>
            <SendTo>~/ST_show.aspx?id=$1</SendTo>
        </RewriterRule>
        <RewriterRule>
            <LookFor>~/BlogIndex_(.*)_(\d+)\.aspx</LookFor>
            <SendTo>~/BlogIndex.aspx?c_id=$2&amp;name=$1</SendTo>
        </RewriterRule>
  <RewriterRule>
    <LookFor>~/BlogIndex_(.*)\.aspx</LookFor>
    <SendTo>~/BlogIndex.aspx?name=$1</SendTo>
  </RewriterRule>
    </Rules>
</RewriterConfig>
```

其中：

```
<section name="RewriterConfig" type="URLRewriter.Config.RewriterConfigSerializerSectionHandler, URLRewriter"/>
```

上述代码用于指定配置节 RewriterConfig 的处理程序类的名称为 URLRewriter.Config.RewriterConfigSerializerSectionHandler，该类存在于 bin 目录下的 URLRewriter .dll 文件中。

在上述 Web.config 配置文件中，最关键的代码类似于以下代码：

```
<LookFor>~/d(\d+)\.aspx</LookFor>
<SendTo>~/default.aspx?id=$1</SendTo>
```

其中，<LookFor></LookFor>标记内的元素表示用户输入的 url，d(\d+)\.aspx 是 url 中文件名匹配的正则表达式（此处为字母 d 开头，后面跟一个或多个数字，并以.aspx 结尾。用户也可根据自己的需要自行设定）；而<SendTo></SendTo>标记内的元素表示当服务器接收到符合上面条件的请求后如何重写 url，此处表示访问 defalutl.aspx 并传入参数 id，其值$1 将通过用户请求的文件名中的第 1 个数字来表示。

例如，用户输入 hostname/d188.aspx，服务器会把它重写为 http://hostname/default.aspx?id=188。换句话说用户输入 http://hostname/d188.aspx，实际访问的是 http://hostname/default.aspx?id=188，这样就可隐藏真实文件名，并便于用户记忆。

2.3.2 Iframe 框架技术

在博客管理员界面首页设计中，主要应用了页面框架布局中的 Iframe 框架技术。应用此技术可以在同一个页面中多次显示同一内容，而不必重复这段内容的代码。通过此框架将网站中各部分独立的网页重新组成一个完整的网页，并显示在浏览器中。<iframe>是框架的一种形式，它与<frame>不同的是，iframe 可以嵌在网页中的任意部分。Iframe 框架就像是“画中画”电视。

与其他设计页面的框架相比，内嵌框架 Iframe 更容易对网站的导航进行控制，最大的优点在于其灵活性。正是基于此原因，Iframe 框架多被用于网站的后台管理。Iframe 框架的标记为<Iframe>（又叫浮动帧标记），可以用它将一个 HTML 文档嵌入在一个 HTML 中显示。它和<Frame>标记的最大区别是在网页中嵌入的<Ifame></Iframe>所包含的内容与整个页面是一个整体，而<Frame></Frame>所包含的内容是一个独立的个体，是可以独立显示的。

设置 Iframe 框架的 Iframe 参数的示例代码如下：

```
<iframe  src="url" name="main"  width="100%" height="30" frameborder="0" border=0 scrolling="no" marginwidth=
"0" marginheight="0">
</iframe>
```

上述代码的各项参数设置及含义如表 2.4 所示。

表 2.4　　Iframe 参数及说明

参　数	说　明
Name	设定框架的名称，须为英文
Src	设置框架中显示的页面路径和名称，为相对路径或绝对路径
Marginwidth	表示框架距离左右边缘的距离
Marginheight	表示框架距离上下边缘的距离
Scrollling	设置是否在框架中显示滚动条，yes 为显示，no 为不显示，auto 表示当框架页中内容超过框架的大小时自动显示滚动条
Frameborder	设置是否显示框架的边框，0 为不显示，1 为显示
Noresize	设定是否可以让使用者改变这个框架的大小，不设置此项可以让浏览者任意拉动框架，改变框架的大小
Framespacing	表示框架与框架之间的距离
Bordercolor	设定框架的边框颜色

2.3.3 GridView 控件中数据实现全选或复选

GridView 控件与 CheckBox 控件结合应用，主要是在 GridView 控件中添加 1 个 CheckBox 列，用于选择 GridView 控件中的数据，实现数据的全选功能，同时也实现删除所选择的多行数据的功能。另外，在 GridView 控件中查找 CheckBox 控件应用了 GridView 控件的 FindControl 方法，该方法主要用来在当前控件中搜索指定 id 的服务器控件，其使用方法如下：

```
Public virtual Control FindControl(string id);
```

其中参数 id 指的是要查找的控件的标识符。

返回值：如果指定的控件存在，则返回该控件，否则返回空。

博客个人文章管理页（ArticleManage.aspx）应用了这种方法实现了数据的批量删除操作，下面详细介绍。其运行效果如图 2.4 所示。

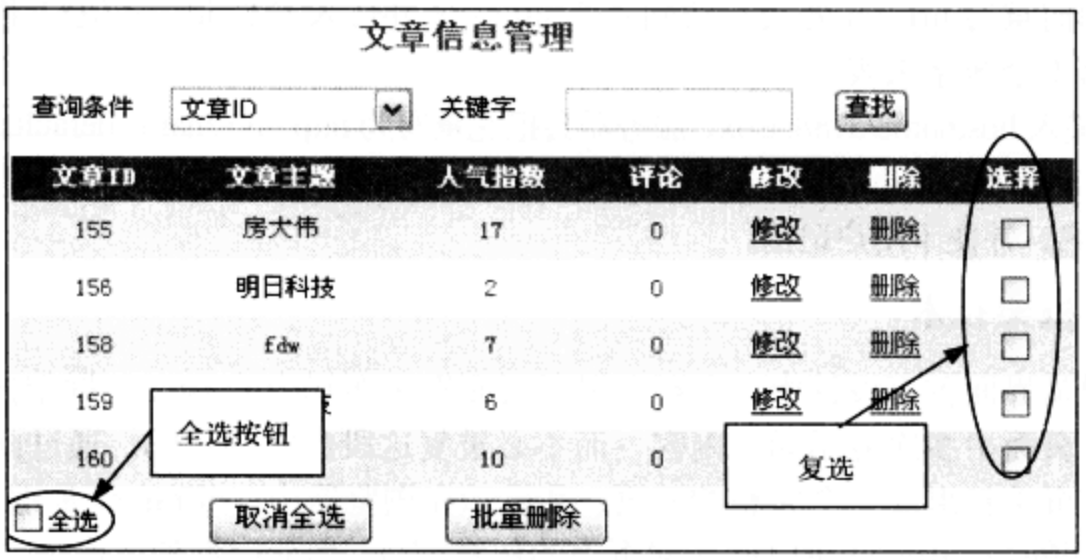


图 2.4　GridView 控件与 CheckBox 控件的应用图

双击博客个人文章管理页(ArticleManage.aspx)中的“批量删除”按钮，触发其 btnAllDelete_Click 事件，该事件首先判断 GridView 控件中哪些行被选中，并对选中行执行删除操作，然后对 GridView 控件进行重新绑定数据，以显示最新信息，具体实现的事件代码如下。

例程 2　代码位置：光盘\mr\02\myBlog\Module \Blog\ ArticleManage.aspx

```
protected void btnAllDelete_Click(object sender, EventArgs e)
 {
     //为btnAllDelete控件添加1个onClick属性
     btnAllDelete.Attributes.Add("onClick","javascript:return confirm('确定删除吗？ ');");
     //应用for循环语句遍历GridView控件中所有行
     for (int i = 0; i <= GridView1.Rows.Count - 1; i++)
     {
         //应用FindControl方法查找GridView控件中CheckBox控件
         CheckBox cbox = (CheckBox)GridView1.Rows[i].FindControl("CheckBox1");
         //判断是否选中GridView控件中的CheckBox控件
         if (cbox.Checked == true)
         {
             //定义删除选中行数据的字符串
             string sqlstr = "delete from ST_news where ST_n_id='" + GridView1.DataKeys[i].Value + "'";
             //调用公共类中的ExceSQL方法执行删除操作的SQL语句
             da.ExceSQL(sqlstr);
             //执行成功，弹出提示框信息
             Response.Write("<script language=javascript>alert('批量删除成功！ ');location='ArticleManage.aspx'</script>");
         }
     }
     //重新绑定数据
     ST_Article_Bind();
 }
```

当“全选”CheckBox 框中选项发生变化时，循环访问 GridView 控件中的每一项，并使用 FindControl 方法查找到添加的 CheckBox 控件，然后建立该控件的引用对象，并使用该对象进行操作，实现全选功能。CheckBox 控件的 CheckedChanged 事件代码如下。

例程 3　代码位置：光盘\mr\02\myBlog\Module \Blog\ ArticleManage.aspx

```
protected void CheckBox2_CheckedChanged(object sender, EventArgs e)
{
  //应用for循环语句遍历GridView控件中所有行
```

```
for(int i=0; i<=GridView1.Rows.Count-1;i++)
{
    //查找GridView控件中的CheckBox1控件
    CheckBox cbox=(CheckBox)GridView1.Rows[i].FindControl("CheckBox1");
    if(checkAll.Checked==true)
    {
        //全选中
        cbox.Checked=true;
    }
    else
    {
        //全取消
        cbox.Checked=false;
    }
}
```

单击“取消全选”按钮时，首先将“全选”按钮的 Checked 属性设为 false，然后使用 FindControl 方法查找到添加的 CheckBox 控件，并使其 Checked 属性设为 false。具体的事件代码如下。

例程 4　代码位置：光盘\mr\02\myBlog\Module \Blog\ ArticleManage.aspx

```
protected void Button1_Click(object sender, EventArgs e)
{
    //全选按钮被取消
    checkAll.Checked = false;
    //应用for循环语句遍历GridView控件中所有行
    for (int i = 0; i <= GridView1.Rows.Count - 1; i++)
    {
        //查找GridView控件中的CheckBox1控件
        CheckBox cbox = (CheckBox)GridView1.Rows[i].FindControl("CheckBox1");
        //取消选择
        cbox.Checked = false;
    }
}
```

2.3.4　母版页技术

母版页的使用基本受益于 ASP.NET Framework 的将一个“超级模板”与用户定义的内容进行合并的能力。换句话说，在 ASP.NET 中，可以将母版页看作是一种具有多项高级功能的页面模板，在这个页面模板中可以将 Web 应用程序中的公用元素如网站标志、广告栏、导航栏、版权声明等内容统一整合到这个模板中。

1. 母版页的分析与使用

除了开头的@Master 指令和一个或多个 ContentPlaceHolder 服务器控件外，母版页基本上类似于标准的 ASP.NET 页。例如，二者在代码结构方面都需要声明<html>、<body>、<form>以及其他 Web 元素等。惟一的区别就是在母版页中使用的 ContentPlaceHolder 控件（普通的.aspx 文件中不允许使用该控件）。此控件起到一个占位符的作用，它能够在母版页中标识出某个区域，可以让相关网页插入定制控件的位置。

注　意

ContentPlaceHolder 控件本身并不包含具体内容设置，仅是一个控件声明而已。在前文讲到的页面非公共部分就是通过此控件来实现占位的，即作为代替页面非公共内容的占位符出现，而具体内容则被放置在内容页中（稍后将对内容页进行分析）。

母版面中的容器控件——ContentPlaceHolder 控件，其内容占位符由一个 ID 惟一标识。例如下面的代码：

```
<asp: ContentPlaceHolder runat="server" ID=" ContentPlaceHolder1">
```

在母版页中，可以根据网页中所需的可定制区域数定义多个内容占位符。而内容页中不必填充所绑定的母版页定义的全部占位符。

说 明

内容占位符可以被赋予默认内容，在内容页没有提供替代内容时，即内容页没有引用母版页中的一个给定占位符，则使用默认内容。如下代码定义了默认内容：

```
<asp: ContentPlaceHolder runat="server" ID=" ContentPlaceHolder1">
  <!—Use the follow markup if no custom content is provided by the content page →
...
</asp: ContentPlaceHolder>
```

但当内容页填充了占位符时，设置的默认内容将被完全忽略，且默认内容决不会与内容页提供的定制标记合并。ContentPlaceHolder 控件（内容占位符）在标准的 ASP.NET 页上是不合法的，只能在母版页上使用。若在普通的 Web 页中发现，则会发生一个解析器错误。

创建一个母版页的过程并不复杂。在解决方案资源管理器中，用鼠标右键单击选择“添加新项”选项，将弹出如图 2.5 所示的窗口。在此对话框中选择母版页图标，并设置文件名为 MasterPage.master。在该窗体中还有一个“将代码放在单独的文件中”复选框项。默认情况下，该复选框处于选中状态。

图 2.5　创建母版页

注 意

如果不选中“将代码放在单独的文件中”复选框，Visual Studio 2008 将不会自动创建与 MasterPage.master 文件相匹配的代码隐藏类文件，即 MasterPage.master.cs 文件。这里建议最好选中此项。

此时单击“添加”按钮，“母版页”将会被添加到解决方案资源管理器，如图 2.6 所示。默认的“母版页”如图 2.7 所示。

图 2.6　将母版页添加到解决方案中

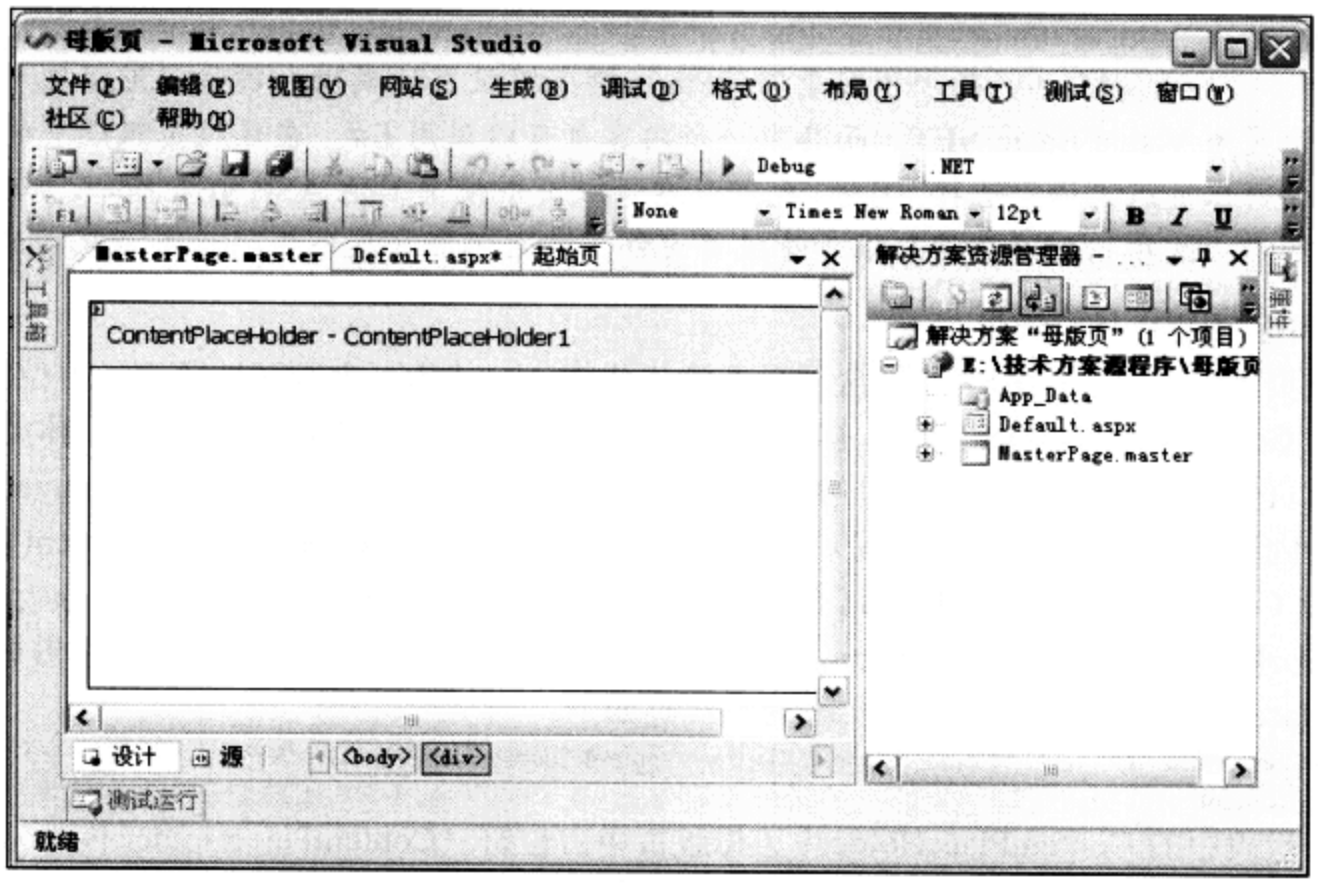


图 2.7 默认的母版页

常见母版页的代码如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
<html >
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:contentplaceholder id="ContentPlaceHolder1" runat="server">
        </asp:contentplaceholder>
    </div>
    </form>
</body>
</html>
```

母版页的页面代码头

ContentPlaceHolder 控件

可以看出母版页 MasterPage.master 的源代码与标准的.aspx 源代码非常相似。但它们之间仍存在一定的差别，其差别在代码块中已经标出，即页面代码头和 ContentPlaceHolder 控件。

最后着重说明一下母版页中应用的@Master 指令，此指令使母版页有别于内容页。@Master 指令的大多属性与@Page 指令的属性相同。表 2.5 将详细描述一下对母版页有特殊含义的属性。

表 2.5　　@Master 指令的属性

属　性	描　述
ClassName	指定为生成母版页而创建的类的名称。该值可以是任何一个有效的类名，但不用包括命名空间
CodeFile	指明包含与母版页关联的任何源代码的文件的 URL
Inherits	指定母版页要继承的代码隐藏类，可以是任何一个派生于 MasterPage 的类
MasterPageFile	指定该母版页引用的母版页的名称。通过使用网页来引用一个母版页的相同方法，一个母版页可以引用另一个母版页。如果设置了该属性，则会得到一个嵌套的母版页

说 明

@Master 指令不会重写@Page 指令级设置的属性。例如，可以把母版页中的语言设置为 Visual Basic.NET，而其中一个内容页可以使用 C#。在母版页级设置的语言决不会影响内容页级的语言选择。同时，也可以在母版页中使用其他 ASP.NET 指令，如@Import 指令。然而，这些指令的范围只限于母版页文件，并且不能扩展到根据母版页生成的子页。

2. 内容页的分析与使用

母版页定义了网页的构架，包含了页面的公共部分，并为可定制区域留下了占位符（即母版页中的 ContentPlaceHolder 控件），接下来内容页就要为这个特定的占位符提供具体的内容了。

内容页是只包含<asp:Content>服务标签的 ASP.NET 页，它的关键部分就是 Content 控件，可称之为内容控件。母版页中的占位符所对应的具体内容就包含在该控件中。Content 控件只能与对应的 ContentPlaceHolder 控件结合使用，它不是一个独立的控件。内容页与母版页的有机结合就是通过内容页中的 Content 控件的 ContentPlaceHolderID 属性绑定到母版页中的 ContentPlaceHolder 控件的 ID 属性实现的，代码如下：

```
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
</asp:Content>
```

以上代码中的 ContentPlaceHolder1 为母版页中占位符（ContentPlaceHolder 控件）的 ID 属性值。代码中的<asp:Content>标签元素的作用就像一个控件容器，非常像 ASP.NET 的 Panel 控件或 HTML 标签。

注 意

内容页是用户通过浏览器调用的资源，它是一种特殊的网页类型（即绑定到一个母版页的网页），它只能包含 Content 控件。并且在<asp:Content>标签外部是不允许有服务器控件或文本文字的。同时在一个内容页中可以有一个或多个 Content 控件，但它们必须指向母版中的不同占位符。

与创建母版页差不多，创建内容页的过程也比较简单。但在创建内容页的过程中，始终要牢记两点：一是内容中所有内容必须包含在 Content 控件中；二是内容页必须绑定母版页。在 Visual Studio 2008 中单击菜单栏中“网站”→“添加新项”命令或在解决方案中的新建网站名称上单击鼠标右键，并选择“添加新项”命令，将会弹出如图 2.8 所示的窗口。

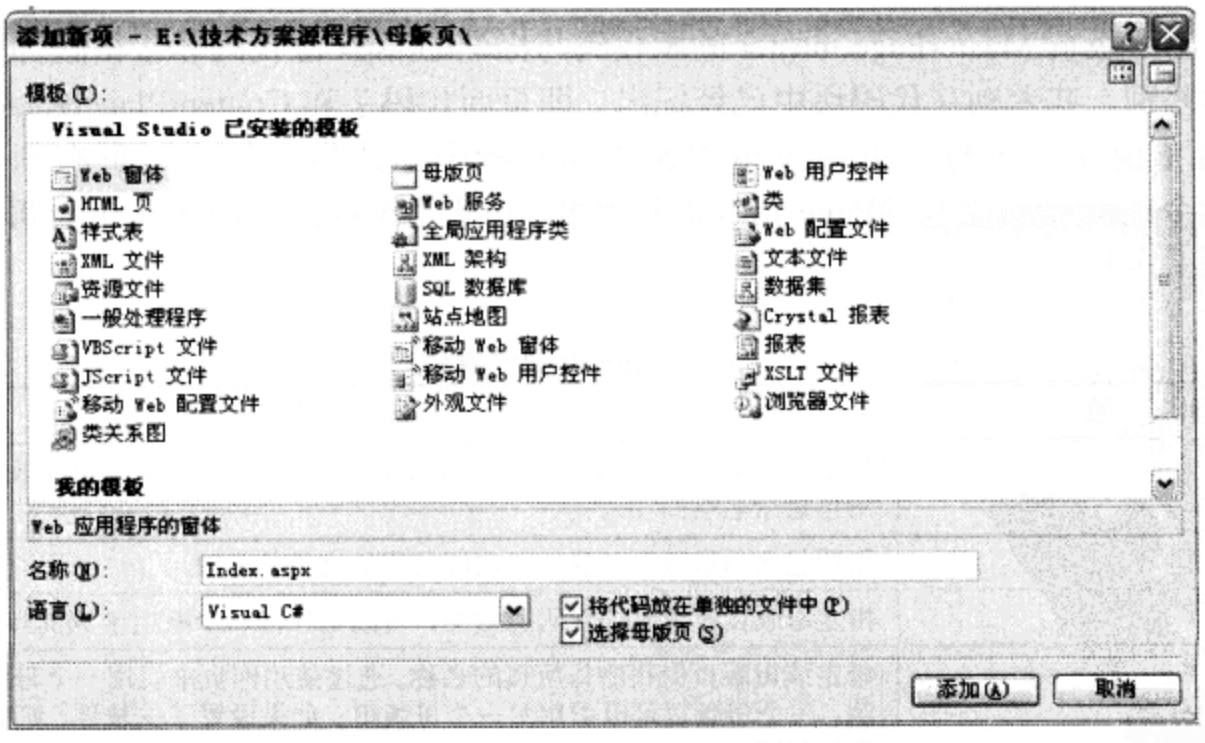


图 2.8 添加内容页

由于内容页与普通.aspx 页面的扩展名相同，因此，在选择新建文件类型中选择 Web 窗体图标，并设置文件名为 Index.aspx。在设置完之后，不要直接单击“添加”按钮，因为还需要对“选择母版页”复选框进行设置，要创建内容页必须选中此选项（内容页必须绑定到母版页才有实际意义）。完成以上操作后，可单击“添加”按钮，此时将会弹出如图 2.9 所示的窗口。

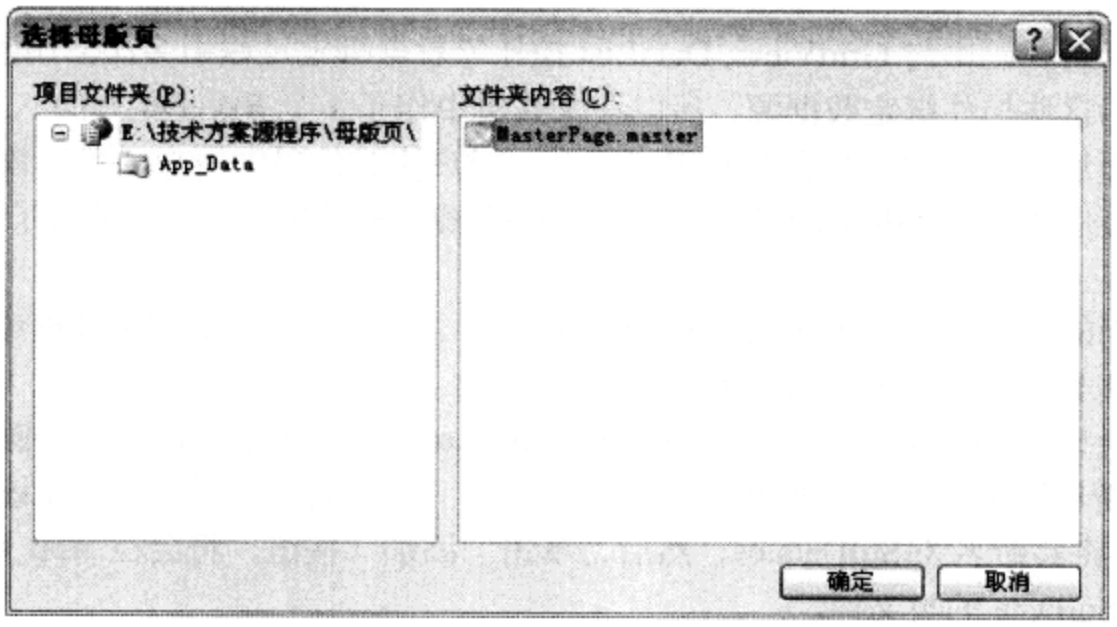


图 2.9 “选择母版页”窗口

在图 2.8 中，窗口左侧是项目文件夹，右侧是文件夹中的母版页列表。此时母版页列表中只有 1 个母版页 MasterPage.master 文件。选中该文件，单击“确定”按钮，即可完成一个绑定到母版页的内容页 Index.aspx。

在实际应用中，为了给整个网站设计一个统一的界面风格，一个母版页可能被多个内容页绑定。只有正确地处理好母版页与内容页之间的对应关系，才能发挥好它们强大的功能。

在前文分析中可知，内容页主要是包含页面中的非公共内容，其扩展名与标准的 ASP.NET 页一样都为.aspx。但是，二者在代码结构上是有很大差异的。在内容页中其代码周围没有<html>、<body>和<form>等关键 Web 元素，这些元素都被放置在母版页中了。

常见内容页的代码如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true" CodeFile="Default2.aspx.cs" Inherits="Default2" Title ="Untitled Page" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
...//省略所要添加的代码段
</asp:Content>
```

由上面的代码可看出，内容页主要分为页面代码头和 Content 控件两个部分。内容页的页面代码头与普通的.aspx 文件代码头很相似，只是增加了 MasterPageFile 和 Title 属性两个属性。属性 MasterPageFile 用于设置内容页所绑定的母版页的路径；属性 Title 用于设置页面 title 值。在创建内容页过程中，由于指定了所绑定的母版页，应用程序中将会自动设置 MasterPageFile 属性值。

2.4 公共类的封装与设计

在网站开发项目中，以类的形式来组织、封装一些常用的方法和事件，在编程过程中起到事半功倍的效果。由于在开发博客网站后台代码中，对于编写的几行或一段代码重复出现两次

以上，就要把这些相同代码编写成一个方法，通过在应用程序中调用这些方法以达到最优代码的编写。

以下是公共类 SqlData 中主要方法的编写思想。

- 在页面后台代码中多次应用了非查询的 SQL 语句，所以在公共类中编写了 1 个 ExeSQL 方法。
- 页面后台代码中与 GridView、DataList 控件等数据绑定过程中多次会用到 DataSet 数据集作为这些控件绑定数据源，所以在公共类中作了 1 个 ExceDS 方法。
- 在页面后台代码中，如在判断用户登录、用户注册代码中用户名等在数据库中是否存在就会应用到 DataReader 对象来读取一下数据，所以在公共类中编写了 1 个 ExceRead 方法。
- 在页面的数据显示中多次用到 GridView 控件，通过该控件来绑定数据库中数据，所以在公共类编写了 1 个 BindData 方法。

创建类文件时，用户可以直接在项目中的 App_Code 文件夹上单击鼠标右键，选择快捷菜单中的“添加新项”选项，将会弹出如图 2.10 所示的“添加新项”对话框，在该对话框中选择“类”选项，并将其命名为 SqlData.cs，然后，单击“添加”按钮，将会在 App_Code 文件夹下创建 1 个名为 SqlData 的类文件。

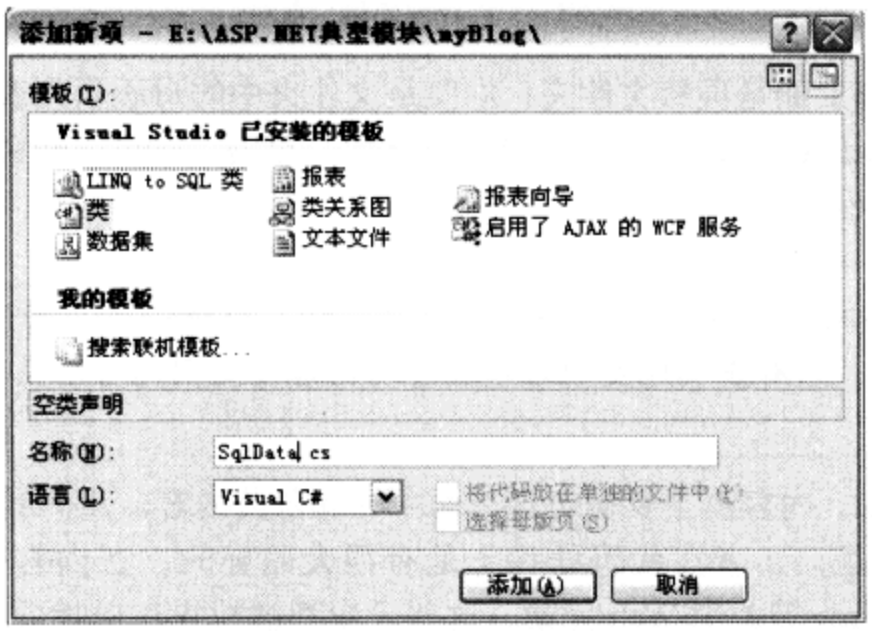


图 2.10　创建类文件

注意　在 ASP.NET 中，App_Code 文件夹专门用来存放一些应用于全局的代码（比如公共类），如果没有，可以在该项目上单击鼠标右键，选择“添加 ASP.NET 文件夹”→“App_Code”选项，添加 1 个 App_Code 文件夹。

2.4.1　Web.config 配置文件

为了使应用程序便于应用，同时为版本控制提供更好的支持，需要对网站进行全局配置（也就是配置 Web.config 文件）。在本例的博客网站中主要配置数据库连接信息，下面为配置连接数据库的代码。

例程 5　代码位置：光盘\mr\02\myBlog\ Web.config

```
<configuration>
<appSettings>
    <add key="conStr" value="Server=(local);DataBase=db_Blog;uid=sa;pwd="/>
</appSettings>
```

```
...
</configuration>
```

注 意

在编写与数据库连接的字符串时，应当使 uid 和 pwd 与本机器上的 SQL Server 服务器的登录名和密码相对应。

2.4.2 公共类中的全局变量

在 SqlData 类中声明了 3 个全局变量，以便在下面的程序代码中能够重复使用。避免重复编写相同的代码段，声明全局变量的位置和变量类型的代码如下。

例程 6 代码位置：光盘\mr\02\myBlog\ App_Code\ SqlData.cs

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
//引用数据库命名空间
using System.Data.SqlClient;
// <summary>
// SqlData 的摘要说明
// </summary>
public class SqlData
{
    //声明1个SqlConnection对象
    private SqlConnection sqlcon;
    //声明1个SqlCommand对象
    private SqlCommand sqlcom;
    //声明1个SqlDataAdapter对象
    private SqlDataAdapter sqldata;
    public SqlData()
    {
        //
        //在此处添加构造函数逻辑
        //
    }
    //以下为该类中的其他方法
    ...
}
```

注 意

#region 预处理器指令在使用 Visual Studio 代码编辑器的大纲显示功能时，指定可展开或折叠的代码块。这个功能非常实用，尤其在编辑复杂的类时，可以使得代码结构更加清晰，在查询代码时可以快速地找到需要的代码行，初学者在学习时一定要学会运用这种预处理器指令的方法，养成良好的编程习惯。

2.4.3 公共类中的构造函数

构造函数中包含连接数据库的字符串，当声明一个类的对象时，将连接数据库，代码如下。

例程 7 代码位置：光盘\mr\02\myBlog\ App_Code\ SqlData.cs

```
#region  构造函数
// <summary>
// 构造函数，初始化时连接数据库
```

```
// </summary>
public SqlData()
{
  //新建一个数据连接，从配置文件中获取连接数据库字符串
sqlcon = new SqlConnection(ConfigurationManager.AppSettings["conStr"]);
}
#endregion
```

2.4.4 执行数据的添加、删除等操作

ExecSQL 方法用来执行 SQL 语句，返回值为 Boolean 型，主要用于对数据库中数据执行添加、修改、删除的操作，相应功能执行成功后返回 True，否则返回 False。

例程 8 代码位置：光盘\mr\02\myBlog\ App_Code\ SqlData.cs

```
// <summary>
// 此方法用来执行SQL语句
// </summary>
// <param name="SqlCom">要执行的SQL语句</param>
// <returns></returns>
public bool ExceSQL(string strSqlCom)
{
    //新建一个数据连接
    SqlCommand sqlcom = new SqlCommand(strSqlCom, sqlcon);
     try
     {
         //判断数据库是否为连接状态
         if (sqlcon.State == System.Data.ConnectionState.Closed)
         {   sqlcon.Open();}
         //执行SQL语句
             sqlcom.ExecuteNonQuery();
         //SQL语句执行成功，返回值为True
          return true;
     }
     catch
     {
          //SQL语句执行失败，返回值为False
         return false;
     }
     finally
     {
          //关闭数据库连接
         sqlcon.Close();
     }
}
#endregion
```

2.4.5 执行数据库查询操作

本程序执行数据库查询操作通过两个方法来实现，一个是 ExecDS 方法，主要用来从数据库中检索数据，并将查询的结果使用 SqlDataAdapter 对象的 Fill 方法填充到 DataSet 数据集，然后返回该数据集的表的集合；另一个是 GetDataSet 方法，该方法与 ExecDS 方法类似，只是多了个 tablename 参数（可以重载这两个方法，读者有兴趣的话，可以独立完成）。其代码如下。

例程 9 代码位置：光盘\mr\02\myBlog\ App_Code\ SqlData.cs

```
#region  返回DataSet类型数据
 // <summary>
 // 此方法返回1个DataSet类型
 // </summary>
 // <param name="SqlCom">要执行的SQL语句</param>
 // <returns></returns>
 public DataSet ExceDS(string SqlCom)
```

```
{
    try
    {
        sqlcon.Open();    //打开链接
        SqlCommand sqlcom = new SqlCommand(SqlCom, sqlcon);
        //创建一个数据适配器
        SqlDataAdapter sqldata = new SqlDataAdapter();
        //打开数据库连接
        sqldata.SelectCommand = sqlcom;
        //创建一个数据集DataSet
        DataSet ds = new DataSet();
        //应用适配器的Fill方法填充DataSet数据集
        sqldata.Fill(ds);
        return ds;
    }
    finally
    {
        //关闭数据库连接
        sqlcon.Close();
    }
}
//定义1个DataSet类型的数据集，参数为2个
 public DataSet GetDataSet(string SqlCom,string tablename)
 {
    try
        {
            sqlcon.Open();    //打开链接
        SqlCommand sqlcom = new SqlCommand(SqlCom, sqlcon);
        SqlDataAdapter sqldata = new SqlDataAdapter();
        sqldata.SelectCommand = sqlcom;
        DataSet ds = new DataSet();
        sqldata.Fill(ds,tablename);
        return ds;
    }
    finally
    {
        sqlcon.Close();
    }
}
#endregion
```

DataSet 对象是支持 ADO.NET 的断开式或分布式数据方案的核心对象。DataSet 对象是数据的内存驻留表示形式，也就是说可以把 DataSet 想象成内存中的数据库。无论数据源是什么，它都会提供一致的关系编程模型。

2.4.6 读取数据库中数据

ExceRead 方法用来返回 SqlDataReader 类型的数据，在该方法中首先定义并初始化 1 个 SqlCommand 命令对象，然后利用该命令对象的 ExecuteReader 方法创建 1 个 SqlDataReader 对象，用来读取数据库数据。其代码如下。

例程 10 代码位置：光盘\mr\02\myBlog\ App_Code\ SqlData.cs

```
#region  返回SqlDataReader类型的数据
// <summary>
// 此方法返回1个SqlDataReader类型的参数
// </summary>
// <param name="SqlCom"></param>
// <returns></returns>
public SqlDataReader ExceRead(string SqlCom)
{
    //定义并初始化命令对象
```

```
        sqlcom = new SqlCommand(SqlCom, sqlcon);
        //创建一个数据阅读器
        SqlDataReader   read = sqlcom.ExecuteReader();
        //返回读取的数据值
        return read;
    }
    #endregion
```

注 意 若要创建 SqlDataReader，必须调用 SqlCommand 对象的 ExecuteReader 方法，而不要直接使用构造函数。在使用 SqlDataReader 时，关联的 SqlConnection 正忙于为 SqlDataReader 服务，对 SqlConnection 无法执行其他操作，只能将其关闭。除非调用 SqlDataReader 的 Close 方法，否则会一直处于此状态。例如，在调用 Close 之前，无法检索输出参数。SqlDataReader 的用户可能会看到在读取数据时另一进程或线程对结果集所做的更改。但是，确切的行为与执行时间有关。当 SqlDataReader 关闭后，只能调用 IsClosed 和 RecordsAffected 属性。尽管当 SqlDataReader 存在时可以访问 RecordsAffected 属性，但是在返回 RecordsAffected 的值之前须调用 Close，以保证返回精确的值。

2.4.7 绑定 GridView 控件中的数据

BindData 方法用来绑定用户控件，返回值为 Boolean 型，主要用来绑定页面中的 GridView 控件，执行成功后返回 True，否则返回 False。

例程 11 代码位置：光盘\mr\02\myBlog\ App_Code\ SqlData.cs

```
#region  绑定用户页面中的GridView控件
// <summary>
// 此方法实现数据绑定到GridView中
// </summary>
// <param name="dl">要绑定的控件</param>
// <param name="SqlCom">要执行的SQL语句</param>
// <returns></returns>
public bool BindData(GridView dl, string SqlCom)
{
    //调用公共类中的ExceDS方法，设置GridView控件数据源
      dl.DataSource = this.ExceDS(SqlCom);
      try
    {
        //调用DataBind方法从数据库中绑定数据
         dl.DataBind();
         return true;
    }
    catch
    {
        return false;
    }
    finally
    {
        //关闭数据库连接
         sqlcon.Close();
    }
}
#endregion
```


2.5 博客主界面设计

2.5.1 概述

在博客主界面的设计过程中，主要应用了母版页（即该页为母版页的一个内容页）。主界面

的内容主要包括博客文章类别列表、热点文章列表和推荐文章列表等，主界面中间主要应用了一个列表控件显示博友发表的所有博客文章，具体包括博客文章的标题、文章摘要、文章发表时间和文章评论及阅读次数。

博客首页运行效果如图 2.11 所示。

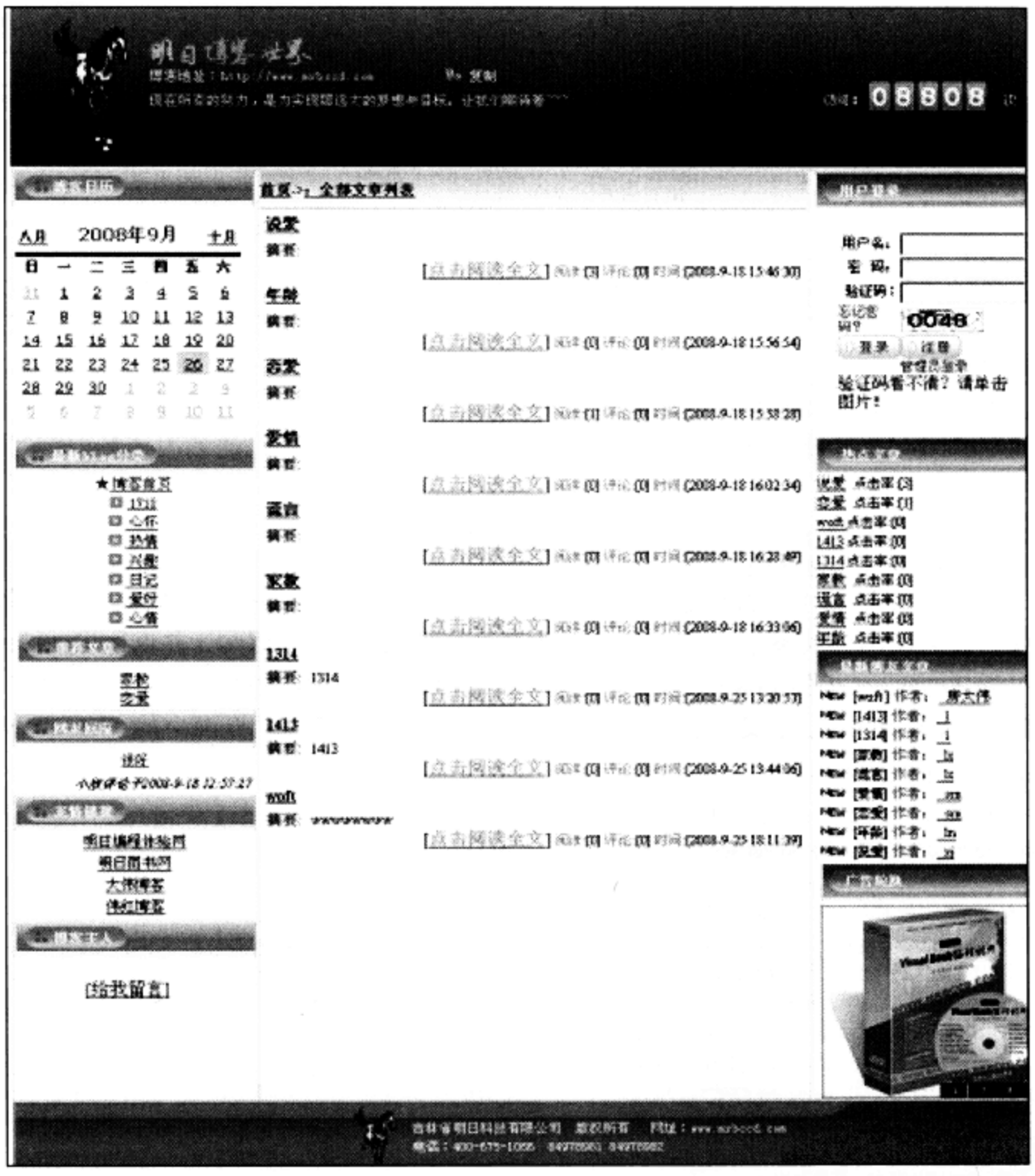


图 2.11　博客首页

2.5.2　实现过程

1. 设计步骤

（1）在应用程序中新建 1 个 Web 窗体，命名为 BlogIndex.aspx，将其作为 MasterPage.master 母版页的内容页，并设置为起始页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 GridView 控件、1 个 Calender 日历控件和 4 个 DataList 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 2.6 所示。

表 2.6　　BlogIndex.aspx 页面用到的主要控件

控件类型	控件名称	主要属性设置	用　途
DataList	ClassList	RepeatDirection 属性设置为 Vertical	用于显示博客类型列表
	TopList	RepeatDirection 属性设置为 Vertical	显示热点文章
	ClassList0	RepeatDirection 属性设置为 Vertical	显示博友发表的最新文章
	CmdList	RepeatDirection 属性设置为 Vertical	用于显示推荐的博客文章
GridView	NewsList	AllowPaging 属性设置为 True（允许分页） AutoGenerateColumns 属性设置为 False（取消自动生成列） PageSize 属性设置为 10（页面显示 10 条数据）	用于显示博客文章列表
Calendar	BlogCalendar	DayNameFormat 属性设置为 Shortest	日历控件，显示博客日期

2. 实现代码

后台页面 BlogIndex.aspx.cs 事件处理过程如下。

在编写 Page_Load 事件前，首先创建 1 个 SqlData 类型实例，以便调用该类中编写的方法，声明如下。

例程 12　代码位置：光盘\mr\02\myBlog\BlogIndex.aspx.cs

```
SqlData myobj = new SqlData();
```

在 Page_Load 事件中，首先定义 4 个 SQL 语句，如定义用来查询博客文章类型等信息的 SQL 语句，然后调用公共类中的 GetDataSet 方法执行定义的 SQL 语句，并返回 DataSet 类型的数据集作为 DataList 控件的数据源，接着调用.DataBind()方法从数据库中绑定相应的 DataList 控件中数据，最后调用自定义方法 NewsBlogList_Bind()绑定博客主界面中的 GridView 控件中数据，主要程序代码如下。

例程 13　代码位置：光盘\mr\02\myBlog\BlogIndex.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    // 在此处放置用户代码以初始化页面
    string ST_cmd_sql = "select top 10 * from ST_news where ST_n_iscmd=1 order by ST_n_date desc";
    string ST_top_sql = "select top 10 * from ST_news order by ST_n_hit desc";
    string personindex = "select top 10* from ST_news order by ST_n_id desc";
    string TitleClass = "select ST_c_id, ST_c_name from ST_class";
    DataSet ST_classds = myobj.GetDataSet(TitleClass,"ST_class");
    DataSet ST_cmdds = myobj.GetDataSet(ST_cmd_sql, "ST_news");
    DataSet ST_topds = myobj.GetDataSet(ST_top_sql, "ST_news");
    //绑定博客文章类型列表
    ClassList.DataSource = new DataView(ST_classds.Tables[0]);
    ClassList.DataBind();
    //绑定推荐文章列表
    CmdList.DataSource = new DataView(ST_cmdds.Tables[0]);
    CmdList.DataBind();
      //绑定热点文章列表
      TopList.DataSource = new DataView(ST_topds.Tables[0]);
      TopList.DataBind();
      //调用自定义NewsBlogList_Bind方法
    NewsBlogList_Bind();
      //绑定最新个人最新发表的博客文章
      DataSet mypersonds = myobj.GetDataSet(personindex, "ST_class");
      ClassList0.DataSource = new DataView(mypersonds.Tables[0]);
```

```
        ClassList0.DataBind();
        if (Request.Cookies["colors"]!=null)
        {
            string ST_test = Request.Cookies["colors"].Value;
            String[] ST_colorList = ST_test.Split(new char[] { ',' });
            ST_bgcolor = ST_colorList[0];
            ST_tcolor = ST_colorList[1];
        }
        else
        {
            ST_bgcolor = "#FFDE94";
            ST_tcolor = "#efe3ce";
        }
        Page.DataBind();
    }
```

自定义 1 个 NewsBlogList_Bind()方法，该方法中首先根据 Web 配置文件中定义的 URL 地址实现两个重定向功能，分别是用户单击博客主界面中的文章类别时重定向 URL 地址和用户单击最新博友文章时重定向 URL 地址。实现 URL 重写后，调用公共类中的 GetDataSet 方法返回 1 个 DataSet 数据集。主要代码如下。

例程 14 代码位置：光盘\mr\02\myBlog\BlogIndex.aspx.cs

```
public void NewsBlogList_Bind()
{
    //===以下代码主要用来在博客主界面中单击“文章类别”时，重定向链接的URL地址==//
    string ST_sql;
    //判断接收的在配置文件中定义的文章类别的c_id号是否为空
        if (Request.QueryString["c_id"]==null)
        {
            //定义1个SQL语句查询博客文章类别信息
            ST_sql = "select   * from ST_news order by ST_n_date desc";
        }
        else
        {
            //如果接收的文章类型编号符合配置文件中定义的文章编号，则执行以下代码
            if (IsSafe(Request.QueryString["c_id"],2)==true)
            {
                ST_sql = "select   * from ST_news where ST_c_id="+ Request.QueryString["c_id"] +" order by
ST_n_date desc";
                }
                else
                {
                    ST_sql="";
                    Response.Write("非法参数");
                    Response.End();
                }
            }
    //==以下代码主要用来在博客主界面中单击博友最新发表的文章时，重定向链接的URL地址==//
            string personindex;
        if (Request.QueryString["name"] == null)
        {
            personindex = "select * from ST_news";
        }
        else
        {
            if(Request.QueryString["name"] != null)
            {
                personindex = "select * from ST_news where ST_n_author= '" + Request.QueryString["name"] + "'";
                }
                else
```

```
                {
                    personindex = "";
                    Response.Write("<div align=center><li>该用户暂时还没有文章！</li><li><a href=javascript:
history.back()>点此返回</a>");
                    Response.End();
                }
            }
        //调用公共类中的GetDataSet方法执行查询文章类型的SQL语句，并返回DataSet类型的数据集
        DataSet personds = myobj.GetDataSet(personindex,"ST_news");
        //绑定文章类型列表中的数据
        NewsList.DataSource = new DataView(personds.Tables[0]);
        NewsList.DataBind();
        //调用公共类中的GetDataSet方法执行查询文章发表作者的SQL语句，并返回DataSet类型的数据集
        DataSet ST_ds = myobj.GetDataSet(ST_sql, "ST_news");
        //绑定博友最新个人文章列表中的数据
        NewsList.DataSource = new DataView(ST_ds.Tables[0]);
        NewsList.DataBind();
    }
```

2.6 博客个人文章管理

2.6.1 概述

在博客主界面的会员登录区内用户成功登录后，将进入个人博客空间管理页。在该页中单击“个人博客管理”节点下的“文章管理”选项，就会显示出博客文章管理的相关信息，包括文章编号、文章主题、人气指数等。这部分内容的实现是由 Blog 文件夹下的 ArticleManage.aspx 页面来实现的，运行结果如图 2.12 所示。

文章信息管理

查询条件 文章ID 关键字 查找

文章ID	文章主题	人气指数	评论	修改	删除	选择
155	房大伟	17	0	修改	删除	☐
156	明日科技	2	0	修改	删除	☐
158	fdw	7	0	修改	删除	☐
159	明日科技	6	0	修改	删除	☐
160	fdw	10	0	修改	删除	☐

☐全选 取消全选 批量删除

图 2.12 文章管理

此页面中主要应用到了 1 个 GridView 控件，用来向用户展示已有文章的信息，同时提供修改、删除（可进行批量删除操作）、添加和查询基本操作。

2.6.2 实现过程

1. 设计步骤

（1）在应用程序的 Module 文件夹下创建 1 个名为 Blog 文件夹，在该文件下创建 1 个 Web 窗体，将其命名为 ArticleManage.aspx。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 GridView 控件、1 个 TextBox、1 个 DropDowList 控件和两个 CheckBox 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 2.7 所示。

表 2.7　　ArticleManage.aspx 页面用到的主要控件

控件类型	控件名称	主要属性设置	用　途
GridView	GridView1	AllowPaging 属性设置为 True（允许分页） AutoGenerateColumns 属性设置为 False（取消自动生成的列） PageSize 属性设置为 6(每页显示 6 条数据)	绑定文章信息
TextBox	txtKey	无	输入查询的关键字
DropDownList	ddlSearch	无	选择文章 ID 号或者文章主题
CheckBox	CheckBox1	无	在 GridView 模板列中对数据进行单选
	CheckBox2	Text 属性设置为“全选”	全选按钮

2. 实现代码

在编写 Page_Load 事件之前，需要创建数据库操作类 SqlData 类的一个实例，以便调用其中编写的方法，声明代码如下。

例程 15　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ArticleManage.aspx.cs

```
SqlData da = new SqlData();
```

在页面的 Page_Load 事件中，首先判断页面是否是首次加载，然后调用两个自定义方法 ST_Article_Bind()和 ST_check_Login()，分别用来绑定 GridView 控件中的数据和判断用户是否登录，接着调用公共类中的 GetDataSet 方法执行指定的 SQL 查询语句，并返回 1 个 DataSet 类型的数据集，最后将获得的会员编号存储在定义的字符型变量 ID 中。详细代码如下。

例程 16　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ArticleManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
     if (!IsPostBack)
     {
         //调用自定义绑定方法绑定数据
         ST_Article_Bind();
         /*判断是否登录*/
         ST_check_Login();
     }
     //定义一个查询用户信息的SQL语句
     string sqlstr = "select * from tb_Blog where UserName='" + Session["UserName"] + "'";
     //调用公共类中的GetDataSet方法执行SQL语句，并返回DataSet类型的数据集
     DataSet mydataset = da.GetDataSet(sqlstr, "tb_Blog");
     //创建数据表的视图
     DataRowView rowview1 = mydataset.Tables["tb_Blog"].DefaultView[0];
     //获取注册博客的编号
     ID = rowview1["BlogID"].ToString();
 }
```

该页的 Page_Load 事件中调用了自定义方法 ST_Article_Bind()。ST_Article_Bind 方法为自定义的无返回值类型方法，该方法主要用来执行 SQL 查询语句，并将查询结果绑定到 GridView 控件上。ST_Article_Bind 方法关键代码如下。

例程 17　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ArticleManage.aspx.cs

```
public void ST_Article_Bind()
 {
     //根据获取的用户ID查询其所有文章
     string sqlstr = "select * from ST_news where BlogID='" + ID + "'";
     //调用公共类中的ExceDS方法，返回1个DataSet类型的数据集，并作为GridView控件的数据源
     GridView1.DataSource = da.ExceDS(sqlstr);
```

```
        //获取GridView控件中的主键字段
        GridView1.DataKeyNames = new string[] { "ST_n_id" };
        //从数据库中绑定数据到列表控件中
        GridView1.DataBind();
    }
```

自定义方法 ST_check_Login()主要用来判断用户是否登录，这里主要应用的是 Session 对象存储用户信息，代码如下。

例程 18　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ArticleManage.aspx.cs

```
public void ST_check_Login()
    {
        //判断存储在Session对象中
        if ((Session["UserName"] == null))
        {
            Response.Write("<script>alert('对不起！ 请先注册再登录！ ');location='../../BlogIndex.aspx'</script>");
            Response.End();
        }
    }
```

双击 ArticleManage.aspx 页面中的“查找”按钮，触发其 btnSearch_Click 事件，在该事件中首先定义一个字符变量 search 来存储下拉列表框中条件值，然后根据下拉列表框中选择的不同条件值来进行模糊查询。关键代码如下。

例程 19　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ArticleManage.aspx.cs

```
protected void btnSearch_Click(object sender, EventArgs e)
{
    //定义一个字符变量search来存储下拉列表框中条件值
    string search = this.ddlSearch.SelectedValue;
    //根据选择的条件不同显示不同结果
    switch (search)
    {
        //根据文章编号进行模糊查询
        case "文章ID":
            da.BindData(this.GridView1, "Select * From ST_news Where ST_n_id   Like   '%" + this.txtKey.Text +
"%' and BlogID='" + ID + "'");
            break;
        //根据文章主题进行模糊查询
        case "文章主题":
            da.BindData(this.GridView1, "Select * From ST_news Where ST_n_title   Like   '%" + this.txtKey.
Text + "%' and BlogID='" + ID + "'");
            break;
        default:
            //查询如果失败，弹出失败对话框
            Response.Write("<script lanuage=javascript>alert('查询失败！ ');location='javascript:history.go(-1)'</script>");
            break;
    }
}
```

该页面中用户可应用 GridView 控件自带的删除功能对个人文章信息进行删除操作，这里主要触发 GridView 控件的 RowDeleting 事件，在该事件中调用了公共类中的 ExceSQL 方法执行非查询的 SQL 语句。主要程序代码如下。

例程 20　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ArticleManage.aspx.cs

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    //调用公共类中的ExceSQL执行删除操作的SQL语句，并将返回值保存在定义的布尔类型的变量 delete中
    bool delete = da.ExceSQL("delete from ST_news where ST_n_id='" + GridView1.DataKeys[e.RowIndex].Value + "'");
    //判断是否删除成功
    if (delete)
    {
```

```
            //删除成功，弹出成功对话框
            Response.Write("<script language=javascript>alert('删除成功！');location='ArticleManage.aspx'</script>");
        }
        else
        {
            //删除失败，弹出失败提示框
            Response.Write("<script language=javascript>alert('删除失败！');location='ArticleManage.aspx'</script>");
        }
    }
```

GridView 控件的 RowDeleting 事件代码中，只可对 GridView 列表框中的信息进行单条数据的删除操作，为了实现批量删除操作，在界面中应用了 CheckBox 控件选中多条信息进行批量删除。读者可参见在本模块关键技术讲解中的“GridView 控件与 CheckBox 控件的应用”，由于篇幅有限，这里不再赘述。

当单击 GridView 控件中的“修改”按钮时，将会链接到 ViewContent.aspx 页面，在该页面初化的 Page_Load 事件中，通过创建 1 个 DataRowView（GridView 控件行的默认视图）的实例对象，将文章的主题和内容添加到指定文本框中。该页的 Page_Load 事件代码如下。

例程 21　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ViewContent.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
    //判断是否是注册用户登录，应用Session对象获取用户名
    if (Session["UserName"] == null)
     {
        Response.Redirect("~/BlogIndex.aspx");
     }
    if (!IsPostBack)//判断页面是否首次加载
     {
      try
{
   string str;
   string str2 = Page.Request["id"].ToString();
    //此处用来实现将数据绑定到前台
   SqlConnection mycon = new SqlConnection(ConfigurationManager.AppSettings["conStr"]);
   mycon.Open();
   SqlDataAdapter myada = new SqlDataAdapter("select * from ST_news
              where ST_n_id='" + str2 + "'", mycon);
   DataSet ds = new DataSet();
   myada.Fill(ds, "ST_news");
   //创建DataRowView对象的一个实例
   DataRowView rowview = ds.Tables["ST_news"].DefaultView[0];
   //将文章主题读取到labSubject文本框中
  this.labSubject.Text = rowview["ST_n_title"].ToString();
    //将文章内容读取到txtContent文本框中
   this.txtContent.Text = rowview["ST_n_content"].ToString();
  //关闭数据库连接
   mycon.Close();
        }
  catch (Exception ex)
   {
      Response.Write(ex.Message);
   }
}
```

当单击该页面（ViewContent.aspx 页）的“修改”按钮时，将触发其 Click 事件对发表的博客文章标题及内容进行修改，代码如下。

例程 22　代码位置：光盘\mr\02\myBlog\ Module \ Blog \ViewContent.aspx.cs

```
protected void btnUpdate_Click(object sender, EventArgs e)
 {
```

```
        string id = Request["id"].ToString();
        string Com = "update ST_news set ST_n_content='" + this.txtContent.Text + "' where ST_n_id='" + id + "'";
            //创建公共类SqlData的一个新的实例对象
            SqlData da = new SqlData();
            bool update = da.ExceSQL(Com);
            if (update)
            {
                Response.Write("<script language=javascript>alert('修改成功！');location='ArticleManage.aspx'</script>");
            }
            else
            {
                Response.Write("<script language=javascript>alert('修改失败！');</script>");
            }
    }
```

2.7 评论信息管理

2.7.1 概述

博客文章管理由网站根目录 Module 文件夹下的 Admin 文件夹中的 ST_admin_replay.aspx 页面实现，通过该页面显示出对博客文章进行评论的相关信息，包括了评论者和评论内容信息。该页面运行结果如图 2.13 所示。

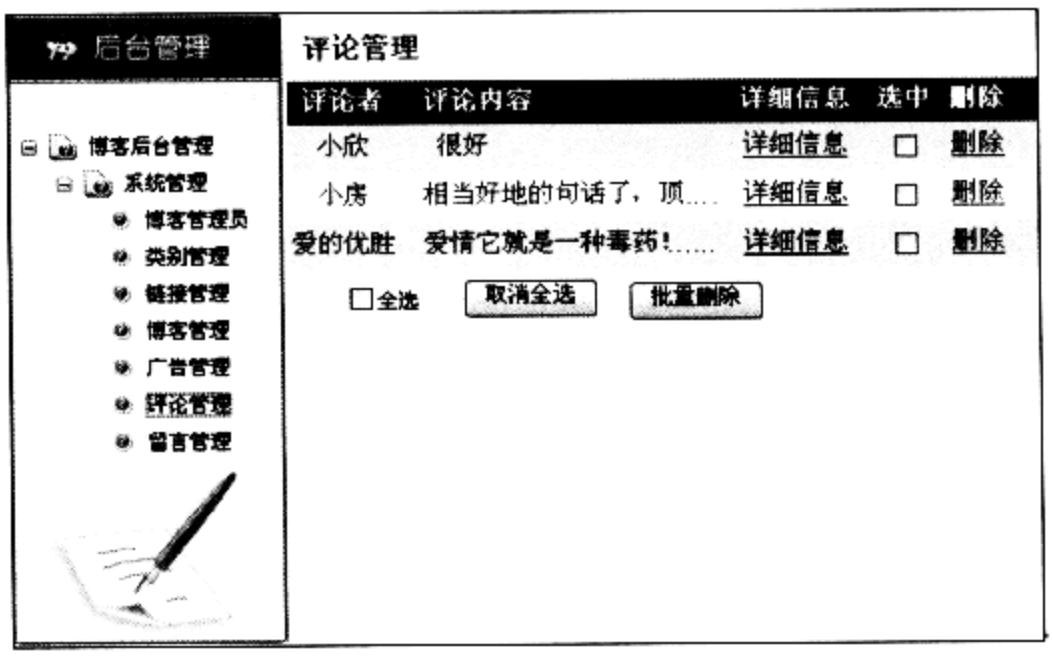


图 2.13 博客评论管理

此界面主要用了 1 个 GridView 控件。该控件用来向用户展示已有的评论信息，同时提供删除（可批量删除）评论信息的操作。

2.7.2 博客评论管理实现过程

1. 设计步骤

（1）在应用程序 Module 文件夹下创建 1 个名为 Admin 文件夹，在该文件下创建 1 个 Web 窗体，将其命名为 ST_admin_replay.aspx。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 GridView 控件、2 个 Button 控件和 2 个 CheckBox 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 2.8 所示。

表 2.8　　ST_admin_replay.aspx 页面用到的主要控件

控件类型	控件名称	主要属性设置	用途
GridView	ReplayList	AllowPaging 属性设置为 True（允许分页） AutoGenerateColumns 属性设置为 False（取消自动生成的列） PageSize 属性设置为 6（每页显示 6 条数据）	在页面中显示博客评论信息
Button	btnDelete	Text 属性设置为“批量删除”	批量删除
	btnCancel	Text 属性设置为“取消全选”	取消全选
CheckBox	CheckBox1	Font 属性设置为 9pt	在 GridView 模板列中对数据进行单选
	CheckBox2	Text 属性设置为“全选”	全选按钮

2. 实现代码

在页面 Page_Load 事件中，首先判断是否是管理员登录，然后建立数据库连接，最后通过调用 ST_Replay_Bind()方法从数据库中获取文章评论信息并绑定到 GridView 控件中，代码如下。

例程 23　代码位置：光盘\mr\02\myBlog\ Module \ Admin\ST_admin_replay.aspx.cs

```
protected void Page_Load(object sender, System.EventArgs e)
    {
      /*判断是否登录*/
      ST_check_Login();
      /*建立链接*/
      string ST_dns = ConfigurationSettings.AppSettings["conStr"];
      ST_myConn = new SqlConnection(ST_dns);
      //判断页面是否首次加载
      if(!Page.IsPostBack)
      {
          //调用自定义方法绑定数据
          ST_Replay_Bind();
       }
  }
```

该页的 Page_Load 事件中调用了自定义方法 ST_Replay_Bind()。ST_Replay_Bind 方法为自定义的无返回值类型方法，该方法主要用来执行 SQL 查询语句，并将查询结果绑定到 GridView 控件上。ST_Replay_Bind 方法关键代码如下。

例程 24　代码位置：光盘\mr\02\myBlog\ Module \ Admin\ST_admin_replay.aspx.cs

```
private void ST_Replay_Bind()
{
      string ST_sql = "select * from ST_replay";
      //调用公共类SqlData中的ExceDS方法，返回1个DataSet类型的数据集
      ReplayList.DataSource = sd.ExceDS(ST_sql);
      ReplayList.DataKeyNames = new string[] { "ST_r_id" };
      //调用自定义绑定方法，绑定数据
      ReplayList.DataBind();
}
```

本页在实现 GridVeiw 控件中信息的删除功能时，调用了该控件自带的删除功能。触发了 GridView 控件的 RowDeleting 事件，在该事件中首先创建公共类 SqlData 的一个实例，并调用其 ExceSQL 方法执行删除操作的 SQL 语句，代码如下。

例程 25　代码位置：光盘\mr\02\myBlog\ Module \ Admin\ST_admin_replay.aspx.cs

```
protected void ReplayList_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
```

```
    //创建公共类SqlData的一个新实例对象
    SqlData da = new SqlData();
    //调用公共类中的ExceSQL，执行删除的SQL语句
    bool delete = da.ExceSQL("delete from ST_replay where
            ST_r_id='" + ReplayList.DataKeys[e.RowIndex].Value.ToString() + "'");
     if (delete)
        {
         Response.Write("<script language=javascript>alert('删除成功！');location='ST_admin_replay.aspx'</script>");
        }
     else
     {
        Response.Write("<script language=javascript>alert('删除失败！');location='ST_admin_replay.aspxx'</script>");
      }
}
```

2.8 友情链接管理

2.8.1 概述

友情链接管理由网站根目录 Module 文件夹下的 Admin 文件夹中的 LinkManage.aspx 页面实现，通过该页面显示出博客中友情链接中的相关信息，包括了链接的网站名称和链接地址信息。该页面运行结果如图 2.14 所示。

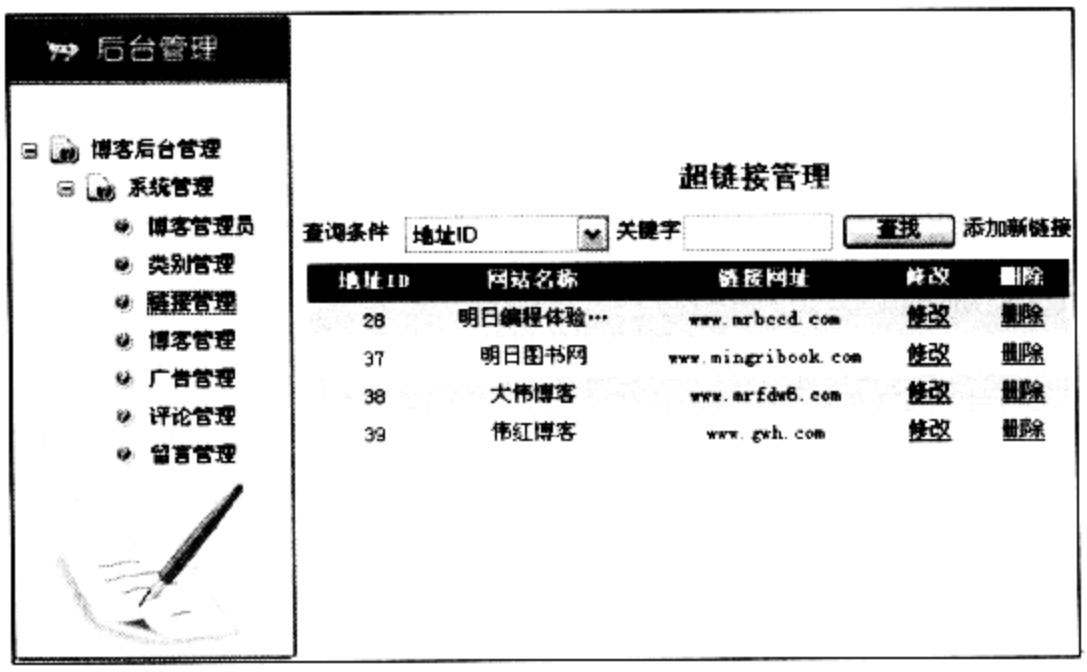


图 2.14 超级链接管理

在此界面主要用了 1 个 GridView 控件和 1 个“添加新链接”的超级链接。GridView 控件用来向用户展示已有的链接信息，同时提供修改和删除评论信息的操作；“查找”按钮则来提供已有链接的查询操作。

2.8.2 友情链接管理实现过程

1. 设计步骤

（1）在应用程序下创建 1 个名为 Blog 文件夹，在该文件下创建 1 个 Web 窗体，将其命名为 LinkManage.aspx。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1

个 GridView 控件、1 个 DropDownList 控件 1 个 Button 控件和 1 个 TextBox 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 2.9 所示。

表 2.9　　LinkManage.aspx 页面用到的控件

控件类型	控件名称	主要属性设置	用　途
GridView	GridView1	AllowPaging 属性设置为 True（允许分页） AutoGenerateColumns 属性设置为 False（取消自动生成的列） PageSize 属性设置为 6（每页显示 6 条数据）	在页面中显示博客评论信息
DropDownList	ddlSearch	Font 属性设置为 9pt	选择链接地址 ID、网站名称或链接网址
Button	btnSearch	Text 属性设置为“查询”	执行查询操作
TextBox	txtKey	Font 属性设置为 9pt	输入查询的关键字

2. 实现代码

在页面 Page_Load 事件中首先判断用户是否登录，然后实例化公共类 SqlData，并调用 ExceDS 返回 1 个 DataSet 类型的数据集作为页面中 GridView 控件数据源，最后调用 DataBind() 从数据库绑定数据。具体实现的代码如下。

例程 26　代码位置：光盘\mr\02\myBlog\ Module \ Admin\LinkManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
    /*判断是否登录*/
    ST_check_Login();
     SqlData da = new SqlData();
    //调用公共类中的ExceDS方法执行SQL语句
    this.GridView1.DataSource = da.ExceDS("select * from ST_link");
    GridView1.DataKeyNames = new string[] { "ST_1_id" };
     GridView1.DataBind();
  }
```

自定义 ST_check_Login 方法，用来判断用户是否登录，该方法中主要应用 Session 对象来存储用户名，代码如下。

例程 27　代码位置：光盘\mr\02\myBlog\ Module \ Admin\LinkManage.aspx.cs

```
public void ST_check_Login()
  {
    if ((Session["UserName"] == null))
     {
        Response.Write("<script>alert('对不起！ 您还没注册，
请先注册再登录！ ');location='../../BlogIndex.aspx'</script>");
        Response.End();
        }
  }
```

双击页面中的“查找”按钮，触发其 Click 事件执行对链接信息的查询操作，这里的查询主要应用了模糊查询，代码如下。

例程 28　代码位置：光盘\mr\02\myBlog\ Module \ Admin\LinkManage.aspx.cs

```
protected void btnSearch_Click(object sender, EventArgs e)
{
SqlData da = new SqlData();
string search = this.ddlSearch.SelectedValue;
//选择查询条件
switch (search)
{
```

```
        case "地址ID":
            da.BindData(this.GridView1, "Select * From ST_link Where ST_l_id
                Like   '%" + this.txtKey.Text + "%'");
            break;
        case "网站名称":
            da.BindData(this.GridView1, "Select * From ST_link Where ST_l_name
                Like   '%" + this.txtKey.Text + "%'");
        break;
            case "链接网址":
             da.BindData(this.GridView1, "Select * From ST_link Where ST_l_url
                Like   '%" + this.txtKey.Text + "%'");
        break;
            default:
            Response.Write("<script lanuage=javascript>alert('查询失败！');location='javascript:history.go(-1)'</script>");
        break;
        }
    }
```

2.9 博客留言信息管理

2.9.1 概述

文章的留言管理主要用于显示留言管理的相关信息。该模块是由网站根目录 Module 文件夹下的 Admin 文件夹中的 ST_admin_message.aspx 页面来实现的，运行结果如图 2.15 所示。

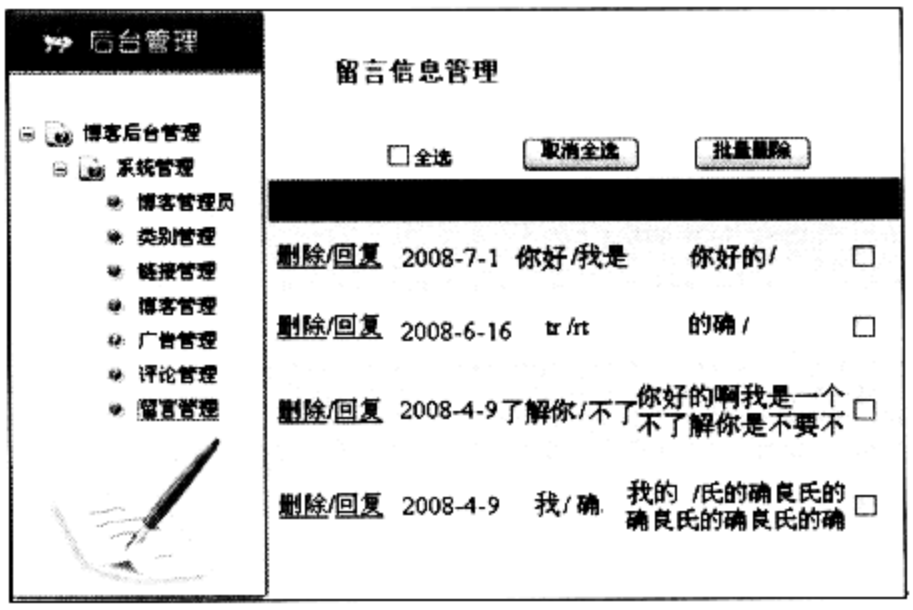


图 2.15 博客留言信息管理

在 ST_admin_message.aspx 页面中主要用到了 1 个 GridView 控件用来向用户展示已有的博客留言信息。同时还提供回复和删除留言信息操作。

2.9.2 实现过程

1. 设计步骤

(1) 找到网站根目录下 Module 文件夹下的 Admin 文件夹，在该文件下创建 1 个 Web 窗体，将其命名为 ST_admin_message.aspx。

(2) 在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 GridView 控件、2 个 Button 控件和 2 个 CheckBox 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 2.10 所示。

表 2.10　　ST_admin_message.aspx 页面用到的主要控件

控件类型	控件名称	主要属性设置	用途
GridView	MessageList	AllowPaging 属性设置为 True（允许分页） AutoGenerateColumns 属性设置为 False（取消自动生成的列） PageSize 属性设置为 6（每页显示 6 条数据）	在页面中显示博客留言信息
Button	btnDelete	Text 属性设置为“批量删除”	批量删除
	btnCancel	Text 属性设置为“取消全选”	取消全选

2. 实现代码

在博客留言回复界面的设计中主要应用了 GridView 控件的模板列 TemplateField。下面将详细介绍下应用过程。

鼠标单击 GridView 控件右上角的▶图标按钮，在弹出的快捷菜单中选择“编辑模板”选项，如图 2.16 所示，此时将会弹出 GridView 控件的模板列，如图 2.17 所示。

图 2.16　GridView 控件的快捷菜单

图 2.17　GridView 控件的模板列

为 GridView 控件设置项模板和头模板的方法如下。

（1）单击图 2.17 中的项模板（ItemTemplate），即可在 GridView 控件的项模板（ItemTemplate）中进行编辑。在此模板中添加 1 个 Table 表格并在此表格中 1 个 CheckBox 控件和一些标识性文字及字符，如图 2.18 所示。

图 2.18　GridView 控件的项模板（ItemTemplate）

（2）设计完项模板后，单击图 2.17 右上角的▶图标按钮，选择 HeaderTemplate 模板，单击该模板将会弹出如图 2.19 所示的模板。

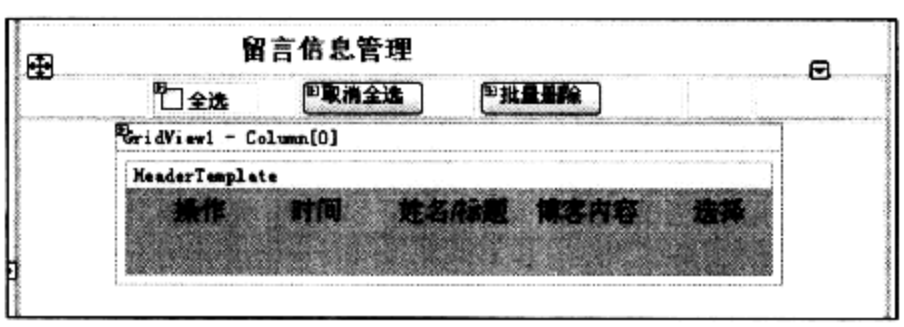


图 2.19　GridView 控件的 HeaderTemplate 模板

应用了 GridView 控件的模板列后，接下来就要对数据库中的数据进行绑定，绑定过程主要应用数据绑定表达式来完成。以下代码是在此页面中应用的数据绑定表达式，分别用来绑定留言回复的时间、留言者昵称、留言的标题及留言的内容。

例程 29 代码位置：光盘\mr\02\myBlog\ Module \ST_admin_message.aspx

```
<ItemTemplate>
    <TABLE class="table" cellSpacing="0" cellPadding="0" border="0" style="width: 100%">
    <TR>
    <td height="20" style="width: 18%">
    <DIV align="center"><A onclick="return confirm('确定删除')" href='ST_edit.aspx?action=delmsg&id=<%#
DataBinder.Eval(Container.DataItem, "ST_id") %>'>
    <span style="font-size: 11pt">删除</span></A><span style="font-size: 11pt">/</span><a href='ST_edit.aspx?
action=hfmsg&id=<%# DataBinder.Eval(Container.DataItem, "ST_id") %>'><span
    style="font-size: 11pt">回复</span></a></DIV>
    </td>
    <TD height="20" style="width: 20%">
        <DIV align="center"><%# DataBinder.Eval(Container.DataItem, "ST_mdate") %></DIV>
    </TD>
<TD width="19%" height="20">
    <DIV align="center"><%# DataBinder.Eval(Container.DataItem, "ST_nickname") %>
        <span style="font-size: 11pt">/<%# DataBinder.Eval(Container.DataItem, "ST_title") %></span></DIV>
    </TD>
    <TD height="20" style="width: 27%">
    <DIV align="center"><%# DataBinder.Eval(Container.DataItem, "ST_content") %>
<span style="font-size: 11pt">/<%# DataBinder.Eval(Container.DataItem, "ST_hf") %></span></DIV>
    </TD>
    <TD height="20" style="width: 57px">
      <asp:CheckBox ID="CheckBox1" runat="server" Font-Size="8pt" Height="14px" Width="1px" /></TD>
    </TR>
        </TABLE>
</ItemTemplate>
```

在上述代码中应用了 DataBinder.Eva 方法在运行时计算数据表达式，其语法格式如下：

```
<%# DataBinder.Eval(Container.DataItem,expression) %>
```

Container.DataItem 表达式引用对该表达式进行计算的对象。DataItem 属性表示当前容器上下文中的对象。在 ASP.NET 2.0 以上版本中上述语法格式可被简化成了以下格式：

```
<%# Eval(expression) %>
```

说 明

Eval 是建立在 DataBinder.Eval 方法之上的一个简单包装，它实现了数据读取的自动化。另外，在 ASP.NET 2.0 以上版本中还支持另一种数据绑定方法——Bind 方法。该方法与 Eval 相比，不仅实现了数据的读取自动化，还实现了数据的写入自动化。只要可以使用 Eval 的地方，就可以使用新的 Bind 方法，并且二者的语法类似。

以上是博客留言管理页的前台数据绑定代码，接下来编写后台代码。

在 Page_Load 事件中调用 ST_Link_Bind()方法来从数据库中获取文章信息并填充 GridView 控件。主要代码如下。

例程 30 代码位置：光盘\mr\02\myBlog\ Module \ST_admin_message.aspx.cs

```
protected void Page_Load(object sender, System.EventArgs e)
{
    //在此处放置用户代码以初始化页面
    /*判断是否登录*/
    ST_check_Login();
    /*建立链接*/
    string ST_dns = ConfigurationSettings.AppSettings["conStr"];
    ST_myConn = new SqlConnection(ST_dns);
if (!IsPostBack)
```

```
    {
        // 调用自定义方法ST_Link_Bind绑定数据库中数据
            ST_Link_Bind();
        }
}
```

ST_Link_Bind 方法为自定义的无返回值类型方法，该方法主要用来执行 SQL 查询语句，并将查询结果绑定到 GridView 控件上。

例程 31 代码位置：光盘\mr\02\myBlog\ Module \ST_admin_message.aspx.cs

```
private void ST_Link_Bind()
{
  string ST_sql = "select * from ST_message order by ST_mdate desc";
  //调用公共类中的ExceDS方法，执行查询语句
  GridView1.DataSource = sd.ExceDS(ST_sql);
  GridView1.DataKeyNames=new string[]{"ST_id"};
  this.GridView1.DataBind();
  }
```

2.10 程序发布与调试

网站开发完成后最终的目地是将其发布到 Internet 上，以提供用户浏览访问。实现的网站发布可以使用两种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍下使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

（1）首先打开要发布的网站，在网站的项目名称上，单击鼠标右键，在弹出的快捷方式菜单中选择“发布网站”选项，如图 2.20 所示。

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择...路径按钮。如图 1.21 所示。

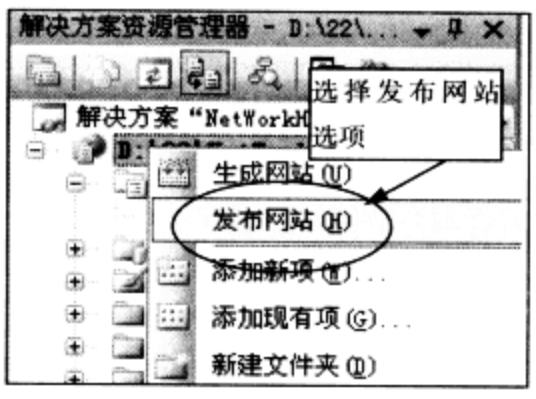


图 2.20 选择发布网站

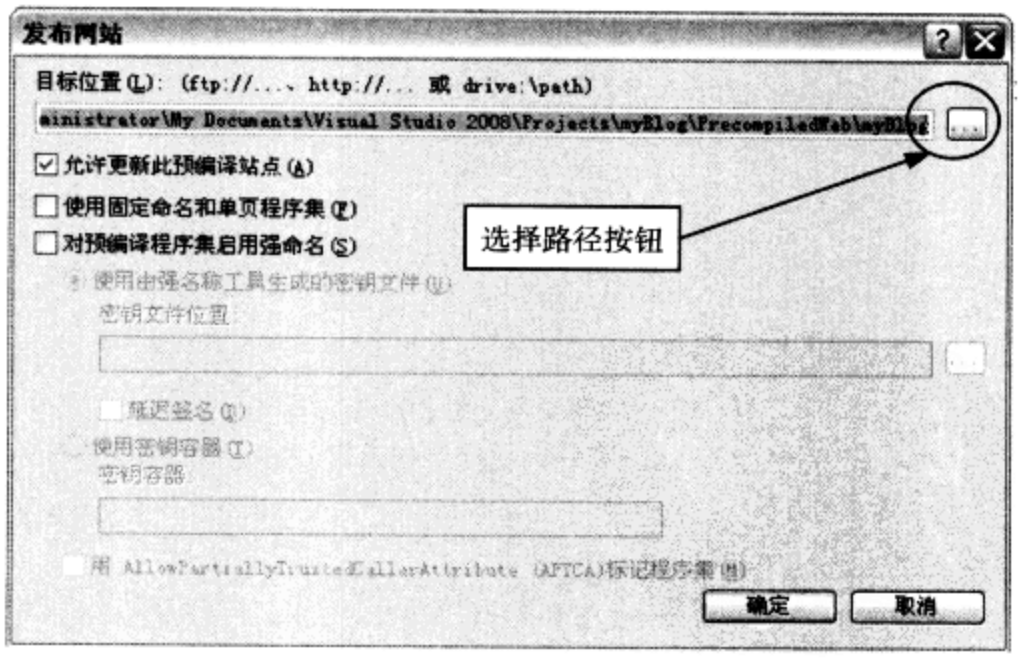


图 2.21 选择路径按钮

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选择中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码如图 1.22 所示。填写完毕后单击“打开”按钮将会返回“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

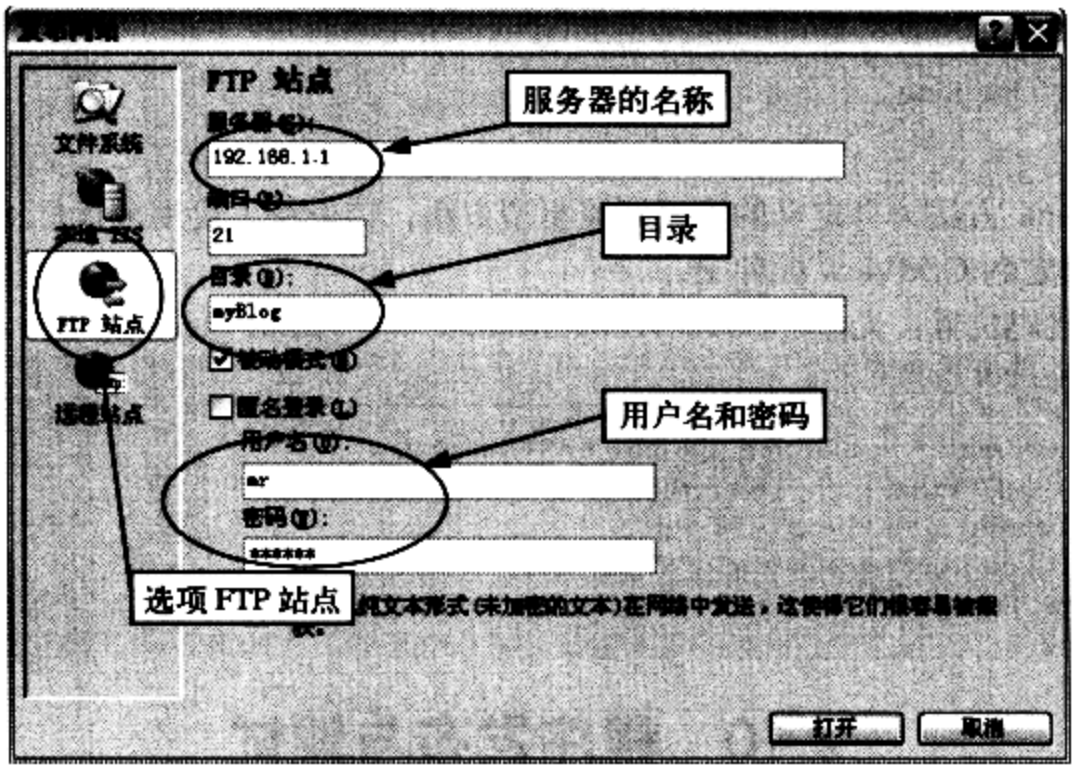


图 2.22 “FTP 站点”选项

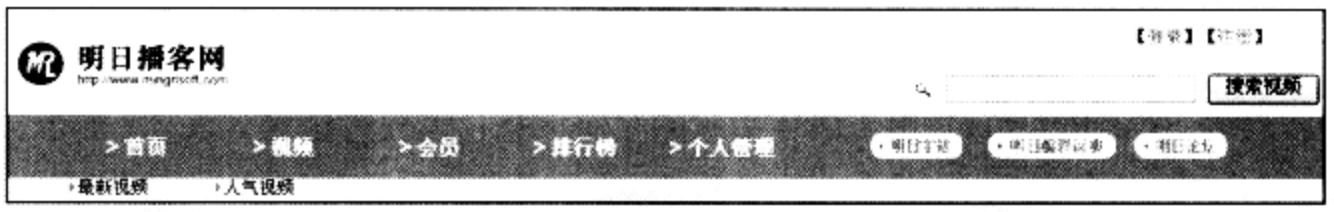
播客

第3章

实例位置：光盘\mr\03\

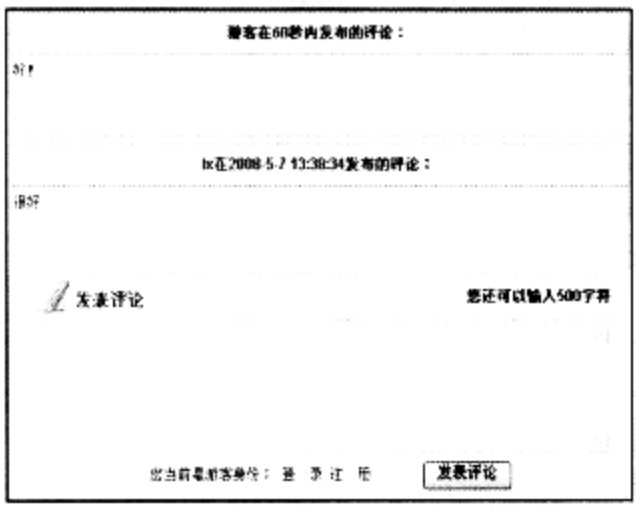
随着因特网的发展，通过文字信息来展现自我，已经不是主流的方式了。播客网站的出现，使得通过视频展现自我方式已经受到了广大网友的青睐。网友可以将自己拍摄或制作的视频上传到播客中，以提供给其他网友浏览，并可以通过评论功能为上传的视频进行评论，以增进网友之间的交流。通过学习本章，读者可以学习以下内容。

导航栏制作

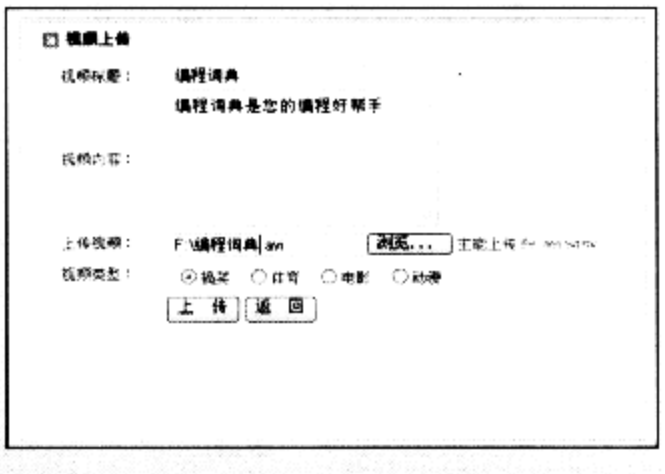


循环广告栏及最新视频显示

发表评论及显示



视频上传



视频欣赏及对其投票功能

3.1 概　　述

3.1.1 功能概述

播客模块主要是用户通过视频的形式来展现自我的平台。在播客模块中用户可以通过注册用户来注册播客网站的会员，成为播客网站的会员后用户就可以在网站中发布自己的视频。其他用户可以在播客网站中欣赏到会员用户所发布的视频。用户在欣赏完视频后还可以发表自己对视频的看法或意见。在该播客网站中还对会员用户进行了积分排名功能，例如某个会员用户发布的视频越多，所得积分也就越高。在网站的后台管理可以对视频进行管理。会员所发布的视频必须通过管理员审核后才可以在前台的页面中显示，如果某个会员发布了违法的视频，管理员还可以使用冻结账号的功能。

3.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建 1 个名为 playVideo 的数据库，在该数据库中创建 7 张表，这 7 张表分别用来记录用户注册信息、用户详细信息、视频详细信等。下面将详细介绍这 7 张表的数据结构。

- 用户注册表

用户注册表（userRegister）用于保存用户的注册信息，该表的结构如表 3.1 所示。

表 3.1　　表 userRegister 的结构

字　段	类　型	长　度	说　明
ID	int	4	自动编号
userName	varchar	30	用户登录名
userPass	varchar	30	用户密码
passQuestion	varchar	50	密码提示问题
passAnswer	varchar	50	密码提示答案
email	varchar	50	E-mail 地址
lock	bit	1	是否锁定
lockCause	varchar	50	锁定原因

- 用户详细信息表

用户详细信息表（userInfo）用于保存用户的详细信息。该表的结构如表 3.2 所示。

表 3.2　　表 userInfo 的结构

字　段	类　型	长　度	说　明
ID	Int	4	自动编号
userName	varchar	30	用户登录名
nickname	varchar	30	用户昵称
sex	char	10	用户性别
img	varchar	100	用户头像路径
city	varchar	50	所在城市
QQ	varchar	20	用户 QQ
speak	varchar	200	用户留言
sumMark	int	4	用户积分
registerDate	datetime	8	注册日期

● 视频详细信息表

视频详细信息表（videoInfo）主要用于保存用户上传视频的详细信息。该表的结构如表 3.3 所示。

表 3.3　　表 videoInfo 的结构

字　段	类　型	长　度	说　明
ID	int	4	自动编号
userName	varchar	30	用户登录名
videoTitle	varchar	30	视频标题
videoContent	varchar	500	视频内容
videoDate	varchar	8	发布视频日期
videoPath	varchar	50	视频路径
videoPicture	varchar	50	视频图片路径
videoType	char	10	视频类型
playSum	int	4	视频点击率
flower	int	4	视频顶人数
tile	int	4	视频踩人数
monthSum	int	4	视频本月点击率
Auditing	bit	1	审核状态

● 视频评论表

视频评论表（videoIdea）主要用于保存视频的评论信息。该表的结构如表 3.4 所示。

表 3.4　　表 videoIdea 的结构

字　段	类　型	长　度	说　明
ID	int	4	自动编号
userName	varchar	50	评论人
content	text	16	评论内容
videoId	int	4	评论视频的编号
issuanceDate	datetime	8	评论时间

● 视频排行表

视频排行表（videoTaxis）主要用于保存视频每月排行信息。该表的结构如表 3.5 所示。

表 3.5　　表 videoTaxis 的结构

字　段	类　型	长　度	说　明
videoId	int	4	视频编号
videoType	char	10	视频类型
videoTitle	varchar	50	视频名称
playSum	int	4	视频点击率
taxisMonth	char	10	视频排行月份

● 视频投票信息表

视频投票信息表（videoPoll）主要用于保存已投票的用户 IP 和视频 ID。该表的结构如表 3.6 所示。

表 3.6　　表 videoPoll 的结构

字　段	类　型	长　度	说　明
ID	int	4	自动编号
IP	varchar	30	投票者的 IP
videoId	Int	4	投票的视频 ID

- 公告信息表

公告信息表（bulletin）主要用于保存公告信息。该表的结构如表 3.7 所示。

表 3.7　　表 bulletin 的结构

字　段	类　型	长　度	说　明
ID	int	4	自动编号
Title	varchar	50	公告标题
content	varchar	100	公告内容
issuanceDate	datetime	8	发布日期

3.2 关 键 技 术

3.2.1 利用 IP 防止重复投票

网友在欣赏完视频后，还可以根据自己的喜好对视频进行投票。如图 3.1 所示。

图 3.1　防止重复投票

为了防止虚假投票的结果，可以采用一个 IP 地址只投一次的方式。在一般的投票模块中都会使用 Cookie 来防止重复投票，但是由于 Cookie 存储在客户端也可能会造成虚假投票。在本程序中是使用数据库来保存投票者的 IP 来防止重复投票。使用 IP 来防止重复投票的主要思路是获取客户端的 IP，判断此 IP 是否已经投过票，如果已投过将给出提示，若未投过则将 IP 保

存到数据库中。获取客户端的 IP 地址可以使用 Request 对象中 UserHostAddress 属性获得。该属性说明如下。

UserHostAddress 属性用于获取客户端的 IP 地址。语法如下：

```
public string UserHostAddress { get; }
```

说 明 使用数据库会使数据库增大，从而影响速度，所以用户在选择防止投票的保存方式时可以根据实际情况来选择。

3.2.2 控制并显示文本框的字符数量

在用户发表评论时使用了限制文本框中输入字符个数的功能。在用户未输入评论前使用 Label 控件显示一个 500 数字的提示。该提示表示用户最多只能输入 500 个字符。当用户输入一个字符 Label 将会显示 500 减去输入的字符数量而获得的结果。当用户输入的字符超过 500 个字符后，光标就会跳转到第 500 个字符后的位置。如图 3.2 所示。

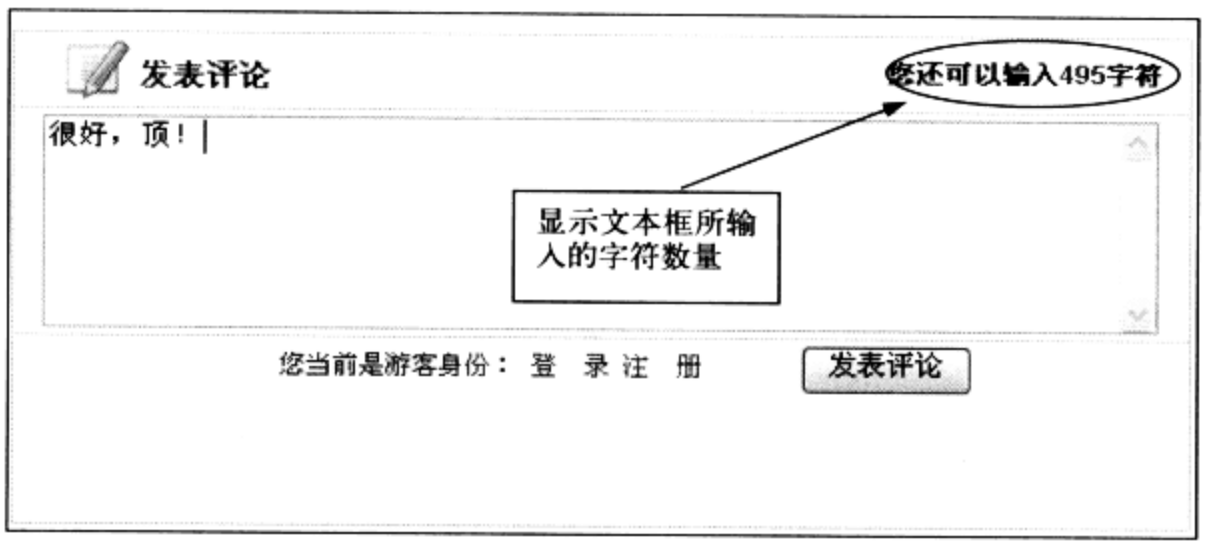


图 3.2 控制并显示文本框输入字符数量

该技术主要是通过文本框的 onKeyUp 事件来实现的。在该事件中调用 JavaScript 自定义函数 change。函数代码如下。

例程 1 代码位置：光盘\mr\03\PlayVideo\play.aspx

```
<script language="javascript">
        function change()
        {
        //获取评论文本框中的值
            var str=document.getElementById('txtContent').value;
            //获取当前还可以输入的字符数量
            var sum=500-str.length;
            //判断是否还可以输入字符
            if(sum<=0)
            {
            //设置Label控件显示文本为红色
              document.getElementById('labCount').style.color="Red";
              //截取文本框中的字符串，从0位置开始截取，截取到500位
             document.getElementById('txtContent').value=document.getElementById('txtContent').value.substring(0,500);
             //显示可以输入的字符数量
            document.getElementById('labCount').innerHTML=sum;
            }else
            {
            //显示可以输入的字符数量
                document.getElementById('labCount').innerHTML=sum;
                //设置Label控件的文本颜色
```

```
                document.getElementById('labCount').style.color="#006FC3";
                }
            }
        </script>
```

3.2.3 使用计时方式显示评论的发表时间

在以前显示发布评论时间都是以日期的形式显示的。例如，某年某月某日的几点几分发布的评论。在本程序中发布评论的时间如果是在 24 小时内，会以在几小时、几分、几秒前发布的评论的形式显示的，如图 3.3 所示。

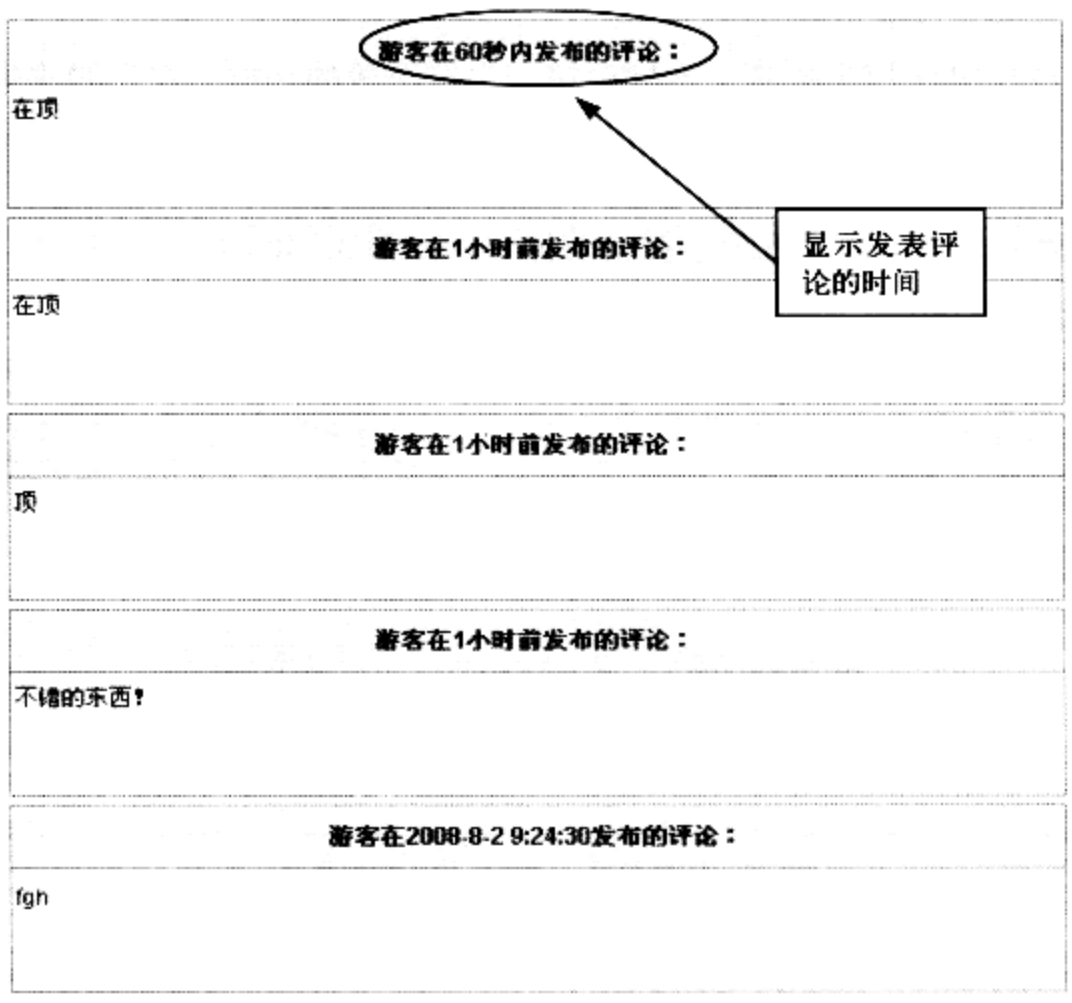


图 3.3 显示发表评论的时间

该功能在 getIsDate 方法中实现。在该方法中主要是通过将两个日期相减后，判断差值是在秒内或分钟内还是在小时内来设置显示的文本。这里使用了 TimeSpan 对象来保存两个日期的间隔，并使用 TotalSeconds 属性将两个日期的差值转换为秒数。

TotalSeconds 属性用于表示以整秒数和秒的小数部分表示的当前 TimeSpan 对象中的两个时间间隔值，语法如下：

```
public double TotalSeconds { get; }
```

返回值：双精度浮点数字，表示总秒数。

获取到总秒数后，将秒数除以 60 来判断是否是在一分钟前发表的评论，如果大于 60 分钟并且小于 1440 分钟说明是在 24 小时内发表的评论，并转换到小时的时间。getIsDate 方法中实现代码如下。

例程 2 代码位置：光盘\mr\03\PlayVideo\play.aspx.cs

```
public string getIsDate(string date)
{
    //转换时间
    DateTime isDate = Convert.ToDateTime(date);
    //获取当前时间
    DateTime nowDate = DateTime.Now;
```

```
        //获取两个时间的差
        TimeSpan ts = nowDate - isDate;
        //将时间差转换为分
        int second = Convert.ToInt32(ts.TotalSeconds) / 60;
        if (second == 0)
        {
            return "60秒内";
        }
        else
            if (second < 60)
            {
                return second.ToString() + "分钟前";
            }
            else if (second > 60 && second < 1440)
            {
                return Convert.ToString(second / 60) + "小时前";
            }
            else
                return date;
    }
```

3.2.4 视频格式转换

目前受大家欢迎的播客网中都是使用 Flash 制作的 flv 播放器。使用 flv 播放器可以使用户上传的文件格式统一、页面的加载速度较快、减少缓冲的等待时间。在用户上传视频时首先需要判读用户上传文件的类型。如果用户上传的文件不是.flv 格式，就需要考虑视频转换。视频转换可以使用 ffmpeg 工具来转换视频，该工具可以转换大多数的视频格式，但是不支持.rm、.rmvb 格式。转换这两种格式可以使用 mencoder 工具来实现。在本程序中不提供 flv 播放器、ffmpeg 工具，读者可以到网上下载。ffmpeg 工具转换视频格式的参数如下：

```
" -i " + Name + " -ab 128 -ar 22050 -qscale 6 -r 29.97 -s " + widthOfFile + "x" + heightOfFile + " " + flv_file;
```

参数说明如下。

- Name：需要转换的视频路径。
- -ab：设置音频码率。
- -ar：设置音频采样率。
- -qscale：设置使用固定的视频量化标度。
- -r：设置帧频，默认值为 25。
- -s：设置帧大小，格式为 WXH 默认值为 160 × 128。
- WidthOfFile、heightOfFile：用来设置帧大小的两个参数。
- flv_file：视频转换后需要保存的路径。

3.2.5 防止 session 丢失

在使用 session 保存数据时，有时可能会因为改写 bin 目录下的某个文件或其他原因而引起 session 中的数据丢失，由于在 web.config 中 session 的配置中 mode 属性是用来设置 sesion 保存状态的。而默认的参数为 Inproc 该参数使 session 的保存状态依赖于 ASP.NET 进程。这个进程不稳定，在某些事件发生时，可能会引起进程的重启，而该进程重启后会导致 session 的丢失。

为了防止 session 的丢失，可以把 mode 属性的参数设置为 StateServer，StateServer 是本机中的一个服务，而该服务除非是在电脑重启或 StateService 崩溃的时候 Session 才会丢失。设置该参数的方法如下。

（1）在 web.config 文件中设置 sessionState 中的 mode 参数为 StateServer。

（2）打开控制面板中管理工具选项。

（3）在管理工具中打开服务选项，在服务选项中找到 ASP.NET 状态服务（asp.NET State Service）并启动该服务

3.3 公共类的封装与设计

设计公共类，可以提高开发效率以及方便以后对程序的维护。对于一个好的程序来说，公共类是不可缺少的一部分。在本程序中编写了两个公共类，这两个公共类分别为数据库操作类 operateData，公共方法类 operateMethod。数据库操作类主要用于编写对数据库常用的一些操作。公共方法类用于编写在程序中比较常用的方法或易于日后修改的方法。下面将详细介绍公共类中的方法。

3.3.1 实现添加、删除和更新操作

execSql 方法用来执行数据表的添加、删除和更新操作，该方法返回一个布尔值用来表示 SQL 语句是否执行成功。该方法编写在数据库操作类 operateData 中。调用该方法时需要传入一个 string 类型的参数，该参数为需要执行的 SQL 语句。实现代码如下。

例程 3 代码位置：光盘\mr\03\PlayVideo\App_Code\operateData.cs

```
public static bool execSql(string sql)
    {
        //创建数据库连接对象
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sql, con);
        //获取ExecuteNonQuery方法返回的值
        int isEx=com.ExecuteNonQuery();
        //关闭数据库连接
        con.Close();
        if (isEx>0)
        {
            return true;
        }else{
            return false;
        }
    }
```

3.3.2 实现返回指定列操作

自定义 getTier 方法用于返回指定的列值，该方法编写在 dataOperate 类中。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法编写在数据库操作类 operateData 中。该方法将返回一个字符串变量，该字符串变量表示查询出的列值。实现代码如下。

例程 4 代码位置：光盘\mr\03\PlayVideo\App_Code\operateData.cs

```
public   static   string getTier(string sql)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sql, con);
    //创建SqlDataReader对象
    SqlDataReader sdr = com.ExecuteReader();
    //读取一条记录
```

```
        sdr.Read();
        //获取首列的值
        string tier=sdr[0].ToString();
        return tier;
    }
```

3.3.3 实现返回表中所有数据

自定义 getRows 方法用来返回表中的所有数据，该方法返回 1 个 DataTable 对象。该方法编写在数据库操作类 operateData 中。调用该方法时需要传入 1 个 string 类型的参数，该参数为需要执行的 SQL 语句。实现代码如下。

例程 5　代码位置：光盘\mr\03\PlayVideo\App_Code\operateData.cs

```
public static   DataTable getRows(string sql)
    {
        //创建DataSet对象
        DataSet ds;
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlDataAdapter对象
        SqlDataAdapter sda = new SqlDataAdapter(sql, con);
        //实例DataSet对象
        ds = new DataSet();
        //填充DataSet对象
        sda.Fill(ds);
        //关闭数据库连接
        con.Close();
        return ds.Tables[0];
    }
```

3.3.4 实现用户登录操作

自定义 login 方法用来实现用户登录查询，主要通过使用 SqlCommand.Parameters 属性的参数传值将非法字符过滤掉，来防止 SQL 注入式攻击。该方法编写在数据库操作类 operateData 中。该方法返回一个布尔值，该值为 True 时表示登录成功，为 False 时为登录失败。调用该方法需要传入 3 个 string 类型的产生，第 1 个 sql 参数表示为需要执行的 SQL 语句。第 2 个 name 参数表示登录名，第 3 个 pass 参数表示登录密码。实现代码如下。

例程 6　代码位置：光盘\mr\03\PlayVideo\App_Code\operateData.cs

```
public static bool login(string sql, string name, string pass)
{
    //创建数据库连接对象
    SqlConnection con = createCon();
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sql, con);
    //设置参数的类型
    com.Parameters.Add(new SqlParameter("@name", SqlDbType.VarChar, 20));
    //设置参数值
    com.Parameters["@name"].Value = name;
    com.Parameters.Add(new SqlParameter("@pass", SqlDbType.VarChar, 20));
    com.Parameters["@pass"].Value = pass;
    //获取ExecuteScalar对象返回的值
    int isEx=Convert.ToInt32(com.ExecuteScalar());
    if (isEx > 0)
    {
        return true;
```

```
        }
        else
        {
            return false;
        }
    }
```

3.3.5 实现转换视频格式

自定义 changeVideoType 方法将上传的视频转换为.flv 格式，并保存到相应的文件夹下。该方法编写在公共方法类 operateMethod 中。该方法返回一个布尔值，该值为 True 表示转变成功，为 False 表示转换失败。调用该方法需要传入 3 个参数。第 1 个参数为需要转换的视频路径，第 2 个参数为视频转换后保存的路径，第 3 个参数为视频抓图后保存的路径。实现代码如下。

例程 7　代码位置：光盘\mr\03\PlayVideo\App_Code\operateMethod.cs

```
public static bool changeVideoType(string fileName, string playFile, string imgFile)
    {
        //获取视频转换工具的路径
        string ffmpeg= System.Web.HttpContext.Current.Server.MapPath("../") + ffmpegtool;
        //获取需要转换的视频路径
        string Name = System.Web.HttpContext.Current.Server.MapPath("../") + upFile + "/" + fileName;
        if ((!System.IO.File.Exists(ffmpeg)) || (!System.IO.File.Exists(Name)))
        {
            return false;
        }
        //获取视频转换后需要保存的路径
        string flv_file = playFile;
        //创建Process对象
        Process pss = new Process();
        //不显示窗口
        pss.StartInfo.CreateNoWindow = false;
        //设置启动程序的路径
        pss.StartInfo.FileName = ffmpeg;
        //设置执行的参数
        pss.StartInfo.Arguments = " -i " + Name + " -ab 128 -ar 22050 -qscale 6 -r 29.97 -s " + widthOfFile + "x" +
heightOfFile + " " + flv_file;

        try
        {
            //启动转换工具
            pss.Start();
            while (!pss.HasExited)
            {
                continue;
            }
            //截取视频的图片
            catchImg(Name, imgFile);
            System.Threading.Thread.Sleep(4000);
            return true;
        }
        catch
        {
            return false;
        }
    }
```

3.3.6 实现截取视频图片

自定义 catchImg 方法用来实现截取视频图片，并保存到相应的文件夹下。该方法编写在公共方法类 operateMethod 中。调用该方法需要传入 2 个参数，第 1 个参数表示需要截取图片的

视频路径，第 2 个参数表示截取图片后保存的路径。实现代码如下。

例程 8　代码位置：光盘\mr\03\PlayVideo\App_Code\operateMethod.cs

```
public static void catchImg(string fileName,string imgFile)
    {
        //获取截图工具路径
        string ffmpeg = System.Web.HttpContext.Current.Server.MapPath("../") + ffmpegtool;
        //获取截图后保存的路径
        string flv_img = imgFile;
        //获取截取图片的大小
        string FlvImgSize = sizeOfImg;
        Process pss = new Process();
        //设置启动程序的路径
        pss.StartInfo.FileName = ffmpeg;
        pss.StartInfo.Arguments = "    -i    " + fileName + "  -y  -f  image2    -ss 2 -vframes 1  -s    " + FlvImgSize + "    " +
flv_img;
        //启动进程
        pss.Start();
    }
```

3.3.7　实现过滤 HTML 字符

自定义 filtrateHtml 方法用来实现过滤 HTML 字符。该方法编写在公共方法类 operateMethod 中。调用该方法需要传入一个字符串变量，该变量表示需要过滤的字符串。该方法返回一个字符串变量，该变量表示过滤后的字符串。实现代码如下。

例程 9　代码位置：光盘\mr\03\PlayVideo\App_Code\operateMethod.cs

```
public static string filtrateHtml(string str)
    {
        str = str.Trim();
        str = str.Replace("'", "&quot;");
        str = str.Replace("<", "&lt;");
        str = str.Replace(">", "&gt;");
        str = str.Replace(" ", " ");
        str = str.Replace("\n", "<br>");
        return str;
    }
```

3.3.8　实现恢复 HTML 字符

自定义 getBrowser 方法用来恢复 HTML 字符。该方法编写在公共方法类 operateMethod 中。调用该方法需要传入一个字符串变量，该变量表示需要恢复的字符串。该方法返回一个字符串变量，该变恢复后的字符串。实现代码如下。

例程 10　代码位置：光盘\mr\03\PlayVideo\App_Code\operateMethod.cs

```
public static string resumeHtml(string str)
    {
        str = str.Trim();
        str = str.Replace("&quot;", "'");
        str = str.Replace("&lt;", "<");
        str = str.Replace("&gt;", ">");
        str = str.Replace(" ", " ");
        str = str.Replace("<br>", "\n");
        return str;
    }
```

3.4　播客模块实现过程

3.4.1　播客首页设计

在播客的首页中用户可以查看到最新发布的视频信息，如最新搞笑类型视频、最新体育类

型视频等。播客首页如图 3.4 所示。

图 3.4　播客模块首页

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 index.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 3.8 所示。

表 3.8　　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
数据/DataList 控件	dlNewVideo	设置属性 RepeatDirection 为 Horizontal	显示最新视频信息
	dlHumour	设置属性 RepeatDirection 为 Horizontal	显示搞笑视频信息
	dlSport	设置属性 RepeatDirection 为 Horizontal	显示体育视频信息
	dlFilm	设置属性 RepeatDirectionv 为 Horizontal	显示电影视频信息
	dlCartoon	设置属性 RepeatDirection 为 Horizontal	显示卡通视频信息
标准/TextBox 控件	txtUserName	均为默认值	用来输入用户登录名
	txtUserPass	设置属性 TextMode 为 Password	用来输入用户登录密码
标准/imageButton 控件	imgBtnEntry	均为默认值	实现用户登录操作
	imgbtnGetPass	设置属性 PostBackUrl 为~/GetPass.aspx	跳转到找回密码页面
	imgbtnRegister	设置属性 PostBackUrl 为~/Register.aspx	跳转到用户注册页面

2. 后台代码编写

在该页面中创建 3 个全局变量，用来保存公告标题、内容和公告发布时间。在该页面的加载事件中通过自定义方法来显示相应的视频信息。实现代码如下。

例程 11　代码位置：光盘\mr\03\PlayVideo\index.aspx.cs

```
//获取公告标题
public string title;
//获取公告内容
public string content;
//获取公告发布时间
public string issuanceDate;
protected void Page_Load(object sender, EventArgs e)
{
    bindNew();                  //调用自定义方法显示最新视频
    bindHumour();               //调用自定义方法显示搞笑视频
    bindCartoon();              //调用自定义方法显示动漫视频
    bindFilm();                 //调用自定义方法显示电影视频
    bindSport();                //调用自定义方法显示体育视频
    bindBulletin();             //调用自定义方法显示公告信息
}
```

自定义 bindSport 方法用来绑定体育视频信息。由于其他视频信息的绑定类似，这里就不再介绍了。在该方法中通过使用 SQL 语句，查询出体育视频的信息并使用公共类型中的 getRows 方法返回的 DataSet 对象绑定到 DataList 控件中。实现代码如下。

例程 12　代码位置：光盘\mr\03\PlayVideo\index.aspx.cs

```
protected void bindSport()
{
    string sqlSel = "select top 10 * from videoInfo where videoType='体育' and Auditing=1 order by videoDate desc";
    dlSport.DataSource = operateData.getRows(sqlSel).DefaultView;
    dlSport.DataBind();
}
```

在“登录”按钮的单击事件中实现了用户登录功能。在该事件中首先将判断用户是否已登录，如果登录将清空 Session 对象。通过获取的用户登录名和密码创建 SQL 语句，来查询

用户登录名和密码是否存在，如果存在表示登录成功。登录成功后还需要判断该用户是否已被管理员冻结。判断用户表中 lock 字段，如果为 True 说明已经冻结并给出相应的提示。实现代码如下。

例程 13　代码位置：光盘\mr\03\PlayVideo\index.aspx.cs

```
protected void imgBtnEntry_Click(object sender, ImageClickEventArgs e)
{
    //判断用户是否登录
    if (Session["userName"] != "")
    {
        Session["userName"] = null;
    }
    //获取登录名
    string userName = txtUserName.Text;
    //获取密码
    string pass = txtUserPass.Text;
    //编写SQL语句，查询用户名和密码是否正确
    string sqlSel = "select * from userRegister where username=@name and userPass=@pass";
    //调用自定义方法执行SQL语句
    if (operateData.login(sqlSel, userName, pass))
    {
        //保存用户名
        Session["userName"] = userName;
        //编写SQL语句，指定用户的信息
        string sql = "select * from userRegister where userName='" + Session["userName"] + "'";
        //调用公共类中的getRow方法执行SQL语句，并接收SqlDataReader对象
        SqlDataReader sdr = operateData.getRow(sql);
        //读取一条记录
        sdr.Read();
        //判断当前用户是否锁定
        if (Convert.ToBoolean(sdr["lock"]))
        {
            //如锁定Session设置为空，并给出提示
            Session["userName"] = null;
            RegisterStartupScript("true", "<script>alert('" + sdr["lockCause"].ToString() + "');location='index.aspx'</script>");

        }
        else
            RegisterStartupScript("true", "<script>alert('登录成功！');location='index.aspx'</script>");
    }
    else
    {
        RegisterStartupScript("false", "<script>alert('用户名或密码错误！')</script>");
    }
    txtUserName.Text = txtUserPass.Text = "";
}
```

3.4.2　个人管理上传设计

个人管理上传页面中用户可以上传自己喜欢的视频供广大网友们欣赏。在该页面中需要填写视频标题、视频内容等信息，最后通过“上传”按钮实现上传操作。个人管理上传页面如图 3.5 所示。

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 upVideo.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 3.9 所示。

图 3.5　个人管理上传页面

表 3.9　　控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说　　明
数据/TextBox 控件	txtTitle	设置属性 RepeatDirection 为 Horizontal	输入视频名称
	txtContent	设置属性 RepeatDirection 为 Horizontal	输入视频内容
标准/ FileUpload 控件	fileupVideo	均为默认值	选择要上传的视频
标准/RadioButtonList 控件	radBtnListType	设置属性 RepeatDirection 为 Horizontal	显示视频的类型
标准/Button 控件	btnUpVideo	均为默认值	实现上传操作
	btnReturn	设置属性 PostBackUrl 为~/user/userIndex.aspx	跳转到用户首页

2. 后台代码编写

在该页面中创建一个全局的字符串数组，该数组中存储着允许上传的视频格式。在页面的加载事件中首先判断用户是否登录，如果未登录将给出提示并返回到首页。实现代码如下。

例程 14　代码位置：光盘\mr\03\PlayVideo\user\upVideo.aspx.aspx.cs

```
//设置上传文件的格式
    string[] videoExtension = new string[] { "flv", "avi", "wmv", };
    protected void Page_Load(object sender, EventArgs e)
    {
        //判断用户是否登录
        if (Session["userName"] == null)
        {
            //未登录给出提示并返回首页
            Response.Write("<script>alert('请您先登录！');location='../index.aspx'</script>");
        }
    }
```

在“上传”按钮的单击事件中，实现了视频的上传操作。在该事件中首先调用自定义 extension 方法判断用户上传文件的格式是否满足要求。如果满足要求将文件保存到指定的目录中。接着判断用户上传文件是否为.flv 格式。如果为.flv 格式将调用公共类中的方法实现截图视频图片操作。如果不为.flv 格式，将调用公共类中的方法将视频转换为.flv 格式，转换后再截取视频的图片。最后将调用自定义 insertVideoInfo 方法，将视频的信息保存到数据库中。实现代码如下。

例程 15 代码位置：光盘\mr\03\PlayVideo\user\upVideo.aspx.aspx.cs

```
protected void btnUpVideo_Click(object sender, EventArgs e)
 {
     //获取上传文件的名称
     string upFileName =fileupVideo.FileName;
     //判断是否选择了文件
     if (this.fileupVideo.HasFile)
     {
         //获取文件的扩展名
         string upExtension = upFileName.Substring(upFileName.LastIndexOf(".") + 1);
         //判断扩展名是否正确
         if (checkExtension(upExtension))
         {
             //获取上传文件所保存的路径
             string upFilePath = Server.MapPath("../upFile/") + upFileName;
             //将文件保存到指定路径中
             fileupVideo.SaveAs(upFilePath);
             //获取当前时间
             string saveName = DateTime.Now.ToString("yyyyMMddHHmmssffff");
             //获取视频转换后所保存的路径及文件名
             string playFile = "playFile/" + saveName + ".flv";
             //获取图片所保存的路径及名称
             string imgFile = "imgFile/" + saveName + ".jpg";
             try
             {
                 //判断上传的文件是否为.flv格式
                 if (upExtension == "flv")
                 {
                     //如果为flv格式直接保存到指定路径下
                     File.Copy(upFileName, playFile + ".flv");
                     //调用公共类中的catchImg方法截取视频图片
                     operateMethod.catchImg(upFileName, imgFile);
                     //调用自定义insertVideoInfo方法将视频的信息保存到数据库中
                     insertVideoInfo(playFile, imgFile);
                 }
                 else
                 {
                     //调用公共类中的changeVideoType方法转换视频格式
                     if (operateMethod.changeVideoType(upFileName, Server.MapPath("../") + playFile, Server.
MapPath("../") + imgFile))
                     {
                         //调用自定义insertVideoInfo方法将视频信息保存到数据库中
                         insertVideoInfo(playFile, imgFile);
                         //删除上传的视频
                         File.Delete(upFilePath);
                     }
                     else
                     {
                         RegisterStartupScript("false", "<script>alert('上传失败！')</script>");
                         //删除上传的视频
                         File.Delete(upFilePath);
                     }
                 }
```

```
                catch (Exception ex)
                {
                    Response.Write(ex.Message.ToString());
                }
            }
            else {
                RegisterStartupScript("false", "<script>alert('文件格式错误！')</script>");
            }
        }

    }
```

自定义 insertVideoInfo 方法用来实现将视频的信息保存到数据库中。在该方法中首先将获取视频的信息，再通过使用 SQL 语句将视频信息插入数据库中，插入成功后将使用 SQL 语句将上传视频的会员积分增加。实现代码如下。

例程 16 代码位置：光盘\mr\03\PlayVideo\user\upVideo.aspx.aspx.cs

```
    private void insertVideoInfo(string playFileh, string imgFile)
    {
        //获取用户名
        string userName =Session["userName"].ToString();
        //获取视频名称
        string videoTitle = txtTitle.Text;
        //获取视频内容
        string videoContent = txtContent.Text;
        //获取当前时间
        string date = DateTime.Now.ToString();
        //获取视频路径
        string videoPath = playFileh;
        //获取图片路径
        string videoPicture = imgFile;
        //获取视频的类型
        string videoType = "";
        int count = RadioButtonList1.Items.Count;
        for (int i = 0; i < count; i++)
        {
            if (RadioButtonList1.Items[i].Selected)
            {
                videoType = RadioButtonList1.Items[i].Value;
                break;
            }
        }
        //编写SQL语句将视频的详细信息添加到数据库中
        string sqlInsert = "insert into videoInfo values('" + userName + "','" + videoTitle + "','" + videoContent + "','" +
date + "','" + videoPath + "','" + videoPicture + "','" + videoType + "','','','','')";
        if (operateData.execSql(sqlInsert))
        {
            RegisterStartupScript("true", "<script>alert('上传成功！')</script>");
            //编写SQL语句将当前用户的积分增加
            string sqlUpd=" update userInfo set sumMark=sumMark+100 where userName='"+userName+"'" ;
            operateData.execSql(sqlUpd);
        }else RegisterStartupScript("true", "<script>alert('上传失败！')</script>");
    }
```

3.4.3 修改个人信息

在修改个人信息页面中，用户可以更改在注册时添加的详细信息，包括用户昵称、用户性别、用户头像、所在城市、个性留言等信息。修改个人信息页面如图 3.6 所示。

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 amendInfo.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 3.10 所示。

图 3.6　修改个人信息页面

表 3.10　　控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说　　明
标准/ TextBox 控件	txtNickname	均为默认值	输入用户昵称
	txtCity	均为默认值	输入所在城市
	txtQQ	均为默认值	输入用户 QQ
	txtSpeak	设置 TextMode 属性为 MultiLine	输入个性留言
标准/ RadioButtonList 控件	rbtnlsSex	均为默认值	选择用户性别
标准/ FileUpload 控件	fupImg	均为默认值	上传用户头像
标准/ Image 控件	ImageHead	均为默认值	显示用户头像
标准/ Button 控件	btnUp	均为默认值	实现上传头像操作
	btnAmend	均为默认值	实现修改信息操作

2. 后台代码编写

定义一个全局变量用来存储图片的路径，在页面的加载事件中判读用户是否登录，如果未登录则给出提示并返回首页。如果登录调用自定义 bindUserInfo 方法，则显示用户的详细信息。实现代码如下。

例程 17　代码位置：光盘\mr\03\PlayVideo\user\amendInfo.aspx

```
public static   string imgpath ="";
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //判断用户是否登录
        if (Session["userName"] == null)
        {
            Response.Write("<script>alert('请您先登录！');location='../index.aspx'</script>");
        }else
        //调用自定义方法显示用户的详细信息
        bindUserInfo();
```

```
        }
    }
```

自定义 bindUserInfo 方法，将通过 SQL 语句查询出当前用户的详细信息，并显示在页面中。实现代码如下。

例程 18 代码位置：光盘\mr\03\PlayVideo\user\amendInfo.aspx

```
//显示用户的详细信息
    protected void bindUserInfo()
    {
     //编写SQL语句查询当前登录用户的详细信息
        string sqlSel = "select * from userInfo where userName='" + Session["userName"].ToString() + "'";
        //调用公共类中的getRow方法并接收返回的SqlDataReader对象
        SqlDataReader sdr = operateData.getRow(sqlSel);
        //读取一条数据
        sdr.Read();
        //显示用户的昵称
        txtNickname.Text = sdr["nickName"].ToString();
        //显示用户性别
        if (sdr["sex"].ToString().Trim() == "男")
        {
            rbtnlsSex.Items[0].Selected=true;
        }
        else
        {
             rbtnlsSex.Items[1].Selected=true;
        }
        //获取图片路径
        imgpath=sdr["img"].ToString();
        //显示用户的图片
        ImageHead.ImageUrl = "../" + imgpath;
        //显示用户所在城市
        txtCity.Text = sdr["city"].ToString();
        //显示用户的QQ
        txtQQ.Text = sdr["qq"].ToString();
        //显示用户的留言
        txtSpeak.Text = sdr["speak"].ToString();
        sdr.Close();
    }
```

在“上传”按钮的单击事件中，修改获取上传图片的名称并保存到指定的路径下，最后将图片显示在页面中。实现代码如下。

例程 19 代码位置：光盘\mr\03\PlayVideo\user\amendInfo.aspx

```
        //获取上传图片的路径
        string imgName = fupImg.FileName;
        //更改上传图片的名称
        imgName = imgName.Replace(imgName.Substring(0, imgName.LastIndexOf(".")), Session["userName"].ToString());
        //设置上传图片的保存路径
        imgpath = "imgHead/" + imgName;
        //保存上传图片
        fupImg.SaveAs(Server.MapPath("../") + imgpath);
        //设置一个参数，使IE重新读取页面显示头像
        string rd = DateTime.Now.Ticks.ToString();
        //显示上传的图片
        ImageHead.ImageUrl = "../" + imgpath+"?rd="+rd;
    }
```

在“修改”按钮的单击事件中，先获取用户修改后的信息，并通过 SQL 语句将数据库中用户的详细信息修改。最后调用自定义 bindUserInfo 方法重新显示用户信息。实现代码如下。

例程 20 代码位置：光盘\mr\03\PlayVideo\user\amendInfo.aspx

```
protected void btnAmend_Click(object sender, EventArgs e)
    {
    //编写SQL语句查询当前登录用户的详细信息
```

```
        string sqlSel = "select * from userInfo where userName='" + Session["userName"].ToString() + "'";
        //调用公共类中的getRow方法并接收返回的SqlDataReader对象
        SqlDataReader sdr = operateData.getRow(sqlSel);
        //读取一条数据
        sdr.Read();
        //显示用户的昵称
        txtNickname.Text = sdr["nickName"].ToString();
        //显示用户性别
        if (sdr["sex"].ToString().Trim() == "男")
        {
            rbtnlsSex.Items[0].Selected=true;
        }
        else
        {
             rbtnlsSex.Items[1].Selected=true;
        }
        //获取图片路径
        imgpath=sdr["img"].ToString();
        //显示用户的图片
        ImageHead.ImageUrl = "../" + imgpath;
        //显示用户所在城市
        txtCity.Text = sdr["city"].ToString();
        //显示用户的QQ
        txtQQ.Text = sdr["qq"].ToString();
        //显示用户的留言
        txtSpeak.Text = sdr["speak"].ToString();
        sdr.Close();
    }
        //获取用户修改后的详细信息并保存到数据库中
    protected void btnAmend_Click(object sender, EventArgs e)
    {
        //获取当前用户名
        string userName = Session["userName"].ToString();
        //获取昵称
        string nickname = txtNickname.Text;
        //获取性别
        string sex="";
        if (rbtnlsSex.SelectedValue =="男")
        {
            sex = "男";
        }else
            if (rbtnlsSex.SelectedValue=="女")
            {
                sex = "女";
            }
        //获取所在城市
        string city = txtCity.Text;
        //获取QQ
        string qq = txtQQ.Text;
        //获取留言
        string speak = txtSpeak.Text;
        //编写SQL语句更新用户的详细信息
        string sqlUp = "update userInfo set nickname='"+nickname+"',sex='"+sex+"',img='"+imgpath+"',city='"+city+"', qq='"+qq+"',
speak='"+speak+"' where userName='"+userName+"'";
        if (operateData.execSql(sqlUp))
        {
            RegisterStartupScript("true", "<script>alert('修改成功!')</script>");
            //调用自定义方法重新显示用户信息
            bindUserInfo();
        }
        else
        {
            RegisterStartupScript("false", "<script>alert('修改失败！')</script>");
```

```
        }
    }
```

3.4.4 播放视频并发表评论设计

在播放视频并发表评论页面中，用户可以欣赏到自己喜欢的视频。还可以对该视频发表自己相关的意见。播放视频并发表评论页面如图 3.7 所示。

图 3.7 播放视频并发表评论页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 play.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 3.11 所示。

表 3.11　　控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说　　明
数据/DataList 控件	dlIdea	均为默认值	用来显示评论信息
标准/ TextBox 控件	txtContent	均为默认值	输入发布的评论
标准/Label 控件	lbeUserName	均为默认值	显示用户登录名
	labCount	均为默认值	记录用户输入评论的数量
标准/LinkButton 控件	lkbtnLogin	PostBackUrl="~/login.aspx"	跳转到用户登录页面
	lkbtnRegister	PostBackUrl="~/Register.aspx"	跳转到用户注册页面
标准/Button 控件	btnIdea	均为默认值	实现发布评论操作
标准/Literal 控件	Literal1	均为默认值	用来显示播放器
标准/Panel 控件	PanelIdea	设置属性 ScrollBars 为 Vertical	设置评论区域滚动条
	PanelHello	均为默认值	显示或隐藏用户登录欢迎区域
	PanelLogin	均为默认值	显示或隐藏用户登录区域

2. 后台代码编写

在该页面中创建几个全局变量来保存视频的信息，以方便在前台显示。在该页面的加载事件中判断用户是否登录，如果登录将显示用户欢迎词，未登录将显示登录区域。如果页面是第一次加载还需要调用自定义 addPlaySum 方法增加视频的点击率。最后将调用自定义 videoInfo 方法和 bidList 方法，实现播放视频、显示视频信息和显示视频的评论。实现代码如下。

例程 21　代码位置：光盘\mr\03\PlayVideo\ play.aspx.cs

```
public string playSum; //保存视频点击率
public string flower;   //保存视频被顶的次数
public string tile;     //保存视频被踩的次数
public string videoDate;     //保存视频发布时间
public string Name;    //发布人
public string videoTitle; //视频名称
public string videoContent; //视频内容
public string videoType;     //视频类型
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //判断用户是否登录
        if (Session["userName"] == null)
        {
            //未登录显示登录panel
            PanelLogin.Visible = true;
            //隐藏欢迎panel
            PanelHello.Visible = false;
        }
        else
        {
            //已登录隐藏登录panel
            PanelLogin.Visible = false;
            //显示欢迎panel
            PanelHello.Visible = true;
            //显示登录名
            lbeUserName.Text = Session["userName"].ToString();
        }
        //调用自定义方法增加视频的点击率
        addPlaySum();
    }
    //播放视频并显示视频详细信息
    videoInfo();
```

```
            //显示评论
            bidList();
        }
```

自定义 videoInfo 方法用来播放视频并显示视频的信息。在方法中首先使用 SQL 语句来查询视频的详细信息。并将详细信息保存到全局变量中。最后将调用公共类中的 GetFlashText 方法来显示并播放视频。实现代码如下。

例程 22 代码位置：光盘\mr\03\PlayVideo\ play.aspx.cs

```
protected void videoInfo()
    {
        //编写SQL语句查询视频的详细信息
        string sql = "select * from videoInfo where id=" + Request["id"];
        SqlDataReader sdr = operateData.getRow(sql);
        sdr.Read();
        //获取视频的路径
        string link = sdr["videoPath"].ToString();
        //获取视频的点击率
        playSum = sdr["playSum"].ToString();
        //获取顶人数
        flower = sdr["flower"].ToString();
        //获取踩人数
        tile = sdr["tile"].ToString();
        //获取视频发布日期
        videoDate = sdr["videoDate"].ToString();
        //获取发布人名称
        Name = sdr["userName"].ToString();
        //获取视频标题
        videoTitle = sdr["videoTitle"].ToString();
        //获取视频内容
        videoContent = sdr["videoContent"].ToString();
        //获取视频类型
        videoType = sdr["videoType"].ToString();
        //判断视频路径开头字符串是否为http://
        if (!link.StartsWith("http://"))
        {
            //获取当前的绝对路径
            string sss = Request.Url.AbsoluteUri;
            //查询play.aspx在字符串中的位置
            int idx = sss.IndexOf("play.aspx");
            //获取指定字符串
            sss = sss.Substring(0, idx);
            link = sss + link;
        }
        //显示播放器并可以播放视频
        this.Literal1.Text = operateMethod.GetFlashText(link);
    }
```

自定义 addPlaySum 方法用来增加用户的点击率和会员的积分。在该方法中通过使用 SQL 语句将视频的点击率增加，在根据查询出的会员名使用 SQL 语句增加会员的积分。实现代码如下。

例程 23 代码位置：光盘\mr\03\PlayVideo\ play.aspx.cs

```
public void addPlaySum()
    {
        //创建SQL语句，增加视频的点击率
        string sql = "update videoInfo set playSum=playSum+1,monthSum=monthSum+1 where id=" + Request["id"];
        operateData.execSql(sql);
        //创建SQL语句，查询出发布视频会员名
        string sqlSel = "select userName from videoInfo where id=" + Request["id"];
        //获取会员名
        string userName = operateData.getTier(sqlSel);
        //创建SQL语句，增加用户的积分
        string sqlUpd = " update userInfo set sumMark=sumMark+1 where userName='" + userName + "'";
        //执行SQL语句
```

```
        operateData.execSql(sqlUpd);
    }
```

自定义 bidList 方法用来显示视频的评论信息。在该方法中使用 SQL 语句查询出评论信息，并判断评论信息是否小于 5 条，如果小于 5 条视频显示评论信息区域的 Panel 的滚动条则为不可见。实现代码如下。

例程 24 代码位置：光盘\mr\03\PlayVideo\user\upVideo.aspx.aspx.cs

```
protected void bidList()
{
    //创建SQL语句，查询出当前视频的所有评论
    string sqlSel = "select * from videoIdea where videoId=" + Request["id"] + " order by issuanceDate desc ";
    //调用数据库操作类中的getRows方法并接收返回值
    DataTable dt = operateData.getRows(sqlSel);
    //判读DataTable中的资料是否小于5行
    if (dt.Rows.Count < 5)
    {
        //隐藏Panel控件的滚动条
        PanelIdea.ScrollBars = ScrollBars.None;
    }
    //设置DataList控件的数据源
    dlIdea.DataSource = dt;
    //设置主键
    dlIdea.DataKeyField = "id";
    //绑定显示
    dlIdea.DataBind();
}
```

3.4.5 体育视频管理设计

体育视频管理页面可以查看到所有用户发布的体育视频信息，管理员可以对其进行相应的管理。体育视频管理页面如图 3.8 所示。

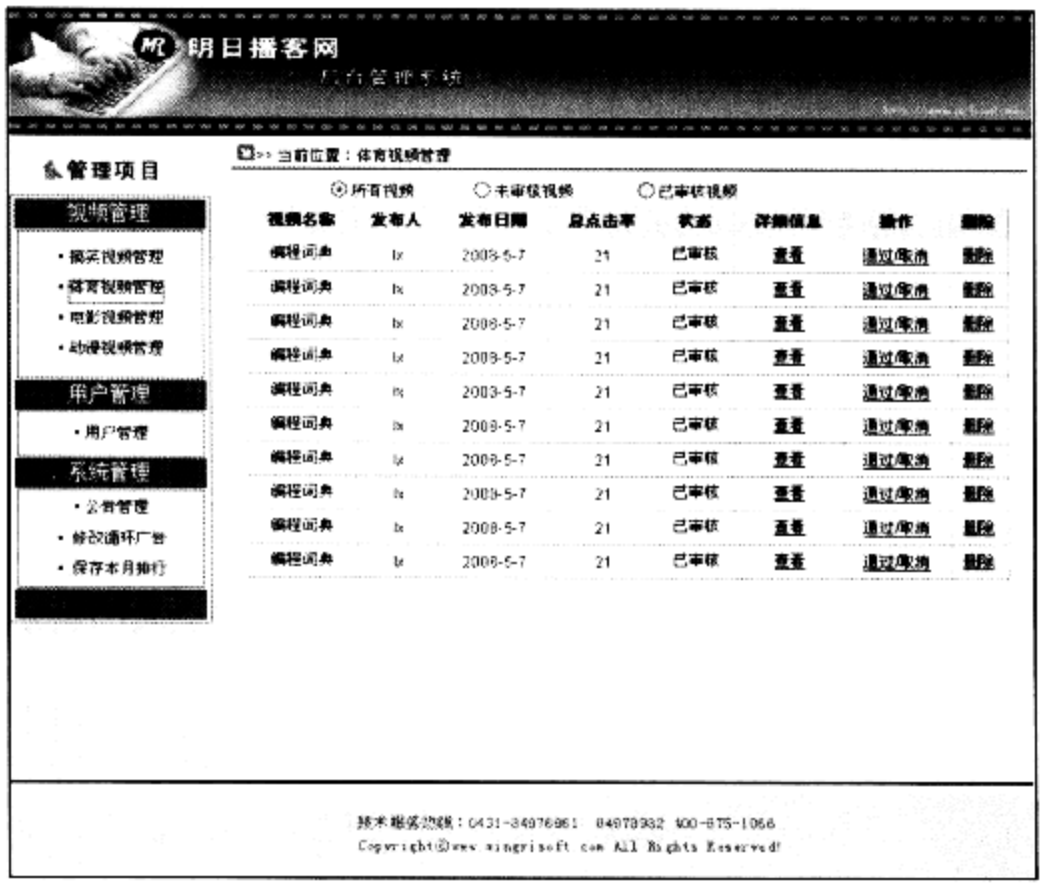


图 3.8 体育视频管理页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 manage_sport.aspx。

(2) 在该窗体中添加 1 个 RadioButtonList 控件，用来选择查看视频的方式。还需要添加 1

个 GridView 控件，该控件用来显示体育视频的详细信息。GridView 控件的前台绑定代码如下。

例程 25　代码位置：光盘\mr\03\PlayVideo\manage\manage_sport.aspx

```
<asp:GridView ID="gvVideo" runat="server" AutoGenerateColumns="False" OnRowDataBound="gvVideo_RowDataBound"
OnRowDeleting="gvVideo_RowDeleting" OnSelectedIndexChanging="gvVideo_SelectedIndexChanging" AllowPaging="True"
OnPageIndexChanging="gvVideo_PageIndexChanging" Width="580px">
                                <Columns>
                                <asp:TemplateField HeaderText="视频名称">
                                        <ItemTemplate>
                                          <a title="<%#Eval("videoTitle") %>"> <%#operateMethod.interceptStr
((string )Eval("videoTitle"),5 ) %> </a>
                                        </ItemTemplate>
                                    </asp:TemplateField>
                                    <asp:BoundField DataField="userName" HeaderText="发布人" />
                                    <asp:BoundField DataField="videoDate" HeaderText="发布日期" DataFormatString=
"{0:d}" HtmlEncode="False" />
                                    <asp:BoundField DataField="playSum" HeaderText="总点击率" />
                                    <asp:BoundField DataField="Auditing" HeaderText="状态" />
                                    <asp:TemplateField HeaderText="详细信息">
                                        <ItemTemplate>
                                            <a href ='../play.aspx?id=<%#Eval("id") %>' target="_blank"     >查看</a>
                                        </ItemTemplate>
                                    </asp:TemplateField>
                                    <asp:CommandField HeaderText="操作" SelectText="通过/取消" ShowSelectButton="True" />
                                    <asp:CommandField HeaderText="删除" ShowDeleteButton="True" />
                                </Columns>
                                  <RowStyle CssClass="huise1" />
                                <HeaderStyle BackColor="#FFC734" CssClass="hongcu" />
                            </asp:GridView>
```

2. 后台代码编写

在该页面的加载事件中将调用自定义 bindGvVideo 方法来显示体育视频信息。在自定义 bindGvVideo 方法中首先判断用户要查看视频的类型，视频的类型分为所有视频、未审核视频和已审核视频。根据相应的视频类型来设置 SQL 语句。最后将查询出的视频信息绑定到 GridView 控件上显示出来。实现代码如下。

例程 26　代码位置：光盘\mr\03\PlayVideo\manage \manage_sport.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //调用自定义方法显示体育视频信息
    bindGvVideo();
}
protected void bindGvVideo()
{
    string sqlSel = "";
    //判断是否选择“所有视频”
    if (RadioButtonList1.SelectedValue == "0")
    {
        sqlSel = "select * from videoInfo where videoType='体育'";
    }
    //判读是否选择“未审核视频”
    else if (RadioButtonList1.SelectedValue == "1")
    {
        sqlSel = "select * from videoInfo where videoType='体育' and Auditing=0 ";
    }
    //判断是否选择“已审核视频”
    else if (RadioButtonList1.SelectedValue == "2")
    {
        sqlSel = "select * from videoInfo where videoType='体育' and Auditing=1 ";
    }
    gvVideo.DataSource = operateData.getRows(sqlSel);
    gvVideo.DataKeyNames = new string[] { "id" };
```

```
        gvVideo.DataBind();
    }
```

改变视频的审核状态是在 GridView 控件的 SelectedIndexChanging 事件中实现的。在该事件中首先将获取视频当前的审核状态，如果当前的审核状态为 True，将其设置为 False。如果为 False，将其设置为 True。最后将审核状态保存到数据库中。实现代码如下。

例程 27　代码位置：光盘\mr\03\PlayVideo\manage \manage_sport.aspx.cs

```
protected void gvVideo_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
    {
        //获取视频的键值
        string id = gvVideo.DataKeys[e.NewSelectedIndex].Value.ToString();
        //编写SQL语句查询当前视频的审核状态
        string sqlSel = "select Auditing from videoInfo where id=" + id;
        //调用公共类中的getTier方法获取视频的审核状态
        string Auditing = operateData.getTier(sqlSel);
        //判断是否未审核
        if (Auditing == "False")
        {
            //将审核状态修改为已审核
            Auditing = "1";
        }
        else
        {
            //将审核状态修改为未审核
            Auditing = "0";
        }
        string sqlUpd = "update videoInfo set Auditing='" + Auditing + "' where id=" + id;
        operateData.execSql(sqlUpd);
        //调用自定义方法重新显示体育视频信息
        bindGvVideo();
    }
```

3.4.6　用户管理设计

用户管理页面用来管理所有的用户，在该页面中可以查看到用户的详细信息并对用户进行管理。用户管理页面如图 3.9 所示。

图 3.9　用户管理页面

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 manage_user.aspx。

（2）在该页面中添加 1 个 GridView 控件，该控件用来显示用户的详细信息。GridView 控件的前台绑定代码如下。

例程 28 代码位置：光盘\mr\03\PlayVideo\manage \manage_user.aspx

```
    <asp:GridView ID="gvUser" runat="server" AllowPaging="True" AutoGenerateColumns="False" OnPageIndexChanging=
"gvUser_PageIndexChanging" OnRowDataBound="gvUser_RowDataBound" OnRowDeleting="gvUser_RowDeleting"
OnSelectedIndexChanging="gvUser_SelectedIndexChanging" Height="322px" Width="502px">
                <Columns>
                    <asp:BoundField DataField="userName" HeaderText="用户名" />
                    <asp:BoundField DataField="nickName" HeaderText="昵称" />
                    <asp:BoundField DataField="sumMark" HeaderText="积分" />
                    <asp:BoundField DataField="registerDate" HeaderText="注册日期" />
                    <asp:BoundField DataField="lock" HeaderText="状态" />
                    <asp:TemplateField HeaderText="详细信息">
                        <ItemTemplate>
                            <a href ='../userInfo.aspx?userName=<%#Eval("userName") %>' target="_blank"    >查看</a>
                        </ItemTemplate>
                    </asp:TemplateField>
                    <asp:CommandField HeaderText="操作" SelectText="锁定/解锁" ShowSelectButton="True" />
                    <asp:CommandField HeaderText="删除" ShowDeleteButton="True" />
                </Columns>
                <RowStyle CssClass="huise1" />
                        <HeaderStyle BackColor="#FFC734" CssClass="hongcu" />
            </asp:GridView>
```

2. 后台代码编写

在页面加载事件中调用自定义 bindGvUser 方法来显示用户的详细信息。在自定义 bindGvUser 方法中使用 SQL 语句查询出所有用户的信息，并将用户信息绑定到 GridView 控件上显示出来。实现的代码如下。

例程 29 代码位置：光盘\mr\03\PlayVideo\manage \manage_user.aspx.cs

```
    protected void Page_Load(object sender, EventArgs e)
        {
            //调用自定义方法显示用户信息
             bindGvUser();
        }
            //自定义方法将用户信息绑定到GridView控件上
        protected void bindGvUser()
        {
            //编写SQL语句，查询用户的详细信息
            string sqlSel = "select a.userName,a.lock,b.nickname,b.sumMark,b.registerDate from userRegister as a join
userInfo as b on a.userName=b.userName";
            //设置GridView控件的数据源
            gvUser.DataSource = operateData.getRows(sqlSel);
            //设置GridView控件的主键
            gvUser.DataKeyNames = new string[] { "userName" };
            gvUser.DataBind();
        }
```

为了管理员方便查询用户信息，在 GridView 控件的 RowDataBound 事件中将表示用户是否锁定的布尔值转换为文本值。实现代码如下。

例程 30 代码位置：光盘\mr\03\PlayVideo\manage \manage_user.aspx.cs

```
    protected void gvUser_RowDataBound(object sender, GridViewRowEventArgs e)
        {
```

```
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            //判断当前用户是否已被锁定
            if (e.Row.Cells[4].Text=="False")
            {
                //显示锁定文字
                e.Row.Cells[4].Text = "未锁定";
            }
            else {
                e.Row.Cells[4].Text = "已锁定";
                //设置当前行的背景颜色
                e.Row.BackColor = System.Drawing.Color.Red;
            }
        }
    }
```

在 GridView 控件的 SelectedIndexChanging 事件中，用来实现用户的锁定操作。首先获取用户的锁定信息，判断用户当前是否被锁定。如果被锁定将改为解锁状态，如果为解锁状态将转换为锁定状态。最后保存到数据库中。实现代码如下。

例程 31　代码位置：光盘\mr\03\PlayVideo\manage \manage_user.aspx.cs

```
protected void gvUser_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
{
    //获取当前行用户的编号
    string userName = gvUser.DataKeys[e.NewSelectedIndex].Value.ToString();
    //编写SQL语句，查询当前行的用户锁定状态
    string sqlSel = "select lock from userRegister where userName='" + userName+"'";
    //获取锁定状态
    string userLock = operateData.getTier(sqlSel);
    //判断是否已被锁定
    if (userLock == "False")
    {
        //打开锁定原因窗口
        RegisterStartupScript("false", "<script>window.open('lockCause.aspx?userName=" +userName + "','','width=310,
height=190')</script>");
        //设置为锁定
        userLock = "1";
    }
    else
    {
        //设置为未锁定
        userLock = "0";
    }
    //编写SQL语句，更新当前用户的锁定状态
    string sqlUpd = "update userRegister set lock='" + userLock + "' where userName='" + userName + "'";
    operateData.execSql(sqlUpd);
    //调用自定义方法显示用户信息
    bindGvUser();
}
```

3.4.7　修改循环广告页面

修改循环广告页面可以修改首页中循环广告的显示图片。修改循环广告页面运行结果如图 3.10 所示。

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 manage_amendLoopAd.aspx。

（2）修改循环广告页面中，添加 1 个 GridView 控件，该控件用来显示并修改循环广告信息。该控件的前台绑定代码如下。

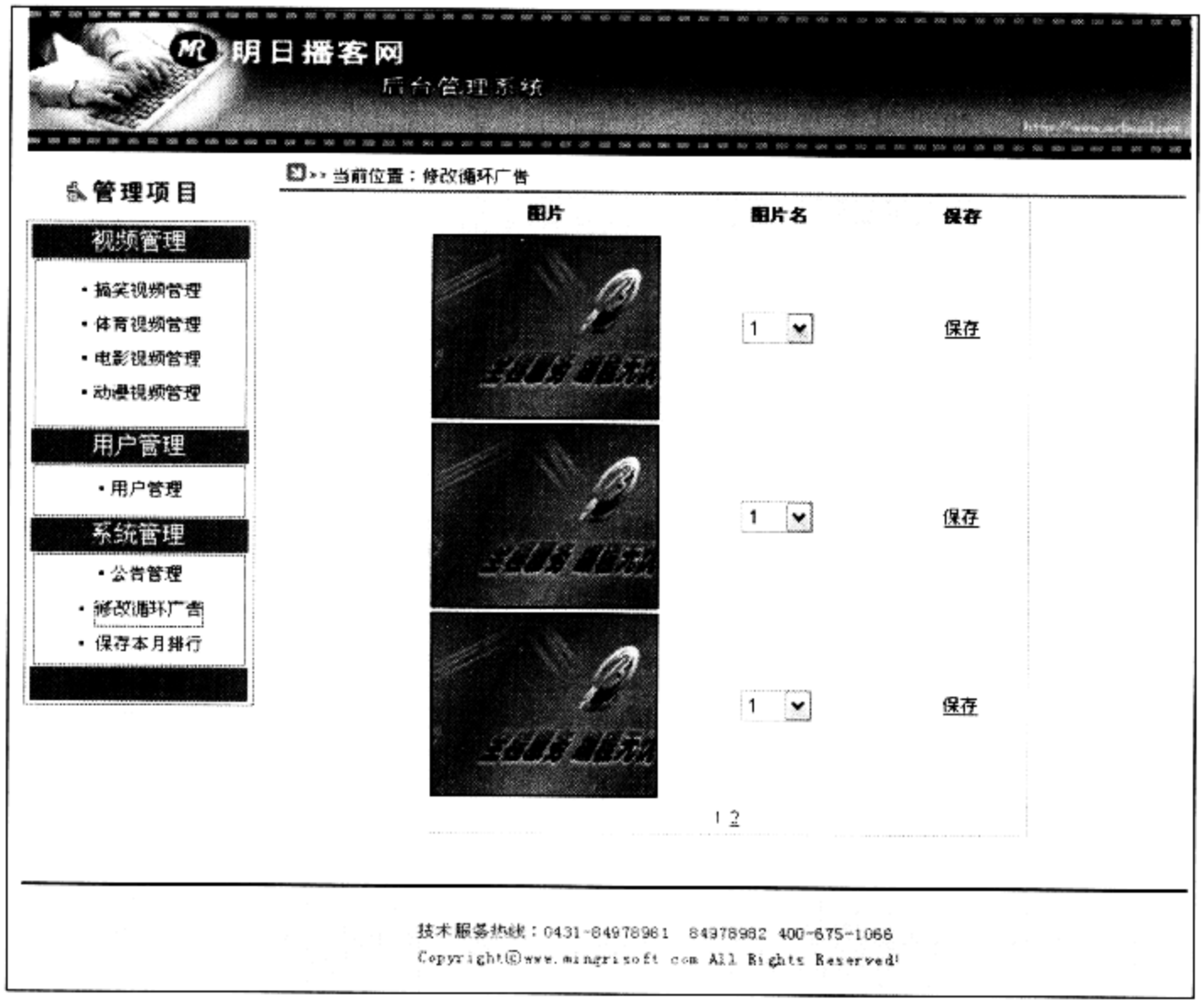


图 3.10 修改循环广告

例程 32 代码位置：光盘\mr\03\PlayVideo\manage \manage_amendLoopAd.aspx

```
<asp:GridView ID="gvImg" runat="server" AutoGenerateColumns="False" Height="182px"
        Width="401px" AllowPaging="True" OnPageIndexChanging="gvImg_PageIndexChanging" PageSize="3"
OnSelectedIndexChanging="gvImg_SelectedIndexChanging">
        <Columns>
         <asp:TemplateField HeaderText="图片">
         <ItemTemplate>
          <img align="middle" border="1"   src='../<%#Eval("videoPicture") %>' style="height: 127px;" width="150" />
        </ItemTemplate>
        <ItemStyle Width="150px" />
        </asp:TemplateField>
          <asp:TemplateField HeaderText="图片名">
            <ItemTemplate>
            <asp:DropDownList ID="DropDownList1" runat="server" Width="48px">
              <asp:ListItem>1</asp:ListItem>
                <asp:ListItem>2</asp:ListItem>
                  <asp:ListItem>3</asp:ListItem>
                  <asp:ListItem>4</asp:ListItem>
               </asp:DropDownList>
           </ItemTemplate>
        </asp:TemplateField>
        <asp:CommandField HeaderText="保存" SelectText="保存" ShowSelectButton="True" />
            </Columns>
          <RowStyle CssClass="huise1" />
          <HeaderStyle BackColor="#E0EEF3" CssClass="hongcu" />
      </asp:GridView>
```

2. 后台代码编写

在页面的加载事件中，调用自定义 bindGvImg 方法显示所有的视频图片。在 bindGvImg 方法通过 SQL 语句查询所有视频的图片并绑定到 GridVidw 控件中。实现代码如下。

例程 33 代码位置：光盘\mr\03\PlayVideo\manage \manage_amendLoopAd.aspx

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            //调用自定义方法显示所有截图信息
            bindGvImg();
        }
    }
    protected void bindGvImg()
    {
        //编写SQL语句查询出所有视频信息
        string sqlSel = "select * from videoInfo order by id desc ";
        //调用公共类中的getRows方法并设置GridView控件的数据源
        gvImg.DataSource = operateData.getRows(sqlSel);
        //设置主键
        gvImg.DataKeyNames = new string[] { "id" };
        gvImg.DataBind();
    }
```

在修改循环广告中的图片后，单击“保存”按钮将图片保存到指定目录下并将视频的编号保存到文本文件中。此功能主要在 GridView 控件的 SelectedIndexChanging 事件中实现。实现代码如下。

例程 34 代码位置：光盘\mr\03\PlayVideo\manage \manage_amendLoopAd.aspx

```
protected void gvImg_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
    {
        //编写SQL语句查询当前视频的图片路径
        string sqlSel = "select videoPicture from videoInfo where id=" + gvImg.DataKeys[e.NewSelectedIndex].Value.ToString();
        //获取当前图片的绝对路径
        string imgPath =Server.MapPath("../")+operateData.getTier(sqlSel);
        //获取当前行中DropDownList对象
        DropDownList ddl = (DropDownList)gvImg.Rows[e.NewSelectedIndex].FindControl("DropDownList1");
        //设置图片的保存路径及名称
        string imgNewPath = Server.MapPath("../")+"img/"+ddl.Text + ".jpg";
        //创建FileInfo对象
        FileInfo fi = new FileInfo(imgPath);
        //判断图片是否存在
        if (fi.Exists)
        {
            //将文件复制到指定路径下
            File.Copy(imgPath, imgNewPath, true);
            //调用自定义方法将视频的ID保存到指定文件中
            addLoopId(gvImg.DataKeys[e.NewSelectedIndex].Value.ToString(), Convert.ToInt32(ddl.Text));
            RegisterStartupScript("true", "<script>alert('保存成功！')</script>");
        }
        else
        {
            RegisterStartupScript("true", "<script>alert('图片不存在！')</script>");
        }
    }
```

自定义 addLoopId 方法，将保存视频编号的文本文件的内容读取出来，修改后在写入文本文件中。实现代码如下。

例程 35 代码位置：光盘\mr\03\PlayVideo\manage \manage_amendLoopAd.aspx

```
protected void addLoopId(string   videoId,int adId)
    {
```

```
        //获取保存视频ID的txt文件
        string path = Server.MapPath("../") + "tool/LoopId.txt";
        //获取txt文件中的内容
            string loopId = File.ReadAllText(path);
        //将txt内容以“,”分隔的字符串保存到数组中
        string[] loopids = loopId.Split(',');
        //修改指定位置的视频ID
        loopids[adId - 1] = "play.aspx?id="+videoId;
        //将数组保存到字符串中
        loopId = loopids[0];
        for (int i = 1; i < loopids.Length;i++ )
        {
            loopId += "," + loopids[i];
        }
        //将字符串中的内容保存到txt文件中
            File.WriteAllText(path,loopId);

    }
```

3.5　网站打包与发布

网站开发完成后最终的目的是将其发布到 Internet 上，以提供用户浏览访问。实现的网站发布可以使用两种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍下使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

（1）首先打开要发布的网站，在菜单栏中选择“生成”选项，在弹出的快捷方式菜单中选择“发布网站”项，如图 3.11 所示。

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择[...]路径按钮。如图 3.12 所示。

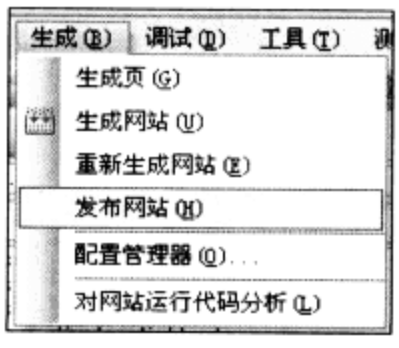


图 3.11　选择“发布网站”

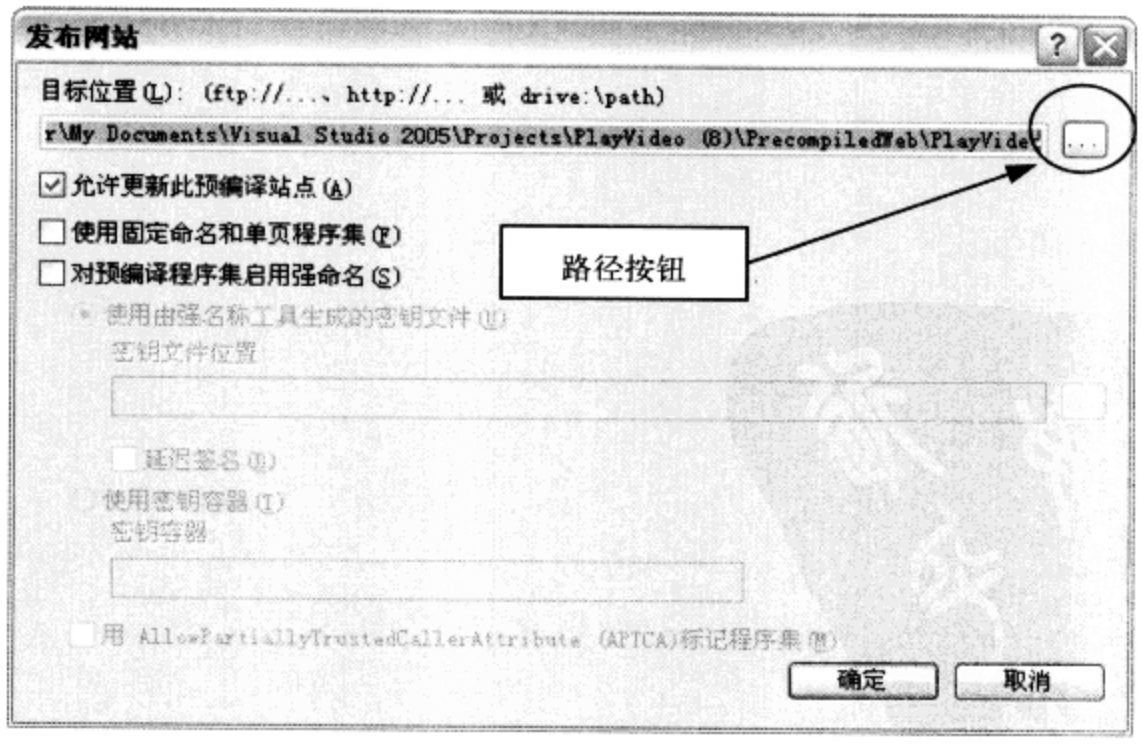


图 3.12　选择路径按钮

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选项中会显示需要填写的相应信息，

如服务器地址、目录、用户名及密码如图 3.13 所示。填写完毕后单击“打开”按钮将会返回“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

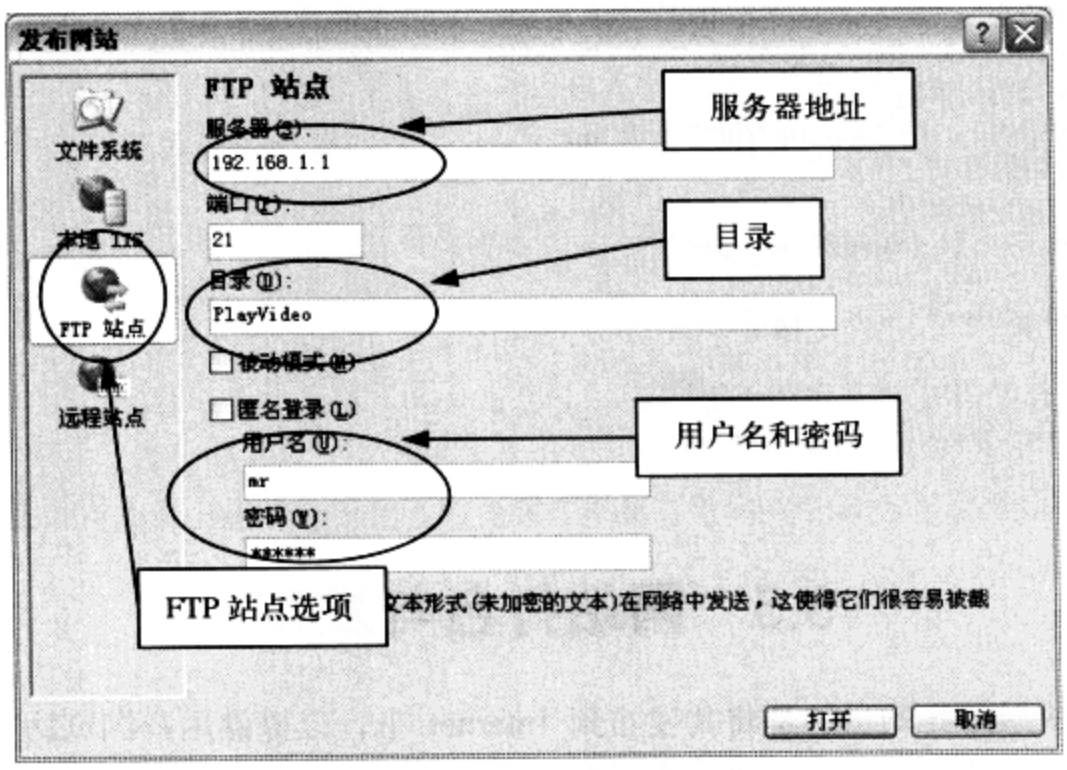


图 3.13　FTP 站点选项

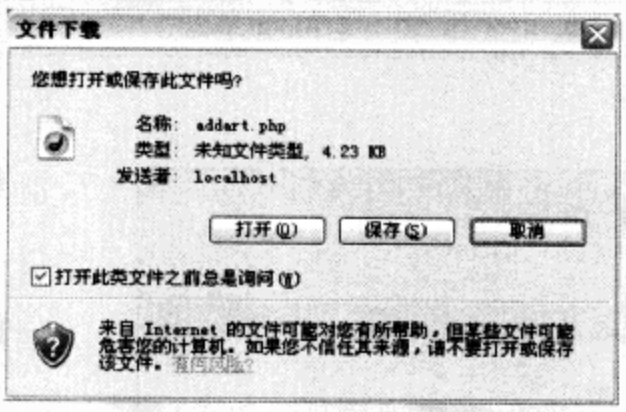
网络硬盘

第4章

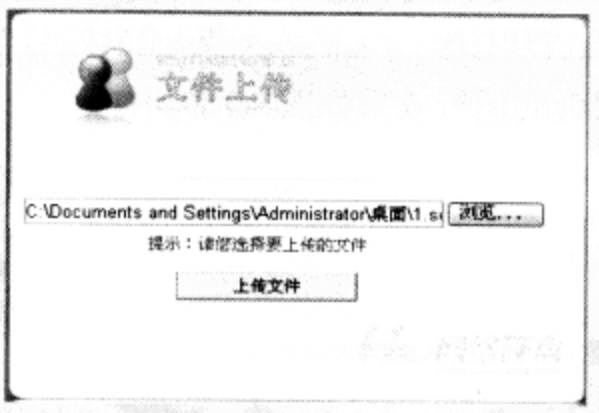
实例位置：光盘\mr\04\

随着计算机逐步深入人们的生活，越来越多的文件被存放到计算机上。为了适应现代社会发展的需要，更大程度地满足广大用户对文件处理需求，本章制作了一个功能强大的网络硬盘。通过网络硬盘，可以对文件进行“管理”、“创建”、“删除”、“修改”、“重命名”和“搜索”等操作以及对文件夹进行“创建”和“编辑”。通过本章的学习，读者能够学到以下知识。

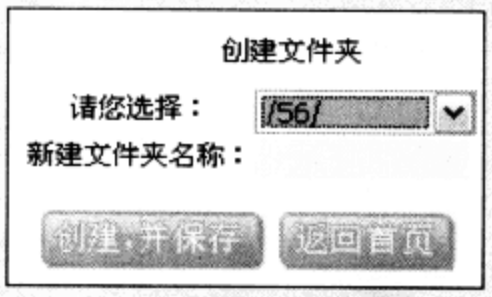
文件下载



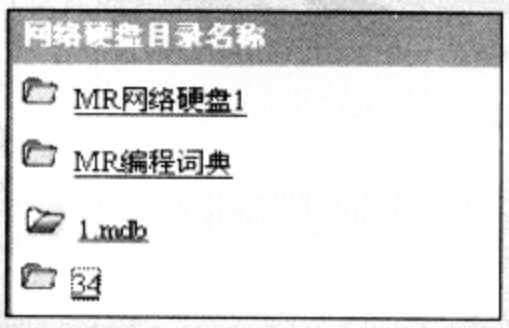
将文件上传到指定的目录中



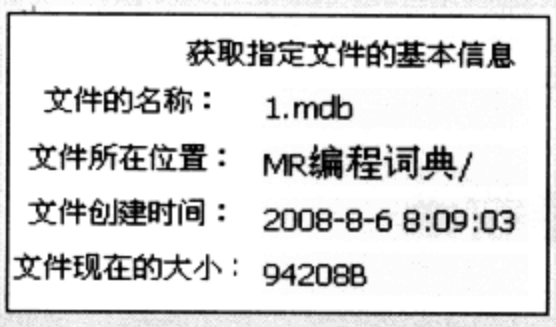
将文件创建到指定的目录下



文件和文件夹以不同的图标显示



获取指定文件的基本信息



搜索指定目录的文件

4.1 网络硬盘概述

网络硬盘能够很方便地管理个人的文件及文件夹，在网络硬盘中能够“下载文件”、“将文件上传到指定的文件夹中”、“删除上传的文件”、“修改文件”、“删除文件夹”和“查看详细的文件信息”等功能。

为了方便读者阅读和有效地利用本书附赠光盘中的实例，网站页面的各部分说明以表格形式，如表 4.1 所示。

表 4.1 网站首页解析

区　域	名　称	说　明	对 应 文 件
1	网站标头	主要用于网站的旗帜广告	header.ascx
2	上一级	主要用于将当前目录返回到上一级目录中	Default.aspx
3	搜索	用于执行搜索网络硬盘中的文件信息	Default.aspx
4	创建文件夹	用于在指定的目录中创建新文件夹	Default.aspx
5	显示硬盘信息	将硬盘信息绑定到 GridView 控件当中	Default.aspx
6	网站脚标	用于显示客服电话等信息内容	footer.ascx

网络硬盘的运行效果如图 4.1 所示。

图 4.1　网络硬盘运行效果图

4.2 网络硬盘关键技术

4.2.1 文件及文件夹处理技术

在网络硬盘中对文件及文件夹进行批量处理时，主要用到了文件流技术，而操作文件流的类都位于 System.IO 命名空间下。主要用到了 File 类、FileInfo 类、Directory 类和 DirectoryInfo 类，下面对它们进行详细讲解。

1. 文件的创建、移动、复制和删除操作

File 类支持对文件的基本操作，它包括用于创建、复制、删除、移动和打开文件的静态方法，并协助创建 FileStream 对象。File 类中一共包含 40 多个方法，这里只列出本模块中用到的几种方法，如表 4.2 所示。

表 4.2　　File 类的常用方法及说明

方　法	说　明
Copy	将现有文件复制到新文件
Create	在指定路径中创建文件
Delete	删除指定的文件。如果指定的文件不存在，则不引发异常
Exists	确定指定的文件是否存在
Move	将指定文件移到新位置，并提供指定新文件名的选项
Open	打开指定路径上的 FileStream
Replace	使用其他文件的内容替换指定文件的内容，这一过程将删除原始文件，并创建被替换文件的备份

说　明

（1）由于 File 类中的所有方法都是静态的，所以如果只想执行一个操作，那么使用 File 类中方法的效率比使用相应的 FileInfo 类中的方法可能更高。（2）File 类的静态方法对所有方法都执行安全检查，因此如果打算多次重用某个对象，可考虑改用 FileInfo 类中的相应方法，因为并不总是需要安全检查。

2. FileInfo 类

FileInfo 类和 File 类之间许多方法的调用都是相同的，但是 FileInfo 类没有静态方法，该类中的方法仅可以用于实例化的对象。File 类是静态类，所以它的调用需要字符串参数为每一个方法调用规定文件位置。因此如果要在对象上进行单一方法调用，则可以使用静态 File 类，在这种情况下静态调用速度要快一些，因为.NET 框架不必执行实例化新对象并调用其方法的过程。如果要在文件上执行几种操作，则实例化 FileInfo 对象使用其方法就更好一些，这样会提高效率，因为对象将在文件系统上引用正确的文件，而静态类就必须每次都寻找文件。

FileInfo 类的常用属性及说明如表 4.3 所示。

表 4.3　　FileInfo 类的常用属性及说明

属　性	说　明
CreationTime	获取或设置当前 FileSystemInfo 对象的创建时间
Directory	获取父目录的实例
DirectoryName	获取表示目录的完整路径的字符串
Exists	获取指示文件是否存在的值
Extension	获取表示文件扩展名部分的字符串
FullName	获取目录或文件的完整目录
IsReadOnly	获取或设置确定当前文件是否为只读的值
LastAccessTime	获取或设置上次访问当前文件或目录的时间
LastWriteTime	获取或设置上次写入当前文件或目录的时间
Length	获取当前文件的大小
Name	获取文件名

3. 创建、移动、删除文件夹

在网络硬盘中使用 Directory 用于创建、删除目录和子目录，这里介绍本模块中用到的该类中的方法，如表 4.4 所示。

表 4.4　　Directory 类的常用方法及说明

方　法	说　明
CreateDirectory	创建指定路径中的所有目录
Delete	删除指定的目录
Exists	确定给定路径是否引用磁盘上的现有目录
GetFiles	返回指定目录中的文件的名称
GetFileSystemEntries	返回指定目录中所有文件和子目录的名称
GetParent	检索指定路径的父目录，包括绝对路径和相对路径
Move	将文件或目录及其内容移到新位置

4. DirectoryInfo 类

DirectoryInfo 类和 Directory 类之间的关系与 FileInfo 类和 File 类之间的关系十分类似，这里不再赘述。下面介绍 DirectoryInfo 类的常用属性。

DirectoryInfo 类的常用属性及说明如表 4.5 所示。

表 4.5　　DirectoryInfo 类的常用属性及说明

属　性	说　明
CreationTime	获取或设置当前 FileSystemInfo 对象的创建时间
Exists	获取指示目录是否存在的值
FullName	获取目录或文件的完整目录
LastAccessTime	获取或设置上次访问当前文件或目录的时间
LastWriteTime	获取或设置上次写入当前文件或目录的时间
Name	获取 DirectoryInfo 实例的名称
Parent	获取指定子目录的父目录
Root	获取路径的根部分

4.2.2　GridView 控件数据绑定

在网络硬盘中用到了大量的数据绑定技术，例如，将查找到的信息显示到 GridView 控件中，下面详细地进行介绍。

数据绑定语法是一种非常灵活的语法。它不仅允许开发人员绑定到数据源，而且还可以绑定到简单属性、表达式、集合，甚至可以从方法调用返回的结果。

数据绑定不仅仅可以直接输出一般的输出语句，也可以输出数据类型，还可以输出任何一种符合数据绑定要求的数据源。并且，对于不同的数据显示控件，对数据源的处理可以不一致，也可以根据情况的改变而动态地发生变化。也就是说，在数据绑定中，数据的显示是由数据源和数据显示控件共同决定的。数据源决定数据的内容，数据显示控件决定数据的显示方式。实际上，数据绑定的作用机制就是由数据显示控件调用数据源的方法得到数据。

数据绑定的语法如下：

```
<语言标记 ...属性='<% 数据绑定表达式 %>' runat="server">
```

如果表达式的结果直接输出到网页上，那么数据绑定的语法如下：

字符串：<% 数据绑定表达式 %>

技巧

所有的数据绑定表达式都必须包含在<%...%>代码块中。位于<%...%>代码块中的代码只有在父控件容器中的 DataBind 方法被调用时才会被执行。DataBind()是页面和所有服务器控件的方法。当在父控件中调用该方法时，它将连接到控件的所有子控件上。DataBind()通常由 Page_Load 事件调用。

下面举个示例说明如何将数据绑定到控件当中，执行程序，运行效果如图 4.2 所示。

本示例实现的主要步骤如下。

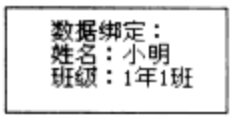


图 4.2　数据绑定

（1）新建 1 个网站，默认主页为 Default.aspx。在 Default.aspx 页的后台代码文件中定义 2 个公共属性，这 2 个属性作为数据绑定时的数据源。具体代码如下：

```
public string name
{
    get
    {
        return "小明";
    }
}
public string classname
{
    get
    {
        return "1年1班";
    }
}
```

（2）设置完数据绑定中的数据源，现在就可以将它与显示控件之间建立绑定关系了。将视图切控到“源”视图，具体代码如下：

```
<body>
    <form id="form1" runat="server">
    <div>
        数据绑定<br />
        姓名：<%# name %><br />
        班级名称：<%# classname %></div>
    </form>
</body>
```

（3）绑定完成后，只需在页面的 Page_Load 事件中调用 Page 类的 DataBind 方法来实现在页面加载时读取数据即可，具体代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    Page.DataBind();
}
```

4.2.3　统一控件的样式使用主题

1. 主题的概述

网站的外观是否美观将直接决定它受欢迎的程度。这就意味着在网站开发过程中，设计和实现美观实用的用户界面的重要性。在 ASP.NET 2.0 版本出现之前，网站开发人员经常使用级联样式表(CSS)来实现。而现在，在 ASP.NET 2.0 版本之后提供了一种称为“主题”的新功能。主题是定义网站中页和控件外观的属性集合。主题可以包括外观文件（定义 ASP.NET Web 服务器控件的属性设置），还可以包括级联样式表文件（.css 文件）和图形。通过应用主题，可以

为网站中的页提供一致的外观。

2. 组成元素

主题由外观、级联样式表（CSS）、图像和其他资源组成。主题中至少包含外观。它是在网站或 Web 服务器上的特殊目录中定义的。

- 外观

外观文件是主题的核心内容，用于定义页面中服务器控件的外观。它包含各个控件（例如，Button 按钮、TextBox 控件 或 Calendar 控件）的属性设置。控件外观设置类似于控件标记本身，但只包含您要作为主题的一部分来设置的属性。

例如，下面定义了 TextBox 控件的外观代码，分别设置了该控件的背景颜色 BackColor 和前景颜色 ForeColor，实现的代码如下：

```
<asp:TextBox runat="server" BackColor="PowderBlue" ForeColor="RoyalBlue"/>
```

控件外观的设置与控件声明代码类似。在控件外观设置中只能包含作为主题的属性定义。上述代码中设置了 TextBox 控件的前景色和背景色属性。如果将以上控件外观应用到单个 Web 页上，那么页面内所有 TextBox 控件都将显示所设置的控件外观。

- 级联样式表（CSS）

主题还可以包含级联样式表（.css 文件）。将.css 文件放在主题目录中时，样式表自动作为主题的一部分应用。使用文件扩展名.css 在主题文件夹中定义样式表。主题中可以包含一个或多个级联样式表。

- 图像和其他资源

主题还可以包含图形和其他资源，例如，脚本文件或视频文件等。通常，主题的资源文件与该主题的外观文件位于同一个文件夹中，但也可以在 Web 应用程序中的其他地方。

3. 文件存储和组织方式

在 Web 应用程序中，主题文件必须存储在根目录的 App_Themes 文件夹下（除全局主题之外），开发人员可以手动或者使用 Visual Studio 2008 在网站的根目录下创建该文件夹。App_Themes 文件夹的示意图如图 4.3 所示。

在 App_Themes 文件夹中包括“主题 1”和“主题 2”两个文件夹。每个主题文件夹中都可以包含外观文件、CSS 文件和图像文件等。通常 APP_Themes 方件夹中，只存储主题文件及与主题有关的文件，尽量不存储其他类型文件。

外观文件是主题的核心部分，每个主题文件夹下都可以包含一个或者多个外观文件，如果主题较多，页面内容较复杂时，外观文件的组织就会出现问题。这样就需要开发人员在开发过程中，根据实际情况对外观文件进行有效的管理。通常根据 SkinID、控件类型及文件 3 种方式组织。具体说明如表 4.6 所示。

表 4.6　　3 种常见的外观文件的组织方式及其说明

组 织 方 式	说　　明
根据 SkinID	在对控件外观设置时，将具有相同的 SkinID 放在同一个外观文件中，这种方式适用于网站页面较多、设置内容复杂的情况
根据控件类型	组织外观文件时，以控件类型进行分类，这种方式适用于页面中包含控件较少的情况
根据文件	组织外观文件时，以网站中的页面进行分类，这种方式适用于网站中页面较少的情况

在网络硬盘程序中，主题文件必须存储在根目录的 App_Themes 文件夹下，如图 4.4 所示。App_Themes 文件夹的示意图。

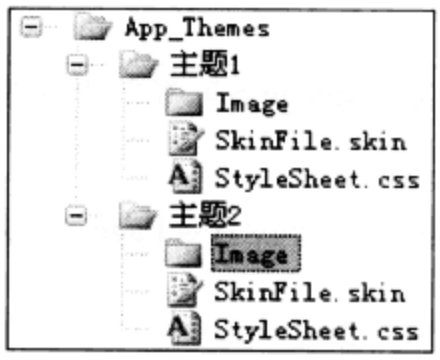


图 4.3　App_Themes 文件夹的示意图

图 4.4　App_Themes 文件夹的示意图

在 web.skin 文件中，编写的代码如下所示：

```
<%-- Button控件的外观 --%>
<asp:Button runat="server" SkinID="btnSkin" BackColor="#ECFAD7" Font-Names="Tahoma" Font-Size="9pt" CssClass="Button" />
<%-- TextBox控件的外观 --%>
<asp:TextBox runat="server" SkinID="tbSkin" BackColor="#ECFAD7" Font-Names="Tahoma" />
<%-- DropDownList控件的外观 --%>
<asp:DropDownList SkinID="ddlSkin" runat="server"
    BackColor="#ECFAD7" Font-Names="Tahoma" Font-Size="9pt" />
```

4. 为单个页面指定和禁用主题

为单个页面指定主题可以将@ Page 指令的 Theme 或 StyleSheetTheme 属性设置为要使用的主题的名称。代码如下：

```
<%@ Page Theme="ThemeName" %>
```

或：

```
<%@ Page StyleSheetTheme="ThemeName" %>
```

StyleSheetTheme 属性的工作方式与普通主题（使用 Theme 设置的主题）类似。不同的是当使用 StyleSheetTheme 时，控件外观的设置可以被页面中声明的同一类型控件的相同属性所代替。例如，如果使用 Theme 属性指定主题，该主题指定所有的 Button 控件的背景都是黄色。那么即使在页面中 Button 控件的背景设置了不同颜色，页面中的所有 Button 控件的背景仍然是黄色。如果需要改变个别 Button 控件的背景，就需要使用 StyleSheetTheme 属性指定主题。

禁用单个页面的主题，只要将@Page 指令的 EnableTheming 属性设置为 False 即可。代码如下：

```
<%@ Page EnableTheming="false" %>
```

如果想要禁用控件的主题，只要将控件的 EnableTheming 属性设置为 False 即可。以 Button 控件为例，代码如下：

```
<asp:Button id="Button1" runat="server" EnableTheming="false" />
```

5. 为应用程序指定和禁用主题

为了能够快速地为整个网站的页面设置相同的主题，可以设置 Web.config 文件中的<pages>配置节的内容。Web.config 文件的配置代码如下：

```
<configuration>
  <system.web >
    <pages theme ="ThemeName"></pages>
  </system.web>
<connectionStrings/>
```

或：

```
<configuration>
  <system.web >
    <pages StylesheetTheme=" ThemeName "></pages>
  </system.web>
<connectionStrings/>
```

说 明

禁用整个应用程序的主题设置，只要将<pages>配置节中的 Theme 属性或者 StylesheetTheme 属性值设置为空（""）即可。

在网络硬盘程序中将一个主题与一个网页相关联，只要设置@Page 指令中的 Theme 或 StyleSheetTheme 属性即可，代码如下：

```
<%@ Page Language="C#"   StylesheetTheme="MRSOFTASPNET" %>
```

一旦指定了使用哪种主题形式，即在页面的头部设置 StyleSheetTheme 属性后，整个页面内容都将自动应用所设置的主题，并将呈现所设置的外观。

上面已经指定了应该使用哪种主题，下面以程序中的 Button 按钮为例。在 Button 按钮的属性中有个 SkinID 属性，在 SkinID 属性旁边的下拉列表中选择“btnSkin”选项，如图 4.5 所示，这是将在 Button 按钮中指定了主题，使用主题后的效果如图 4.6 所示。

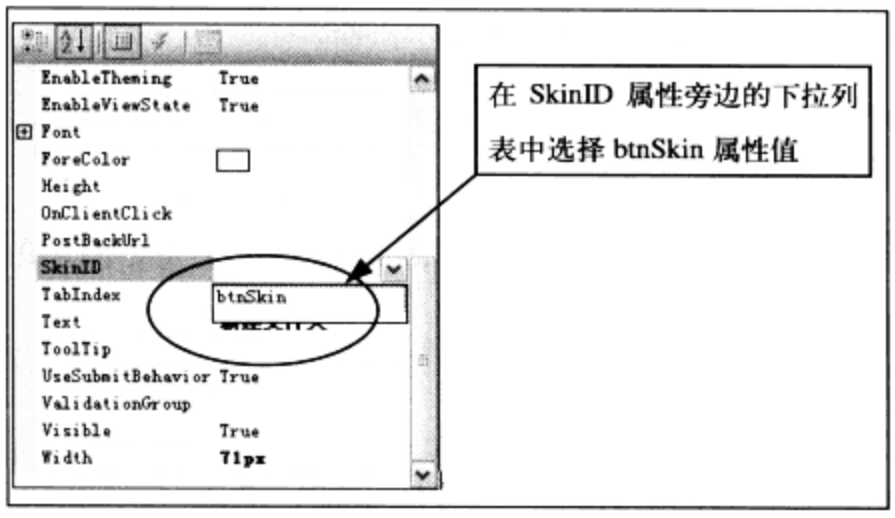


图 4.5　在 Button 按钮中指定主题

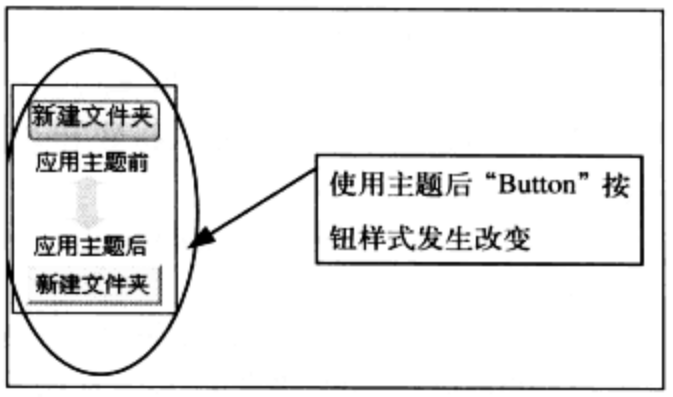


图 4.6　在 Button 按钮中应用主题前后运行效果图

4.3　网络硬盘实现过程

4.3.1　选择不同的文件夹进行文件上传

文件上传页面（Up_WJ.aspx）是用于将选定的文件上传到指定的文件夹中，同时将上传的文件保存到数据库中。该页运行效果如图 4.7 所示。

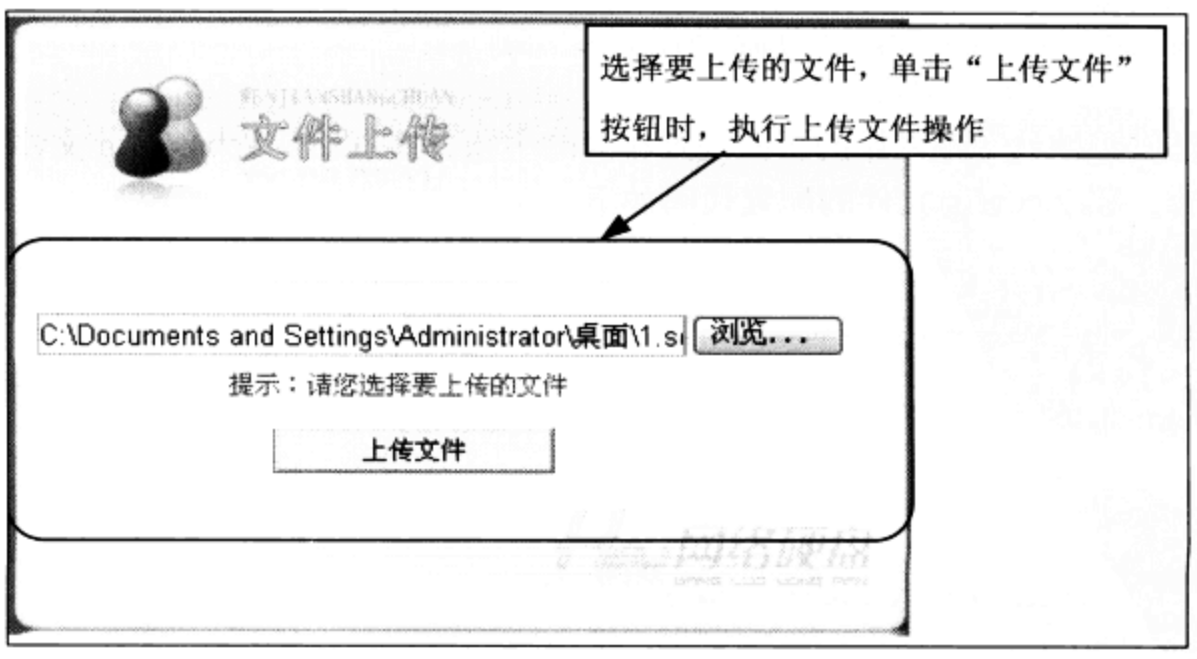


图 4.7　运行效果图

控件的属性设置可以通过前台代码实现，也可以在“控件属性”对话框中设置实现，本页中使用的控件属性设置请参见如下代码。

例程 1 代码位置：光盘\mr\04\NetWorkHD\Default.aspx

```
<table border="0" cellpadding="0" cellspacing="0" style="background-image: url(Images/wjsc.bmp);
                                width: 408px; height: 293px">
                                <tr>
                                    <td style=" width: 105px; text-align: center">
                                        <div style="text-align: center">
                                            <div style="text-align: center">
                                                <table cellpadding="0" cellspacing="0" style="width: 192px; height: 160px">
                                                    <tr>
                                                        <td align="left" style="width: 100px; height: 42px; text-align: left">
                                                        </td>
                                                    </tr>
                                                <tr>
                                                    <td align="left" style="width: 100px; height: 30px; text-align: left">
                            <asp:Button Runat="server"    Text="上传文件" ID="SureBtn" OnClick="SureBtn_Click"
Width="128px" SkinID="btnSkin"></asp:Button></td>
                                                </tr>
                                                <tr>
                                                    <td align="left" style="width: 100px; height: 37px; text-align: left">
                                                        <asp:Label ID="Label1" runat="server" Font-Bold="False" Font-Size=
"9pt" ForeColor="Red"
                                                            Height="20px" Width="201px">提示：请您选择要上传的文件
</asp:Label></td>
                                                </tr>
                                                <tr>
                                                    <td align="left" style="width: 100px; height: 30px; text-align: center">
                                                    </td>
                                                </tr>
                                                <tr>
                                                    <td style="width: 100px; height: 30px">
                                                        <table id="F" runat="server" cellpadding="0" cellspacing="0"
enableviewstate="true">
                                                            <tr>
                                                                <td style="width: 100px; height: 30px">
                                                                <input type="file" name="File" style="width: 330px" id=
"File1" language="javascript" onclick="return File1_onclick()" /></td>
                                                            </tr>
                                                        </table>
                                                    </td>
                                                </tr>
                                                <tr>
                                                    <td style="width: 100px; height: 21px">
                                </td>
                                                </tr>
                                            </table>
                                        </div>
                                    </div>
                                </td>
                            </tr>
                        </table>
```

1. 前台页面设计

(1) 在应用程序中创建 1 个 Web 窗体，命名为 Up_WJ.aspx，作为上传文件页。Up_WJ.aspx 的设计界面如图 4.8 所示。

图 4.8　页面 Up_WJ.aspx 的设计界面图

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Button 控件、1 个 Label 控件和一个 Input(File)控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 4.7 所示。

表 4.7　　选择不同的文件夹进行文件上传页面中控件的说明

控 件 类 型	控 件 名 称	主要属性设置	用　　途
Table	无	无	页面布局
Input (File)	控件的名称为 file1	将控件的 Name 属性设置为 file	用于浏览要上传的文件
A Label	控件的名称为 lbltishi	将控件的Text属性设置为“提示：请您选择要上传的文件”	用于显示提示信息
Button	将控件的名称设置为 BtnUp	将控件的 SkinID 属性设置为 btnSkin	用于执行上传文件操作

说　明

由于 Input (File) 控件是 HTML 控件，所以不能在后台中获取该控件的属性和方法。如果需要编写后台代码，需要将 Input (File) 控件转换成服务器控件，在该控件中单击鼠标右键从下拉列表框中选择“作为服务器控件运行”选项，如图 4.9 所示。

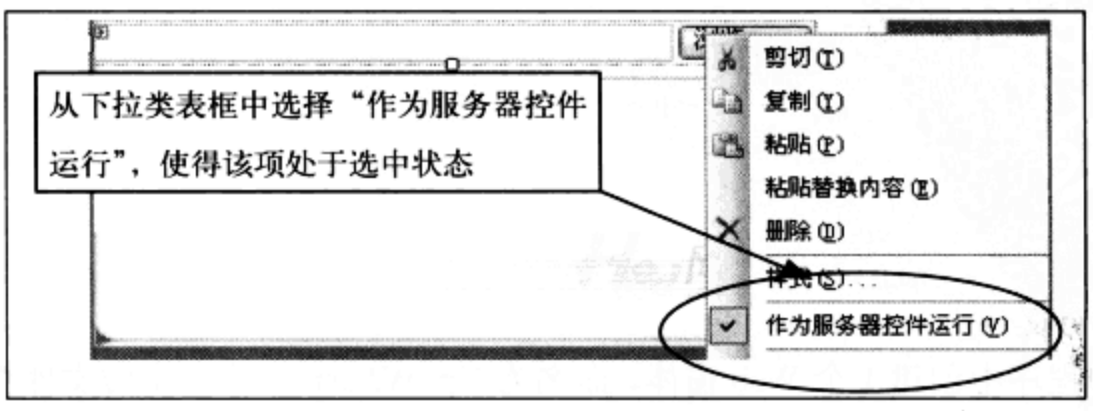


图 4.9　使 Html 控件作为服务器控件运行

2. 后台代码实现

在命名空间区域中，首先引入一个命名空间，实现的代码如下。

例程 2 代码位置：光盘\mr\04\NetWorkHD\Up_WJ.aspx.cs

```
//引入命名空间
using System.Text;
using System.IO;
```

在 Page_Load 事件外声明一个 WebHard 类的对象 directory，目的是调用类中的方法，实现的代码如下。

例程 3 代码位置：光盘\mr\04\NetWorkHD\Up_WJ.aspx.cs

```
WebHard directory = new WebHard();
```

在 Page_Load 页加载事件中，验证用户是否登录，如果未登录，弹出提示信息并跳转到“tishi.aspx”提示信息页面，否则将变量 ID 的值传递到该页面中，实现的代码如下。

例程 4 代码位置：光盘\mr\04\NetWorkHD\Up_WJ.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["UserName"] == null && Session["UserPwd"] == null)
    {
        Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
    }
    else
    {
        if (Request.Params["id"] != null)
        {
            if (Int32.TryParse(Request.Params["id"].ToString(), out ID1) == false)
            {
                return;
            }
        }
        this.btn();
    }
    }
```

在用户自定义的 Up 方法中，可实现将上传文件保存到指定的文件夹中，并且将上传的文件信息保存到数据库中，实现的代码如下。

例程 5 代码位置：光盘\mr\04\NetWorkHD\Up_WJ.aspx.cs

```
public void Up(int id)
 {
  HttpFileCollection UpLoad = HttpContext.Current.Request.Files;
  for (int i = 0; i < UpLoad.Count; i++)
  {
    HttpPostedFile file = UpLoad[i];
    string fileName = Path.GetFileName(file.FileName);
    if (fileName != null)
        {
          file.SaveAs(MapPath("mr/") + fileName);
          directory.AddFile(fileName, id, file.ContentLength, "mr/" + fileName, file.ContentType);
        }
    }
    Response.Write("<script>alert('恭喜您！ 文件上传成功！ ');location='Default.aspx'</script>");
  }
```

单击“上传文件”按钮，触发 Click 事件，在该事件中判断 File1 控件中是否有要上传的文件，如果没有要上传的文件将弹出“很遗憾，上传文件不能为空!”的字样，否则调用自定义的 Up 方法将文件上传到指定的位置中，并且将上传的文件信息保存到数据库中，实现的代码如下。

例程 6 代码位置：光盘\mr\04\NetWorkHD\Up_WJ.aspx.cs

```
protected void BtnUp_Click(object sender, EventArgs e)
{
```

```
        if (this.File1.PostedFile.FileName=="")
        {
            Response.Write("<script>alert('很遗憾，上传文件不能为空！')</script>");
        }
        else
        {
            Up(ID1);
        }
    }
```

4.3.2 修改文件名称

修改文件名称页是在 Update_Wj.aspx 页面中实现的，在该页面中可实现修改文件的名称，同时将修改后的文件的信息保存到数据库中。修改文件名称页运行结果如图 4.10 所示。

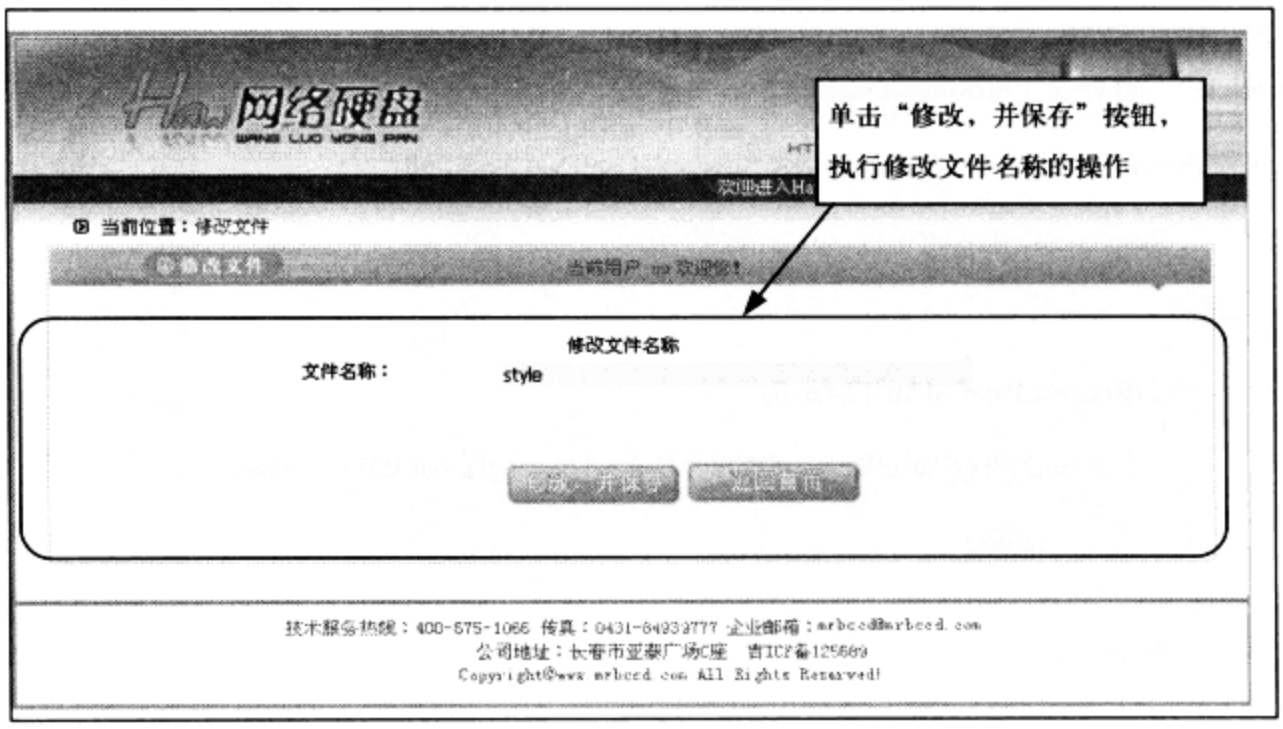


图 4.10 修改文件夹名称

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 Update_Wj.aspx，作为修改文件夹名称页。页面 Update_Wj.aspx 的设计界面如图 4.11 所示。

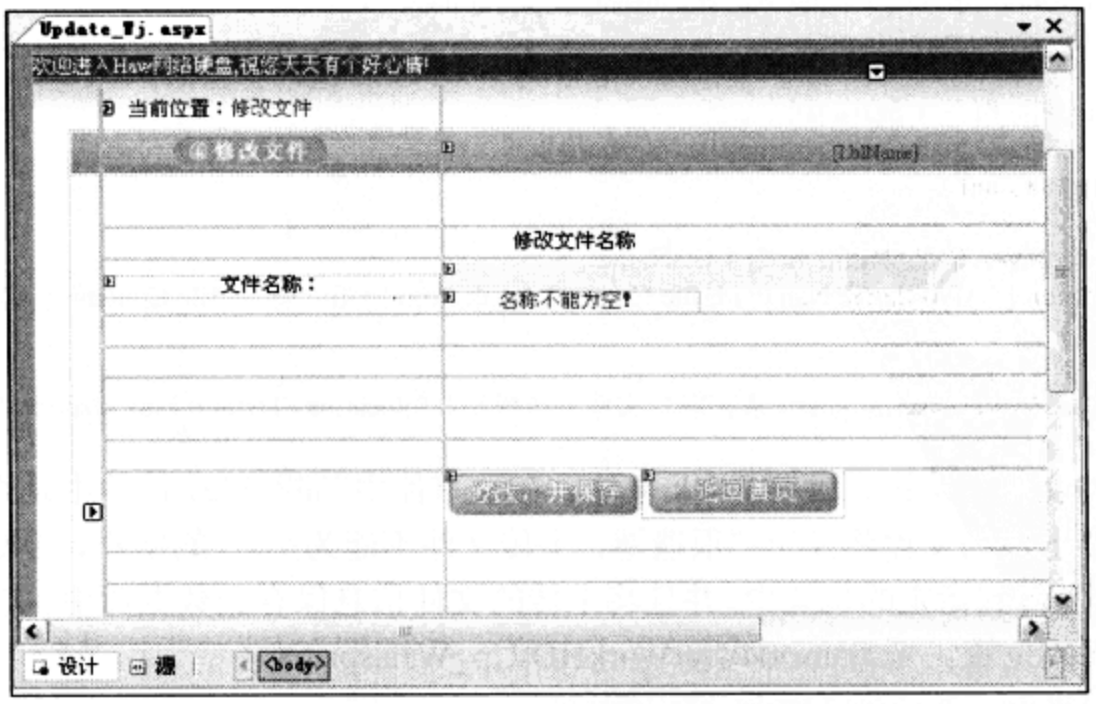


图 4.11 Update_WJ.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放两个 ImageButton 控件和 1 个 TextBox 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 4.8 所示。

表 4.8　　修改文件名称页面中控件的说明

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
ImageButton	控件的名称分别设置为 ImgBtnUpdate 和 ImgBtnSY	分别将两个控件的 ImageUrl 属性设置为“~/Images/update.JPG”和“~/Images/shouye.JPG”	这两个控件分别用于执行“修改文件名称”和将页面跳转到网站的“Default.aspx”首页当中
TextBox	控件的名称为 TxtName	将控件的 SkinID 属性设置为“tbSkin”，目的是引用主题	输入要修改的文件的名称

2. 后台代码实现

在命名空间区域中，首先引入命名空间，实现的代码如下。

例程 7　代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

实例化 WebHard 类的一个 directory 变量，目的是用来调用类中方法，代码如下。

例程 8　代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
WebHard directory = new WebHard();
```

声明一个整数类型的变量 dir 和整数类型的变量 mrid，实现的代码如下。

例程 9　代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
public int dir;
public int mrid ;
```

在 Page_Load 页加载事件中，验证用户是否登录，如果未登录，弹出提示信息并跳转到“tishi.aspx”提示信息页面，否则将变量值传递到该页面中，实现的代码如下。

例程 10　代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["UserName"] == null && Session["UserPwd"] == null)
    {
        Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
    }
    else
    {
        this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
        if (Request.Params["id"] != null)
        {
            if (Int32.TryParse(Request.Params["id"].ToString(), out dir) == false)
            {
                return;
            }
        }
        if (Request.Params["mrid"] != null)
        {
            if (Int32.TryParse(Request.Params["mrid"].ToString(), out mrid) == false)
            {
                return;
            }
        }
```

```
            if (!Page.IsPostBack)
            {
                if (dir > -1)
                {
                    xianshi(dir);
                }
            }
        }
    }
```

在用户自定义的 xianshi 方法中，使用 WebHard 类中的 StrSelect 方法，实现的是查找文件的信息，然后使用 SqlDataReader 方法将文件的信息显示到相应的文本框中，实现的代码如下。

例程 11 代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
public void xianshi(int nid)
{
    SqlDataReader dr = directory.StrSelect(dir);
    if (dr.Read())
    {
        TxtName.Text = dr["mrming"].ToString();
        TxtDaxiao.Text = dr["size"].ToString() + "B";
        TxtTime.Text = dr["time"].ToString();
        TxtDirectory.Text = Insert(Convert.ToInt32(dr["id"].ToString()));
    }
    dr.Close();
}
```

在用户自定义的 Insert 方法中，实现的是按照编号查找修改文件的信息，实现的代码如下。

例程 12 代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
public string Insert(int nid)
{
StringBuilder str = new StringBuilder();
DataTable dt = WebHard.dt(directory.strsql4());
DataRow[] dr = dataTable.Select("id='" + nid.ToString() + "'");
if (rowList.Length != 1) return ("");
Insert1(dataTable, Convert.ToInt32(rowList[0]["mrid"].ToString()), dirSB);
return (dirSB.ToString());
}
```

在用户自定义的 Insert1 方法中，实现的是查找相应文件的信息，实现的代码如下。

例程 13 代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
public void Insert1(DataTable dt, int mrid, StringBuilder S)
{
if (mrid <= -1)
  {
      return;
  }
DataRow[] dr = dt.Select("id='" + mrid.ToString() + "'");
if (rowList.Length != 1) return;
S.Insert(0, rowList[0]["mrming"].ToString() + "/");
Insert1(dt, Convert.ToInt32(rowList[0]["mrid"].ToString()), S);
}
```

单击“返回首页”按钮，触发 Click 事件，在该事件中使用 Response 的 Redirect 方法，将页面跳转到“Default.aspx”网站首页当中，实现的代码如下。

例程 14 代码位置：光盘\mr\04\NetWorkHD\Update_Wj.aspx.cs

```
protected void ImgBtnSY_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Default.aspx?mrid=" + mrid.ToString());
}
```

4.3.3 获取指定文件的基本信息

获取指定文件的基本信息页是在 Information.aspx 页面中实现的，在该页面中将显示“文件的名称”、“文件所在位置”、“文件创建时间”和“文件现在的大小”。修改文件名称页运行结果如图 4.12 所示。

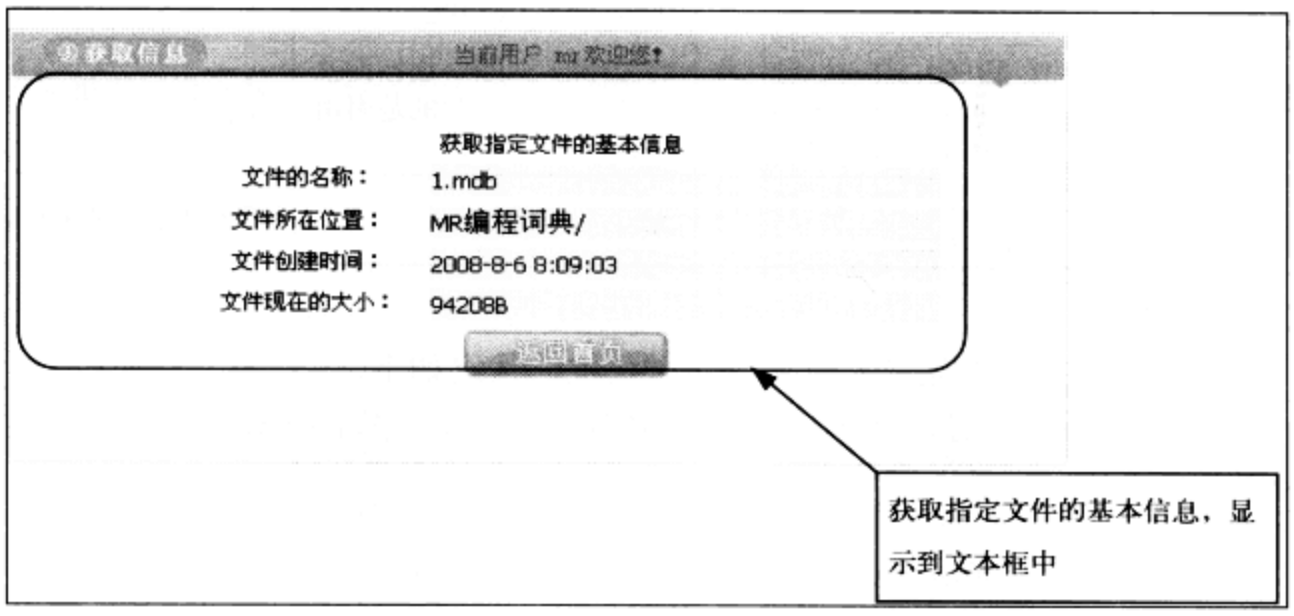


图 4.12 修改文件夹名称

1. 前台页面设计

(1) 在应用程序中创建 1 个 Web 窗体，命名为 Information.aspx，作为获取指定文件的基本信息页。页面 Information.aspx 的设计界面如图 4.13 所示。

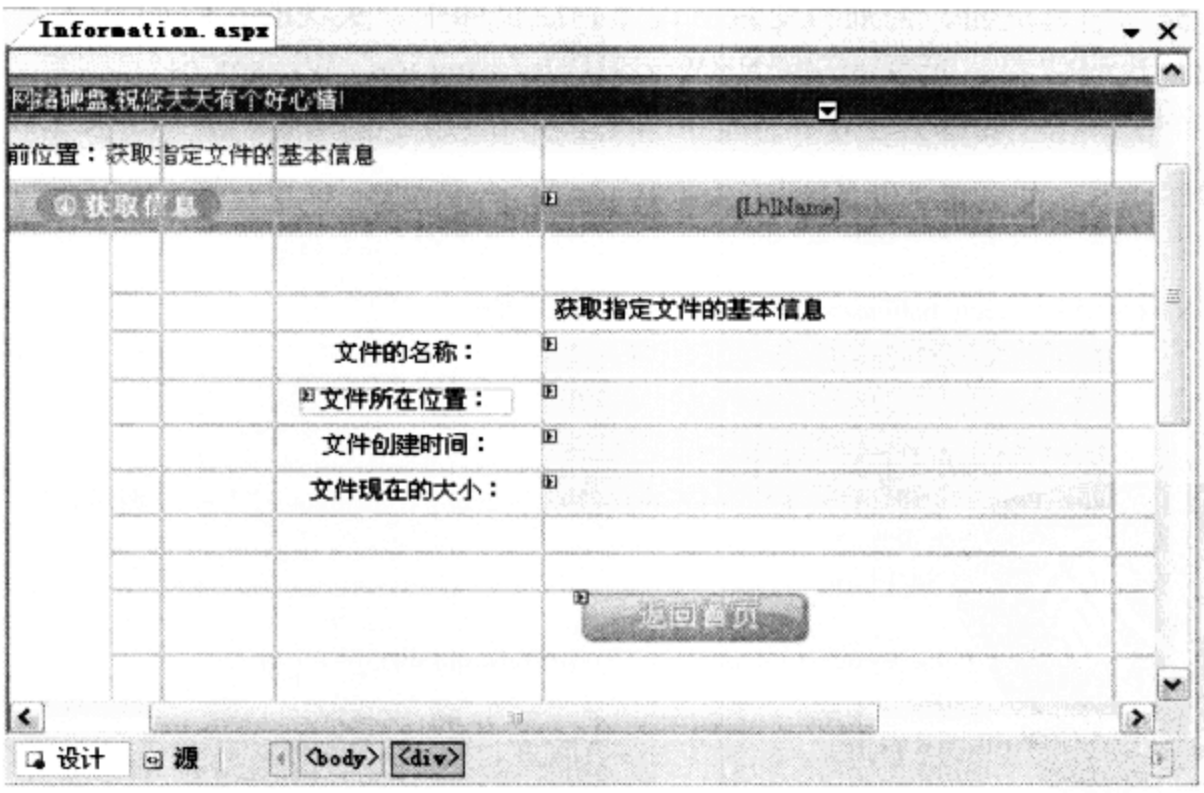


图 4.13 Information.aspx 的设计界面

(2) 在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 ImageButton 控件和 4 个 TextBox 控件通过属性窗口和 1 个 Label 控件，设置控件的属性。页面中各个控件的属性设置及其用途如表 4.9 所示。

表 4.9　　修改文件名称页面中控件的说明

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
ImageButton	控件的名称分别设置为 ImgBtnUpdate 和 ImgBtnSY	分别将两个控件的 ImageUrl 属性设置为“~/Images/update.JPG”和“~/Images/shouye.JPG”	这两个控件分别用于执行“修改文件名称”和将页面跳转到网站的“Default.aspx”首页当中
TextBox	控件的名称为 TxtName、TxtDirectory、TxtTime 和 TxtDaxiao	将控件的 SkinID 属性设置为“tbSkin”，目的是引用主题	显示获取的指定文件的基本信息
Label	控件的名称为 LblName	无	用于显示登录用户名的信息

2．后台代码实现

在命名空间区域中， 首先引入命名空间，实现的代码如下。

例程 15　代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

实例化 WebHard 类的一个的 directory 变量，目的是用来调用类中方法，代码如下。

例程 16　代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
WebHard directory = new WebHard();
```

定义 2 个 int 类型的量“dir”和“mrid”，实现的代码如下。

例程 17　代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
//定义两个变量
public int dir;
public int mrid ;
```

在 Page_Load 页加载事件中，验证用户是否登录，如果未登录，弹出提示信息并跳转到“tishi.aspx”提示信息页面，否则将变量值传递到该页面中，实现的代码如下。

例程 18　代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //判断用户是否登录，如果没有登录将页面跳转到tishi.aspx页面中
    if (Session["UserName"] == null && Session["UserPwd"] == null)
    {
        Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
    }
    else
    {
        //将登录用户的用户名称显示出来
      this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
        //判断传来的ID值是否为空
        if (Request.Params["id"] != null)
        {
            if (Int32.TryParse(Request.Params["id"].ToString(), out dir) == false)
            {
                //return;
            }
        }
        if (Request.Params["mrid"] != null)
        {
            if (Int32.TryParse(Request.Params["mrid"].ToString(), out mrid) == false)
            {
                //return;
            }
```

```
            }
            if (!Page.IsPostBack)
            {
                if (dir > -1)
                {
                    xianshi(dir);
                }
            }
        }
    }
```

在用户自定义的 xianshi 方法中，使用 WebHard 类中的 StrSelect 方法，实现的是查找文件的信息，然后使用 SqlDataReader 方法将文件的信息显示到相应的文本框中，实现的代码如下。

例程 19 代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
#region 将基本信息按照编号查找出来，并且将查找出来的信息显示到相应的文本框中
/// <summary>
/// 将基本信息按照编号查找出来，并且将查找出来的信息显示到相应的文本框中
/// 返回值：返回整数类型
/// 参数：nid 按照编号来查找要获取的文件的信息
/// </summary>
/// <param name="nid"></param>
public void xianshi(int nid)
{
    SqlDataReader dr = directory.StrSelect(dir);
    if (dr.Read())
    {
        TxtName.Text = dr["mrming"].ToString();
        TxtDaxiao.Text = dr["size"].ToString() + "B";
        TxtTime.Text = dr["time"].ToString();
        TxtDirectory.Text = Insert(Convert.ToInt32(dr["id"].ToString()));
    }
    dr.Close();
}
#endregion
```

在用户自定义的 Insert 方法中，实现的是按照编号查找文件的信息，实现的代码如下。

例程 20 代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
#region 按照编号来查找文件的信息
    /// <summary>
    ///按照编号来查找文件的信息
    /// 返回值：返回字符串类型
    /// 参数：nid 按照编号来查找要获取的文件的信息
    /// </summary>
    /// <param name="nid"></param>
    /// <returns></returns>
    public string Insert(int nid)
    {
        StringBuilder str = new StringBuilder();//定义并且实例化1个StringBuilder类型的变量str
        DataTable dt = WebHard.dt(directory.strsql4());
        DataRow[] dr = dt.Select("id='" + nid.ToString() + "'");//按照id查找文件
        if (dr.Length != 1)
        return ("");//返回一个空字符串
        Insert1(dt, Convert.ToInt32(dr[0]["mrid"].ToString()), str);
        return (str.ToString());//返回一个str字符串
    }
    #endregion
```

在用户自定义的 Insert1 方法中，实现的是查找相应文件的信息，实现的代码如下。

例程 21 代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
    #region
     /// <summary>
```

```
/// 按照编号来查找文件的信息
/// 返回值：返回空类型
/// 参数：nid 按照编号来查找要获取的文件的信息
/// </summary>
/// <param name="dt"></param>
/// <param name="mrid"></param>
/// <param name="S"></param>
public void Insert1(DataTable dt, int mrid, StringBuilder S)
{
    if (mrid <= -1)
    {
        return;
    }
    DataRow[] Dr = dt.Select("id='" + mrid.ToString() + "'");
    if (Dr.Length != 1) return;
    S.Insert(0, Dr[0]["mrming"].ToString() + "/");
    Insert1(dt, Convert.ToInt32(Dr[0]["mrid"].ToString()), S);
}
#endregion }
```

单击“返回首页”按钮，触发其 Click 事件，在该事件中使用 Response 的 Redirect 方法，将页面跳转到“Default.aspx”网站首页当中，实现的代码如下。

例程 22 代码位置：光盘\mr\04\NetWorkHD\Information _Wj.aspx.cs

```
#region 将页面跳转到Default.aspx网站页面中
/// <summary>
/// 将页面跳转到Default.aspx网站页面中
/// 返回值：返回空类型
/// 代码中的传递参数 mrid 将文件编号传递给Default.aspx网站页面中
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
protected void ImgBtnSY_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Default.aspx?mrid=" + mrid.ToString());
}
#endregion
```

4.3.4 修改文件夹名称

修改文件夹名称是在“Update_WJJ.aspx”页面中实现的，在文本框中输入文件夹的名称，并且单击“修改，并保存”按钮，即可修改文件夹的名称，同时将修改后的数据信息保存到数据库中。该页运行结果如图 4.14 所示。

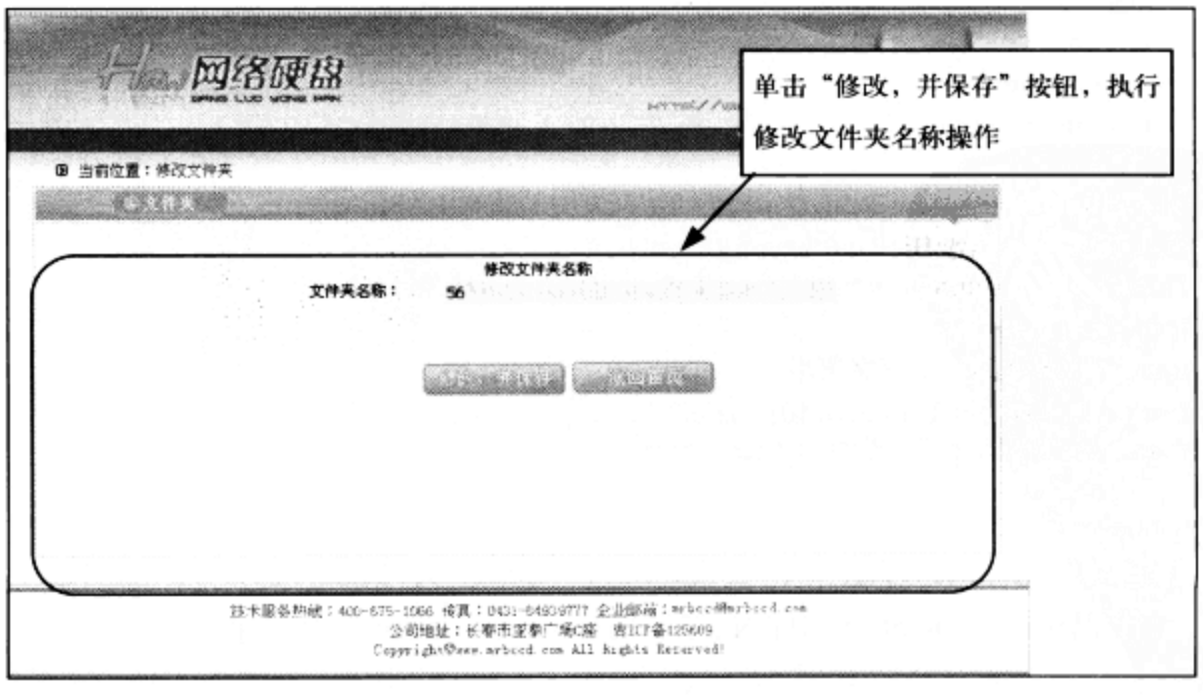


图 4.14 修改文件夹名称

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 Update_WJJ.aspx，作为修改文件夹名称页。页面 Update_WJJ.aspx 的设计界面如图 4.15 所示。

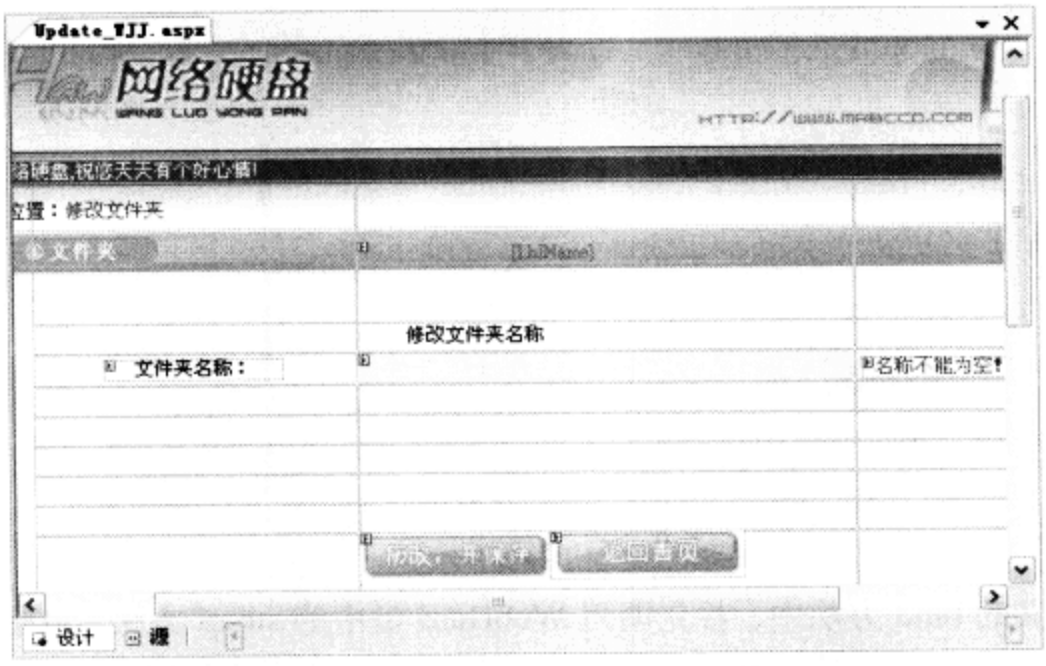


图 4.15　页面 Update_WJJ.aspx 的设计界面图

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 ImageButton 控件和 1 个 TextBox 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 4.10 所示。

表 4.10　　修改文件夹名称页面中控件的说明

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
ImageButton	控件的名称分别设置为 ImgBtnUpdate 和 ImgBtnSY	分别将两个控件的 ImageUrl 属性设置为～/Images/update.JPG 和～/Images/shouye.JPG	这两个控件分别用于执行“修改文件夹名称”和将页面跳转到网站的“Default.aspx”首页当中
TextBox	控件的名称为 TxtName	将控件的 SkinID 属性设置为 tbSkin，目的是引用主题	输入要修改的文件夹的名称

2. 后台代码实现

在命名空间区域中，首先引入命名空间，实现的代码如下。

例程 23　代码位置：光盘\mr\04\NetWorkHD\Update_WJJ.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

实例化 1 个 WebHard 类型的变量，目的是调用类中方法，实现的代码如下。

例程 24　代码位置：光盘\mr\04\NetWorkHD\Update_WJJ.aspx.cs

```
WebHard directory = new WebHard();
```

在 Page_Load 页装载事件中，验证用户是否登录，如果未登录，弹出提示信息并跳转到登录页面，否则在 LblName 控件中显示当前用户信息，并且在文本框中显示要修改的文件夹的名称，实现的代码如下。

例程 25　代码位置：光盘\mr\04\NetWorkHD\Update_WJJ.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["UserName"] == null && Session["UserPwd"] == null)
        {
```

```
            Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
        }
        else
        {
            if (Request.Params["id"] != null)
            {
                if (Int32.TryParse(Request.Params["id"].ToString(), out ID) == false)
                {
                    return;
                }
            }
            if (!Page.IsPostBack)
            {
                if (ID > -1)
                {
                    bind(ID);
                }
            }
        }
    }
```

在用户自定义的 bind 方法中，首先调用 WebHard 类中的 strsql 方法，可将查找到的数据读取出来，并且将读取到的数据显示到 TxtName 控件当中，实现的代码如下。

例程 26 代码位置：光盘\mr\04\NetWorkHD\Update_WJJ.aspx.cs

```
public void bind(int id)
{
SqlDataReader dr = directory.strsql(id);
    if(dr.Read())
    {
        TxtName.Text = dr["mrming"].ToString();
    }
    dr.Close();
}
```

单击“修改，并保存”按钮，触发其 Click 事件，在该事件中调用 WebHard 类中的 update 方法实现的是按照编号修改文件夹名称，并且使用 Response 对象的 Write 方法弹出一个“恭喜您！修改数据成功！！”字样的提示对话框，将页面跳转到“Default.aspx”网站首页当中，实现的代码如下。

例程 27 代码位置：光盘\mr\04\NetWorkHD\Update_WJJ.aspx.cs

```
protected void ImgBtnUpdate_Click(object sender, ImageClickEventArgs e)
{
    directory.update(ID, TxtName.Text.Trim());
    Response.Write("<script>alert('恭喜您！ 修改数据成功！！ ');</script>");
}
```

单击“返回首页”按钮，触发其 Click 事件，在该事件中使用 Response 的 Redirect 方法，将页面跳转到“Default.aspx”网站首页当中，实现的代码如下。

例程 28 代码位置：光盘\mr\04\NetWorkHD\Update_WJJ.aspx.cs

```
protected void ImgBtnSY_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Default.aspx");
}
```

4.3.5 添加文件夹到指定的目录中

添加文件夹到指定的目录 Insert_Wjj.aspx 页面，在该页面实现将文件夹添加到指定的指定的位置中，同时将新添加的文件夹的信息添加到数据库中。该页运行结果如图 4.16 所示。

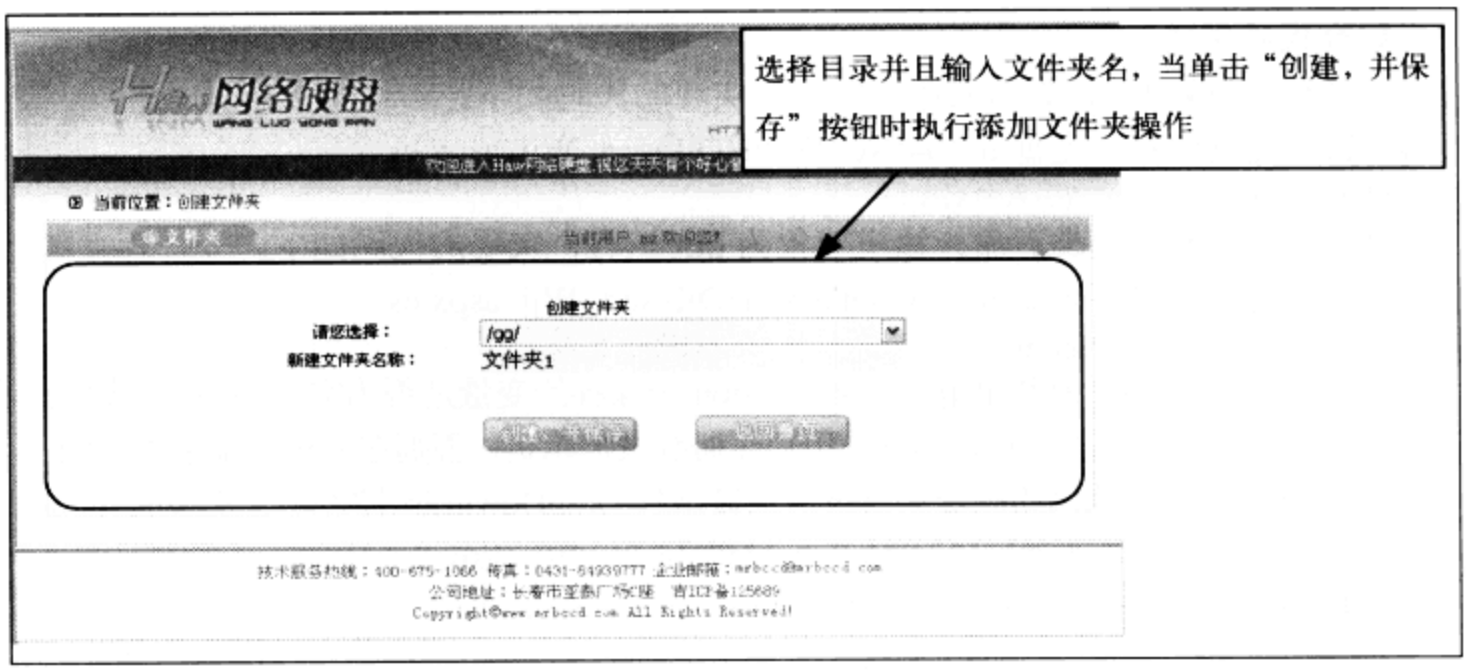


图 4.16　运行效果图

1. 前台页面设计

(1) 在应用程序中创建 1 个 Web 窗体，命名为 Insert_Wjj.aspx，实现将文件夹添加到指定的目录下的功能。页面 Insert_Wjj.aspx 的设计界面如图 4.17 所示。

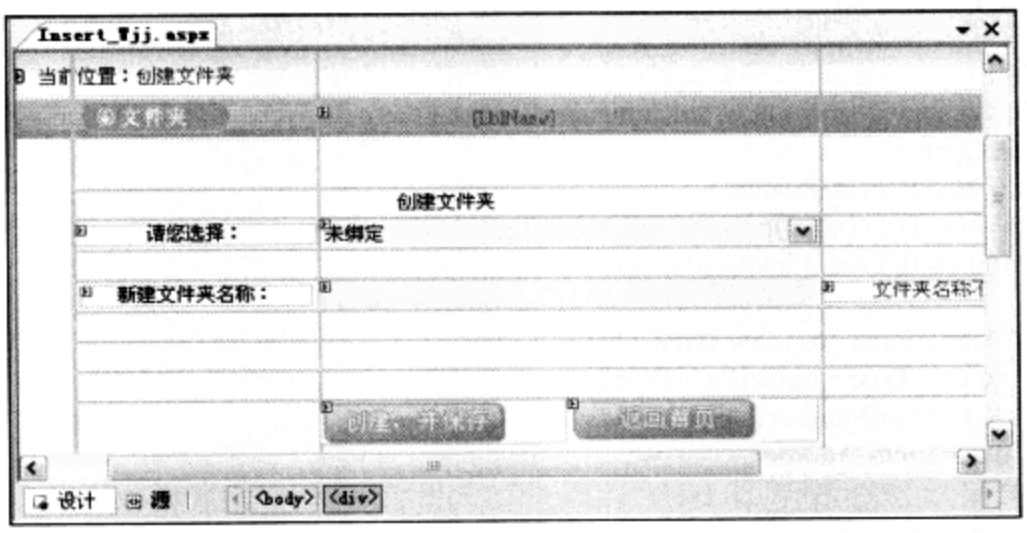


图 4.17　页面 Insert_Wjj.aspx 的设计界面

(2) 在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 ImageButton 控件，1 个 DropDownList 控件和 1 个 TextBox 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 4.11 所示。

表 4.11　　添加文件夹到指定的目录中控件的说明

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
ImageButton	将控件的名称分别设置为 ImgBtnCreate 和 ImgBtnSY	将控件的 ImageUrl 属性分别设置为～/Images/Create.JPG 和 ImageUrl=～/Images/shouye.JPG	这两个控件分别用于执行“创建文件夹”和将页面跳转到网站的“Default.aspx”首页当中
TextBox	控件的名称为 TxtName	将控件的 SkinID 属性设置为 tbSkin	用于输入要创建的文件夹的名称
DropDownList	控件的名称为 DDLdir	将控件的 SkinID 属性设置为 ddlSkin	用于绑定目录的名称

2. 后台代码实现

在命名空间区域中，首先引入命名空间，实现的代码如下。

例程 29 代码位置：光盘\mr\04\NetWorkHD\Insert_Wjj..aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

实例化 1 个 WebHard 类对象，该类对象为 directory，实现的代码如下。

例程 30 代码位置：光盘\mr\04\NetWorkHD\Insert_Wjj..aspx.cs

```
WebHard directory = new WebHard();
```

在 Page_Load 事件中，利用 if 语句判断 session 中保存的变量是否为空，如果为空说明该用户没有登录过，则将页面跳转到“tishi.aspx”用户注册登录提示页，否则在 LblName 控件中显示当前用户信息，并且将查找到的网络硬盘的目录信息显示到 DropDownList 控件中，实现的代码如下。

例程 31 代码位置：光盘\mr\04\NetWorkHD\Insert_Wjj..aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["UserName"] == null && Session["UserPwd"] == null)
    {
        Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
    }
    else
    {
        this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
        if (!Page.IsPostBack)
         {
            SqlConnection con = new SqlConnection(WebHard.GetConStr());
            SqlCommand cmd = new SqlCommand("select * from tb_Webhard where Mark='1'", con);
            con.Open();
            dr = cmd.ExecuteReader();
            dt = new DataTable();
           for (int i = 0; i < dr.FieldCount; i++)
           {
               DataColumn dc = new DataColumn();
               dc.DataType = dr.GetFieldType(i);
               dc.ColumnName = dr.GetName(i);
               dt.Columns.Add(dc);
           }
           while (dr.Read())
           {
               DataRow dr1 = dt.NewRow();
               for (int i = 0; i < dr.FieldCount; i++)
               {
                   dr1[i] = dr[i].ToString();
               }
               dt.Rows.Add(dr1);
               dr1 = null;
           }
           dr.Close();
           this.DDLdir.Items.Clear();
          DataRow[] drdir = dt.Select("mrid='-1'");
          if (drdir.Length <= 0)
           return;
          this.DDLdir.Items.Add(new ListItem("/", drdir[0]["id"].ToString()));
          insert(this.DDLdir, dt, Convert.ToInt32(drdir[0]["id"].ToString()), "/");
    }
  }
}
```

在用户自定义的 insert 方法中，实现根据编号查找相关的信息，并且将查找到的数据信息绑定到 DropDownList 控件当中，实现的代码如下。

例程 32　代码位置：光盘\mr\04\NetWorkHD\Insert_Wjj..aspx.cs

```
public void insert(DropDownList ddl, DataTable dt, int id, string s)
{
    DataRow[] dr = dt.Select("mrid='" + id.ToString() + "'");
    foreach (DataRow dr1 in dr)
    {
        string mrming1 = s + dr1["mrming"].ToString() + "/";
        ListItem listdir = new ListItem(mrming1, dr1["id"].ToString());
        ddl.Items.Add(listdir);
        insert(ddl, dt, Convert.ToInt32(dr1["id"].ToString()), mrming1);
    }
}
```

单击“返回首页”按钮，触发其 Click 事件，在该事件中使用 Response 的 Redirect 方法，将页面跳转到“Default.aspx”网站首页当中，实现的代码如下。

例程 33　代码位置：光盘\mr\04\NetWorkHD\Insert_Wjj.aspx.cs

```
protected void ImgBtnSY_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Default.aspx");//跳转到网站首页
}
```

单击“创建，并保存”按钮，触发其 Click 事件，在该事件中将实现文件夹信息添加到指定的网络硬盘的目录中，并且使用 insert 插入语句将信息添加到数据库中，实现的代码如下。

例程 34　代码位置：光盘\mr\04\NetWorkHD\Insert_Wjj.aspx.cs

```
protected void ImgBtnCreate_Click(object sender, ImageClickEventArgs e)
{
    SqlConnection con = new SqlConnection(WebHard.GetConStr());
    SqlCommand cmd = new SqlCommand("insert into tb_Webhard (mrming,mrid,size,mrto,mrtotle,Mark,time) values('"
+ TxtName.Text.Trim() + "','" + Convert.ToInt32(DDLdir.SelectedValue).ToString() + "','0','0','0','1',GetDate())", con);
    con.Open();
    cmd.ExecuteNonQuery();
    SqlCommand cmd1 = new SqlCommand("update tb_Webhard set mrto = mrto + 1 where id='" + Convert.ToInt32
(DDLdir.SelectedValue).ToString() + "'", con);
    cmd1.ExecuteNonQuery();
    Response.Write("<script>alert('恭喜您，创建成功！')</script>");
    con.Close();
}
```

4.3.6　搜索文件并显示

搜索文件并显示页面（Select_WJ.aspx）实现了将从文件夹中搜索文件，同时将搜索出来的文件显示到 GridView 控件中。该页运行结果如图 4.18 所示。

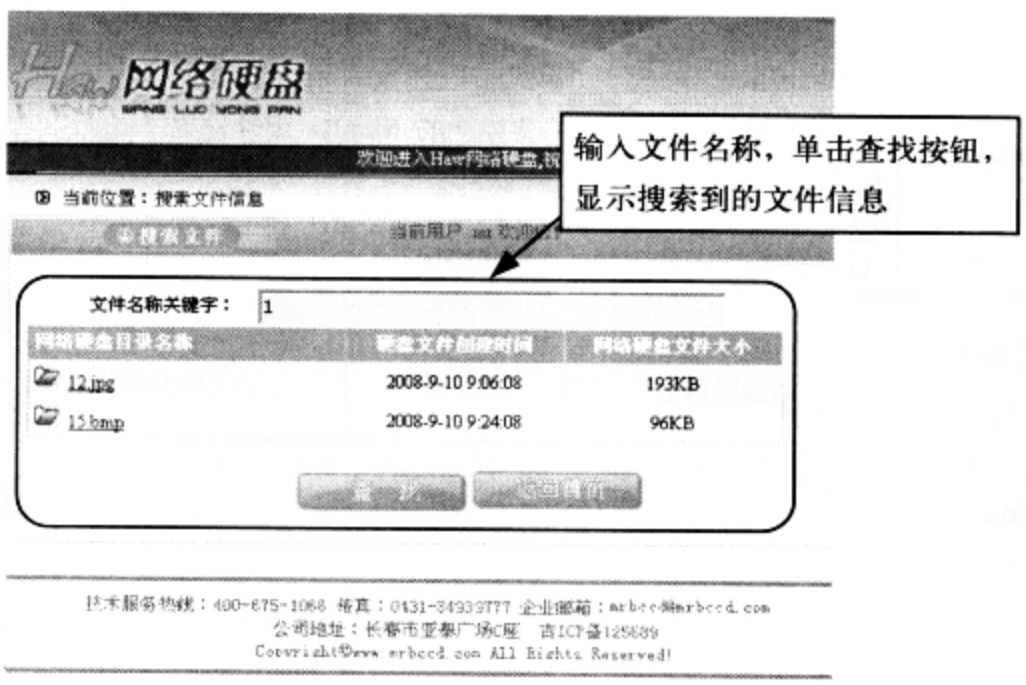


图 4.18　搜索文件并显示运行结果图

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 Select_WJ.aspx，作为检索文件并显示页。页面 Select_WJ.aspx 的设计界面如图 4.19 所示。

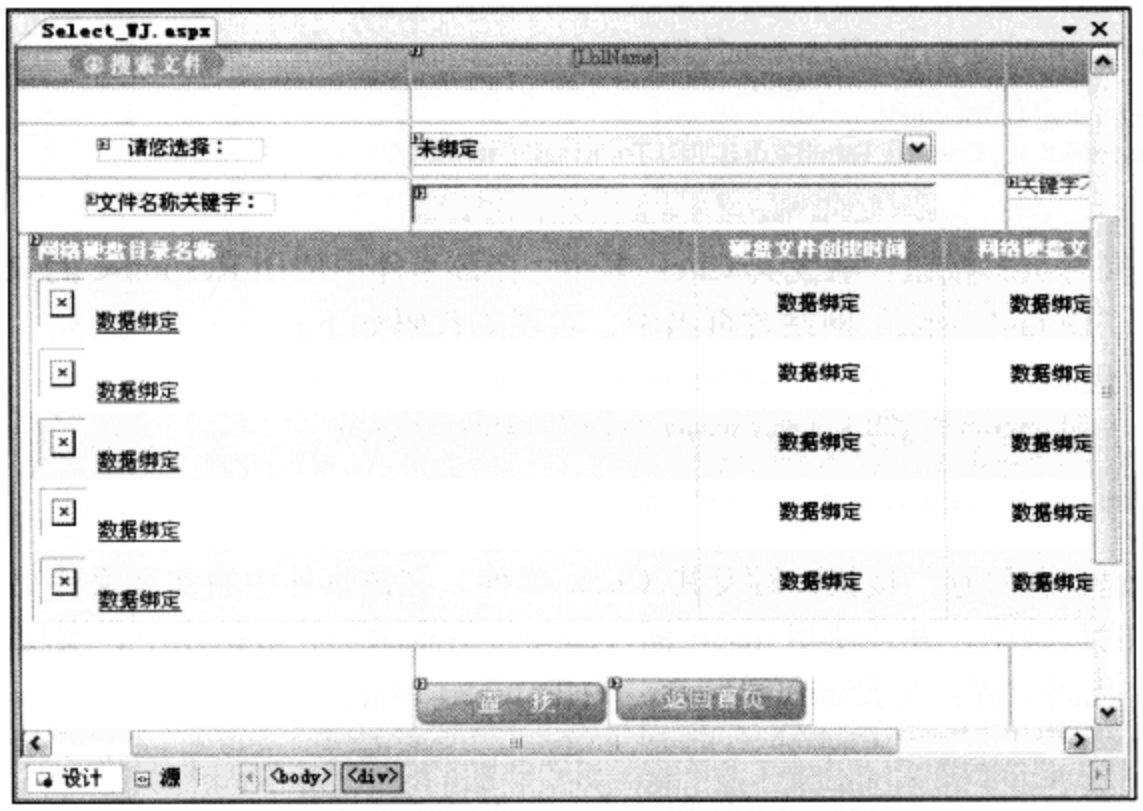


图 4.19　页面 Select_WJ.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 ImageButton 控件、1 个 GridView 控件、1 个 DropDownList 控件和 1 个 TextBox 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 4.12 所示。

表 4.12　　搜索文件并显示页面控件的说明

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
ImageButton	将控件的名称分别设置为 ImgBtnSelect 和 ImgBtnSY	将控件的 ImageUrl 属性分别设置为 ~ /Images/search.JPG 和~/Images/shouye.JPG	这两个控件分别用于执行检索文件和将页面跳转到网站的“Default.aspx”首页当中
GridView	将控件的名称设置为 GvWebHard	无	用于显示检索到的文件的信息
DropDownList	将控件的名称设置为 DDLWebHard	将控件的 SkinID 属性设置为 ddlSkin	用户显示目录信息
TextBox	将控件的名称设置为 TxtName	将控件的 SkinID 属性设置为 tbSkin	用于输入要查询的文件的名称

2. 后台代码实现

首先引入命名空间，实现的代码如下。

例程 35　代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
using System.Data.SqlClient;
```

实例化一个 WebHard 类对象，该类对象为 directory，实现的代码如下。

例程 36　代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
WebHard directory = new WebHard();
```

在 Page_Load 事件中，利用 if 语句判断 session 中存的变量是否为空，如果为空说明没有登

录，则将页面跳转到“tishi.aspx”用户注册登录提示页，否则在 LblName 控件中显示当前用户信息，并且将网络硬盘的目录信息显示到 DropDownList 控件当中，实现的代码如下。

例程 37 代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
protected void Page_Load(object sender,EventArgs e)
{
    if (Session["UserName"] == null && Session["UserPwd"] == null)
    {
        Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
    }
    else
    {
        this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
        if (!Page.IsPostBack)
         {
            bind();
         }
      }
}
```

在用户自定义的 bind 方法中，调用 WebHard 类中的 dir 方法，将从数据库查找到的数据信息显示到 DropDownList 控件的 ID 属性为 DDLWebHard 的控件当中，实现的代码如下。

例程 38 代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
public void bind()
{
    directory.dir(DDLWebHard,-1);
}
```

在用户自定义的 img 方法中，将文件夹和文件显示不同的图标，实现的代码如下。

例程 39 代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
public string img(bool truefalse,string s)
{
    if(truefalse == true)
    {
        return ("Images/文件夹.ico");
    }
    else
    {
        switch(s)
        {
            default: return ("Images/打开.png");
        }
    }
        return ("");
    }
```

在用户自定义的 str 方法中，根据编号将页面跳转到不同的页面中，实现的代码如下。

例程 40 代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
public string str(int n,int i,bool falsetrue)
{
    if(falsetrue == true)
    {
        return ("ShowDisk.aspx?id=" + n.ToString() + "&mrid=" + i.ToString());
    }
    else
    {
        return ("ViewDisk.aspx?id=" + n.ToString() + "&mrid=" + i.ToString());
    }
}
```

单击“返回首页”按钮，触发其 Click 事件，在该事件中使用 Response 的 Redirect 方法，将页面跳转到“Default.aspx”网站首页当中，实现的代码如下。

例程 41 代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
protected void ImgBtnSY_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Default.aspx");
}
```

单击“查找”按钮，触发其 Click 事件，首先在该事件中调用 WebHard 类中的 select 方法，实现的是查找文件的信息，然后使用 DataBind 方法将数据绑定到 GridView 控件当中，实现的代码如下。

例程 42 代码位置：光盘\mr\04\NetWorkHD\Select_WJ.aspx.cs

```
protected void ImgBtnSelect_Click(object sender, ImageClickEventArgs e)
{
    SqlDataReader dr = directory.select(this.TxtName.Text.Trim());
    GvWebHard.DataSource = dr;
    GvWebHard.DataBind();
    dr.Close();
}
```

4.3.7 提示信息页

提示信息页面（tishi.aspx）是当用户没有登录就访问网络硬盘程序时所显示的页面，该页面提示您还没有登录或注册字样，并且有两个跳转按钮，目的是让您进行登录或者注册操作。该页运行结果如图 4.20 所示。

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 tishi.aspx，作为提示信息页。页面 tishi.aspx 的设计界面如图 4.21 所示。

图 4.20 提示信息页运行效果图

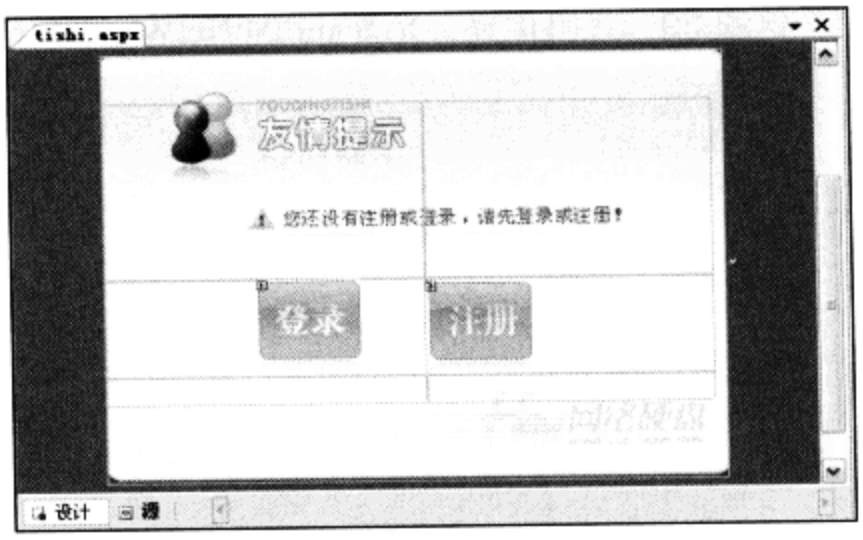


图 4.21 页面 tishi.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 ImageButton 控件，然后用鼠标右健单击这两个控件，在弹出的快键菜单中选择“属性”选项来进行控件的属性设置。页面中各个控件的属性设置及其用途如表 4.13 所示。

表 4.13 提示信息页控件的说明

控件类型	控件名称	主要属性设置	用途
Table	无	无	页面布局
ImageButton	控件的名称分别为 ImgBtnDenglu 和 ImgBtnZHuce	将控件的 ImageUrl 属性分别设置为"～/Images/denglu.bmp" ImageUrl="～/Images/yhzhuce.bmp"	这两个控件分别用于将页面跳转到登录界面和将页面跳转到网站的用户注册页面中

控件的属性设置可以通过前台代码实现，也可以在“控件属性”对话框中设置实现，本页

中使用的控件属性设置请参见如下代码。

例程 43 代码位置：光盘\mr\04\NetWorkHD\tishi.aspx

```
<table border="0" cellpadding="0" cellspacing="0" style="width: 408px; height: 291px; background-image: url(Images/
bg7.bmp);">
    <tr>
        <td style="background-image: url(Images/bg7.bmp); width: 103px; text-align: center;">
            <div style="text-align: center">
                <div style="text-align: center">
                    <table border="0" cellpadding="0" cellspacing="0" style="width: 388%">
                        <tr>
                            <td style="width: 211px; height: 120px">
                            </td>
                            <td style="height: 120px" colspan="2">
                            </td>
                        </tr>
                        <tr>
                            <td style="width: 211px">
                                              
                                <asp:ImageButton ID="ImgBtnDenglu" runat="server"
ImageUrl="~/Images/denglu.bmp"
                                    OnClick="ImgBtnDenglu_Click" /></td>
                            <td style="text-align: left;" colspan="2">
                                <asp:ImageButton ID="ImgBtnZHuce" runat="server" ImageUrl=
"~/Images/yhzhuce.bmp"
                                    OnClick="ImgBtnZHuce_Click" /></td>
                        </tr>
                        <tr>
                            <td style="width: 211px">
                            </td>
                            <td colspan="2">
                            </td>
                        </tr>
                    </table>
                </div>
            </div>
        </td>
    </tr>
</table>
```

2. 后台代码实现

单击“登录”按钮，触发其 Click 事件，在该事件中使用 Response 的 Redirect 方法，可实现将页面跳转到“Login.aspx”登录页面中，实现的代码如下。

例程 44 代码位置：光盘\mr\04\NetWorkHD\tishi.aspx.cs

```
protected void ImgBtnDenglu_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Login.aspx");//跳转到Login.aspx页面中
}
```

单击“注册”按钮，触发其 Click 事件，在该事件中使用 Response 的 Redirect 方法，可实现将页面跳转到“reg_zhuce.aspx”注册页面中，实现的代码如下。

例程 45 代码位置：光盘\mr\04\NetWorkHD\tishi.aspx.cs

```
protected void ImgBtnZHuce_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("reg_zhuce.aspx");//跳转到reg_zhuce.aspx页面中
}
```


4.4 网站打包与发布

网站开发完成后最终的目的是将其发布到 Internet 上，以提供用户浏览访问。实现网站发

布可以使用两种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

（1）首先打开要发布的网站，在网站的项目名称上，单击鼠标右键，在弹出的快捷方式菜单中选择“发布网站”项，如图 4.22 所示。

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择“...”路径按钮。如图 4.23 所示。

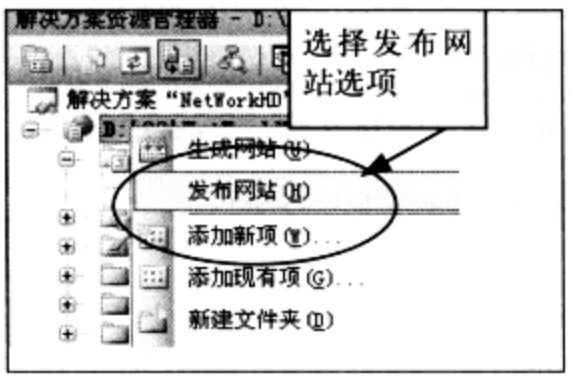


图 4.22　选择发布网站

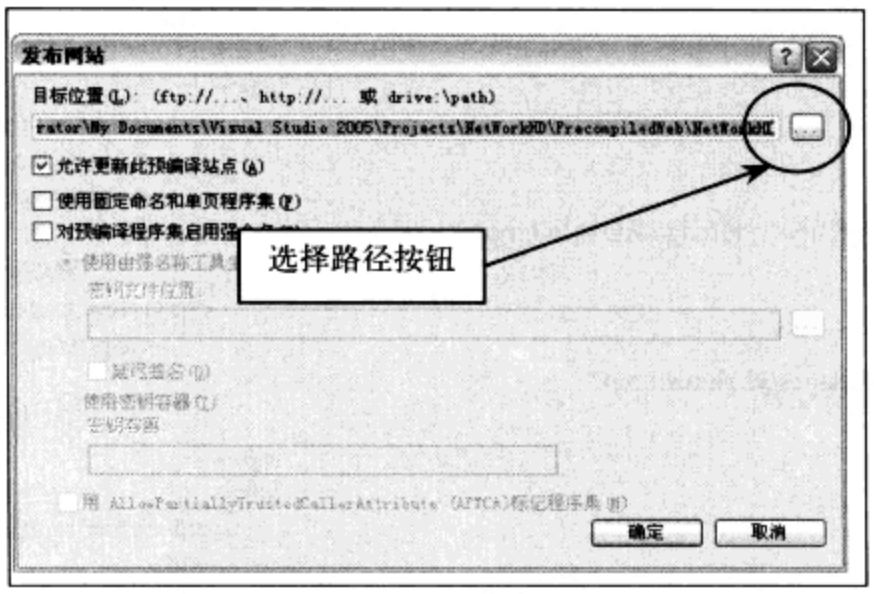


图 4.23　选择路径按钮

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选择中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码如图 4.24 所示。填写完毕后选择“打开”按钮将会返回“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

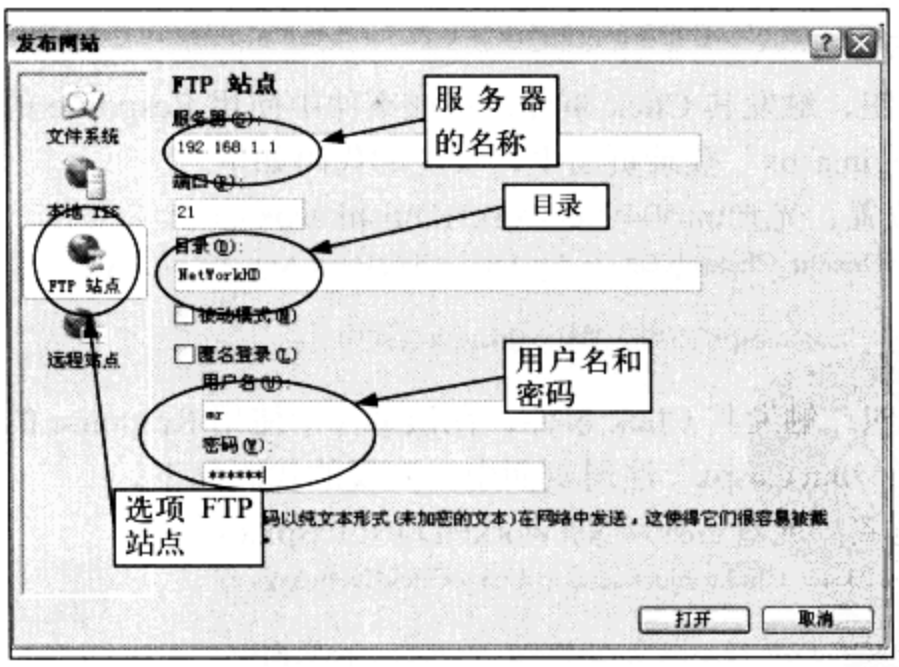


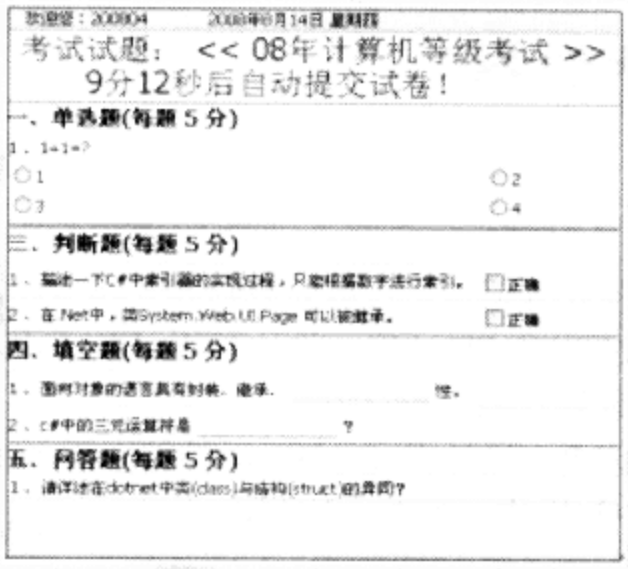
图 4.24　FTP 站点选项

第5章 在线考试模块

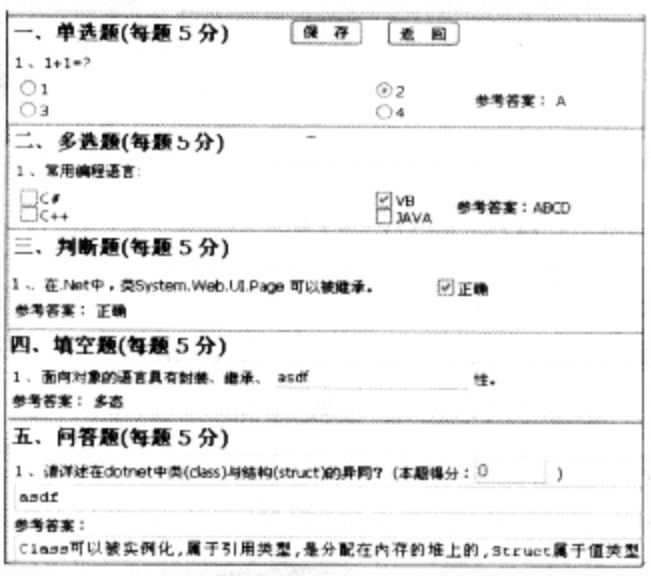
实例位置：光盘\mr\05\

传统的考试由于涉及组织命题、试卷印刷、考场安排、组织阅卷等诸多环节，考试时间周期长、效率低下；同时人工批卷等主观因素也影响到考试的公正性。随着网络技术在教育领域应用的普及，应用现代信息技术构架的网络在线考试系统展现出了越来越多的优越性。本章将详细介绍一个功能较为全面的在线考试模块的编写过程，通过本章的学习，读者能够学到以下内容。

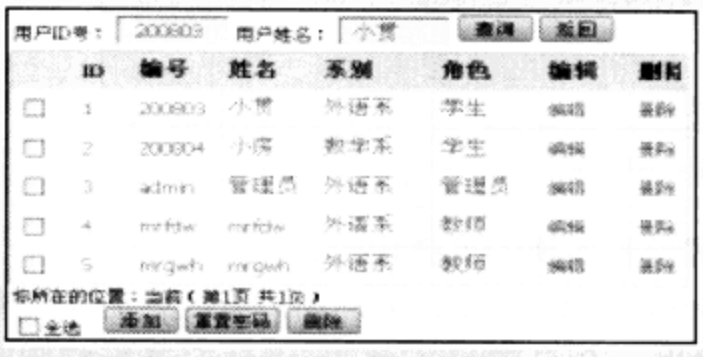
在线考试页试



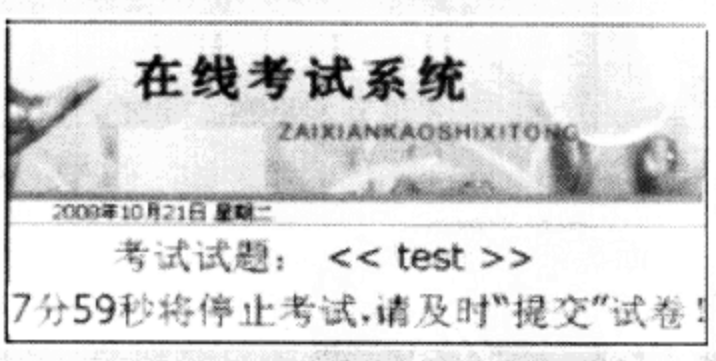
考生试卷评阅



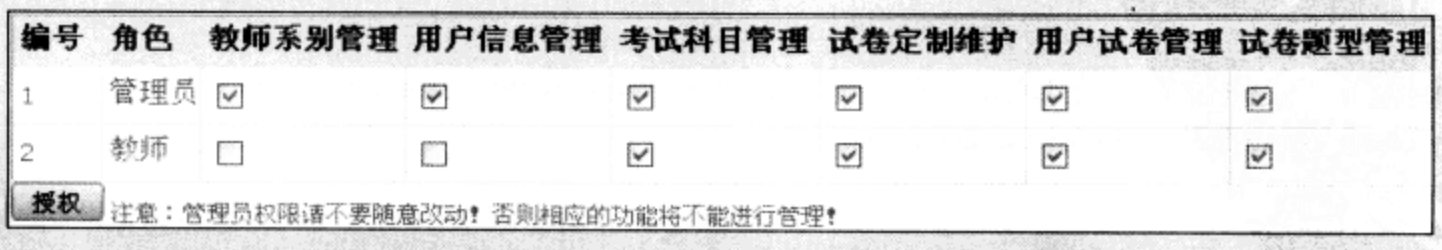
用户管理



考试时间倒计时



用户权限分配设置



编号	角色	教师系别管理	用户信息管理	考试科目管理	试卷定制维护	用户试卷管理	试卷题型管理
1	管理员	☑	☑	☑	☑	☑	☑
2	教师	☐	☐	☑	☑	☑	☑

授权 注意：管理员权限请不要随意改动！否则相应的功能将不能进行管理！

5.1 在线考试模块概述

网络考试系统的开发及应用，避免了以往学校里考试纸张、笔、监考和巡考等各种资源的浪费，并且后台管理是管理员单一的管理，确保了程序的安全性。网络考试管理员登录后，对考试内容、专业、科目和考试题目等都做了详细的分类，只要考生通过自己的学生编号和密码进入前台，按步骤依次进行选题和答题，答题完毕后系统会自动判卷，并核对出最后考卷的分数，从而解决了学生在学校考试后等待考卷分数通知的问题。

本在线考试模块中应用了 Ajax 无刷新环境，如从数据库中检测试卷名称时实现的是无刷新效果。另外，实现考试倒计时应用了 Ajax 中的 Time 控件。在考生试卷页面中可提供单选题、多选题、判断题、填空题和问答题 5 种题型，考生答完题提交试卷后，教师在后台对考生所答试卷进行评阅，5 种题型中除了问答题需要教师根据考生答题情况酌情给分外，其他 4 种题型都是系统自动评分，这样设计使得本考试模块更加贴近实际、更为人性化。另外在考试模块后台管理中，管理员可对教师进行权限分配和对用户（考生和教师）进行其他更人性化的管理，如考生忘记密码可为考生重设密码等。

5.2 关键技术详解

5.2.1 用户管理权限设置

该在线考试系统主要包括 3 种角色，即考生、教师和管理员，可以为教师和管理员设置相应的管理权限。默认情况下，用户拥有所有权限，通过取消 CheckBox 选中情况可禁用用户某个权限，如通过设置禁用教师部门管理、用户信息管理等权限。

用户管理权限设置由 RoleManage.aspx 页实现，该页中通过在 GridView 控件中添加 1 个 CheckBox 控件来对相应的权限进行选中（或取消选中）操作，如图 5.1 所示。

编号	角色	教师部门管理	用户信息管理	考试科目管理	试卷定制维护	用户试卷管理	试卷题型管理
1	管理员	☑	☑	☑	☑	☑	☑
2	教师	☐	☐	☑	☑	☑	☑

授权　注意：管理员权限请不要随意改动！否则相应的功能将不能进行管理！

图 5.1　设置用户权限图

下面具体讲解 RoleManage.aspx 页对用户权限设置实现的关键代码。

首先在页 Page_Load 事件中，首先应用 Session 对象获取用户 ID 号，然后实例化公共类 Users，来判断用户是否已经登录，最后调用一个自定义方法 InitData()进行权限设置的选择，如屏蔽教师的用户管理和教师部门管理权限等操作。Page_Load 事件实现代码如下。

例程 1　代码位置：光盘\mr\05\在线考试模块\WEB\RoleManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //应用Session记录用户ID号
        string loginName = Session["userID"].ToString();
```

```
        //实例化公共类Users
        Users user = new Users();
        user.LoadData(loginName);
        //显示登录的用户角色
        labUser.Text = user.UserName;
        //调用自定义方法InitData()进行权限设置选择
        InitData();
        }
}
```

自定义方法 InitData()主要用来进行用户权限设置的选择操作，在该自定义方法中首先创建 1 个 DataTable 类型的变量存储哈希表中数据，然后将创建的内存表做为表格控件的数据源，并绑定数据库中数据，最后应用 for 语句循环查找 GridView 控件中的 CheckBox 控件，并判断其是否为选中状态，详细代码如下。

例程 2 代码位置：光盘\mr\05\在线考试模块\WEB\RoleManage.aspx.cs

```
//自定义方法InitData()进行权限设置选择
private void InitData()
{
    //创建1个DataTable类型的变量存储哈希表中数据
    DataTable dt = Role.Query(new Hashtable());
    //将创建的dt作为数据源
    GV.DataSource = dt;
    //从数据库中绑定GridView控件中数据
    GV.DataBind();
    //循环GridView控件中的CheckBox控件
    for (int i = 0; i < dt.Rows.Count; i++)
    {
        //部门管理
        if (OnLineExam.DataAccessHelper.GetSafeData.ValidateDataRow_N(dt.Rows[i], "HasDuty_Department Manage") == 1)
            ((CheckBox)GV.Rows[i].FindControl("chkDepartmentManage")).Checked = true;
        //用户管理
        if (OnLineExam.DataAccessHelper.GetSafeData.ValidateDataRow_N(dt.Rows[i], "HasDuty_User Manage") == 1)
            ((CheckBox)GV.Rows[i].FindControl("chkUserManage")).Checked = true;
        //考试科目管理
        if (OnLineExam.DataAccessHelper.GetSafeData.ValidateDataRow_N(dt.Rows[i], "HasDuty_CourseManage") == 1)
            ((CheckBox)GV.Rows[i].FindControl("chkCourseManage")).Checked = true;
        //试卷制定维护
        if (OnLineExam.DataAccessHelper.GetSafeData.ValidateDataRow_N(dt.Rows[i], "HasDuty_PaperSetup") == 1)
            ((CheckBox)GV.Rows[i].FindControl("chkPaperSetup")).Checked = true;
        //用户试卷管理
        if (OnLineExam.DataAccessHelper.GetSafeData.ValidateDataRow_N(dt.Rows[i], "HasDuty_UserPaperList") == 1)
            ((CheckBox)GV.Rows[i].FindControl("chkUserPaperList")).Checked = true;
        //试题类别管理
        if (OnLineExam.DataAccessHelper.GetSafeData.ValidateDataRow_N(dt.Rows[i], "HasDuty_SingleSelectManage") == 1)
            ((CheckBox)GV.Rows[i].FindControl("chkTypeManage")).Checked = true;
    }
}
```

设置完相应的用户权限后，可单击该页面中的“授权”按钮完成权限设置操作，在该按钮的 ImageButtonGiant_Click 事件中，主要应用 foreach 语句循环 GridView 控件中的 CheckBox 控件，并设置其选中的状态值（1 代表为选中，0 代表为不选），事件详细代码如下。

例程 3 代码位置：光盘\mr\05\在线考试模块\WEB\RoleManage.aspx.cs

```
protected void ImageButtonGiant_Click(object sender, ImageClickEventArgs e)
{
```

```
        //定义一个哈希表ht
        Hashtable ht = new Hashtable();
        string where = "";
        //应用foreach循环GridView控件中的CheckBox控件
        foreach (GridViewRow row in GV.Rows)
        {
            //先清空哈希表中的数据
            ht.Clear();
            //应用FindControl方法查找GridView控件中CheckBox控件，并判断是否选中了用户权限
            ht.Add("HasDuty_DepartmentManage", ((CheckBox)row.FindControl("chkDepartmentManage")).Checked ==
true ? 1 : 0);
            ht.Add("HasDuty_UserManage", ((CheckBox)row.FindControl("chkUserManage")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_RoleManage", ((CheckBox)row.FindControl("chkUserManage")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_Role", ((CheckBox)row.FindControl("chkUserManage")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_CourseManage", ((CheckBox)row.FindControl("chkCourseManage")).Checked ==
true ? 1 : 0);
            ht.Add("HasDuty_PaperSetup", ((CheckBox)row.FindControl("chkPaperSetup")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_PaperLists", ((CheckBox)row.FindControl("chkPaperSetup")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_UserPaperList", ((CheckBox)row.FindControl("chkUserPaperList")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_UserScore", ((CheckBox)row.FindControl("chkUserPaperList")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_SingleSelectManage", ((CheckBox)row.FindControl("chkTypeManage")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_MultiSelectManage", ((CheckBox)row.FindControl("chkTypeManage")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_FillBlankManage", ((CheckBox)row.FindControl("chkTypeManage")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_JudgeManage", ((CheckBox)row.FindControl("chkTypeManage")).Checked == true ? 1 : 0);
            ht.Add("HasDuty_QuestionManage", ((CheckBox)row.FindControl("chkTypeManage")).Checked == true ? 1 : 0);
            //定义一个查询的条件语句
            where = " Where RoleId=" + row.Cells[0].Text;
            //调用Role公共类中的Update方法修改角色权限信息
            Role.Update(ht, where);
        }
    }
```

5.2.2 考试时间倒计时

该在线考试系统倒计时主要应用了 Ajax 服务器中的 Timer 控件，并实现了考试页面的无刷新效果，关于 Ajax 服务器中的 Timer 控件稍后会给予详细介绍。实际运行效果如图 5.2 所示。

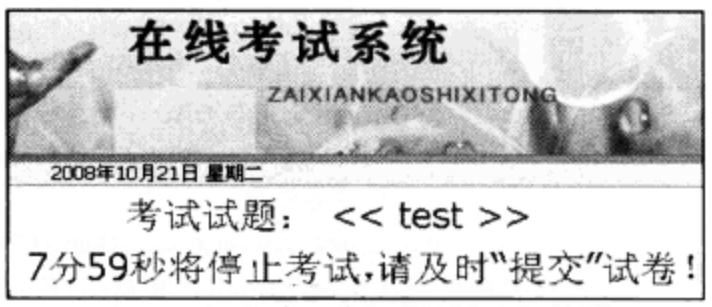


图 5.2 考试倒计时（Ajax 实现）

Ajax 环境中的 Timer 控件在实现考试倒计时主要应用在考生答题页 StudentIndex.aspx 中，在 Timer 控件的 Timer1_Tick 事件中编写的代码如下。

例程 4 代码位置：光盘\mr\05\在线考试模块\WEB\UserText.aspx.cs

```
this.index--;
            //ss
            if (this.index == 0)
            {
                //设置Timer控件不可见
                this.Timer1.Enabled = false;
                BindData();
                Response.Redirect("Loginout.aspx");
            }
```

```
            else
            {
                //显示考试剩余时间
                this.lbtime.Text = this.index / 60 + "分" + this.index % 60 + "秒将停止考试，请及时“提交”
试卷！ ";
            }
```

自定义一个私有变量 index，用来定义考试的总时间，实现的代码如下：

例程 5 代码位置：光盘\mr\05\在线考试模块\WEB\StudentIndex.aspx.cs

```
        /// <summary>
        /// 定义在线考试总时间变量index,
        /// 并设置读写属性
        /// </summary>
        get
            {
                object o = ViewState["index "];
                return (o == null) ? 600 : (int)o;
            }
            set
            {
                ViewState["index "] = value;
            }
```

5.2.3 大量数据查询进度等待

在用户信息管理页（UserManage.aspx）中查询用户信息时将会显示等待进度，这里主要应用了 Thread 类中的 Sleep()方法，显示查询等待进度条。读者可以将此技术应用在网页本身比较大或用户登录系统时可能需要较长的等待时间的情况下，让其显示一个等待进度。实际运行效果如图 5.3 所示。

图 5.3 查询时显示进度条

双击用户信息管理页（UserManage.aspx）中的“查询”按钮，触发其 ImageButtonQuery_Click 事件，实现代码如下。

例程 6 代码位置：光盘\mr\05\在线考试模块\WEB\UserManage.aspx.cs

```
protected void ImageButtonQuery_Click(object sender, ImageClickEventArgs e)
{
    Response.Write("<div id='mydiv' >");
    Response.Write("_");
    Response.Write("</div>");
    Response.Write("<script>mydiv.innerText = ';</script>");
    Response.Write("<script language=javascript>;");
    Response.Write("var dots = 0;var dotmax = 10;function ShowWait()");
    Response.Write("{var output; output = '正在查询，请稍候';dots++;if(dots>=dotmax)dots=1;");
    Response.Write("for(var x = 0;x < dots;x++){output += '·';}mydiv.innerText =  output;}");
    Response.Write("function StartShowWait(){mydiv.style.visibility = 'visible'; ");
    Response.Write("window.setInterval('ShowWait()',1000);}");
    Response.Write("function HideWait(){mydiv.style.visibility='hidden';");
    Response.Write("window.clearInterval();}");
    Response.Write("StartShowWait();</script>");
    Response.Flush();
    Thread.Sleep(10000);
   //这里可以添加处理搜索用户信息的代码，省略不计
}
```

注意 这里主要使用了 Response 对象中的 Write()方法输出进度条显示信息。另外，需要引入添加线程的命名空间，如 using System.Threading。

5.2.4 智能记忆登录用户名

智能记忆登录用户名主要应用在线考试登录窗口中，这里主要实现记忆最后一次用户的登录名。智能记忆登录可以使相同用户不必每次都输入相同用户名，实现了一种智能登录（在登录窗口中该功能可以设置为不选）。实现该智能记忆功能主要应用了 Cookie 对象。Cookie 对象是 HttpCookieCollection 类的一个实例，用于保存客户端浏览器请求的服务器页面，也可用存放非敏感性的用户信息，信息保存的时间可以根据用户的需求进行设置。实际运行效果如图 5.4 所示。

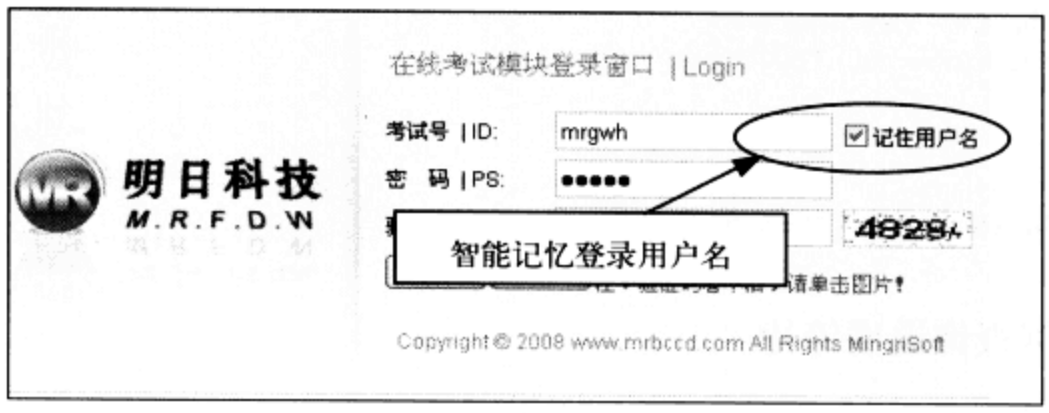


图 5.4 智能记忆登录用户名

下面详细介绍一下在登录窗口页 Login.aspx 页面中实现该功能的过程，首先在页 Page_Load 事件中判断页面是否首次加载，然后判断请求的用户登录名是否为空，如果用户名不为空，则创建 1 个 Cookie 对象的实例将登录用户名保存在该对象中。具体实现代码如下。

例程 7 代码位置：光盘\mr\05\在线考试模块\WEB\Login.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
     if (!IsPostBack)//判断页面是否首次加载
     {
         if (!Object.Equals(Request.Cookies["UserID"], null))
         {
             //创建一个Cookie对象，实现记住用户名
             HttpCookie readcookie = Request.Cookies["UserID"];
             this.txtUserID.Text = readcookie.Value;
         }
     }
 }
```

接着在登录按钮事件中，首先判断用户输入的验证码及密码是否正确，如果正确则调用一个自定义方法 CreateCookie()来实现智能记忆登录用户名，并通过添加的 1 个 CheckBox 控件来设置该功能的可选择性，该方法具体实现的代码如下。

例程 8 代码位置：光盘\mr\05\在线考试模块\WEB\Login.aspx.cs

```
private void CreateCookie()
{
    //创建1个Cookie对象
    HttpCookie cookie = new HttpCookie("UserID");
    //判断Checkbox控件是否被选中
    if (this.cbxRemeberUser.Checked)
    {
        //将用户编号存储到创建的Cookie对象中
        cookie.Value = this.txtUserID.Text;
```

```
        }
        //获取创建的Cookie对象的过期时间
        cookie.Expires = DateTime.MaxValue;
        //将创建的Cookie对象添加到内部Cookie集合中
        Response.AppendCookie(cookie);
    }
```

上述自定义 CreateCookie()方法中，主要是应用了 Cookie 对象。Cookie 对象是 HttpCookieCollection 类的实例，用于保存客户端浏览器请求的服务器页面，也可用于存放非敏感性的用户信息，信息保存的时间可以根据用户的需要进行设置。如果没有设置 Cookie 的失效日期，那么仅保存到关闭浏览器为止。如果将 Cookie 对象的 Expires 属性设置为 MinValue，则表示 Cookie 永远不会过期。Cookie 存储的数据量受限制，大多数浏览器支持的最大容量为 4096 字节，因此，一般不要用 Cookie 对象来保存数据集或其他大量数据。并非所有浏览器都支持 Cookie，并且数据信息是以明文文本的形式保存在客户端计算机中，因此最好不要保存未加密的数据，否则会影响网络安全性。要存储一个 Cookie 变量，可以使用 Response 对象的 Cookies 集合，使用语法如下：

```
Response.Cookie[Name].Value="资料";
```

要取回 Cookie，使用 Response 对象的 Cookies 集合，并将指定的 Cookies 集合返回，其使用语法如下：

```
变量名= Response.Cookie[Name].Value;
```

下面介绍一下 Cookie 对象中的 Expires 属性。

- Expires 属性

获取或设置 Cookie 的过期日期和时间。

- 其语法如下：

```
public DataTime Expires{get;set};
```

- 属性值：Cookie 的过期时间（在客户端）。

例如，将 Cookie 的过期时间设置为当前时间之后 20s，代码如下：

```
HttpCookie cookie = new HttpCookie("UserName");
Cookie.Value="明日科技";
DataTime time = DataTime.Now;
TimeSpan time = new TimeSpan(0,0,0,20);
Cookie. Expires=time.Add();
```

5.2.5 GridView 控件中更改试卷可用状态

创建的考试试卷题目有时要求考生并不是全答，可针对不同考试要求来选择几套试题，这样就需要设置试卷状态，通过试卷的可用或不可用来选择性地出卷，这样使在线考试系统更加人性化、合理化。试卷状态的更改由 PaperLists.aspx 页实现，在该页面中应用了 1 个 GridView 控件来绑定试卷的维护信息，如试卷状态更改等。单击该 GridView 控件自带的编辑选项，在“试卷状态”一栏中会现一个下拉列表框，该列表框中提供了两种选项，即可用和不可用，可人工选择性地出卷。实际运行效果如图 5.5 所示。

考试科目	试卷名称	试卷状态	详细...	编辑	删除
C#技能考试	测试	可用	详细...	编辑	删除
C#技能考试	明日在线考试	可用	详细...	更新 取消	
C#技能考试	08年计算机等级考试	可用 不可用	详细...	编辑	删除

图 5.5 试卷状态更改效果图

GridView 控件中实现该功能的代码主要编写在 GridView 控件中 RowUpdating 事件中，这

里主要是应用FindControl查找GridView控件中的DropDownList，并将其列表中的值转换成byte类型赋予公共类 Paper 中的 PaperState 试卷状态变量。实现代码如下。

例程 9 代码位置：光盘\mr\05\在线考试模块\WEB\PaperLists.aspx.cs

```
//GridView控件RowUpdating事件
    protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
    {
        //取出要删除记录的主键值
        int ID = int.Parse(GridView1.DataKeys[e.RowIndex].Values[0].ToString());
        //创建Paper对象
        Paper paper = new Paper();
        //应用FindControl查找GridView控件中的DropDownList，并将其列表中的值转换成byte类型赋予公共类
Paper中的PaperState试卷状态变量
        paper.PaperState = byte.Parse(((DropDownList)GridView1.Rows[e.RowIndex].FindControl("ddlPaperState")).SelectedValue);
        //使用Paper类UpdateByProc方法修改试卷状态
        if (paper.UpdateByProc(ID))
        {
            Response.Write("<script language=javascript>alert('试卷状态修改成功!')</script>");
        }
        else
        {
            Response.Write("<script language=javascript>alert('试卷状态修改失败!')</script>");
        }
        //取消编辑操作
        GridView1.EditIndex = -1;
        //调用自定义方法InitData()重新绑定GridView控件中信息
        InitData();
    }
```

实现试卷状态更改操作主要是用到了 GridView 控件的模板列和其 FindControl 方法。下面分别作简单介绍。

- GridView 控件模板列

GridView 控件中的模板列由 TemplateField 列来提供，TemplateField 列用来在数据绑定控件中显示自定义的字段，该列支持的模板如表 5.1 所示。

表 5.1　　GridView 控件中模板列支持的模板

模　板	用　途
AlternatingItemTemplate	为 TemplateField 模板列中的交替项指定要显示的内容
EditItem Template	为 TemplateField 模板列中处于编辑模式中的项指定要显示的内容
FooterTemplate	为 TemplateField 模板列的脚注部分指定要显示的内容
HeaderTemplate	为 TemplateField 模板列的标头部分指定要显示的内容
ItemTemplate	为 TemplateField 模板列中的项指定要显示的内容

- FindControl 方法

GridView 控件的 FindControl 方法主要用来在当前控件中搜索指定 id 的服务器控件，其使用方法如下：

```
Public virtual Control FindControl(string id);
```

- id：要查找的控件的标识符。
- 返回值：如果指定的控件存在，则返回该控件，否则返回空。

5.2.6 Ajax 服务器控件的应用

在实现如考试倒计时、无刷新检测试卷名称时都应用到了 Ajax 环境，下面来着重介绍 Ajax 主要的服务器控件及其应用。

Visual Studio 2008 开发环境的“工具箱”中自带了 Ajax 服务器端控件，如图 5.6 所示。下面主要讲解几个重要的 Ajax 服务器控件。

- ScriptManager 控件。
- UpdatePanel 控件。
- Timer 控件。

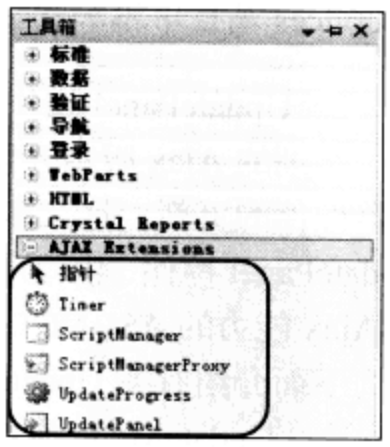


图 5.6　Atlas 服务器控件

1. ScriptManager 控件

ScriptManager 控件负责管理 Page 页面中所有的 Ajax 服务器控件，是 Ajax 的核心，有了 ScriptManager 控件才能够让 Page 局部更新起作用，所需要的 JavaScript 才会自动管理。因此开发 Ajax 网站时，每个页面中必须添加 ScriptManager 控件作为管理。

下面分别介绍 ScriptManager 控件的基本属性。

- EnablePageMethods 属性

EnablePageMethods 属性用来返回或设置 1 个 bool 值，默认值为 False，表示在客户端 JavaScript 代码中是否以一种简单、直观的形式直接调用服务器端的某个静态 Web Method。EnablePageMethods 属性设置代码如下：

```
<asp:ScriptManager ID="ScriptManager1"  runat="server" EnablePageMethods="True" />
```

其中在服务器端的一个方法，必须是静态的方法，必须添加[System.Web.Services.WebMethod]属性。例如下面的代码：

```
[System.Web.Services.WebMethod]
public static string hello(string str)
{
    return str + "      " + DateTime.Now.ToString();
}
```

- EnablePartialRendering 属性

EnablePartialRendering 属性用来返回或设置一个 bool 值，默认值为 True，表示 Ajax 允许改变原有的 ASP.NET 回送模式，不再是整个页面的回送，而是只回送页面中的一部分。待服务器端返回之后，也仅仅更新了局部页面。若要启动页面的局部更新模式，则应该将 EnablePartialRendering 属性值设置为 Ture。EnablePartialRendering 属性设置代码如下：

```
<asp:ScriptManager ID="ScriptManager1"  runat="server" EnablePartialRendering="true" />
```

EnablePartialRendering 属性只针对 UpdatePanel 局部来使用。

- EnableScriptComponents 属性

EnableScriptComponents 属性用于设置是否传送除了 Ajax 核心以外的其他组件，包括 Client 控件、数据绑定、XML 声明式 Script、用户接口组件。

- Scripts 属性

Scripts 属性用于取得 ScriptReference 对象的集合，ScriptReference 对象的集合通过 Ajax 将用户的 Script 文件送到 Client 端进行对象引用。

- Services 属性

Services 属性用于取得 1 个 ServiceRefence 对象的集合，ServiceRefence 对象的集合通过 Ajax 为每个 Web Service 在 Client 端公开 1 个 Proxy 对象引用。

注意

如果有自定义的 Script 或 Web Service 要传送到前端，需要在 Scripts 及 Services 属性中添加相应的设置。

2. UpdatePanel 控件

早期的 Ajax 版本开发出很多的 Ajax 服务器控件，例如 TextBox、Button 等，随着.NET 服务器控件的更新，这么多的 Ajax 服务器控件并不能满足实际需要，最后微软开发出了 Ajax 的 UpdatePanel 控件，由程序人员将 ASP.NET 服务器控件拖放到 UpdatePanel 控件中，使原本不具备 Ajax 能力的 ASP.NET 服务器控件，都具有 Ajax 异步的功能。

下面介绍有关 UpdatePanel 控件的属性。

- UpdateMode 属性

UpdateMode 属性共有两种模式：Always 与 Conditional，Always 是每次 Postback 后，UpdatePanel 会连带被更新；相反，Conditional 只针对特定情况才被更新。

Mode 模式通常在一个 Page 页面同时包含多个 UpdagePanel，若设置为 Always 模式，则即使只有一个 UpdatePanel 做异步更新，其他 UpdatePanel 也会连带受到影响。如果将 Mode 设置为 Conditional，就可以避免连带受到其他 UpdatePanel 影响。

设置 UpdateMode 的属性值，如图 5.7 所示。

- RenderMode 属性

若 RenderMode 的属性值为 Block，则以<DIV>标签来 Render 程序段；若为 Inline，则以<Span>标签来 Render。

设置 RenderMode 的属性值，如图 5.8 所示。

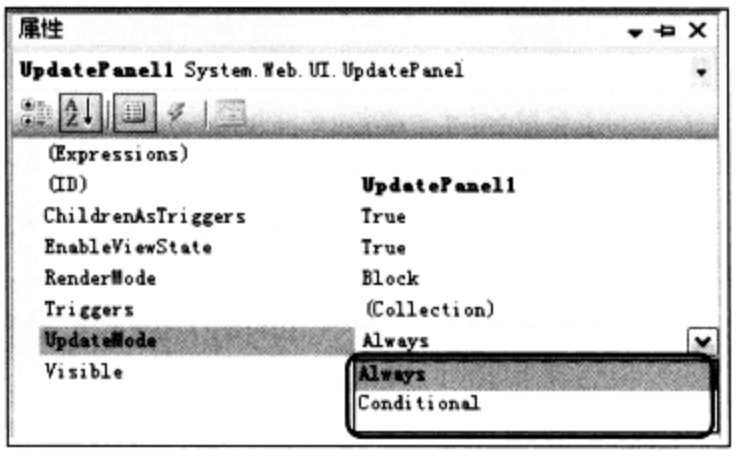


图 5.7 设置 UpdateMode 属性值

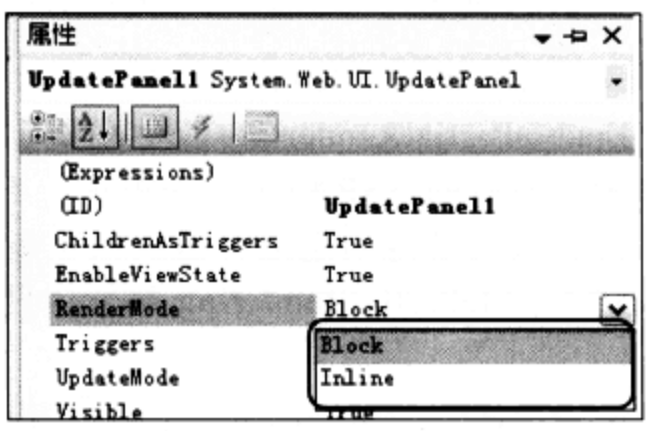


图 5.8 RenderMode 属性设置

- Triggers 属性

Triggers 属性用于设置 UpdatePanel 的触发事件。

Triggers 有两种类型。第 1 种是控件属性改变；第 2 种是控件属性引发事件。针对这两种不同情况可以设置 Trigger 要监视的对象，而 1 个 UpdatePanel 可以同时设置多个 Triggers 属性。

设置 Triggers 的属性值，如图 5.9 所示。

3. Timer 控件

Timer 定时器用 JavaScript 构建非常容易，但在 ASP.NET 中实现 Timer 定时器不但困难，而且运作起来非常麻烦，还会损耗计算机资源。但 Ajax 直接构建了一个 Timer 服务器控件，让程序开发人员可以设置时间间隔来触发特定事件的操作。

下面对 Timer 控件的相关属性和事件进行介绍。

- Interval 属性

用于设置 Timer 时间控件的 Tick 事件间隔时间，单位为 ms（1000ms 等于 1s）。

例如，如果要设置触发 Tick 事件的时间间隔为 3s，将 Interval 属性设置为 3000 即可，如图 5.10 所示。

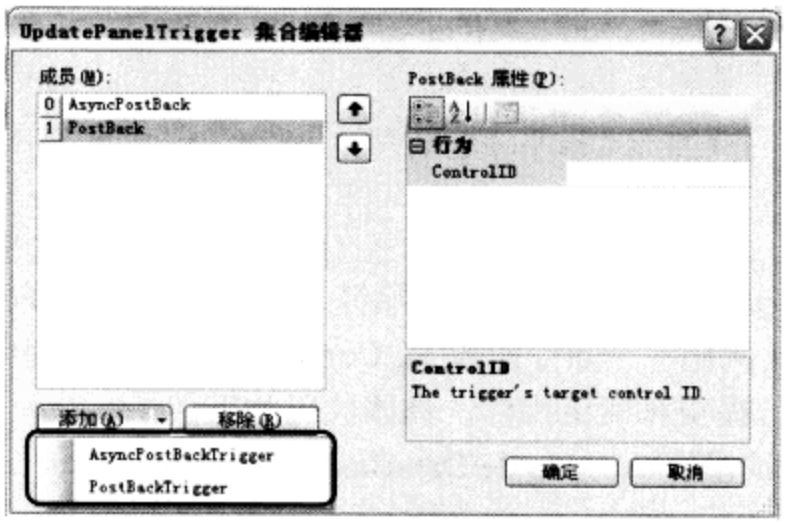


图 5.9　Triggers 属性值设置

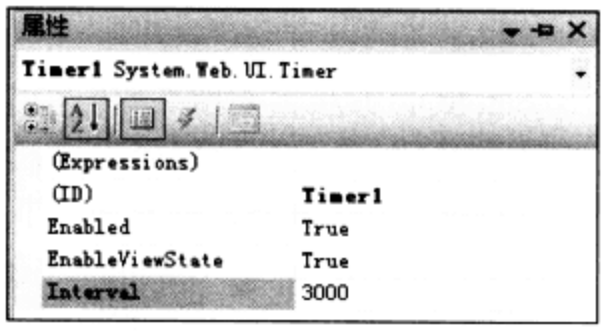


图 5.10　设置 Interval 属性为 3s

- Tick 事件

该事件用于在指定的时间间隔进行触发。

5.3　公共类的封装与设计

在网站开发项目中通常以类的形式来组织、封装一些常用的方法和事件，公共类的编写可以减少重复代码的编写，有利于代码维护，在编程时可以起到事半功倍的效果。同时也大大方便了代码的管理。在线考试网中主要介绍数据库连接类 DataBase 和字符串处理类 SQLString，其他公共类请读者参见本书附带的光盘中代码。

5.3.1　数据库连接类

数据库的连接操作主要编写在公共类 DataBase 中，在该公共类创建了多个与数据库操作的方法，如调用存储过程（带参数）的 RunProcGetCount 方法、ExecuteSQL 方法执行 SQL 语句、GetDataSetSql 方法根据 Sql 语句，返回一个结果数据集等，下面简单介绍下该公共类中的几个方法。

在此公类中编写了 Open()和 Close()方法，这两种方法分别用来打开数据库连接和关闭数据库连接，具体代码如下。

例程 10　代码位置：光盘\mr\05\在线考试模块\App_Code\DataBaseClass\DBClass.cs

```
//保护方法，打开数据库连接
private void Open()
{
  //判断数据库连接是否存在
if (Connection == null)
{
        //不存在，新建并打开数据库连接
          Connection = new SqlConnection(ConnectionString);
          Connection.Open();
      }
      else
      {
        //存在，判断是否处于关闭状态
        if (Connection.State.Equals(ConnectionState.Closed))
            //连接处于关闭状态，重新打开
            Connection.Open();
```

```
        }
    }
    //公有方法，关闭数据库连接
    public void Close()
    {
        if (Connection.State.Equals(ConnectionState.Open))
        {
            //连接处于打开状态，关闭连接
            Connection.Close();
        }
    }
```

ExecuteSQL 方法主要用来执行指定的 SQL 语句，如修改、删除等 SQL 语句，在该方法中首先调用 open()方法打开数据库连接，然后实例化一个命令对象 SqlCommad，在该方法中还应用了 ADO.NET 中的事务，可分别对事务进行提交和回滚操作。具体代码如下。

例程 11 代码位置：光盘\mr\05\在线考试模块\App_Code\DataBaseClass\DBClass.cs

```
public bool ExecuteSQL(String[] SqlStrings)
{
    bool success = true;
    //打开数据库连接
    Open();
    //定义一个命令对象
    SqlCommand cmd = new SqlCommand();
    //通过SqlConnection的BeginTransaction方法创建名为st的对象Transaction
    SqlTransaction trans = Connection.BeginTransaction();
    //建立数据库连接
    cmd.Connection = Connection;
    //将SqlTransaction对象分配给SqlCommand对象的Transaction属性
    cmd.Transaction = trans;
    int i = 0;
    try
    {
        foreach (String str in SqlStrings)
        {
            cmd.CommandText = str;
            //执行指定的SQL语句
            cmd.ExecuteNonQuery();
            i++;
        }
        //提交事务
        trans.Commit();
    }
    catch
    {
        success = false;
        //关闭数据库连接
        Close();
        //事务回滚
        trans.Rollback();
    }
    finally
    {
        //关闭数据库连接
        Close();
    }
    return success;
}
```

GetDataSet 方法主要用来调用存储过程，返回 1 个 DataSet 类型的数据集，在该方法中首先调用 Open()方法打开数据库连接，然后实例化 1 个命令对象 SqlCommand 执行存储过程操作，最后应用数据适配器从存储过程中读取数据添充到创建的 DataSet 数据集中。具体代码如下。

例程 12　代码位置：光盘\mr\05\在线考试模块\App_Code\DataBaseClass\DBClass.cs

```
//公有方法，调用存储过程（不带参数）
 //输入：
 //ProcName存储过程名
 //输出：
 //将执行结果以DataSet返回
 public DataSet GetDataSet(string ProcName, SqlParameter[] Params)
 {
     //调用Open()方法打开数据库连接
      Open();
     //实例化一个命令对象SqlCommand，并调用存储过程
      SqlCommand Cmd = CreateCommand(ProcName, Params);
     //创建一个数据适配器，读取存储过程中的数据
      SqlDataAdapter adapter = new SqlDataAdapter(Cmd);
     //创建一个DataSet对象
      DataSet dataset = new DataSet();
     //将从数据库中读取的数据添充到DataSet中
      adapter.Fill(dataset);
     //关闭数据库连接
      Close();
      return dataset;
 }
```

5.3.2　Ajax 环境中的对话框类

在普通的的 ASP.NET 环境中，要在 Web 窗体页上弹出一个对话框可以使用如下实例代码：

```
Response.Write("<script>alert('这个是一个对话框！')</script>");
```

然而在 ASP.NET 环境中要在 Web 窗体页上弹出一个对话框，却不能使用上述代码，否则会报错！但 Ajax 的 ScriptManager 类提供了在 Web 窗体上弹出对话框功能。该类使用 RegisterClientBlock()方法能够直接向 Web 窗体页中注册对话框的脚本，从而实现弹出对话框功能。实现这一功能代码编写在了创建的公共类 AjaxCommond 中，具体代码如下。

例程 13　代码位置：光盘\mr\05\在线考试模块\App_Code\DataBaseClass\AjaxCommond.cs

```
//<summary>
//在ASP.NET   Ajax环境中，为Button控件弹出一个提示对话框
//</summary>
//<param name="button">Button控件</param>
//<param name="message">对话框中的消息</param>
public void OpenDialogForButton(Button button, string message)
{
    ScriptManager.RegisterClientScriptBlock(
          button,
          typeof(Button),
          DateTime.Now.ToString().Replace(":", " "),//使用当前时间作为标识
        "alert('" + message + "')",
        true);
}
//<summary>
//在ASP.NET   Ajax环境中，为Page对象弹出一个提示对话框
//</summary>
//<param name="button">Page对象</param>
//<param name="message">对话框中的消息</param>
public void OpenDialogForPage(Page page, string message)
{
    ScriptManager.RegisterClientScriptBlock(
        page,
        typeof(Page),
        DateTime.Now.ToString().Replace(":", " "),//使用当前时间作为标识
      "alert('" + message + "')",
      true);
}
```

5.4 在线考试页设计

5.4.1 在线考试页概述

考生通过登录窗口成功登录后，将进入试卷选择页。选择某套试卷后考生就进入在线考试页 UserTest.aspx 页，在该页中设置了考试时间及多种题型，如单选、多选、填空等，该页运行效果如图 5.11 所示。

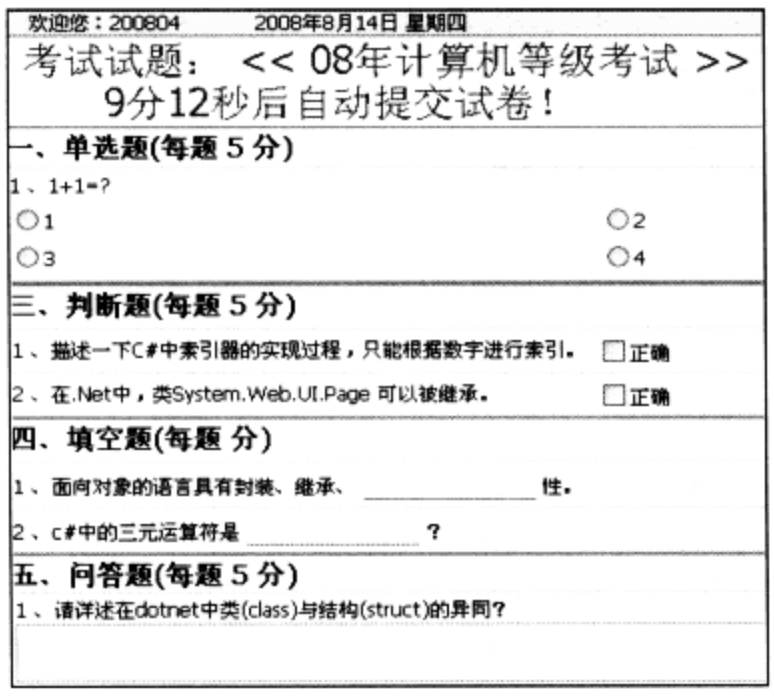


图 5.11 在线考试页

5.4.2 在线考试页实现过程

1. 设计步骤

（1）在应用程序中创建的名为 Web 文件夹下，创建 1 个 Web 窗体，命名为 UserTest.aspx，作为考生在线考试页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 5 个 GridView 控件，1 个 ImageButton 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 5.2 所示。

表 5.2 在线考试页面涉及的主要控件

控件类型	控件名称	主要属性设置	用途
Button	imgBtnSubmit	无	用于提交在线考试答案
GridView	GridView1	AutoGenerateColumns 属性设置为 False	用于绑定单选题试题信息
	GridView2	AutoGenerateColumns 属性设置为 False	用于绑定多选题试题信息
	GridView3	AutoGenerateColumns 属性设置为 False	用于绑定判断题试题信息
	GridView4	AutoGenerateColumns 属性设置为 False	用于绑定填空题试题信息
	GridView5	AutoGenerateColumns 属性设置为 False	用于绑定问答题试题信息

2. 实现代码

在该页面的 Page_Load 事件中，首先判断页面是否是首次加载，然后应用 Session 对象获取考生用户名、试卷名称和在线考试时间等信息，并在该事件中调用自定义方法 InitData()从数据库中提取出相应的各种题型试题，其实现代码如下：

例程 14 代码位置：光盘\mr\05\在线考试模块\WEB\UserTest.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
     if (!IsPostBack)//判断页面是否首次加载
     {
         //应用session对象存储用户名
         labUser.Text = Session["userID"].ToString();
         //单击弹出“提交”按钮时弹出提示框
         imgBtnSubmit.Attributes.Add("OnClick", "javascript:return confirm('确实要交卷吗？')");
         //应用Session对象显示试卷名称
         lblPaperName.Text = Session["PaperName"].ToString();
         //调用自定义方法InitData()初始化试卷，从数据库中将试题取出
         InitData();
         //显示在线考试时间
         this.lbtime.Text = this.index / 60 + "分" + this.index % 60 + "秒后自动提交试卷！";
         this.lblend.Visible = false;
     }
 }
```

自定义方法 InitData()，用于从数据库中提取出相应试卷题目，如单选题、多选题、判断题和问答题等，其具体实现代码如下。

例程 15 代码位置：光盘\mr\05\在线考试模块\WEB\UserTest.aspx.cs

```
//初始化试卷，从数据库中将试题取出
    protected void InitData()
    {
        //提取试卷中的单选题
        SqlParameter[] Params1 = new SqlParameter[2];
        DataBase DB = new DataBase();
        int paperID = int.Parse(Session["PaperID"].ToString());
         //试卷编号
        Params1[0] = DB.MakeInParam("@PaperID", SqlDbType.Int, 4, paperID);
         //题目类型
        Params1[1] = DB.MakeInParam("@Type", SqlDbType.VarChar, 10, "单选题");
        DataSet ds1 = DB.GetDataSet("Proc_PaperDetail",Params1);
        GridView1.DataSource = ds1;
        GridView1.DataBind();
       ((Label)GridView1.HeaderRow.FindControl("Label27")).Text = ((Label)GridView1.Rows [0].FindControl
("Label4")).Text;
        //提取试卷中的多选题
        SqlParameter[] Params2 = new SqlParameter[2];
         //试卷编号
        Params2[0] = DB.MakeInParam("@PaperID", SqlDbType.Int, 4, paperID);
        //题目类型
        Params2[1] = DB.MakeInParam("@Type", SqlDbType.VarChar, 10, "多选题");
        DataSet ds2 = DB.GetDataSet("Proc_PaperDetail", Params2);
        GridView2.DataSource = ds2;
        GridView2.DataBind();
        ((Label)GridView2.HeaderRow.FindControl("Label28")).Text = ((Label)GridView2.Rows[0].FindControl
("Label8")).Text;
        //提取试卷中的判断题
        SqlParameter[] Params3 = new SqlParameter[2];
        //试卷编号
        Params3[0] = DB.MakeInParam("@PaperID", SqlDbType.Int, 4, paperID);
          //题目类型
```

```
            Params3[1] = DB.MakeInParam("@Type", SqlDbType.VarChar, 10, "判断题");
            DataSet ds3 = DB.GetDataSet("Proc_PaperDetail", Params3);
            GridView3.DataSource = ds3;
            GridView3.DataBind();
            ((Label)GridView3.HeaderRow.FindControl("Label29")).Text = ((Label)GridView3.Rows[0].FindControl
("Label12")).Text;
            //提取试卷中的填空题
            SqlParameter[] Params4 = new SqlParameter[2];
             //试卷编号
            Params4[0] = DB.MakeInParam("@PaperID", SqlDbType.Int, 4, paperID);
             //题目类型
            Params4[1] = DB.MakeInParam("@Type", SqlDbType.VarChar, 10, "填空题");
            DataSet ds4 = DB.GetDataSet("Proc_PaperDetail", Params4);
            GridView4.DataSource = ds4;
            GridView4.DataBind();
            ((Label)GridView4.HeaderRow.FindControl("Label30")).Text = ((Label)GridView4.Rows[0].FindControl
("Label17")).Text;
            //提取试卷中的问答题
            SqlParameter[] Params5 = new SqlParameter[2];
             //试卷编号
            Params5[0] = DB.MakeInParam("@PaperID", SqlDbType.Int, 4, paperID);
            //题目类型
            Params5[1] = DB.MakeInParam("@Type", SqlDbType.VarChar, 10, "问答题");
            DataSet ds5 = DB.GetDataSet("Proc_PaperDetail", Params5);
            GridView5.DataSource = ds5;
            GridView5.DataBind();
            ((Label)GridView5.HeaderRow.FindControl("Label31")).Text = ((Label)GridView5.Rows[0].FindControl
("Label37")).Text;
        }
```

单击“交卷”按钮，首先通过 Session 变量获取学生 ID 和其选择的科目及套题信息，然后调用 getCom 方法来执行提交考卷操作，其关键代码如下。

例程 16 代码位置：光盘\mr\05\在线考试模块\WEB\UserTest.aspx.cs

```
protected void imgBtnSubmit_Click(object sender, ImageClickEventArgs e)
{
    //调用自定义方法tijiao()提交试卷
    this.tijiao();
    //隐藏提交按钮
    imgBtnSubmit.Visible = false;
}
```

自定义方法 tijiao()，用于提交考生试卷并计算各种题型的分值和对考生答案与数据库答案进行对比等操作，具体实现代码如下。

例程 17 代码位置：光盘\mr\05\在线考试模块\WEB\UserTest.aspx.cs

```
public void tijiao()//提交试卷的方法
{
    int paperid = Convert.ToInt32(Session["PaperID"].ToString());
    //实例化公共类DataBase
    DataBase db = new DataBase();
    //取出单选题的每题分值
    int singlemark = int.Parse(((Label)GridView1.Rows[0].FindControl("Label4")).Text);
    foreach (GridViewRow dr in GridView1.Rows)
    {
  string str = "";
  if (((RadioButton)dr.FindControl("RadioButton1")).Checked)
  {
      str = "A";
  }
  else if (((RadioButton)dr.FindControl("RadioButton2")).Checked)
  {
      str = "B";
```

```
        }
        else if (((RadioButton)dr.FindControl("RadioButton3")).Checked)
        {
            str = "C";
        }
        else if (((RadioButton)dr.FindControl("RadioButton4")).Checked)
        {
            str = "D";
        }
        //应用FindControl查找到ID为Label40的Lable控件
        int titleid = int.Parse(((Label)dr.FindControl("Label40")).Text);
        //定义一个将考生答的单选题信息存储到数据库的SQL语句
        string single = "insert into UserAnswer(UserID,PaperID,Type,TitleID,Mark,UserAnswer,ExamTime) values('" + lab
User.Text + "','" + paperid + "','单选题','" + titleid + "','" + singlemark + "','" + str + "','" + DateTime.Now.ToString() + "')";
            //调用公共类中的Insert执行SQL语句
            db.Insert(single);
        }
        //取出多选题每题分值
        int multimark = int.Parse(((Label)GridView2.Rows[0].FindControl("Label8")).Text);
        //对多选题每题进行判断用户选择答案
        foreach (GridViewRow dr in GridView2.Rows)
        {
            string str = "";
            if (((CheckBox)dr.FindControl("CheckBox1")).Checked)
            {
                str += "A";
            }
            if (((CheckBox)dr.FindControl("CheckBox2")).Checked)
            {
                str += "B";
            }
            if (((CheckBox)dr.FindControl("CheckBox3")).Checked)
            {
                str += "C";
            }
            if (((CheckBox)dr.FindControl("CheckBox4")).Checked)
            {
                str += "D";
                    }
                    int titleid = int.Parse(((Label)dr.FindControl("Label41")).Text);
                    string Multi = "insert into UserAnswer(UserID,PaperID,Type,TitleID,Mark,UserAnswer,ExamTime) values('" +
labUser.Text + "','" + paperid + "','多选题','" + titleid + "','" + multimark + "','" + str + "','" + DateTime.Now.ToString() + "')";
                    db.Insert(Multi);
                }
                //取出判断题每题分值
                int judgemark = int.Parse(((Label)GridView3.Rows[0].FindControl("Label12")).Text);
                //对判断题每题进行判断用户选择答案
                foreach (GridViewRow dr in GridView3.Rows)
                {
                    string str = Convert.ToString(false);
                    if (((CheckBox)dr.FindControl("CheckBox5")).Checked)
                    {
                        str = Convert.ToString(true);
                    }
                    int titleid = int.Parse(((Label)dr.FindControl("Label42")).Text);
                    string Judge = "insert into UserAnswer(UserID,PaperID,Type,TitleID,Mark,UserAnswer,ExamTime) values('" +
labUser.Text + "','" + paperid + "','判断题','" + titleid + "','" + judgemark + "','" + str + "','" + DateTime.Now.ToString() + "')";
                    db.Insert(Judge);
                }
                //取出填空题每题分值
                int fillmark = int.Parse(((Label)GridView4.Rows[0].FindControl("Label17")).Text);
```

```
                foreach (GridViewRow dr in GridView4.Rows)
                {
                    string str = "";
                    str = ((TextBox)dr.FindControl("TextBox1")).Text.Trim();
                    int titleid = int.Parse(((Label)dr.FindControl("Label43")).Text);
                    string Fill = "insert into UserAnswer(UserID,PaperID,Type,TitleID,Mark,UserAnswer,ExamTime) values('" +
labUser.Text + "','" + paperid + "','填空题','" + titleid + "','" + fillmark + "','" + str + "','" + DateTime.Now.ToString() + "')";
                    db.Insert(Fill);
                }
                //取出问答题每题分值
                int quemark = int.Parse(((Label)GridView5.Rows[0].FindControl("Label37")).Text);
                foreach (GridViewRow dr in GridView5.Rows)
                {
                    string str = "";
                    str = ((TextBox)dr.FindControl("TextBox2")).Text.Trim();
                    int titleid = int.Parse(((Label)dr.FindControl("Label44")).Text);
                    string Que = "insert into UserAnswer(UserID,PaperID,Type,TitleID,Mark,UserAnswer,ExamTime) values('" +
labUser.Text + "','" + paperid + "','问答题','" + titleid + "','" + quemark + "','" + str + "','" + DateTime.Now.ToString() + "')";
                    db.Insert(Que);
                }
                Response.Write("<script language=javascript>alert('试卷已提交!');location='StudentIndex.aspx'</script>");
            }
```

5.5 用户信息管理页

5.5.1 用户信息管理页概述

用户信息管理页（UserManage.aspx）可对 3 种用户进行管理，即考生、教师和管理员进行管理（该页只能由管理员进行管理，教师无此管理权限）。该页面可根据用户 ID 及用户姓名进行模糊及精确查询，用户查询时会显示查询过程中的进度条。可对用户信息进行批量删除操作。还可以添加新的用户角色。另外，“在用户管理”这页中，还增加了一个“重置密码”的功能，如果用户（考生或教师）忘记密码了，管理员则可以把用户的密码重置为 6 位的随机数字。

该页面的运行效果如图 5.12 所示。

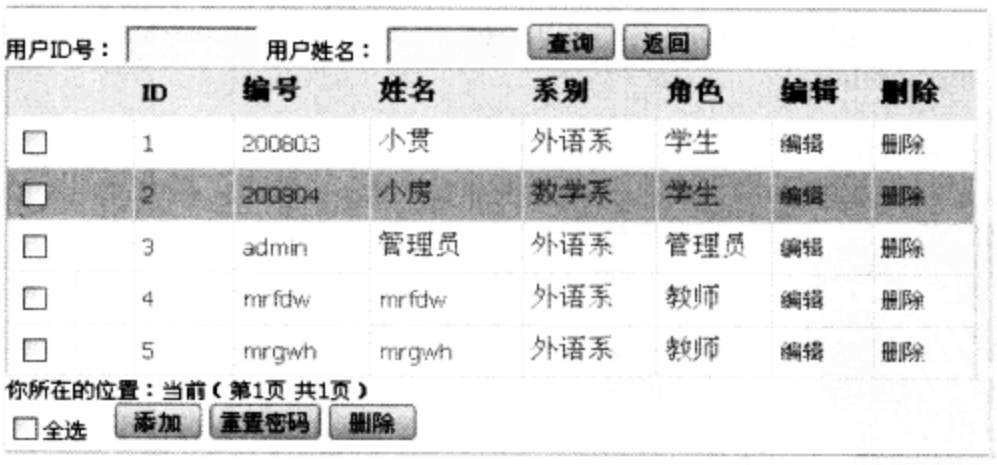


图 5.12 考试题目管理页

5.5.2 用户信息管理页实现过程

1. 设计步骤

（1）在应用程序中新建 1 个文件夹，将其命名为 Web。在该文件夹创建 1 个 Web 窗体，命名为 UserManage.aspx，用于管理考生、教师和管理员信息。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 5 个 ImageButton 控件、2 个 TextBox 控件、1 个 GridView 控件、1 个 Hyperlink 控件和 1 个 CheckBox 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 5.3 所示。

表 5.3　　kaoshi_timu.aspx 页面用到的主要控件

控件类型	控件名称	主要属性设置	用　途
ImageButton	ImageButtonQuery	无	根据用户查询条件查询相关信息
	ImageButton1	无	返回到用户管理页
	ImageButtonResetPassword	无	重置用户密码
	ImageButtonDelete	无	执行删除操作
HyperLink	HyperLinkAdd	无	链接用户添加页面
GridView	GridView1	AutoGenerateColumns 属性设置为 False、AllowPaging 属性设置为 True、DataKeyNames 属性设置为 UserID	用来绑定考生、教师和管理员信息
CheckBox	ChkSelectAll	AtuoPostBack 设置为 True、Text 属性设置为“全选”	执行全选操作

2. 实现代码

在页面 Page_Load 事件中，主要调用自定义方法 GridViewBind()绑定 GridView 控件中的数据，具体实现代码如下。

例程 18　代码位置：光盘\mr\05\在线考试模块\WEB\UserManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)//判断用户是否首次加载
    {
        //应用Session对象存储用户登录名
        string loginName = Session["userID"].ToString();
        //实例化公共类User
        Users user = new Users();
        //调用公共类中的LoadData()方法获取用户信息
        user.LoadData(loginName);
        labUser.Text = user.UserName;
        //调用自定义方法绑定GreidView控件中信息
        GridViewBind();
    }
    //当单击“重置密码”按钮时弹出确认对话框
    ImageButtonResetPassword.Attributes.Add("onClick", "javascript:return confirm('确定重置密码?');");
   //当单击GridView控件的“删除”列时弹出确认对话框
 ImageButtonDelete.Attributes.Add("onclick", "javascript:return confirm('确定删除?');");
}
```

自定义方法 GridViewBind()用于从数据库中获取数据绑定到 GridView 控件中，并显示当前分页信息。具体的实现代码如下。

例程 19　代码位置：光盘\mr\05\在线考试模块\WEB\UserManage.aspx.cs

```
private void GridViewBind()
    {
        //创建Users类对象user
        Users user = new Users();
```

```
            //使用Users类QueryUsers方法查询所有用户信息
            DataSet myds = user.QueryUsers();
            //设置GridView控件数据源
            GridView1.DataSource = myds;
            //GridView控件绑定数据
            GridView1.DataBind();
            //分页显示当前页信息量
            LabelPageInfo.Text = "你所在的位置：当前（第" + (GridView1.PageIndex + 1).ToString() + "页 共" +
GridView1.PageCount.ToString() + "页）";
        }
```

单击 GridView 控件中的编辑选项，可对用户列表中的用户姓名、部门和角色进行更改操作，其中用户角色和部门的更改可通在 GridView 控件模板列中添加的 DropDownList 控件来实现，绑定用户角色和部门的下拉列表信息主要编写在 GridView 控件的 RowDataBound 事件中，具体的实现代码如下。

例程 20 代码位置：光盘\mr\05\在线考试模块\WEB\UserManage.aspx.cs

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    //更改用户角色操作
    if (((DropDownList)e.Row.FindControl("myddlrole")) != null)
    {
        DropDownList myddlrole = (DropDownList)e.Row.FindControl("myddlrole");
        //生成 DropDownList 的值，绑定数据
        string connStr = "Data Source=(local);DataBase=OnLineExam;User ID=sa;Password=;";
        DataSet myds = new DataSet();
        //建立数据库连接
        SqlConnection conn = new SqlConnection(connStr);
        if (conn.State.ToString() == "Closed") conn.Open();
        SqlDataAdapter da = new SqlDataAdapter("Proc_RoleList", conn);
        da.Fill(myds);
        if (conn.State.ToString() == "Open") conn.Close();
        //设置用户角色表的默认视图作为 DropDownList控件的数据源
        myddlrole.DataSource = myds.Tables[0].DefaultView;
        myddlrole.DataTextField = "RoleName";
        myddlrole.DataValueField = "RoleId";
        //绑定 DropDownList中数据
        myddlrole.DataBind();
    }
    //更改用户部门操作
    if (((DropDownList)e.Row.FindControl("ddlDepartment")) != null)
    {
        DropDownList ddldepartment = (DropDownList)e.Row.FindControl("ddlDepartment");
        //生成 DropDownList 的值，绑定数据
        string connStr = "Data Source=(local);DataBase=OnLineExam;User ID=sa;Password=;";
        //创建一个数据集ds
        DataSet ds = new DataSet();
        //打开数据库连接
        SqlConnection conn = new SqlConnection(connStr);
        if (conn.State.ToString() == "Closed") conn.Open();
        //创建并实例化数据适配器
        SqlDataAdapter da = new SqlDataAdapter("Proc_DepartmentList", conn);
        //应用数据适配器的Fill方法填充DataSet数据集
        da.Fill(ds);
        if (conn.State.ToString() == "Open") conn.Close();
        //设置用户部门表的默认视图作为 DropDownList控件的数据源
        ddldepartment.DataSource = ds.Tables[0].DefaultView;
        ddldepartment.DataTextField = "DepartmentName";
```

```
            ddldepartment.DataValueField = "DepartmentId";
            ddldepartment.DataBind();
        }
        int i;
        //执行循环，保证每条数据都可以更新
        for (i = 0; i < GridView1.Rows.Count; i++)
            {
                //首先判断是否是数据行
                if (e.Row.RowType == DataControlRowType.DataRow)
                {
                    //当鼠标停留时更改背景色
                    e.Row.Attributes.Add("onmouseover", "c=this.style.backgroundColor;this.style.backgroundColor='Aqua'");
                    //当鼠标移开时还原背景色
                    e.Row.Attributes.Add("onmouseout", "this.style.backgroundColor=c");
                }
            }
        }
```

双击设计页面中的“查询”按钮，触发 ImageButtonQuery_Clickg 事件，主要执行用户查询操作，在执行查询用户操作时，将显示等待进度条（该技术在本模块的关键技术中已列出，由于篇幅有限，这里省去其代码），实现代码如下：

例程 21 代码位置：光盘\mr\05\在线考试模块\WEB\UserManage.aspx.cs

```
protected void ImageButtonQuery_Click(object sender, ImageClickEventArgs e)
    {
        ..... //省略等待进度条代码
        //创建一个哈希表myqueryItems
        Hashtable myqueryItems = new Hashtable();
        //向创建的哈希表中添加用户名及用户编号
        myqueryItems .Add("UserID", tbxUserID.Text.Trim());
        myqueryItems .Add("UserName", tbxUserName.Text.Trim());
        //创建一个内存表dt，并调用公共类中的Users的QueryUsers方法执行用户查询工作
        DataTable dt = OnLineExam.BusinessLogicLayer.Users.QueryUsers(myqueryItems );
        if (dt.Rows.Count >= 1)
        {
            //将创建的内存表作为GridView控件的数据源
            GridView1.DataSource = dt;
            //绑定控件数据
            GridView1.DataBind();
        }
        else
        {
            //弹出提示框
            Response.Write("<script language=javascript>alert('没有这个用户!')</script>");
        }
    }
```

在该页面中，管理员可以为用户重设密码，如果考生或教师忘记了自己的用户密码，可以与管理员进行联系，管理员将会给用户重新设置用户密码，重设的密码为随机产生的 6 位数字，具体实现的代码如下。

例程 22 代码位置：光盘\mr\05\在线考试模块\WEB\UserManage.aspx.cs

```
protected void ImageButtonResetPassword_Click(object sender, ImageClickEventArgs e)
{
    int numOfChecked = 0;
    //在GridView控件中循环查找CheckBox控件
    for (int i = 0; i < this.GridView1.Rows.Count; i++)
    {
        bool isChecked = ((CheckBox)GridView1.Rows[i].FindControl("chkSelected")).Checked;
```

```
                if (isChecked)
                {
                    numOfChecked++;
                }
            }
            //判断CheckBox是否被选中
        if (numOfChecked == 1)
        {
            //用for语句循环查找GridView控件中的CheckBox控件
    for (int i = 0; i < this.GridView1.Rows.Count; i++)
            {
                bool isChecked = ((CheckBox)GridView1.Rows[i].FindControl("chkSelected")).Checked;
                    if (isChecked)
                    {
                        string UserID = ((Label)GridView1.Rows[i].FindControl("Label1")).Text;
                        //创建随机数
                        Random ran = new Random();
                        string newPassword = (ran.Next(999999).ToString().PadLeft(6, '8'));      //随机生成一个密码
                        //创建Users对象user
                        Users user = new Users();
                        //取出数据库中加密的用户密码
                        string pwdMd5 = System.Web.Security.FormsAuthentication.HashPasswordForStoringIn ConfigFile
(newPassword, "MD5").ToString();
                        //将用户密码与数据库中的MD5加密密码进行比较
                        user.UserPwd = pwdMd5.ToString().Trim();
                        //更改用户密码
                        if (user.ModifyPassword(UserID))
                        {
                            Response.Write("<Script language=JavaScript>alert('" + UserID + "的密码已经重置，新密
码为【" + newPassword + "】。');</Script>");
                        }
                        else//修改密码失败
                        {
                            Response.Write("<Script language=JavaScript>alert('" + UserID + "重置密码失败!');</Script>");
                        }
                    }
                }
            }
        }
        else
            {
                Response.Write("<Script language=JavaScript>alert('您只能选择一个用户!');</Script>");
                return;
            }
        }
```

5.6 试卷出题页

5.6.1 试卷出题页概述

试卷制定页由 PaperSetup.aspx 页实现，该页主要实现的是人工制定试卷，试卷的试题类型共分为单选、多选、判断、填空和问答 5 类，每种类型的试题在该试题题型下全部显示，在制定试卷时，可通过是否选中试题前的 CheckBox 控件来选择性地出题（该页还设置了一个全选按钮可全选所列试卷试题）。另外，此页还应用了 Ajax 无刷新环境，在从数据库中检测试卷名称是否存在时，页面实现的是无刷新效果。该页实际运行效果如图 5.13 所示。

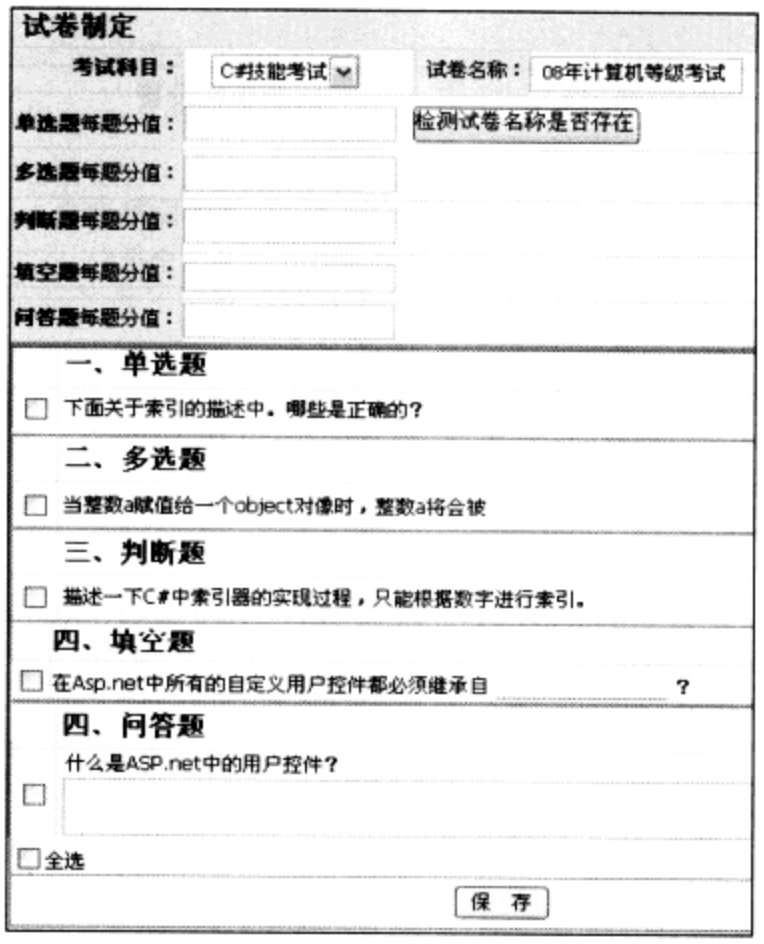


图 5.13　试卷制定页

5.6.2　试卷出题页实现过程

1. 设计步骤

（1）在应用程序中找到命名为 Web 文件夹。在该文件夹创建 1 个 Web 窗体，命名为 PaperSetup.aspx，用于人工制定试卷信息。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 5 个 ImageButton 控件、2 个 TextBox 控件、1 个 GridView 控件、1 个 Hyperlink 控件和 1 个 CheckBox 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 5.4 所示。

表 5.4　**kaoshi_timu.aspx** 页面用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用　途
TextBox	txtSingleFen	无	用来设置单选题分数值
	txtMultiFen	无	用来设置多选题分数值
	txtJudgeFen	无	用来设置判断题分数值
	txtFillFen	无	用来设置填空题分数值
	txtQuestionFen	无	根据考生答题情况计算分数
	txtPaperName	无	添加试卷名称
GridView	GridView1	AutoGenerateColumns 属性设置为 False	绑定数据库中单选题信息
	GridView2	AutoGenerateColumns 属性设置为 False	绑定数据库中多选题信息
	GridView3	AutoGenerateColumns 属性设置为 False	绑定数据库中判断题信息
	GridView4	AutoGenerateColumns 属性设置为 False	绑定数据库中填空题信息
	GridView5	AutoGenerateColumns 属性设置为 False	绑定数据库中问答题信息

续表

控件类型	控件名称	主要属性设置	用途
Button	imgBtnSave	无	用于保存试卷制定信息
CheckBox	chkSelectAll	AutoPostBack 属性设置为 True Text 属性设置为“全选”	全选试题功能

2. 实现代码

在页面 Page_Load 事件中，首先应用 Session 对象保存登录用户名，然后实例化公共类 Users，调用该公共类中的 LoadData 判断用户是否登录。如果用户登录，则分别调用 PaperData()和 BindData()两个自定义方法，实现对考试试卷的初使化和根据设置自动生成考试试卷，具体的实现的代码如下。

例程 23 代码位置：光盘\mr\05\在线考试模块\WEB\PaperSetup.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //应用Session对象存储用户名
        string loginName = Session["userID"].ToString();
        //实例化公共类Users
        Users user = new Users();
        user.LoadData(loginName);
        labUser.Text = user.UserName;
        //调用自定义方法PaperData()初始化考试科目
        PaperData();
        //根据设置调用自定义方法BindData()自动生成试卷
        BindData();
    }
}
```

自定义方法 PaperData()，主要用来初始化考试科目，考试科目通过绑定下拉列表框中数据来实现，实现代码如下。

例程 24 代码位置：光盘\mr\05\在线考试模块\WEB\PaperSetup.aspx.cs

```
protected void PaperData()
{
    //实例化考试科目对象Course
    Course papercs = new Course();
    //调用Course公共类中QueryCourse()来查询考试科目信息并保存在创建的DataSet对象ds中
    DataSet ds = papercs.QueryCourse();
    //设置考试科目列表框的数据源
    MyddlCourse.DataSource = ds;
    //DataTextField显示Name字段值
    MyddlCourse.DataTextField = "Name";
    //DataValueField显示ID字段值
    MyddlCourse.DataValueField = "ID";
    //绑定数据
    MyddlCourse.DataBind();
}
```

自定义方法 BindData()根据用户设置自动生成考试试卷，在该方法中，首先实例化公共类 DataBase，接着根据参数分别设置单选题、多选题、判断题、填空题和问答题的 SQL 语句，然后调用 DataBase 类方法 GetDataSetSql 方法执行这些查询数据，最后从数据库中绑定数据到相应的 GridView 控件中，具体实现的代码如下。

例程 25 代码位置：光盘\mr\05\在线考试模块\WEB\PaperSetup.aspx.cs

```
protected void BindData()
 {
     //创建DataBase类对象
```

```
DataBase database = new DataBase();
        //根据参数设置查询单选题SQL语句
        string GridView1StrSQL = "select * from SingleProblem";
        //调用DataBase类方法GetDataSetSql方法查询数据
        DataSet myds1 = database.GetDataSetSql(GridView1StrSQL);
        //为单选题GridView控件设置数据源
        GridView1.DataSource = myds1.Tables[0].DefaultView;
        //绑定数据
        GridView1.DataBind();
        //根据参数设置查询多选题SQL语句
        string GridView2StrSQL = "select * from MultiProblem";
        //调用DataBase类方法GetDataSetSql方法查询数据
        DataSet myds2 = database.GetDataSetSql(GridView2StrSQL);
        //为多选题GridView控件设置数据源
        GridView2.DataSource = myds2.Tables[0].DefaultView ;
        //绑定数据
        GridView2.DataBind();
        //根据参数设置查询判断题SQL语句
        string GridView3StrSQL = "select * from JudgeProblem";
        //调用DataBase类方法GetDataSetSql方法查询数据
        DataSet myds3 = database.GetDataSetSql(GridView3StrSQL);
        //为判断题GridView控件设置数据源
        GridView3.DataSource = myds3.Tables[0].DefaultView;
        //绑定数据
        GridView3.DataBind();
        //根据参数设置查询填空题SQL语句
        string GridView4StrSQL = "select * from FillBlankProblem";
        //调用DataBase类方法GetDataSetSql方法查询数据
        DataSet myds4 = database.GetDataSetSql(GridView4StrSQL);
        //为填空题GridView控件设置数据源
        GridView4.DataSource = myds4.Tables[0].DefaultView;
        //绑定数据
        GridView4.DataBind();
        //根据参数设置查询问答题SQL语句
        string GridView5StrSQL = "select * from QuestionProblem";
        //调用DataBase类方法
        DataSet myds5 = database.GetDataSetSql(GridView5StrSQL); GetDataSetSql方法查询数据
        //为问答题GridView控件设置数据源
        GridView5.DataSource = myds5.Tables[0].DefaultView;
        //绑定数据
        GridView5.DataBind();
    }
```

双击该页面中的“保存”按钮，触发其 imgBtnSave_Click 事件，该事件代码中主要是将自动生成的试卷保存到数据库中，如生成的单选题信息、多选题信息和判断题信息等。具体实现的代码如下。

例程 26 代码位置：光盘\mr\05\在线考试模块\WEB\PaperSetup.aspx.cs

```
    protected void imgBtnSave_Click(object sender, ImageClickEventArgs e)
    {
        //实例化公共类DataBase
        DataBase database = new DataBase();
        //实例化公共类AjaxCommond
        AjaxCommond ac = new AjaxCommond();
        string insertExamPage = "insert into Paper(CourseID,PaperName,PaperState) values(" + int.Parse(MyddlCourse.Selected
Value) + ",'" + txtPaperName.Text + "',1) SELECT @@IDENTITY as id";
            //保存试卷，并返回自动生成的试卷编号
            int autopageID = GetIDInsert(insertExamPage);
            if (autopageID > 0)
            {
                //生成单选题信息，并保存在数据库中
```

```
                for (int i = 0; i < this.GridView1.Rows.Count; i++)
                {
                    bool isChecked = ((CheckBox)GridView1.Rows[i].FindControl("chkSelect1")).Checked;
                    if (isChecked)
                    {
                        string sqlstr1 = ((Label)GridView1.Rows[i].FindControl("Label3")).Text;
                        string singlepaper = "insert into PaperDetail(PaperID,Type,TitleID,Mark) values(" + autopageID + ",'
单选题'," + sqlstr1 + "," + int.Parse(txtSingleFen.Text) + ")";
                        database.Insert(singlepaper);
                    }
                }
                //生成多选题信息，并保存在数据库中
                for (int i = 0; i < this.GridView2.Rows.Count; i++)
                {
                    bool isChecked = ((CheckBox)GridView2.Rows[i].FindControl("chkSelect2")).Checked;
                    if (isChecked)
                    {
                        string sqlstr2 = ((Label)GridView2.Rows[i].FindControl("Label6")).Text;
                        string multipaper = "insert into PaperDetail(PaperID,Type,TitleID,Mark) values(" + autopageID
+ ",'多选题'," + sqlstr2 + "," + int.Parse(txtMultiFen.Text) + ")";
                        database.Insert(multipaper);
                    }
                }
                //生成判断题信息，并保存在数据库中
                for (int i = 0; i < this.GridView3.Rows.Count; i++)
                {
                    bool isChecked = ((CheckBox)GridView3.Rows[i].FindControl("chkSelect3")).Checked;
                    if (isChecked)
                    {
                        string sqlstr3 = ((Label)GridView3.Rows[i].FindControl("Label7")).Text;
                        string judgepaper = "insert into PaperDetail(PaperID,Type,TitleID,Mark) values(" + autopageID
+ ",'判断题'," + sqlstr3 + "," + int.Parse(txtJudgeFen.Text) + ")";
                        database.Insert(judgepaper);
                    }
                }
                //生成填空题信息，并保存在数据库中
                for (int i = 0; i < this.GridView4.Rows.Count; i++)
                {
                    bool isChecked = ((CheckBox)GridView4.Rows[i].FindControl("chkSelect4")).Checked;
                    if (isChecked)
                    {
                        string sqlstr4 = ((Label)GridView4.Rows[i].FindControl("Label8")).Text;
                        string fillpaper = "insert into PaperDetail(PaperID,Type,TitleID,Mark) values(" + autopageID +
",'填空题'," + sqlstr4 + "," + int.Parse(txtFillFen.Text) + ")";
                        database.Insert(fillpaper);
                    }
                }
                //生成问答题信息，并保存在数据库中
                for (int i = 0; i < this.GridView5.Rows.Count; i++)
                {
                    bool isChecked = ((CheckBox)GridView5.Rows[i].FindControl("chkSelect5")).Checked;
                    if (isChecked)
                    {
                        string sqlstr5 = ((Label)GridView5.Rows[i].FindControl("Label23")).Text;
                        string quepaper = "insert into PaperDetail(PaperID,Type,TitleID,Mark) values(" + autopageID +
",'问答题'," + sqlstr5 + "," + int.Parse(txtQuestionFen.Text) + ")";
                        database.Insert(quepaper);
                    }
                }
```

```
        }
        //弹出Ajax环境中的提示对话框
        ac.OpenDialogForButton((Button)sender, "数据保存成功！");
        //跳转到试卷维护页
        Response.Redirect("PaperLists.aspx");
    }
```

5.7 试卷评审页

5.7.1 试卷评审页概述

考生考完试提交试卷后，教师即可对考生试卷进行评阅，试卷评阅由 UserPaper.aspx 页实现，教师进入该试卷评审页后只需对考生所答的问答题情况酌情给分，然后单击“考生问答题计分”按钮计算出问答题分数。考生所答的其他类型试题，如单选题、多选题、填空题等系统会自动计算出，同时教师可对考生试卷进行简单的评语，完成以上操作后，单击页面中的“保存”按钮，计算出考生的总成绩。

该页面实际运行效果如图 5.14 所示。

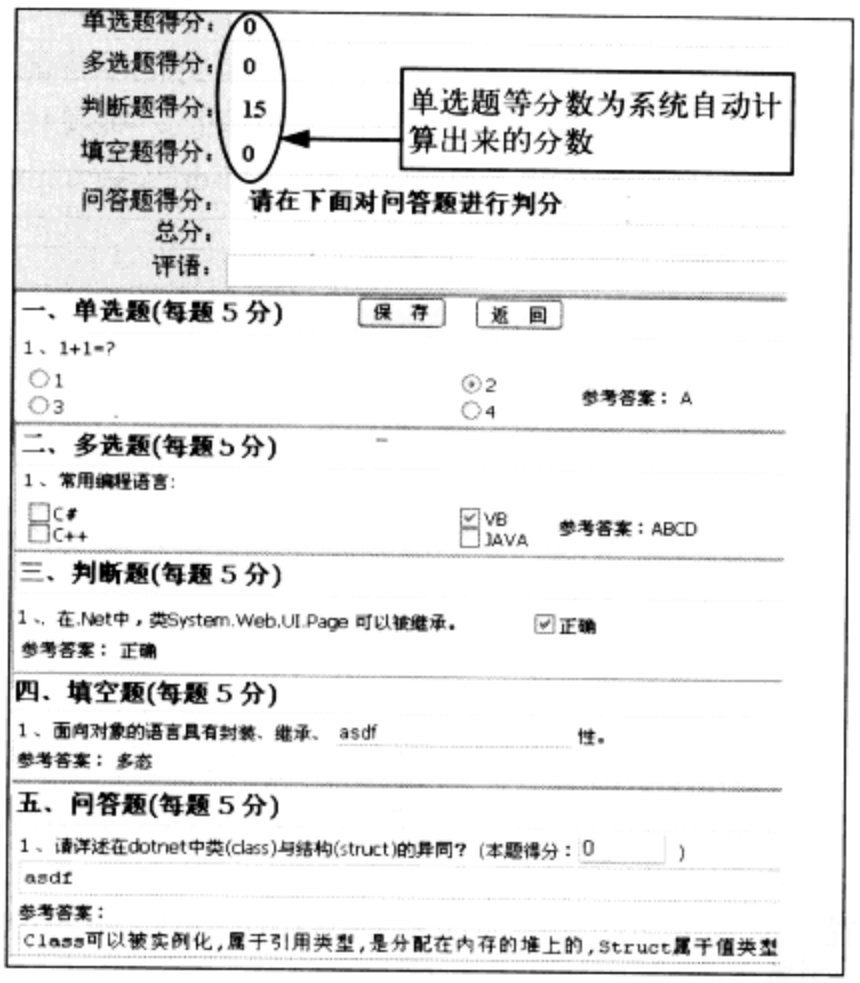


图 5.14 试卷评阅页面

5.7.2 试卷评审页实现过程

1. 设计步骤

（1）在应用程序中找到命名为 Web 的文件夹。在该文件夹创建 1 个 Web 窗体，命名为 UserPaper.aspx，用于对考生试卷进行评阅。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 5 个 GridView 控件和 3 个 ImageButton 控件，通过属性窗口设置控件的属性。页面中各个控件的属性设置及其用途如表 5.5 所示。

表 5.5　　kaoshi_timu.aspx 页面用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用　途
GridView	GridView1	AutoGenerateColumns 属性设置为 False	绑定考生存储到数据库中单选题信息
	GridView2	AutoGenerateColumns 属性设置为 False	绑定考生存储到数据库中多选题信息
	GridView3	AutoGenerateColumns 属性设置为 False	绑定考生存储到数据库中判断题信息
	GridView4	AutoGenerateColumns 属性设置为 False	绑定考生存储到数据库中填空题信息
	GridView5	AutoGenerateColumns 属性设置为 False	绑定考生存储到数据库中问答题信息
Button	ImgBtnSave	无	用于保存考生试卷成绩信息
	imgBtnReturn	无	返回试卷评阅页
	Button1	无	用于计算考生问答题分数

2. 实现代码

在页面 Page_Load 事件中，首先判断页面是否为首次加载，然后实例化公共类 Users，并调用该类中的 LoadData 来判断用户是否登录，如果用户登录将把用户显示在指定的 Lable 控件中，最后调用自定义方法 PaperData()来绑定数据库中试题信息，实现代码如下。

例程 27　代码位置：光盘\mr\05\在线考试模块\WEB\UserPaper.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //应用Session对象存储用户名
        string loginName = Session["userID"].ToString();
        //实例化公共类Users
        Users user = new Users();
        user.LoadData(loginName);
        labUser.Text = user.UserName;
        //调用自定义方法绑定数据库中试题信息
        PaperData();
    }
}
```

双击页面中的“问答题计分”按钮，触发其 Button1_Click 事件，教师根据考生答题的具体情况进行给分，触发此按钮事件后将计算出问答题分数，考生最终的问答题分数将显示在问答题得分区域内，实现代码如下。

例程 28　代码位置：光盘\mr\05\在线考试模块\WEB\UserPaper.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
    int quemark = 0;
    int maxMark = int.Parse(((Label)GridView5.Rows[0].FindControl("Label21")).Text);
    int flag = 1;
    foreach (GridViewRow dr in GridView5.Rows)
    {
            if (int.Parse(((TextBox)dr.FindControl("tbxqueScore")).Text.Trim()) > maxMark)
            {
                lblQuestion.Text = "问答题每道的得分不能超过每道题的分数!";
                 flag =0;
            }
        }
       if(flag==1)
```

```
        {
            foreach (GridViewRow dr in GridView5.Rows)
            {
                    quemark = quemark + int.Parse(((TextBox)dr.FindControl("tbxqueScore")).Text.Trim());
                    queScore.Text = Convert.ToString(quemark);
                    sumScore.Text = Convert.ToString(Convert.ToInt32(sinScore.Text) + Convert.ToInt32(mulScore.Text) +
Convert.ToInt32(judScore.Text) + Convert.ToInt32(filScore.Text) + Convert.ToInt32(queScore.Text));
                }
                lblQuestion.Text = "";

            }
        }
```

双击页面中的“保存”按钮，触发其 imgBtnSave_Click 事件，将考生成绩保存到数据库中，教师计算完考生成绩后还可以给出考生简单的评语，具体实现的代码如下。

例程 29 代码位置：光盘\mr\05\在线考试模块\WEB\UserPaper.aspx.cs

```
protected void imgBtnSave_Click(object sender, ImageClickEventArgs e)
{
    //创建Scores类对象
    Scores insertExamScore = new Scores();
    insertExamScore.UserID = Request.QueryString["UserID"].ToString();
    insertExamScore.ExamTime = Convert.ToDateTime(lblExamtime.Text);
    insertExamScore.PaperID = int.Parse(Request.QueryString["PaperID"].ToString());
    insertExamScore.Score = Convert.ToInt32(sumScore.Text);
    insertExamScore.PingYu = tbxPingyu.Text;
    //实例化公共类Paper
    Paper mypaper = new Paper();
    mypaper.UserID= Request.QueryString["UserID"].ToString();
    mypaper.PaperID = int.Parse(Request.QueryString["PaperID"].ToString());
    mypaper.state = "已评阅";
    //使用CheckScore方法验证成绩是否存在
    if (!insertExamScore.CheckScore(insertExamScore.UserID, insertExamScore.PaperID))
    {
        //调用InsertByProc方法向数据库中插入成绩
        if (insertExamScore.InsertByProc())
        {
            mypaper.UpdateByProc(mypaper.UserID, mypaper.PaperID,mypaper.state);
            //给出成功提示信息
            lblMessage.Text = "考生成绩保存成功!";
        }
        //考试成绩保存失败
        else
        {
            lblMessage.Text = "考生成绩保存失败!";
        }
    }
    else
    {
        lblMessage.Text = "该用户的成绩已存在，请先删除成绩再评阅!";
    }
}
```

5.8 程序发布与调试

网站开发完成后最终的目的是将其发布到 Internet 上，以提供用户浏览访问。实现网站发布可以使用两种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

1．首先打开要发布的网站，在网站的项目名称上，单击鼠标右键，在弹出的快捷方式菜单中选择“发布网站”选项，如图 5.15 所示。

2．在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具，需要选择“...”路径按钮。如图 5.16 所示。

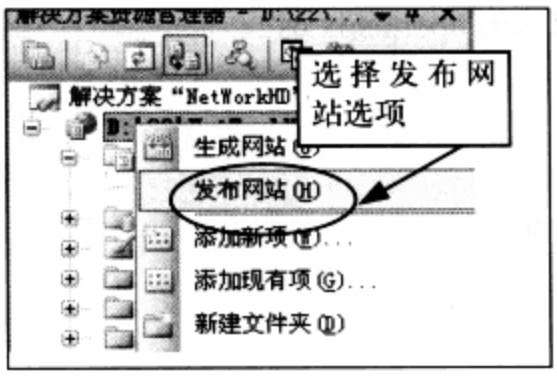


图 5.15　选择发布网站

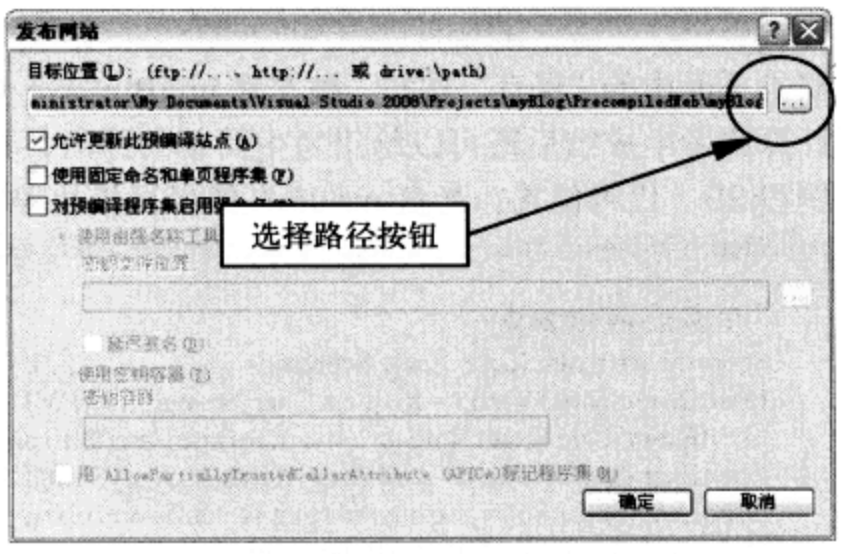


图 5.16　选择路径按钮

3．在弹出的窗口中，选择“FTP 站点”选项，在该选项中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码，如图 5.17 所示。填写完毕后，单击“打开”按钮将会返回“发布网站”窗口，在该窗口中单击“确定”按钮，网站就会发布到 Internet 上。

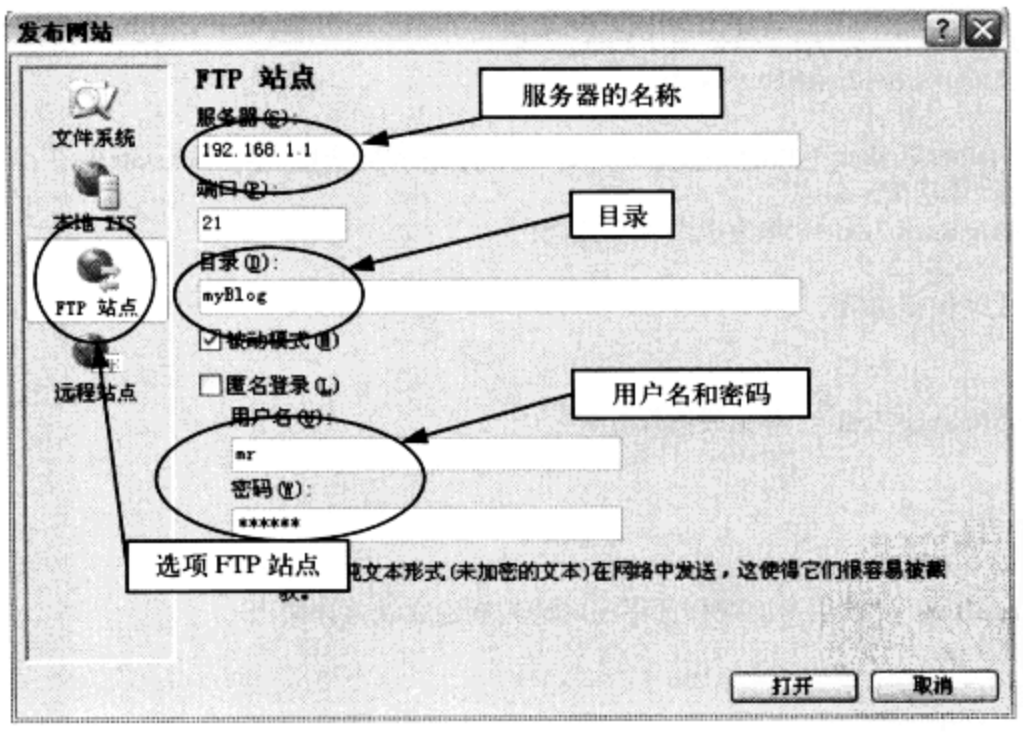


图 5.17　FTP 站点选项

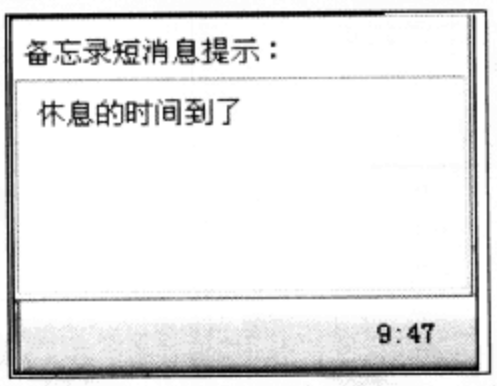
网站备忘录

第6章

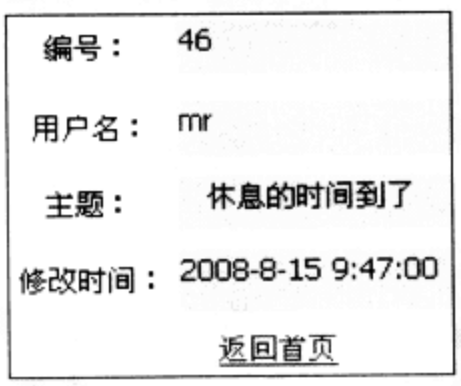
实例位置：光盘\mr\06\

随着网络技术的飞速发展，网络备忘录在网络上越来越受到人们的青睐。备忘录也是一个比较简单而且常见的功能，现在许多网站都有备忘录。但是使用 Ajax 技术实现的为数不多，并且弹出的备忘录的智能提示信息使用 JavaScript 脚本实现的，和以往弹出一个对话框相比备忘录中的智能提示信息更实用些。本章主要介绍如何利用 ASP.NET 3.5+Ajax+SQL Server 2000 快速开发一个备忘录程序。通过本章学习，读者可以学到以下内容。

右下脚弹出备忘提示信息

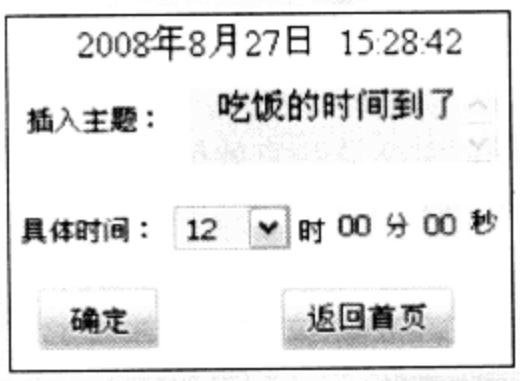


将数据绑定到文本控件中



单击“”按钮弹出日历控件选择日期

时间每分钟刷新一次



6.1 网站备忘录模块概述

6.1.1 功能概述

本模块首先通过登录页面进入网站备忘录的首页中，在首页中通过网站备忘录的网站上方的导航功能，进入“新建”、“检索”、“查询”、“修改”、“删除”、“重新登录”和“用户注册”详细的页面中。本模块主要介绍了一个简单的网站备忘录的具体实现过程，运行结果如图 6.1 所示。

图 6.1 网站备忘录首页

为了方便读者阅读和有效地利用本书附赠光盘中的实例，这里将给出网站首页面的各部分说明，如表 6.1 所示。

表 6.1 网站首页解析

区域	名 称	说 明	对 应 文 件
1	网站标头	主要用于网站的旗帜广告	Default.aspx
2	网站导航	主要用于网站的导航	Logo.ascx
3	显示登录用户名	主要用于显示当前登录的用户的用户名称	Default.aspx
4	日历控件	主要用于日历上的时间，当单击具体的日期将跳转到新建网站备忘录页面中	Default.aspx
5	网站脚标	主要用于显示技术服务热线等信息内容	Default.aspx

6.1.2 数据库设计

本系统采用了 SQL Server 2000 数据库，该数据库的名称为“db_memo”，在数据库中主要创建了 2 个数据表，分别为网站备忘录信息表（tb_memo）和用户信息表（tb_Login）。

tb_Login（用户信息表）表主要用于保存注册用户的基本信息，其结构如表 6.2 所示。

表 6.2　　用户信息表（tb_Login）的结构

字　段	类　型	长　度	是否可为空	说　明
id	int	4	否	用户编号
name	varchar	50	是	用户姓名
pwd	varchar	50	是	用户密码
truepwd	varchar	50	是	确认密码
E-mail	varchar	50	是	电子邮件
QQ	int	4	是	QQ
beizhu	text	16	是	备注信息

tb_ memo（网站备忘录信息表）表主要用于存储网站备忘录的基本信息，其结构如表 6.3 所示。

表 6.3　　网站备忘录信息表（tb_ memo）的结构

字　段	类　型	长　度	是否可为空	说　明
id	int	4	否	网站备忘录编号
subject	varchar	50	是	网站备忘录主题
alertTime	varchar	50	是	网站备忘录时间
name	varchar	50	是	姓名
pwd	varchar	50	是	密码

6.2　网站备忘录模块关键技术

6.2.1　向网站中添加公共类

在网站开发项目中以类的形式来组织、封装一些常用的方法和事件，将会在编程中起到事半功倍的效果。创建公共类可以减少重复代码的编写，并有利于代码维护。下面对 Memobwl 类做详细的介绍。

创建公共类的方法如下。

在解决方案资源管理器中右键单击项目文件名称，在弹出的快捷菜单中选择“添加新项”命令，打开“添加新项”对话框，在该对话框中选择“类”选项，在“名称”文本框中输入“SqlData.cs”，单击“确定”按钮即可。如图 6.2 所示。

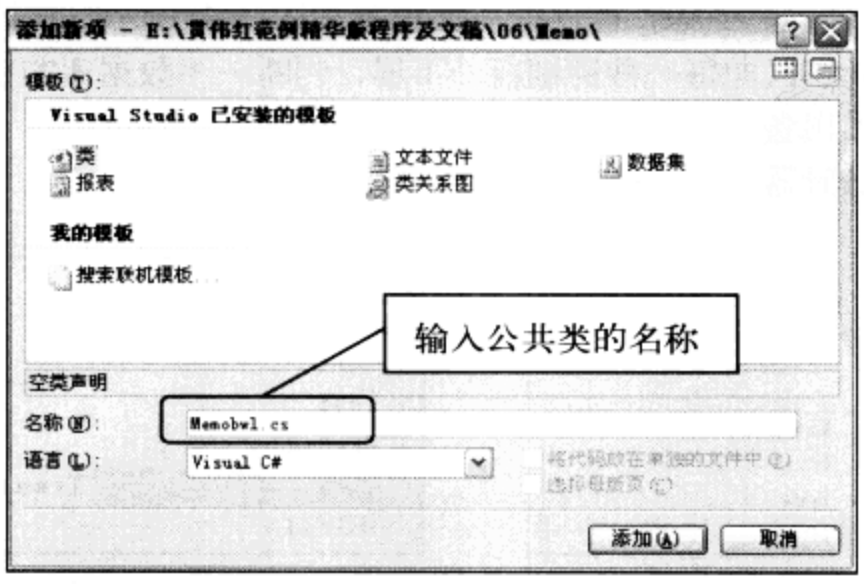


图 6.2　创建类文件

6.2.2 定时自动提示网站备忘信息

在网站备忘录程序中，当数据库的时间和系统的时间相同时，将在程序的右下角弹出一个备忘提示信息框。该备忘提示消息框将显示用户设置的备忘信息。运行效果如图 6.3 所示。

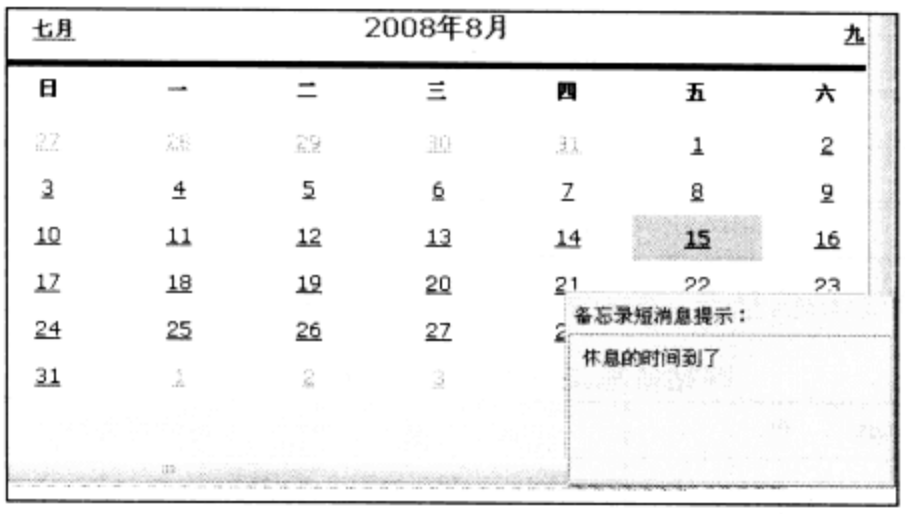


图 6.3 运行效果图

该提示框是在 JavaScript 中利用 window 对象中的 CreatePopup 对象来实现的。该代码在 blebHint.js 文件中实现。

调用提示框是在后台代码中实现的。在后台中首先判断数据库的时间是否和系统的时间相同。如果相同将弹出网站备忘录的提示对话框。实现每一秒钟进行时间的比较操作是在 Ajax 中提供的计时器 Timer 控件实现的，该计时器 Timer 控件，能够指定一个时间间隔和一个 Tick 事件，在每一次时间间隔到达之后，都触发 Tick 事件。下面将详细介绍计时器 Timer 控件的常用属性和方法及有关事件。

- 在计时器 Timer 控件中两个常用的属性。
- Enabled 属性 ：该属性表示控件是否可用。
- Interval 属性：该属性表示时间的间隔（单位为：毫秒）。
- 在计时器 Timer 控件中 2 个常用的方法。
- GetScriptReferences()：该方法获取与控件相关的、引用的脚本资源。
- RaisePostBackEvent()：当网页提交到服务器时，使控件产生一个 Tick 事件。
- 在计时器 Timer 控件中 1 个常用的事件。
- Tick 事件：按照一定事件来触发该事件。

在网站备忘录中计时器 Timer 控件的属性设置如图 6.4 所示，在该图中读者可以看到将 Interval 属性设置为 1000（即每一秒钟刷新一下屏，判断一下数据库的信息是否和当前系统的时间相同，如果相同弹出备忘提示框）。

在网站备忘录中计时器 Timer 控件的事件设置如图 6.5 所示。

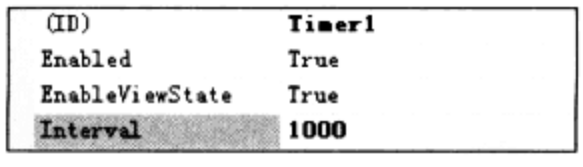


图 6.4 Timer 控件属性

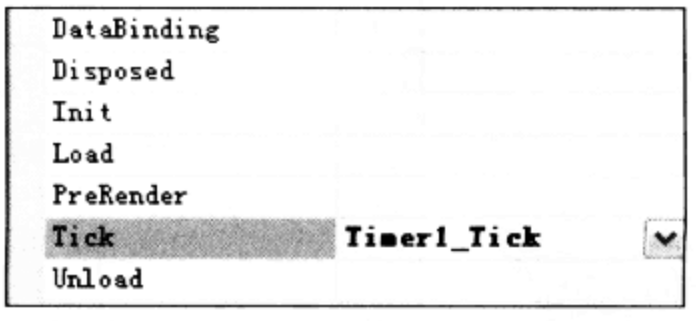


图 6.5 Timer 控件的事件设置

Timer1_Tick 事件每隔一秒钟触发一次 Timer1_Tick 事件，该事件的代码如下：

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    this.Label1.Text = System.DateTime.Now.ToString();
    SqlDataReader sdr1 = ep.ExceRead("select * from tb_memo where name='" + Session["UserName"] + "'and pwd='" +
Session["UserPwd"] + "'");
    while (sdr1.Read())
    {
        this.Label2.Text = sdr1["alerttime"].ToString();
        if (this.Label1.Text == sdr1["alerttime"].ToString())
        {
            this.Label2.Text = sdr1["subject"].ToString();
            ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var MSG1 = new PopBubble('备忘
录短消息提示：','" + this.Label2.Text + "'); MSG1.show();", true);
        }
    }
        sdr1.Close();
}
```

6.2.3 使用 Web 用户控件实现页面导航

在网站备忘录中除了使用 Web 服务器控件外，还应用了用户自定义控件。下面着重介绍用户自定义控件。

1. Web 用户控件的概述

用户控件是一种复合控件，其工作原理非常类似于 ASP.NET 网页，可以向用户的控件中添加现有的 Web 服务器控件和标记，并定义控件的属性和方法，然后将其嵌入网站备忘录网页中充当一个单元。

2. 创建 Web 用户控件

尽管 ASP.NET 提供的服务器控件具有十分强大的功能，但在实际应用中，遇到的问题总是复杂多样的。为了满足不同的功能需求，ASP.NET 允许程序开发人员根据实际需要制作适用的控件。通过本节的学习，读者将会了解到如何创建 Web 用户控件，如何将制作好的 Web 用户控件添加到网页中以及 Web 用户控件在实际开发中的应用。

创建用户控件的方法与创建 Web 网页大致相同，其主要操作步骤如下。

（1）打开解决方案资源管理器，在项目名称中单击鼠标右键，然后在弹出的快捷菜单中选择“添加新项”选项，打开如图 6.6 所示的“添加新项”对话框。在该对话中，选择“Web 用户控件项”选项，并为其命名，单击“添加”按钮将 Web 用户控件添加到项目中。

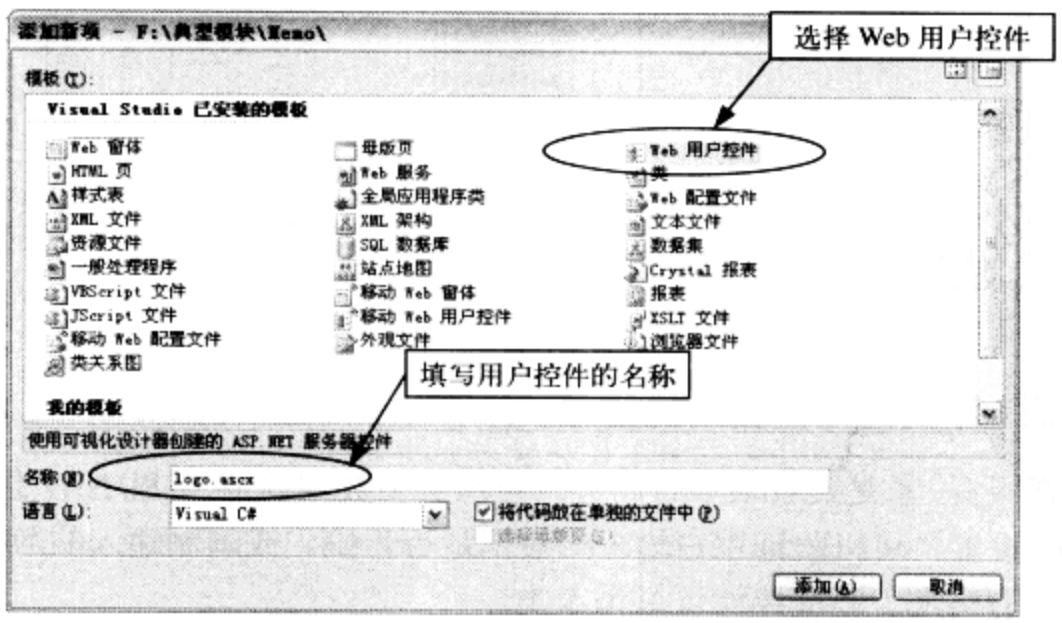


图 6.6 添加新对话框

（2）打开已创建好的 Web 用户控件（用户控件的文件扩展名为.ascx），在.ascx 文件中可以直接往页面上添加各种服务器控件以及静态文本、图片等。

（3）双击页面上的任何位置，或者直接按下快捷键<F7>，可以将视图切换到后台代码文件，程序开发人员可以直接在文件中编写程序控制逻辑，包括定义各种成员变量、方法以及事件处理程序等。

注 意

创建好用户控件后，必须添加到其他 Web 页中才能显示出来，不能直接作为一个网页来显示，因此也就不能设置用户控件为“起始页”。

3. 将 Web 用户控件添加至网页

如果已经设计好了 Web 用户控件，可以将其添加到一个或者多个网页中。在同一个网页中也可以重复使用，各个用户控件会以不同 ID 来标识。将用户控件添加到网页可以使用“Web 窗体设计器”直接添加。

使用“Web 窗体设计器”可以在“设计”视图下，将用户控件以拖放的方式直接添加到网页上，其操作与将内置控件从工具箱中拖放到网页上一样。在网页中添加用户控件的步骤如下。

（1）在解决方案资源管理器中，用鼠标单击要添加至网页的用户控件。

（2）按住鼠标左键，移动鼠标到网页上，然后松开鼠标左键即可，如图 6.7 所示。

图 6.7　将 Web 用户控件添加至网页

（3）在已添加的用户控件上，单击鼠标右键，选择“属性”选项，打开属性窗口，如图 6.8 所示。用户可以在属性窗口中修改用户控件的属性。

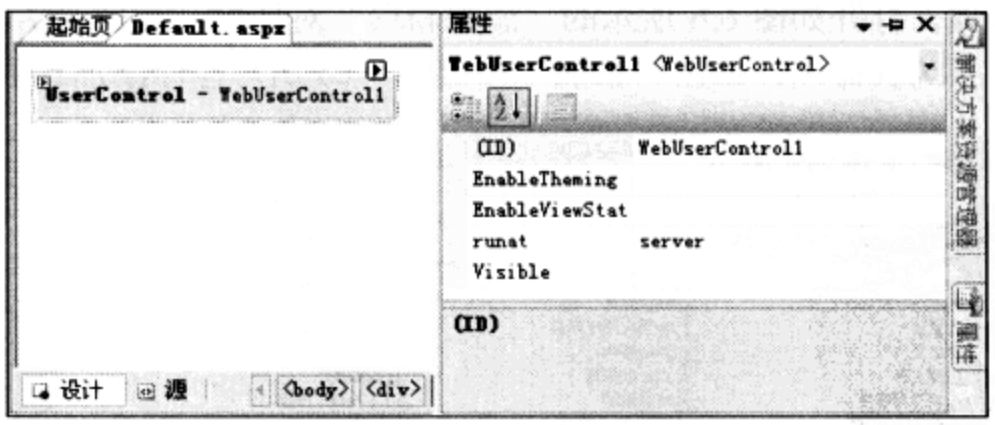


图 6.8　用户控件的属性窗口

6.2.4　使用验证控件验证用户输入的信息

以往开发动态网页，总是要费很多时间进行特殊类型数据验证。ASP.NET 提供了验证控件，通过对控件属性的设置，可以验证用户输入的数据是否正确，或限制输入的数据只能在某个范围或符合某种特定的格式。验证控件可分为 5 类。

- RequiredFieldValidator：验证是否不为空。

- CompareValidator：对比两个项目的数据。如在更改用户密码时，为避免用户输入错误密码，需要重复输入密码内容，然后通过设置 CompareValidator 控件的属性比较两者是否相等。
- RangeValidator：验证用户输入的内容是否在某个范围之内。如输入月份时，只能输入1～12的值。
- RegularExpressionValidator：使用 RequlerExpressionValidator 设计可接受的数据样式。
- CustomValidator：自定义验证逻辑，编写验证程序代码。

除了上述5个验证控件外，还有1个 ValidationSummary 控件，用来显示 Web 页上所有验证错误的列表。

1. 非空数据验证（RequiredFieldValidator）

当某个字段不能为空时，可以使用 RequiredFieldValidator 控件。该控件常用于文本框的非空验证。在网页提交到服务器前，该控件验证控件的输入值是否为空。如果为空，则显示错误信息和提示信息。

RequiredFieldValidator 控件部分常用属性的介绍如表 6.4 所示。

表 6.4　RequiredFieldValidator 控件最常用的属性

属　性	描　述
ID	控件 ID，控件惟一标识符
ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的控件必须和验证控件在相同的容器中
ErrorMessage	表示当验证不合法时，出现错误的信息
IsValid	获取或设置一个值，该值指示控件验证的数据是否有效。默认值为 True
Display	设置错误信息的显示方式
Text	如果 Display 为 Static，不出错时，显示该文本

下面对比较重要的属性进行详细介绍。

（1）ControlToValidate 属性

该属性指定验证控件对哪一个控件进行验证。

例如，要验证 TextBox 控件的 ID 属性为 txtPwd，只要将 RequiredFieldValidator 控件的 ControlToValidate 属性设置为 txtPwd。实现的代码如下：

```
this.RequiredFieldValidator1.ControlToValidate = "txtUserName";
```

（2）ErrorMessage 属性

该属性用于指定页面中使用 RequiredFieldValidator 控件时显示的错误消息文本。

例如，将 RequiredFieldValidator 控件的错误消息文本设为“*”。实现的代码如下：

```
this.RequiredFieldValidator1.ErrorMessage = "*";
```

2. 数据比较验证（CompareValidator）

比较验证将输入控件的值同常数值或其他输入控件的值相比较，以确定这两个值是否与比较运算符（小于、等于、大于等）指定的关系相匹配。

CompareValidator 控件部分常用属性的介绍如表 6.5 所示。

表 6.5　CompareValidator 控件最常用的属性

属　性	描　述
ID	控件 ID，控件的惟一标识符
ControlToCompare	获取或设置用于比较的输入控件的 ID。默认值为空字符串（""）

续表

属　性	描　述
ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的空间必须和验证控件在相同的容器中
ErrorMessage	表示当验证不合法时，出现错误的信息
IsValid	获取或设置一个值，该值指示控件验证的数据是否有效。默认值为 True
Operator	获取或设置验证中使用的比较操作。默认值为 Equal
Display	设置错误信息的显示方式
Text	如果 Display 为 Static，不出错时，显示该文本
Type	获取或设置比较的两个值的数据类型。默认值为 string
ValueToCompare	获取或设置要比较的值

下面对比较重要的属性进行详细介绍。

（1）ControlToCompare 属性

该属性用于指定要对其进行值比较的控件的 ID。

例如，ID 属性为 txtRePwd 的 TextBox 控件与 ID 属性为 txtPwd 的 TextBox 控件进行比较验证。代码如下：

```
this. CompareValidator1.ControlToCompare= "txtPwd";
this. CompareValidator1. ControlToValidate = "txtRePwd";
```

（2）Operator 属性

该属性用于指定要对其进行比较验证中使用的比较操作。ControlToValidate 属性必须位于比较运算符的左边，ControlToCompare 属性位于右边，才能有效进行计算。

例如，要验证 ID 属性为 txtRePwd 的 TextBox 控件与 ID 属性为 txtPwd 的 TextBox 控件是否相等。代码如下：

```
this.CompareValidator1.Operator = ValidationCompareOperator.Equal;
```

（3）Type 属性

该属性用于指定要对其进行比较的两个值的数据类型。

例如，要验证 ID 属性为 txtRePwd 的 TextBox 控件与 ID 属性为 txtPwd 的 TextBox 控件的值类型为 string 类型。代码如下：

```
this.CompareValidator1.Type = ValidationDataType.String;
```

（4）ValueToCompare 属性

该属性指定要比较的值。如果 ValueToCompare 和 ControlToCompare 属性都存在，则使用 ControlToCompare 属性的值。

例如，设置比较的值为“你好”。代码如下：

```
this.CompareValidator1.ValueToCompare = "你好";
```

3. 数据类型验证（CompareValidator）

CompareValidator 控件还可以对照特定的数据类型来验证用户的输入，以确保用户输入的是数字、日期等。

例如，如果要在用户信息页上输入出生日期信息，就可以使用 CompareValidator 控件确保该页在提交之前对输入的日期格式进行验证。

主要通过 CompareValidator 控件的 ControlToValidate 属性、Operator 属性和 Type 属性验证用户输入的出生日期与日期类型是否匹配。执行程序，如果出生日期不是日期类型，单击“验证”按钮，示例运行结果如图 6.9 所示。

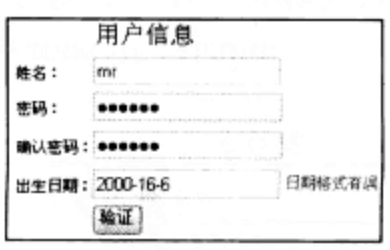


图 6.9　数据类型验证

下面介绍本程序实现的主要步骤。

新建一个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 4 个 TextBox 控件、1 个 RequiredFieldValidator 控件、2 个 CompareValidator 控件和 1 个 Button 控件，它们的属性设置如表 6.6 所示。

表 6.6　　Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性设置	用　途
标准/TextBox 控件	txtName		输入姓名
	txtPwd	TextMode 属性设置为 Password	设置为密码格式
	txtRePwd	TextMode 属性设置为 Password	设置为密码格式
	txtBirth		输入出生日期
标准/Button 控件	btnCheck	Text 属性设置为“验证”	执行页面提交的功能
验证/ RequiredField Validator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为 txtName	要验证的控件的 ID 为 txtName
		ErrorMessage 属性设置为“姓名不能为空”	显示的错误信息为“姓名不能为空”
		SetFocusOnError 属性设置为 True	验证无效时，在该控件上设置焦点
验证/ CompareValidator 控件	CompareValidator1	ControlToValidate 属性设置为 txtRePwd	要验证的控件的 ID 为 txtRePwd
		ControlToCompare 属性设置为 txtPwd	进行比较的控件 ID 为 txtPwd
		ErrorMessage 属性设置为“确认密码与密码不匹配”	显示的错误信息为“确认密码与密码不匹配”
	CompareValidator2	ControlToValidate 属性设置为 txtBirth	要验证的控件的 ID 为 txtBirth
		ErrorMessage 属性设置为“日期格式有误”	显示的错误信息为“日期格式有误”
		Operator 属性设置为“DataTypeCheck”	对值进行数据类型验证
		Type 属性设置为“Date”	进行日期比较

4. 数据格式验证（RegularExpressionValidator）

使用 RegularExpressionValidator 控件可以验证用户输入是否与预定义的模式相匹配，这样就可以对电话号码、邮编、网址等进行验证。RegularExpressionValidator 控件允许有多种有效模式，每个有效模式使用“|”字符来进行分隔。预定义的模式需要使用正则表达式定义。RegularExpressionValidator 控件部分常用属性的介绍如表 6.7 所示。

表 6.7　　RegularExpressionValidator 控件最常用的属性

编　号	属　性	描　述
1	ID	控件 ID，控件惟一标识符
2	ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的空间必须和验证控件在相同的容器中
3	ErrorMessage	表示当验证不合法时，出现错误的信息
4	IsValid	获取或设置一个值，该值指示控件验证的数据是否有效。默认值为 True
5	Display	设置错误信息的显示方式
6	Text	如果 Display 为 Static，不出错时，显示该文本
7	ValidationExpression	获取或设置被指定为验证条件的正则表达式。默认值为空字符串（""）

RegularExpressionValidator 控件的属性与 RequiredFieldValidator 控件大致相同，这里只对 ValidationExpression 属性进行具体介绍。

ValidationExpression 属性用于指定验证条件的正则表达式。常用的正则表达式字符及其含义如表 6.8 所示。

表 6.8　　常用正则表达式字符及其含义

编　　号	正则表达式字符	含　　义
1	[……]	匹配括号中的任何一个字符
2	[^……]	匹配不在括号中的任何一个字符
3	\w	匹配任何一个字符（a～z、A～Z 和 0～9）
4	\W	匹配任何一个空白字符
5	\s	匹配任何一个非空白字符
6	\S	与任何非单词字符匹配
7	\d	匹配任何一个数字（0～9）
8	\D	匹配任何一个非数字（^0～9）
9	[\b]	匹配一个退格键字母
10	{n,m}	最少匹配前面表达式 n 次，最大为 m 次
11	{n,}	最少匹配前面表达式 n 次
12	{n}	恰恰匹配前面表达式为 n 次
13	?	匹配前面表达式 0 或 1 次{0,1}
14	+	至少匹配前面表达式 1 次{1,}
15	*	至少匹配前面表达式 0 次{0,}
16	|	匹配前面表达式或后面表达式
17	(…)	在单元中组合项目
18	^	匹配字符串的开头
19	$	匹配字符串的结尾
20	\b	匹配字符边界
21	\B	匹配非字符边界的某个位置

下面再来列举几个常用的正则表达式。

- 验证电子邮件
- \w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*。
- \S+@\S+\．\S+。
- 验证网址
- http(s)?://([\w-]+\.)+[\w-]+(/[\w-./?%&=]*)?。
- 验证邮政编码
- \d{6}。
- 其他常用正则表达式
- [0-9]：表示 0～9 十个数字。
- \d*：表示任意个数字。
- \d{3,4}-\d{7,8}：表示中国大陆的固定电话号码。
- \d{2}-\d{5}：验证由两位数字、一个连字符再加 5 位数字组成的 ID 号。

- <\s*(\S+)(\s[^>]*)?>[\s\S]*<\s*\/\1\s*>：匹配 HTML 标记。

5. 数据范围验证（RangeValidator）

使用 RangeValidator 控件验证用户输入是否在指定范围之内。可以通过对 RangeValidator 控件的上、下限属性，以及指定控件要验证的值的数据类型的设置完成这一功能。如果用户的输入无法转换为指定的数据类型，例如，无法转换为日期，则验证将失败。如果用户将控件保留为空白，则此控件将通过范围验证。若要强制用户输入值，则还要添加 RequiredFieldValidator 控件。

RangeValidator 控件部分常用属性的介绍如表 6.9 所示。

表 6.9　　RangeValidator 控件最常用的属性

属　性	描　述
ID	控件 ID，控件惟一标识符
ControlToValidate	表示要进行验证的控件 ID，此属性必须设置为输入控件 ID。如果没有指定有效输入控件，则在显示页面时引发异常。另外该 ID 的控件必须和验证控件在相同的容器中
ErrorMessage	表示当验证不合法时，出现错误的信息
IsValid	获取或设置一个值，该值指示控件验证的数据是否有效。默认值为 True
Display	设置错误信息的显示方式
MaximumValue	获取或设置要验证的控件的值，该值必须小于或等于此属性的值。默认值为空字符串("")
MinimumValue	获取或设置要验证的控件的值，该值必须大于或等于此属性的值。默认值为空字符串("")
Text	如果 Display 为 Static，不出错时，显示该文本
Type	获取或设置一种数据类型，用于指定如何解释要比较的值

下面对比较重要的属性进行详细介绍。

- MaximumValue 属性和 MinimumValue 属性

该属性指定用户输入范围的最小值和最大值。

例如，要验证用户输入的值在 20～70 之间。代码如下：

```
this. RangeValidator1. MaximumValue= "70";
this. RangeValidator1. MaximumValue= "20";
```

- Type 属性

该属性用于指定进行验证的数据类型。在进行比较之前，值被隐式转换为指定的数据类型。如果数据转换失败，数据验证也会失败。

例如，将 RequiredFieldValidator 控件的错误消息文本设为“*”。代码如下：

```
this. RangeValidator1.Type = ValidationDataType.Integer;
```

6. 验证错误信息显示（ValidationSummary）

使用 ValidationSummary 控件可以为用户提供将窗体发送到服务器时所出现错误的列表。错误列表可以通过列表、项目符号列表或单个段落的形式进行显示。

ValidationSummary 控件中为页面上每个验证控件显示的错误信息，是由每个验证控件的 ErrorMessage 属性指定的。如果没有设置验证控件的 ErrorMessage 属性，将不会在 ValidationSummary 控件中为该验证控件显示错误信息。还可以通过设置 HeaderText 属性，在 ValidationSummary 控件的标题部分指定一个自定义标题。

通过设置 ShowSummary 属性，可以控制 ValidationSummary 控件是显示还是隐藏。如果将 ShowMessageBox 属性设置为 True，就可以在消息框中显示摘要。

ValidationSummary 控件部分常用属性的介绍如表 6.10 所示。

表 6.10　　ValidationSummary 控件最常用的属性

属　性	描　述
HeaderText	控件汇总信息
DisplayMode	设置错误信息的显示格式
ShowMessageBox	是否以弹出方式显示每个被验证控件的错误信息
ShowSummary	是否使用错误汇总信息
EnableClientScript	是否使用客户端验证，系统默认值为 True
Validate	执行验证并且更新 IsValid 属性

下面对比较重要的属性进行详细介绍。

- DisplayMode 属性

使用该属性指定 ValidationSummary 控件的显示格式。摘要可以按列表、项目符号列表或单个段落的形式显示。

例如，设置 ValidationSummary 的显示模式为项目符号列表，代码如下：

```
this.ValidationSummary1.DisplayMode = ValidationSummaryDisplayMode.BulletList;
```

- ShowMessageBox 属性

当 ShowMessageBox 属性设为 True 时，网页上的错误信息不在网页本身上显示，而是以弹出对话框的形式来显示错误信息，如图 6.10 所示。

- ShowSummary 属性

除了 ShowMessageBox 属性外，该属性也可用于控制验证摘要的显示位置。如果该属性设置为 True，则在网页上显示验证摘要。

注意　如果 ShowMessageBox 和 ShowSummary 属性都设置为 True，则在消息框和网页上都显示验证摘要。

7. 自定义验证控件（CustomValidator）

如果现有的 ASP.NET 验证控件无法满足需求，那么可以定义一个自定义的服务器端验证函数，然后使用 CustomValidator 控件来调用它。

例如，通过使用 CustomValidator 控件实现服务器端验证，此时只需将验证函数与 ServerValidate 事件相关联。执行程序，如果输入的数字不是偶数或不符合数据类型要求，单击“验证”按钮，示例运行结果如图 6.11 所示。

图 6.10　ShowMessageBox 属性

图 6.11　验证错误信息显示

本例实现的主要步骤如下。

新建 1 个网站，默认主页为 Default.aspx，在 Default.aspx 页面上添加 1 个 TextBox 控件、1 个 CustomValidator 控件和 1 个 Button 控件，它们的属性设置如表 6.11 所示。

表 6.11　　Default.aspx 页控件属性设置及说明

控件类型	控件名称	主要属性及事件设置	用　途
标准/TextBox 控件	txtNum		输入偶数
标准/Button 控件	btnCheck	Text 属性设置为“验证”	执行页面提交的功能
验证/ CustomValidator 控件	RequiredFieldValidator1	ControlToValidate 属性设置为“txtNum”	要验证的控件的 ID 为 txtNum
		ErrorMessage 属性设置为“您输入的不是偶数”	显示的错误信息为“您输入的不是偶数”
		ServerValidate 事件设置为“ValidateEven”	与自定义函数相关联，以在服务器上执行验证

6.3　网站备忘录实现过程

6.3.1　新建网站备忘录

新建网站备忘录主要是插入新的提示主题和提示的具体时间，到时即可在程序右下角中自动弹出所插入的主题的提示信息，新建网站备忘录页的运行的效果如图 6.12 所示，弹出提示框如图 6.13 所示。

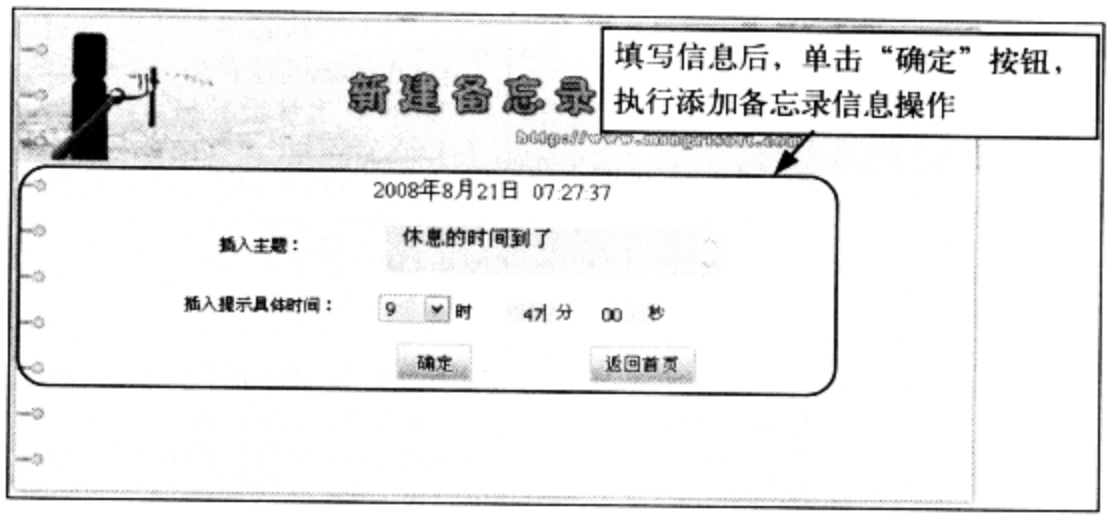


图 6.12　运行效果图

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 add.aspx，作为网站备忘录的登录页，页面 add.aspx 的设计界面如图 6.14 所示。

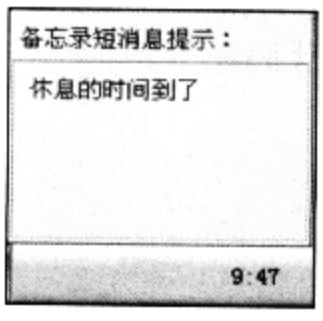


图 6.13　弹出提示信息页的运行效果

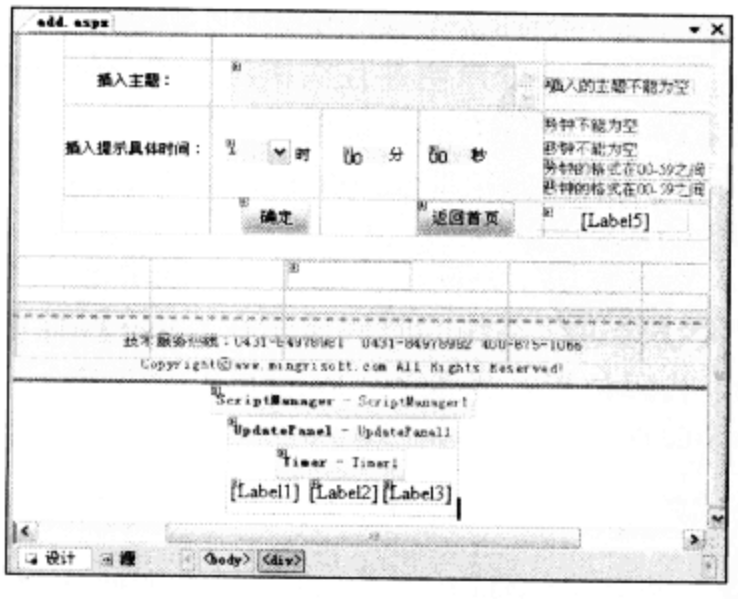


图 6.14　页面“add.aspx”的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Web 用户控件，3 个 TextBox 控件和 3 个 RequiredFieldValidator 控件、2 个“RangeValidator”和 2 个“ImageButton”控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 6.12 所示。

表 6.12　　网站备忘录新建页控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称为 logo.ascx	无	logo.ascx 实现网站的页面导航功能
TextBox	控件的名称分别为 TxtSubject、TxtMinute 和 TxtSecond	例如，将 TxtSubject 控件的 SkinID 属性设置为 tbSkin。 将 TxtSubject 控件的 TextMode 属性设置为 MultiLine	用于输入网站备忘录的信息
RequiredFieldValidator	控件的名称分别设置为 rfvSubject、rfvminute 和 rfvsecond	例如，将 rfvSubject 控件的 ControlToValidate 属性设置为 TxtSubject，将 Display 属性设置为 Dynamic，将 ErrorMessage 属性设置为“插入的主题不能为空”	用于验证文本框输入的信息是否为空
RangeValidator	控件的名称分别设置为“RanVMinute”和“RanVSecond”	例如，将 RanVMinute 控件的 ControlToValidate 属性设置为 TxtMinute，将 Display 属性设置为 Dynamic，将 ErrorMessage 属性设置为“分钟的格式在 00-59 之间”，将 MaximumValue 属性设置为“59”，将 MinimumValue 属性设置为“00”，将 Type 属性设置为 Integer	用于验证来自用户输入的值是否在指定的范围内
ImageButton	控件的名称为 ImgQueDing 和 ImgReturn	例如，将“ImgReturn”控件的 ImageUrl 属性设置为“~/image/返回首页.jpg”，将 OnClick 属性设置为 ImgReturn_Click，将 Causes Validation 属性设置为 False	ImgQueDing 控件用于执行新建操作 ImgReturn 控件用于将页面跳转到网站首页中

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下：

例程 1　代码位置：光盘\mr\06\memo\add.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化 1 个 Memobwl 类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下：

例程 2　代码位置：光盘\mr\06\memo\add.aspx.cs

```
Memobwl ep = new Memobwl();
```

Page_Load 事件用于判断用户是否已经登录过。如果没有登录过跳转到 tishi.aspx 登录页面中，如果已经登录过将系统的时间显示在 Label 控件中，实现的代码如下。

例程 3　代码位置：光盘\mr\06\memo\add.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
     if (Session["UserName"] == null && Session["UserPwd"] == null)
       {
            Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
       }
```

```
                else
                {
                    this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
                    string date;
                    if (Request.QueryString["name"] == null)
                    {

                        date = System.DateTime.Now.ToShortDateString() + " " + this.ddlhour.SelectedItem.Text + ":" +
this.TxtMinute.Text + ":" + this.TxtSecond.Text;
                    }
                    else
                    {
                        date = Request.QueryString["name"] + " " + this.ddlhour.SelectedItem.Text + ":" + this.TxtMinute.Text + ":" +
this.TxtSecond.Text;
                    }
                    this.Label2.Text = date;
                }
                }
```

在 Timer1_Tick 事件中，每一秒钟判断一下当前系统的时间是否与数据库中 alerttime 网站备忘录的时间相同，如果相同则弹出提示信息，实现的代码如下。

例程 4 代码位置：光盘\mr\06\memo\add.aspx.cs

```
        protected void Timer1_Tick(object sender, EventArgs e)
        {
            this.Label1.Text = System.DateTime.Now.ToString();
            SqlDataReader sdr1 = ep.ExceRead("select * from tb_memo where name='" + Session["UserName"] + "'and
pwd='" + Session["UserPwd"] + "'");
            while (sdr1.Read())
            {
                this.Label2.Text = sdr1["alerttime"].ToString();
                if (this.Label1.Text == sdr1["alerttime"].ToString())
                {
                    this.Label2.Text = sdr1["subject"].ToString();
                    ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var MSG1 = new Pop
Bubble('备忘录短消息提示：','" + this.Label2.Text + "'); MSG1.show();", true);
                }

            }
            sdr1.Close();
        }
```

双击“确定”按钮，触发其 ImgQueDing_Click 事件，在该单击事件中，实现将备忘信息添加到“tb_memo”数据表中，实现的代码如下。

例程 5 代码位置：光盘\mr\06\memo\add.aspx.cs

```
    protected void ImgQueDing_Click(object sender, ImageClickEventArgs e)
    {
        string insertSql = "insert into tb_memo(subject,alertTime,name,pwd) values('" + this.TxtSubject.Text + "','" +
this.Label2.Text + "','" + Session["UserName"] + "','" + Session["UserPwd"] + "')";
        Memobwl.EXECCommand(insertSql);
        this.TextBox1.Text = Request.QueryString["name"];
        Response.Write("<script>alert('恭喜您！ 网站备忘录信息添加成功！ ');location='javascript:history.go
(-1)'</script>");
        this.TxtSubject.Text = "";
        this.TxtMinute.Text = "00";
        this.TxtSecond.Text = "00";
    }
```

双击“返回首页”按钮，触发其 ImgReturn_Click 事件，在该单击事件中，实现将页面跳转到 Default.aspx 网站首页，实现的代码如下。

例程 6　代码位置：光盘\mr\06\memo\add.aspx.cs

```
protected void ImgReturn_Click(object sender, ImageClickEventArgs e)
    {
      Response.Redirect("Default.aspx");//跳转到Default.aspx页面中
    }
```

6.3.2　检索网站备忘录信息

在网站备忘录程序首页中，当单击网站头的导航中的“检索”按钮时，页面将跳转网站备忘录的检索页，检索网站备忘录的信息实现在“Search.aspx”页面中，当在日历控件中选择指定日期时，将页面跳转到“Search.aspx”页面中，检索网站备忘录的运行效果图如图 6.15 所示。

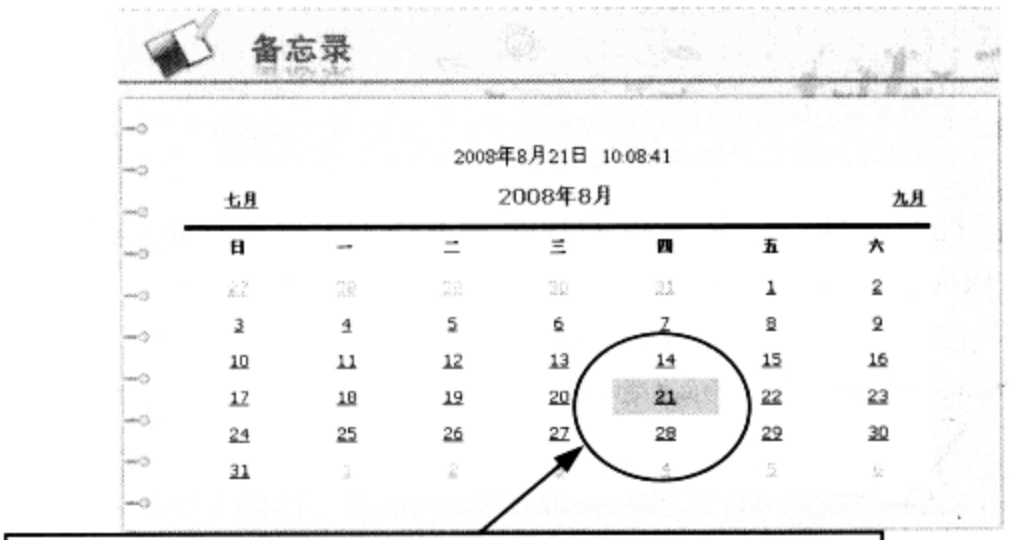


图 6.15　发表留言信息运行效果图

1．前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 Search.aspx，作为检索网站备忘录的信息页。页面 Search.aspx 的设计界面如图 6.16 所示。

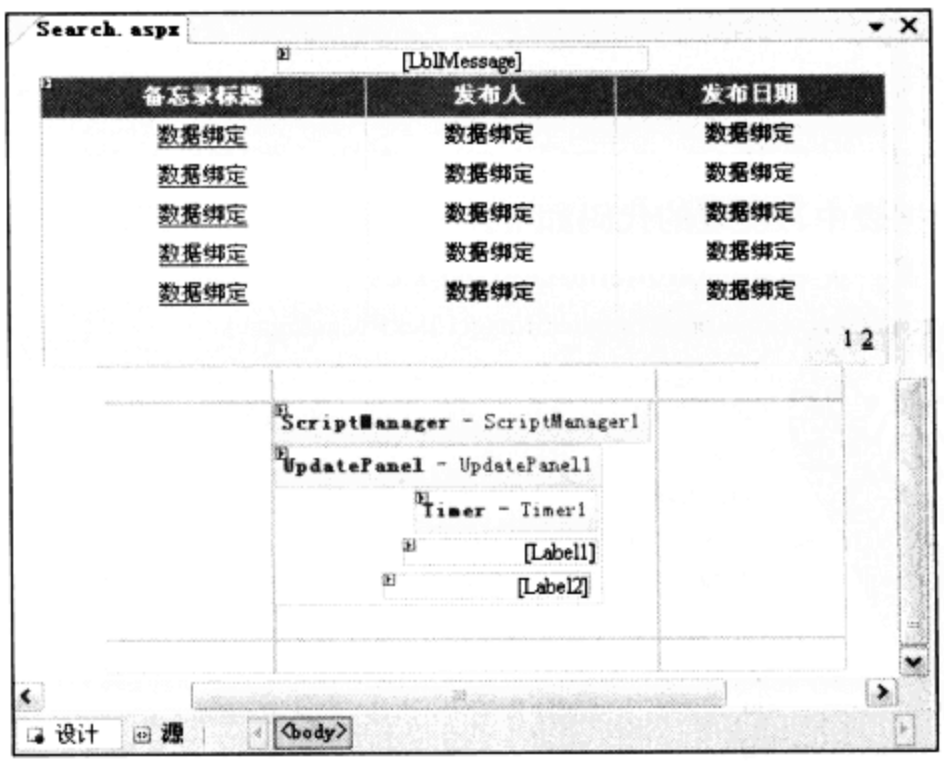


图 6.16　页面 Search.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1

个 Web 用户控件和 1 个 Calendar 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 6.13 所示。

表 6.13　　检索网站备忘录信息页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　　途
Table	无	无	页面布局
Web 用户控件	控件的名称为 logo.ascx	无	logo.ascx 实现网站的页面导航功能
Calendar	控件的名称为“Calendar1”	无	用于显示日期时间

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 7　代码位置：光盘\mr\06\memo\Calendar.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化 1 个 Memobwl 类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下。

例程 8　代码位置：光盘\mr\06\memo\Calendar.aspx.cs

```
Memobwl ep = new Memobwl();
```

在 Page_Load 页面加载的事件当中，将当前登录的用户信息显示到 Label 控件中。实现的代码如下。

例程 9　代码位置：光盘\mr\06\memo\Calendar.aspx.cs

```
    protected void Page_Load(object sender, EventArgs e)
    {
        this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
    }
```

在 Timer1_Tick 事件中，每一秒钟判断一下当前系统的时间是否和数据库中 alerttime 网站备忘录的时间相同，如果相同则弹出提示信息，实现的代码如下。

例程 10　代码位置：光盘\mr\06\memo\Calendar.aspx.cs

```
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        this.Label1.Text = System.DateTime.Now.ToString();
        SqlDataReader sdr1 = ep.ExceRead("select * from tb_memo where name='" + Session["UserName"] + "'and
pwd='" + Session["UserPwd"] + "'");
        while (sdr1.Read())
        {
            this.Label2.Text = sdr1["alerttime"].ToString();
            if (this.Label1.Text == sdr1["alerttime"].ToString())
            {
                this.Label2.Text = sdr1["subject"].ToString();
                  ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var MSG1 = new
PopBubble('备忘录短消息提示：','" + this.Label2.Text + "'); MSG1.show();", true);
            }
        }
        sdr1.Close();
    }
```

在页面的日历控件的 Calendar1_SelectionChanged 事件中，主要实现的是当单击 Calendar 控件中的日期，则会跳转到相应的检索网站备忘录的详细页面中，实现的代码如下。

例程 11　代码位置：光盘\mr\06\memo\Calendar.aspx.cs

```
    protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        DateTime dateValue;
```

```
        dateValue =Convert.ToDateTime ( Calendar1.SelectedDate);
        Session["NextdateValue"]=Convert.ToDateTime(Calendar1.SelectedDate).AddDays(+1).ToShortDateString();
        Response.Redirect("Search.aspx?dateValue=" + dateValue);
    }
```

6.3.3 详细信息页

网站备忘录详细信息实现在“info.aspx”页面中，该页实现的是查看备忘录的详细信息，该页面运行的效果如图 6.17 所示。

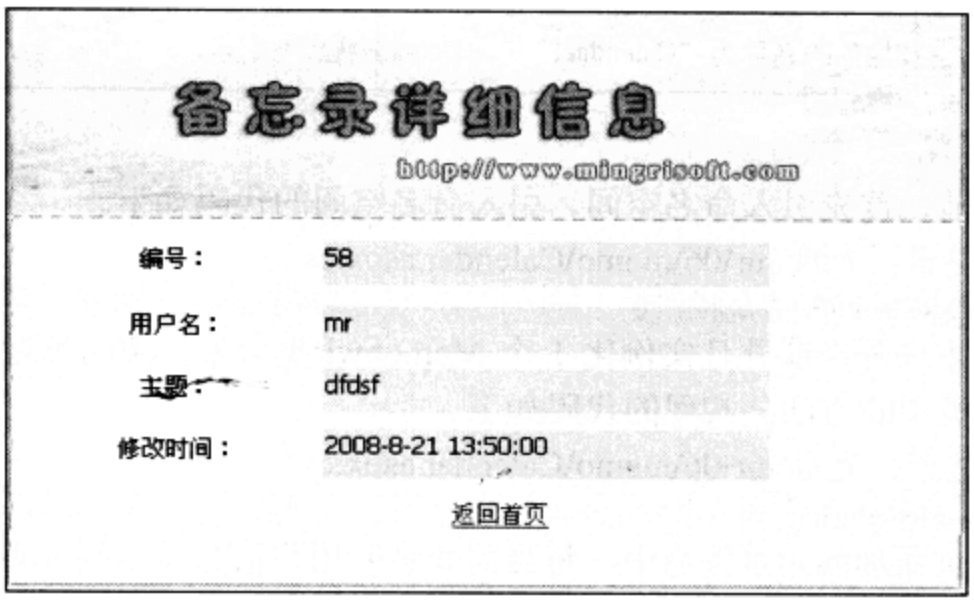


图 6.17　发表留言信息运行效果图

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 Search.aspx，作为检索网站备忘录的信息页。页面 Search.aspx 的设计界面如图 6.18 所示。

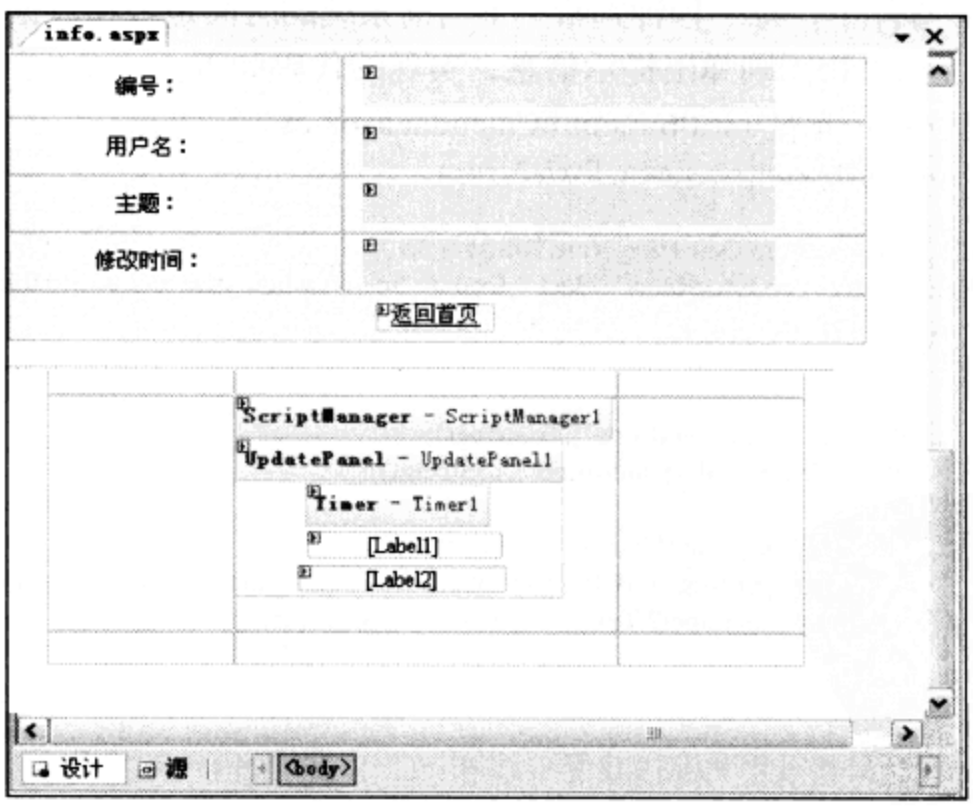


图 6.18　页面 Search.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Web 用户控件、1 个 LinkButton 控件和 4 个 TextBox 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 6.14 所示。

表 6.14　　详细信息页控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称为 logo.ascx	无	logo.ascx 实现网站的页面导航功能
LinkButton	控件的名称为 LinkButton1	无	当单击该控件时，将页面跳转到网站首页中
TextBox	控件的名称分别为 TxtId、TxtName、Txtsubject 和 TxtalertTime	将这 4 个文本框控件的 SkinID 属性设置为 tbSkin	用于显示备忘录的详细信息

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 12　代码位置：光盘\mr\06\memo\Calendar.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化 1 个 Memobwl 类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下。

例程 13　代码位置：光盘\mr\06\memo\info.aspx.cs

```
Memobwl ep = new Memobwl();
```

在 Page_Load 页面加载的事件当中，使用 Session 判断是否已成功登录。如果用户没有登录，将页面跳转到 tishi.aspx 页面中，否则将备忘录的详细信息显示到 TextBox 控件当中，实现的代码如下。

例程 14　代码位置：光盘\mr\06\memo\info.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        //使用Session判断是否已成功登录
        if (Session["UserName"] == null && Session["UserPwd"] == null)
        {
            Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
        }
        else
        {
            if (!IsPostBack)
            {
                if (Request["id"] != null)
                {
                    this.lblMessage.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！";
                    DataSet ds = ep.ReturnDataSet("select * from tb_memo where id='" + Request["id"] + "'", "tb_memo");
                    DataRowView rv = ds.Tables["tb_memo"].DefaultView[0];
                    this.TxtId.Text = rv["id"].ToString();
                    this.TxtName.Text = rv["Name"].ToString();
                    this.Txtsubject.Text = rv["subject"].ToString();
                    this.TxtalertTime.Text = rv["alertTime"].ToString();
                }
            }
        }
    }
```

双击页面中的“返回首页”按钮，触发其 LinkButton1_Click 事件，在该事件中实现的是将页面跳转到网站首页中“Default.aspx”，实现的代码如下。

例程 15　代码位置：光盘\mr\06\memo\info.aspx.cs

```
protected void LinkButton1_Click(object sender, EventArgs e)
{
    Response.Redirect("Default.aspx");//跳转到Default.aspx页面中
}
```

在页面的日历控件的 Calendar1_SelectionChanged 事件中，主要实现的是当单击 Calendar 控件中的日期时，则会跳转到相应的检索网站备忘录的详细页面中，实现的代码如下。

例程 16 代码位置：光盘\mr\06\memo\Calendar.aspx.cs

```
protected void Timer1_Tick(object sender, EventArgs e)
    {
        this.lblMessage.Text = System.DateTime.Now.ToString();
        SqlDataReader sdr1 = ep.ExceRead("select * from tb_memo where name='" + Session["UserName"] + "'and
pwd='" + Session["UserPwd"] + "'");
        while (sdr1.Read())
        {
            this.Label2.Text = sdr1["alerttime"].ToString();
            if (this.lblMessage.Text == sdr1["alerttime"].ToString())
            {
                this.Label2.Text = sdr1["subject"].ToString();
                ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var MSG1 = new Pop
Bubble('备忘录短消息提示：','" + this.Label2.Text + "'); MSG1.show();", true);
            }

        }
        sdr1.Close();
    }
```

6.3.4 按日期查看当天信息

在按日期查看当天信息页面中单击日历控件上的日期，将跳转到该备忘录标题的详细页面中，如果该日期有网站备忘录信息，将网站备忘录的检索页的详细信息显示到 GridView 控件中，如图 6.19 所示。如果该日期没有备忘录信息，将提示“您指定的日期没有发布任何网站备忘录信息!”，如图 6.20 所示。

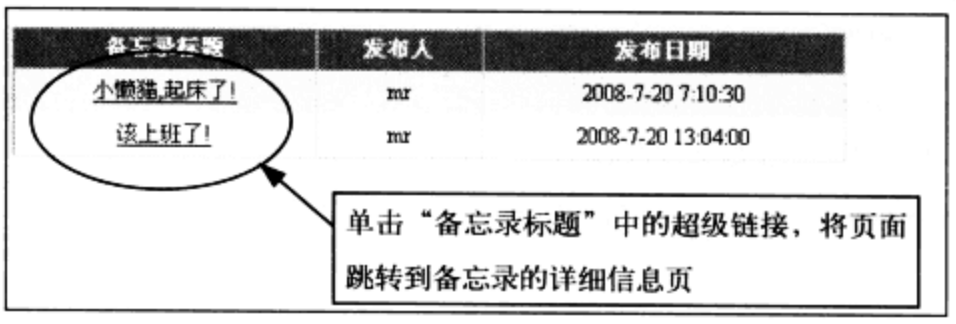


图 6.19 按照日期查看备忘录信息运行效果图

1. 前台页面设计

(1) 在应用程序中创建 1 个 Web 窗体，命名为 Search.aspx，作为网站备忘录检索的信息页。页面 Search.aspx 的设计界面如图 6.21 所示。

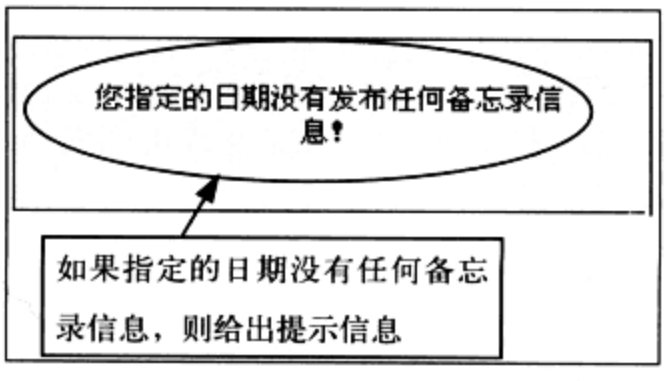


图 6.20 指定的日期没有发布任何网站备忘录信息图

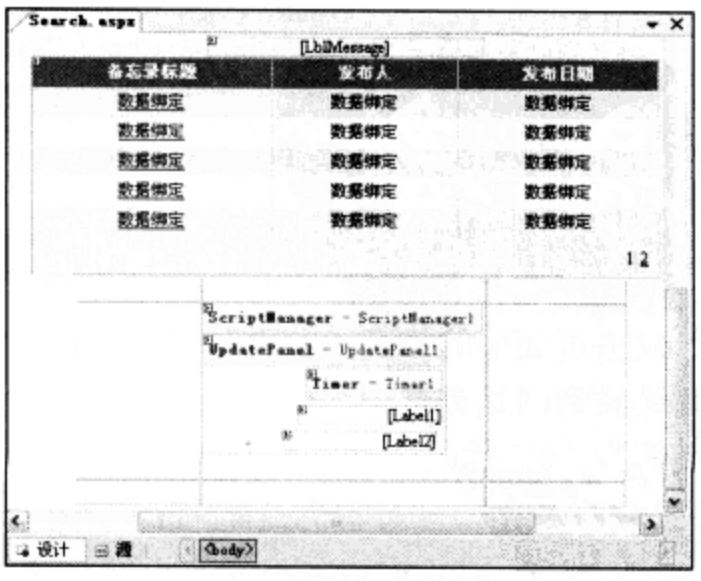


图 6.21 页面 Search.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Web 用户控件和 1 个 GridView 控件和 1 个 Label 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 6.15 所示。

表 6.15　网站备忘录检索信息页的控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称为 logo.ascx	无	logo.ascx 实现网站的页面导航功能
GridView	控件的名称设置为 MyDataView	将该控件的 AllowPaging 属性设置为 True，将 OnPageIndexChanging 属性设置为 MyDataView_PageIndexChanging，将 PageSize 属性设置为 5	实现分页功能，并且每页显示 5 条数据
A Label	控件的名称设置为 LblMessage	无	显示时间

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 17　代码位置：光盘\mr\06\memo\Search.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化 1 个 Memobwl 类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下。

例程 18　代码位置：光盘\mr\06\memo\Search.aspx.cs

```
Memobwl ep = new Memobwl();
```

在 Page_Load 事件中判断用户是否已经登录过。如果没有登录过则跳转到 tishi.aspx 登录页面中，如果已经登录过将系统的时间显示在 Label 控件中，并调用自定义方法 DataGridDataBind() 将数据绑定到 GridView 控件中，实现的代码如下。

例程 19　代码位置：光盘\mr\06\memo\Search.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
            //使用Session判断是否已成功登录
            if (Session["UserName"] == null && Session["UserPwd"] == null)
            {
                Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
            }
            else
            {
                this.DataGridDataBind();
            }
    }
}
```

在自定义的 DataGridDataBind 方法中，将指定日期时间的信息绑定到 GridView 控件中，如果指定的时间没有网站备忘录信息，将显示“您指定的日期没有发布任何网站备忘录信息!”，实现的代码如下。

例程 20　代码位置：光盘\mr\06\memo\Search.aspx.cs

```
        private void DataGridDataBind()
        {
            SqlConnection conn = new SqlConnection("Data Source=(local);DataBase=db_memo;UID=sa;pwd=");
            string sqlstr;
            sqlstr = "select * from tb_memo where alertTime between'" + Convert.ToDateTime(Request["dateValue"]).
ToShortDateString() + "'and '" + Session["NextdateValue"].ToString() + "'";
            SqlDataAdapter da = new SqlDataAdapter(sqlstr, conn);
            DataSet ds = new DataSet();
            da.Fill(ds, "tb_memo");
            try
            {
                if (!(ds.Tables["tb_memo"].Rows.Count > 0))

                this.LblMessage.Text="您指定的日期没有发布任何网站备忘录信息！";
            else
            {
                MyDataView.DataSource = ds.Tables["tb_memo"];
                MyDataView.DataBind();
                this.LblMessage.Text="";
            }
        }
        catch (Exception error)
        {
            Response.Write(error.ToString());
        }
    }
```

该页中 GridView 控件的分页功能，主要编写在“MyDataView_PageIndexChanging”事件中，实现的代码如下。

例程 21　代码位置：光盘\mr\06\memo\Search.aspx.cs

```
protected void MyDataView_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    this.MyDataView.PageIndex = e.NewPageIndex;
    this.MyDataView.DataBind();
}
```

在 Timer1_Tick 事件中，每一秒钟判断一下当前系统的时间是否和数据库中 alerttime 网站备忘录的时间相同，如果相同则弹出提示信息，实现的代码如下。

例程 22　代码位置：光盘\mr\06\memo\Search.aspx.cs

```
        protected void Timer1_Tick(object sender, EventArgs e)
        {
            this.Label1.Text = System.DateTime.Now.ToString();
            SqlDataReader sdr1 = mome.ExceRead("select * from tb_memo where name='" + Session["UserName"] +
"'and pwd='" + Session["UserPwd"] + "'");
            while (sdr1.Read())
            {
                this.Label2.Text = sdr1["alerttime"].ToString();
                if (this.Label1.Text == sdr1["alerttime"].ToString())
                {
                    this.Label2.Text = sdr1["subject"].ToString();
                    ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var MSG1 = new
PopBubble('备忘录短消息提示：','" + this.Label2.Text + "'); MSG1.show();", true);
                }
            }
            sdr1.Close();
        }
```


6.3.5　网站备忘录修改信息页

网站备忘录修改信息页，修改备忘录信息页主要是“Update.aspx”页实现的，该页面主要是将备忘录的信息显示在 GridView 控件中，当单击 GridView 控件上的修改按钮将页面跳转到

修改备忘录信息的详细页面中，运行效果如图 6.22 所示。

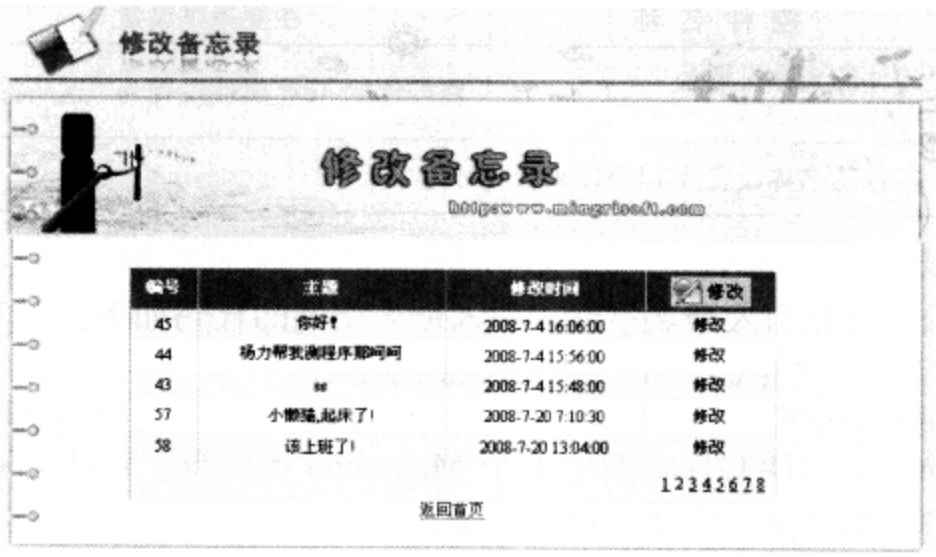


图 6.22　网站备忘录的修改信息页运行效果图

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 Update.aspx，作为网站备忘录修改信息页。页面 Update.aspx 的设计界面如图 6.23 所示。

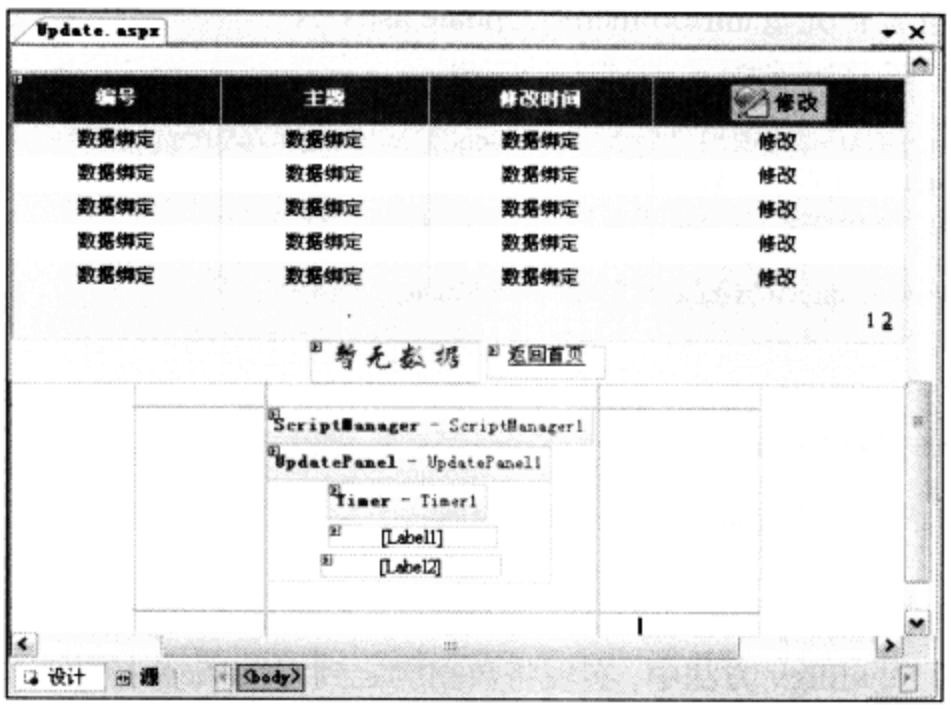


图 6.23　页面 Update.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Web 用户控件和 1 个 GridView 控件、1 个 LinkButton 控件和 1 个 Label 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 6.16 所示。

表 6.16　　网站备忘录修改信息页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称为 logo.ascx	无	logo.ascx 实现网站的页面导航功能
GridView	控件的名称设置为 MyDataView	将该控件的 AllowPaging 属性设置为 True，将 OnPageIndexChanging 属性设置为 MyDataView_PageIndexChanging，将 PageSize 属性设置为 5	实现分页功能，并且每页显示 5 条数据

续表

控 件 类 型	控 件 名 称	主要属性设置	用　途
ab LinkButton	控件的名称设置为 LinkButton1	将控件的 Text 属性设置为“返回首页”	实现将页面跳转到网站首页中
A Label	控件的名称设置为 LblMessage	无	显示时间

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 23　代码位置：光盘\mr\06\memo\Update.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化 1 个 Memobwl 类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下。

例程 24　代码位置：光盘\mr\06\memo\Update.aspx.cs

```
Memobwl ep = new Memobwl();
```

在 Page_Load 事件中判断用户是否已经登录过。如果没有登录过跳转到 tishi.aspx 页面中，如果已经登录过将系统的时间显示在 Label 控件中，并且调用用户自定义 Newfillgv 方法将数据绑定到 GridView 控件中，实现的代码如下。

例程 25　代码位置：光盘\mr\06\memo\Update.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    this.LblName.Text = "当前用户" + " " + Session["UserName"].ToString() + " " + "欢迎您！ ";
    //使用Session判断是否已成功登录
    if (Session["UserName"] == null && Session["UserPwd"] == null)
    {
        Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
    }
    else
    {
        if (!this.IsPostBack)
        {
            this.Newfillgv();//调用用户自定义空间
        }
    }
}
```

在用户自定义的 Newfillgv 方法中，实现将数据绑定到 GridView 控件当中，实现的代码如下。

例程 26　代码位置：光盘\mr\06\memo\Update.aspx.cs

```
public void Newfillgv()                    //用户自定义的方法
{
    string SqlStr = "select * from tb_memo where name='" + Session["UserName"] + "' order by alertTime desc ";
    SqlConnection con = new SqlConnection(Memobwl.GetConStr());
    con.Open();//打开数据库连接
    SqlDataAdapter sda = new SqlDataAdapter(SqlStr, con);
    DataSet ds = new DataSet();//声明1个DataSet的对象,并将该对象ds实例化
    sda.Fill(ds, "tb_memo");//将tb_memo填充到数据库中
    this.gvInfo.DataSource = ds;
    this.gvInfo.DataBind();//将数据绑定到GridView控件中
    for (int i = 0; i <= this.gvInfo.Rows.Count - 1; i++)
    {
        DataRowView mydrv;   //声明1个DataRowView的对象
        string gintro;
        if (this.gvInfo.PageIndex == 0)
        {
            Memobwl.GetConStr();
            mydrv = ds.Tables["tb_memo"].DefaultView[i];
```

```
                gintro = Convert.ToString(mydrv["subject"]);
                this.gvInfo.Rows[i].Cells[1].Text = ep.SubStr(gintro, 10);
            }
        }
        if (this.gvInfo.PageCount == 0)
        {
            this.Lblmessage.Visible = true;
        }
        con.Close();//关闭数据库连接
    }
```

在“gvUserInfo_PageIndexChanging”事件中，实现 GridView 控件的分页功能，实现的代码如下。

例程 27 代码位置：光盘\mr\06\memo\Update.aspx.cs

```
    protected void gvUserInfo_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
        this.gvInfo.PageIndex = e.NewPageIndex;
        this.Newfillgv();
    }
```

在 Timer1_Tick 事件中，每秒判断一下当前系统的时间是否和数据库中 alerttime 网站备忘录的时间相同，如果相同则弹出提示信息，实现的代码如下。

例程 28 代码位置：光盘\mr\06\memo\Update.aspx.cs

```
    protected void Timer1_Tick(object sender, EventArgs e)
    {
        this.Label1.Text = System.DateTime.Now.ToString();
        SqlDataReader sdr1 = ep.ExceRead("select * from tb_memo where name='" + Session["UserName"] + "'and
pwd='" + Session["UserPwd"] + "'");
        while (sdr1.Read())
        {
            this.Label2.Text = sdr1["alerttime"].ToString();
            if (this.Label1.Text == sdr1["alerttime"].ToString())
            {
                this.Label2.Text = sdr1["subject"].ToString();
                ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var MSG1 = new
PopBubble('备忘录短消息提示：'," + this.Label2.Text + "); MSG1.show();", true);
            }
        }
        sdr1.Close();
    }
```

6.3.6 新用户注册

网站备忘录的注册页（zhuce.aspx），该页用于实现新用户的注册功能，新用户注册成功后，需要记住您的注册信息，通过已注册的用户名称和密码能够进入网站中。网站备忘录注册页面的运行效果如图为 6.24 所示。

图 6.24 网站备忘录注册页面图

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 zhuce.aspx，作为网站备忘录注册页。页面 zhuce.aspx 的设计界面。如图 6.25 所示。

图 6.25　页面 zhuce.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Web 用户控件、6 个 TextBox 控件、1 个 RadioButton 控件、1 个 DropDownList 控件、3 个 Button 控件和 5 个 RequiredFieldValidator 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 6.17 所示。

表 6.17　网站备忘录注册页控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称为 logo.ascx	无	Head.asox 实现网站的页面导航功能，foot.ascx 页面布局功能
TextBox	控件的名称分别为 Txtname、TxtPwd、TxtTruePwd、Txthomepage、TxtEmail、TxtQQ	其中 TxtPwd 和 TxtTrue Pwd 的 TextMode 的属性都设置为 password	用于输入注册信息
RadioButton	控件的名称为 radSex	无	实现在 RadioButton 控件中选择“男”或者“女”的选项
DropDownList	控件的名称为 ddlquan	向 Itmes 属性中填充两个成员分别为“普通成员”和“管理员”	在 DropDownList 下拉列表中选择相应的选项
Button	控件的名称分别设置为 BtnCheck、BtnZc 和 BtnCz	将 Text 属性值分别设置为“审核用户名”、“注册”和“重置”	这几个按钮分别用于审核用户输入的用户名是否已经被占用、审核用户注册信息和清空文本框的信息
RequiredFieldValidator	控件的名称分别设置为 rfvMm、rfvPwd、rfvhomepage、rfvEmail 和 rfvQQ	例如，将 rfvMm 的 ControlToValidate 属性设置为“TxtTruePwd”、Display 属性设置为“Dynamic”、Error Message 属性设置为“密码不能为空”	验证文本框输入的内容是否为空，如果为空将提示错误信息

续表

控件类型	控件名称	主要属性设置	用途
CompareValidator	控件的名称设置为CvPwd	将控件的ControlToCompare属性设置为TxtPwd、ControlToValidate的属性设置为TxtTruePwd、将Display属性设置为Dynamic和将ErrorMessage属性设置为两次输入的密码不一致	该控件用于验证两次输入的密码是否相同，如果不相同将弹出错误提示信息，直到输入两次相同的密码为止
RegularExpressionValidator	控件的名称分别设置为RevEmail、revydtel和revZy	例如将控件RevEmail的ControlToValidate属性设置为TxtEmail、将Display属性设置为Dynamic、将ErrorMessage属性设置为“格式不正确”和将ValidationExpression属性设置为\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*	用来验证用户输入的信息内容格式是否正确

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 29 代码位置：光盘\mr\06\memo\Zhuce.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化 1 个 LeaveWord 类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。声明的代码如下。

例程 30 代码位置：光盘\mr\06\memo\Zhuce.aspx.cs

```
LeaveWord lw = new LeaveWord();
```

双击“注册/登录”按钮，触发 ImgBtnLogin_Click 事件，在该事件中主要实现将注册信息插入 tb_Login 数据表中，注册成功弹出“恭喜您!! 注册成功!”字样。否则弹出“很遗憾!! 注册失败!”。实现的代码如下。

例程 31 代码位置：光盘\mr\06\memo\Zhuce.aspx.cs

```
protected void ImgBtnLogin_Click(object sender, ImageClickEventArgs e)
    {
        try
        {
            //将信息插入到注册登录信息表中
            string InsertStr = "insert into tb_login values('" + this.TxtUserName.Text + "','" + this.TxtPwd.Text + "','" +
this.TxttruePwd.Text + "','" + this.TxtEmail.Text + "','" + this.TxtQQ.Text + "','" + this.TxtBeizhu.Text + "')";
            Memobwl.EXECCommand(InsertStr);//调用EP类中的静态EXECCommand方法,执行SQL语句
            //注册成功后弹出对话框,显示"恭喜您!! 注册成功！"
            Response.Write("<script lanuage='javaScript'>alert('恭喜您!!注册成功!');location='javascript:history.go
(-1)'</script>");//弹出对话框显示“恭喜您!! 注册成功!”
            this.clearText();//调用清空文本框的方法
        }
        catch
        {
            //注册失败后弹出对话框,显示"很遗憾!! 注册失败！"
            Response.Write("<script lanuage='javaScript'>alert('很遗憾!! 注册失败！');location='javascript:history.go(-1)'
</script>");//弹出对话框显示“很遗憾!! 注册失败!”
        }
    }
```

双击“重置”按钮，触发“ImgBtnCZ_Click”事件，在该事件中主要实现的是调用自定义的 clearText 方法，清空文本框的功能，实现的代码如下。

例程 32 代码位置：光盘\mr\06\memo\Zhuce.aspx.cs

```
protected void ImgBtnCZ_Click(object sender, ImageClickEventArgs e)
    {
        this.clearText();
    }
```

在自定义的 clearText 方法中实现，主要用来实现清空文本框的功能，实现的代码如下：

例程 33 代码位置：光盘/mr/06/memo/Zhuce.aspx.cs

```
public void clearText()
{
    /*清空文本框中的内容*/
    this.TxtUserName.Text = "";
    this.TxtPwd.Text = "";
    this.TxttruePwd.Text = "";
    this.TxtQQ.Text = "";
    this.TxtEmail.Text = "";
    this.TxtBeizhu.Text = "";
}
```

6.3.7 用户登录

在进入网站备忘录之前用户得先登录才能进入网站备忘录中，只有填写正确的用户名和密码才能进入登录成功后将弹出提示对话框，否则将不能进入该系统。用户登录页的运行效果如图 6.26 所示，弹出登录成功的提示对话如图 6.27 所示。

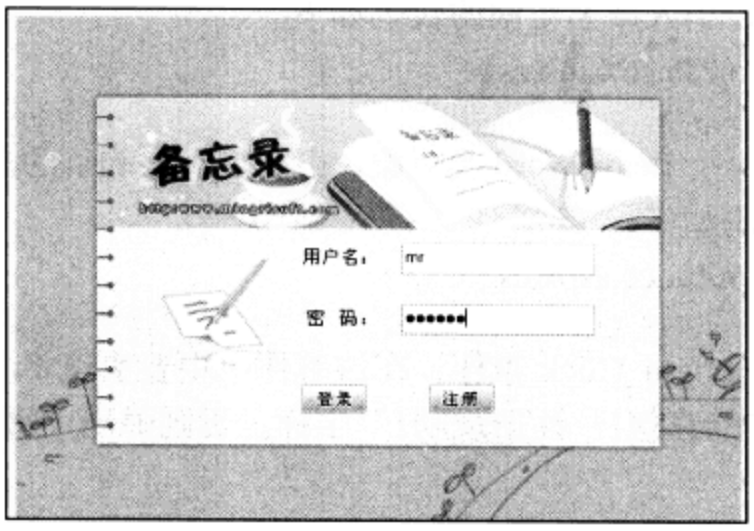


图 6.26 用户登录页的运行效果图

图 6.27 恭喜您，登录成功

1. 前台页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为 Login.aspx，作为网站备忘录的登录页。页面 Login.aspx 的设计界面如图 6.28 所示。

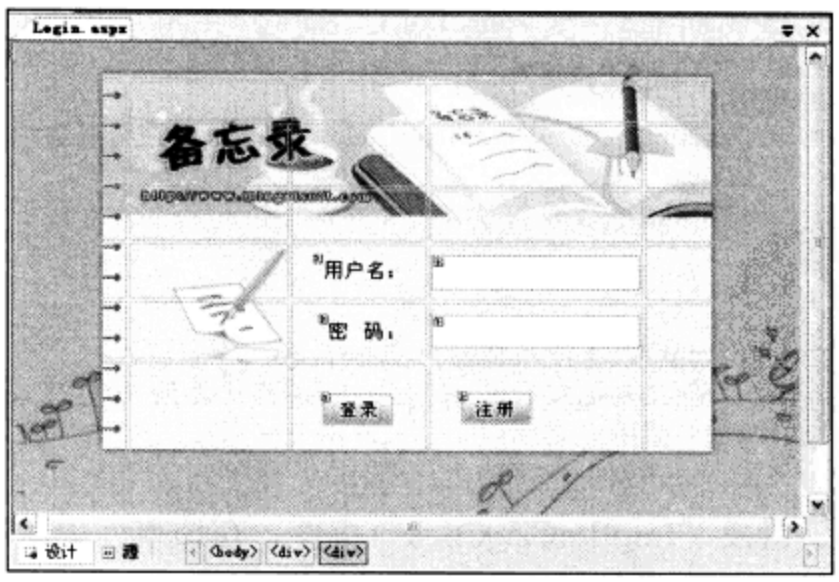


图 6.28 页面“login.aspx”的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Web 用户控件，2 个 TextBox 控件和 2 个 ImageButton 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 6.18 所示。

表 6.18　登录页控件列表

控件类型	控件名称	主要属性设置	用途
Table	无	无	页面布局
Web 用户控件	控件的名称为 logo.ascx	无	logo.ascx 实现网站的页面导航功能
TextBox	控件的名称分别为 TxtName、TxtPwd	其中 TxtPwd 的 TextMode 属性设置为 password	用于输入用户登录时的用户姓名和用户密码信息
ImageButton	控件的名称分别为 ImgBtnLogin 和 ImgBtnzhuce	将控件的 ImageUrl 属性分别设置为～/image/login.jpg 和～/image/zhuce.jpg	用于执行登录操作和将页面跳转到注册页面中

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下：

例程 34　代码位置：光盘\mr\06\memo\Login.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化一个类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下：

例程 35　代码位置：光盘\mr\06\memo\Login.aspx.cs

```
Memobwl ep = new Memobwl();
```

双击“登录”按钮，触发其 BtnLogin_Click 事件，在该事件中，首先调用用户自定义的 checkLogin 方法，该方法中设置了 2 个参数主要用来传递用户名及密码，并且将返回的结果赋值给整型变量 int 中，然后使用 if 语句判断变量 I 是否大于 0，如果大于 0 说明方法调用成功，即用户登录成功，并将用户名和用户密码分别赋值给 Session 变量。系统弹出登录成功的提示对话框“恭喜您，登录成功!”，并将页面跳转到 Default.aspx 页面中，否则调用方法失败，并弹出提示对话框显示“很遗憾，用户名和密码错误!”，实现的代码如下。

例程 36　代码位置：光盘\mr\06\memo\Login.aspx.cs

```
protected void ImgBtnLogin_Click(object sender, ImageClickEventArgs e)
{
    int i = this.checkLogin(this.TxtName.Text, this.TxtPwd.Text);

    if (i > 0)
    {
      Session["UserName"] =this.TxtName.Text;
      Session["UserPwd"] = this.TxtPwd.Text;
      Response.Redirect("Default.aspx");//登录成功后跳转到Default.aspx页面中
    }
    else//调用方法调用失败，弹出提示对话框
    {
        Response.Write("<script language=javascript>alert('很遗憾，用户名称或密码错误！');location='javascript:history.go
(-1)'</script>");
        }
    }
```

上述代码中调用了用户自定义的 checkLogin 方法，该方法主要用来判断用户输入到文本框中的内容是否和数据库中的数据一致，并且返回一个整数类型的变量。实现的代码如下：

例程 37　代码位置：光盘\mr\06\memo\Login.aspx.cs

```
#region
/// <summary>
/// 该方法用于判断用户输入到文本框中的内容是否和数据库中的数据一致
/// </summary>
/// <param name="loginName">传递用户名</param>
/// <param name="loginPwd">传递用户密码</param>
/// <returns></returns>
public int checkLogin(string loginName, string loginPwd)
```

```
        {
            SqlConnection con = lw.getcon();
             SqlCommand myCommand = new SqlCommand("select count(*) from tb_User where name=@loginName
and pwd=@loginPwd", con);
            myCommand.Parameters.Add(new SqlParameter("@loginName", SqlDbType.VarChar, 50));
            myCommand.Parameters["@loginName"].Value = loginName;
            myCommand.Parameters.Add(new SqlParameter("@loginPwd", SqlDbType.VarChar, 50));
            myCommand.Parameters["@loginPwd"].Value = loginPwd;
            myCommand.Connection.Open();//打开数据库连接
            int i = (int)myCommand.ExecuteScalar();//执行SQL语句
            myCommand.Connection.Close();//关闭数据库连接
            return i;
        }
        #endregion
```

说 明

上述代码中的 Check Login 方法（判断用户是否合法登录）主要通过带@的参数进行传值，这样可有效地防止 SQL 注入式攻击。

6.4 网站打包与发布

网站开发完成后最终的目的是将其发布到 Internet 上，以供用户浏览访问。实现的网站发布可以使用两种方法。第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍下使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

图 6.29 选择发布网站

（1）首先打开要发布的网站，在网站的项目名称上，单击右键后弹出的快捷方式菜单中选择“发布网站”选项，如图 6.29 所示。

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮，网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择“...”路径按钮。如图 6.30 所示。

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选项中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码，如图 6.31 所示。填写完毕后选择“打开”按钮返回到“发布网站”窗口，在该窗口中单击“确定”按钮，网站就会发布到 Internet 上。

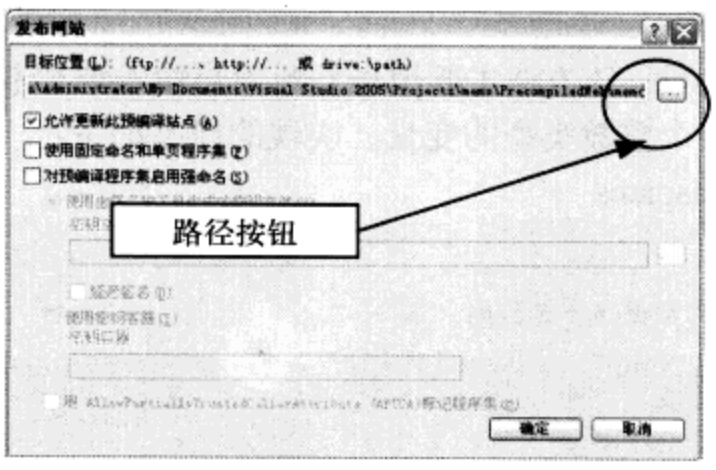


图 6.30 选择路径按钮

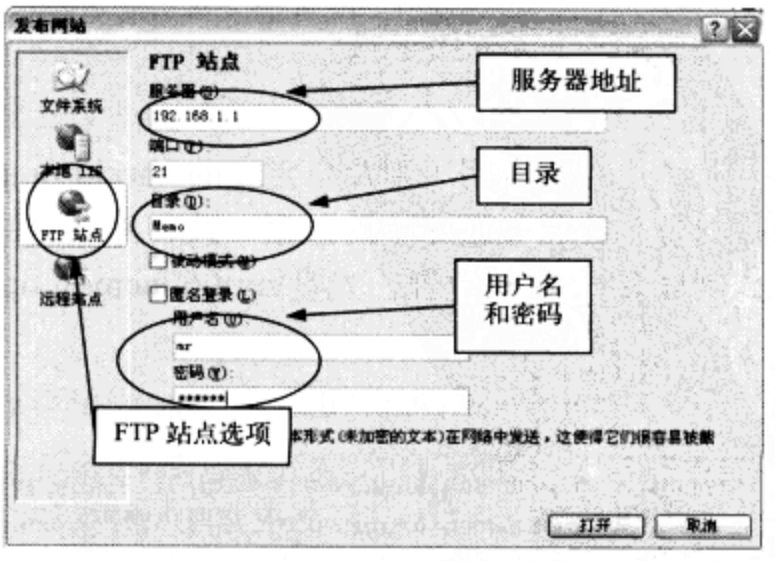


图 6.31 FTP 站点选项

电子邮件发送与接收模块

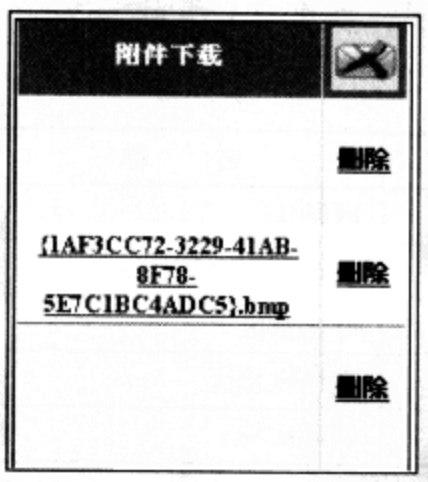
第7章

实例位置：光盘\mr\07\

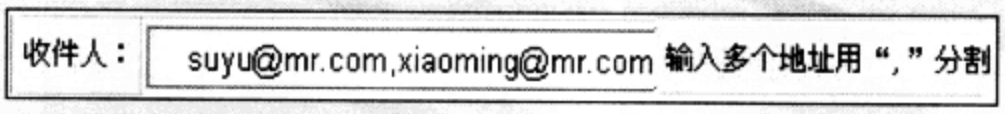
随着 Internet 技术的飞速发展，网络已经成为生活中不可缺少的一部分。网络中的通信大多是通过 E-mail 来实现的，因此邮件的使用尤为重要。本章主要利用 ASP.NET+ JMAIL 开发一个电子邮件发送与接收的程序。通过本章学习，读者将学到以下内容。

连接邮件

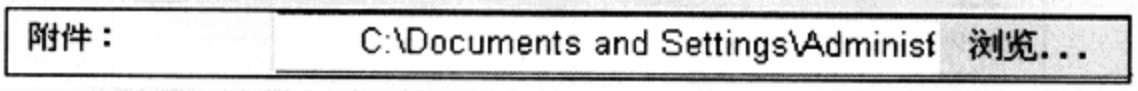
附件下载



群发邮件使用“,”隔开



发送附件



7.1 电子邮件发送模块功能概述

电子邮件已经成为大家最普遍的联系方式之一，在网站程序中往往需要使用电子邮件与客户取得联系，因此电子邮件模块也就成为网站的重要组成部分。本模块将通过 Jmail 组件实现收发电子邮件。

本模块实现的具体功能如下。

- 发送电子邮件。
- 发送带附件的电子邮件。
- 自动回复邮件。
- 群发电子邮件。
- 接收电子邮件。
- 接收电子邮件的附件并下载。

为了方便读者阅读和有效地利用本书附赠光盘中的实例，这里将网站页面的各部分说明以列表形式给出，如表 7.1 所示。

表 7.1　　网站首页解析

区　域	名　称	说　明	对 应 文 件
1	网站头	主要用于网站的旗帜广告	sendOutEmail.aspx
2	发送和接收邮件区	主要用于邮件发送与接收操作	sendOutEmail.aspx
3	好友录区	执行添加好友和管理好友操作	sendOutEmail.aspx
4	友情链接区	显示友情链接	sendOutEmail.aspx
5	网站脚标	显示客服热线电话等一些信息	sendOutEmail.aspx

电子邮件发送运行效果图 7.1 所示。

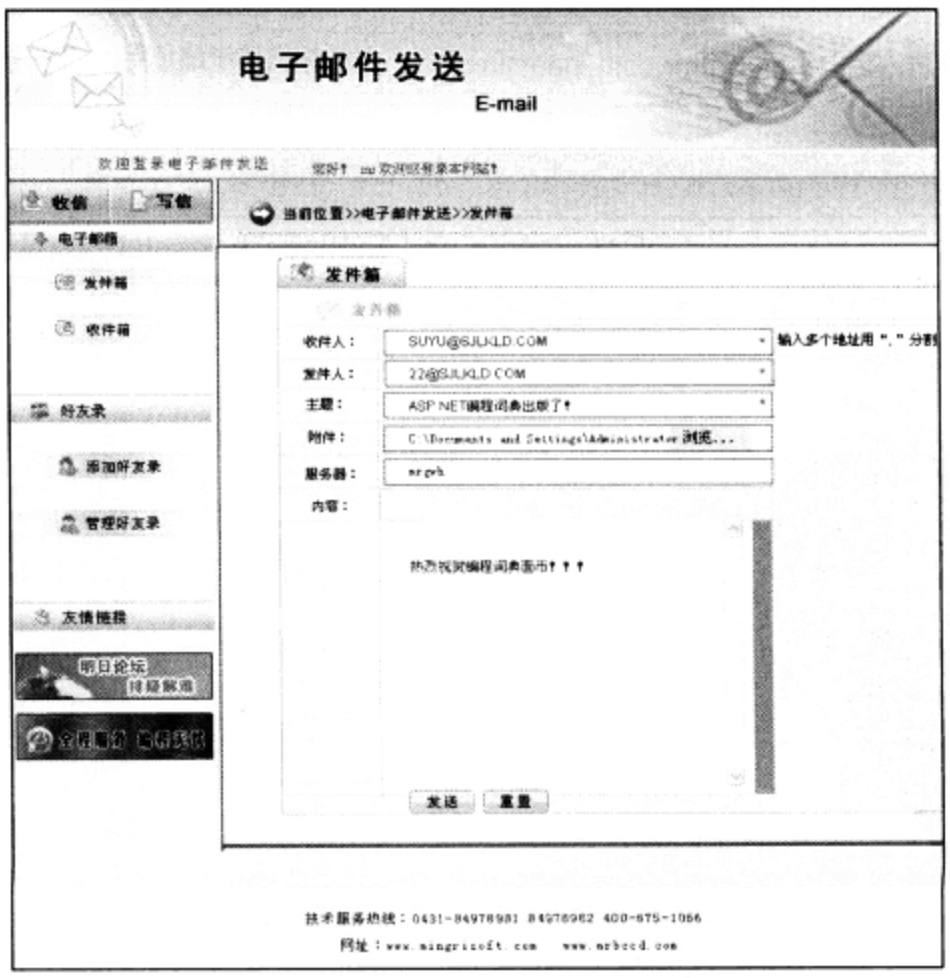


图 7.1　电子邮件发送运行效果图

7.2 实现电子邮件发送与接收的关键技术

7.2.1 引入 Jmail 组件到 ASP.NET 中

Jmail 组件是由 Dimac 公司开发的，用来完成邮件的发送、接收、加密和集群传输等工作。它支持从 POP3 邮件服务器收取邮件，支持加密邮件的传输，其发送邮件的速度快，功能丰富，并不需要 Outlook 之类的邮件客户端，而且是免费使用的，是使用非常广泛的邮件发送组件。

在使用 Jmail 组件发送电子邮件之前，首先需要添加对 Jmail 组件的引用，具体步骤如下。

（1）在“解决方案资源管理器”中找到要添加引用的网站项目，单击鼠标右键，在弹出的快捷菜单中选择“添加引用”命令，在项目中添加引用，如图 7.2 所示。

（2）在打开的“添加引用”对话框中选择“浏览”选项卡并选择要引用的 jmail.dll 文件，单击“确定”按钮，将 Jmail 组件添加到网站项目的引用中，然后就可以直接在后台代码中使用其属性和方法，“添加引用”对话框如图 7.3 所示。

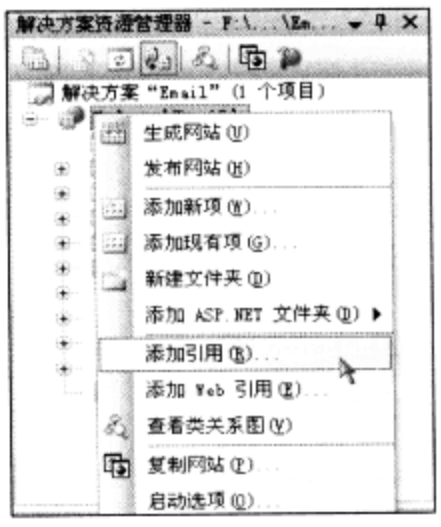


图 7.2 在项目中添加引用

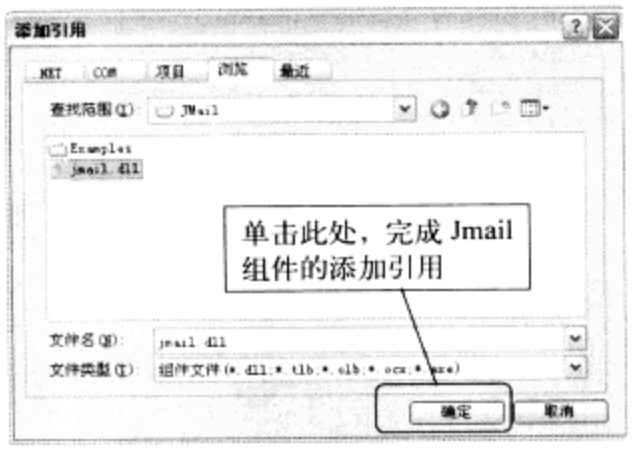


图 7.3 “添加引用”对话框

说 明 Jmail 组件不是 ASP.NET 3.5 中自带的组件，使用时需要安装，并且在本地计算机上要注册该组件。例如，该组件放在 C:\Jmail\Jmail.dll 下，注册时只需在“运行”里运行“Regsvr32 C:\Jmail\Jmail.dll”即可。

7.2.2 配置 POP3 服务

实现电子邮件的接收功能时，首先需要配置 POP3 服务，配置 POP3 服务实现的具体步骤如下。

（1）在“控制面板”中双击“添加或删除程序”选项，如图 7.4 所示，将弹出“添加或删除程序”窗口，如图 7.5 所示。

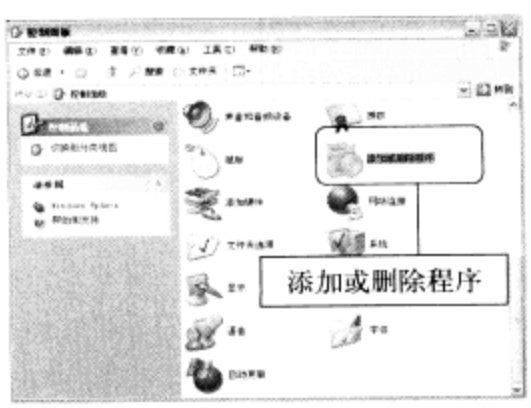


图 7.4 双击“添加或删除程序”

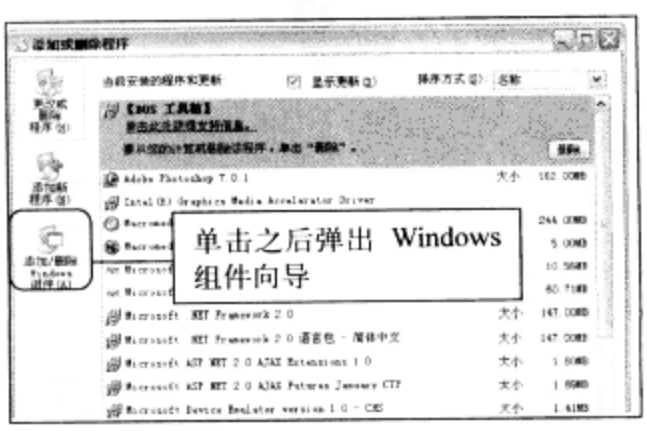


图 7.5 “添加或删除程序”窗口

（2）单击“添加/删除 Windows 组件”图标按钮，会弹出“Windows 组件向导”对话框。在“Windows 组件向导”对话框中选择“电子邮件服务”复选框，如图 7.6 所示。电子邮件服务对话框如图 7.7 所示。依次单击“确定”按钮，完成“电子邮件服务”的添加操作。

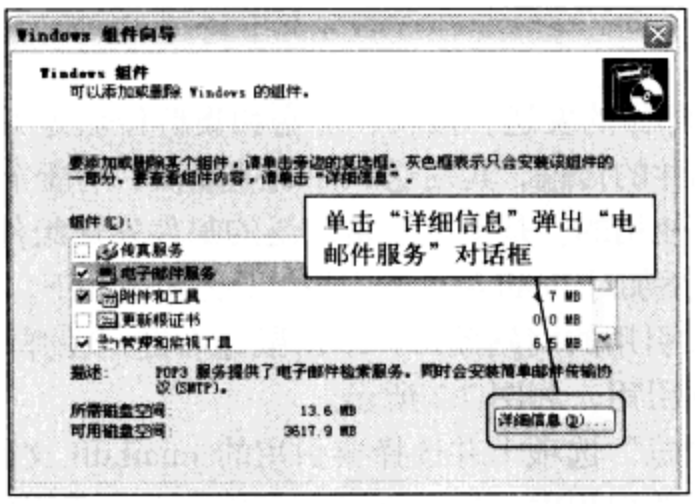


图 7.6 “Window 组件向导”对话框

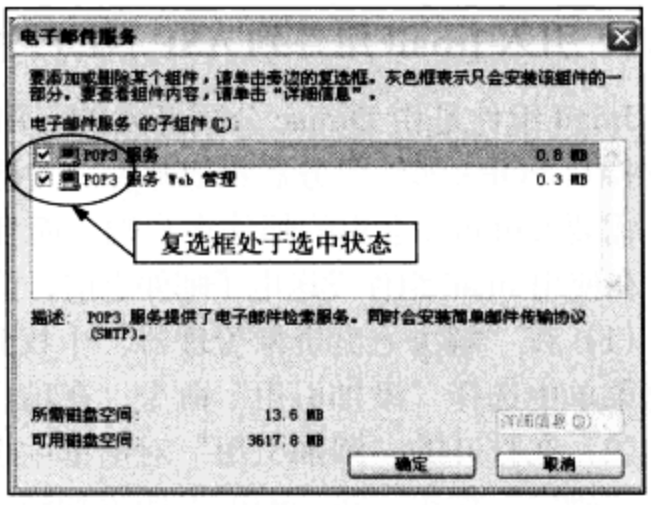


图 7.7 “电子邮件服务”对话框

（3）当添加完“电子邮件服务”后，打开“管理工具”窗口，这时将会出现“POP3 服务”功能，如图 7.8 所示。

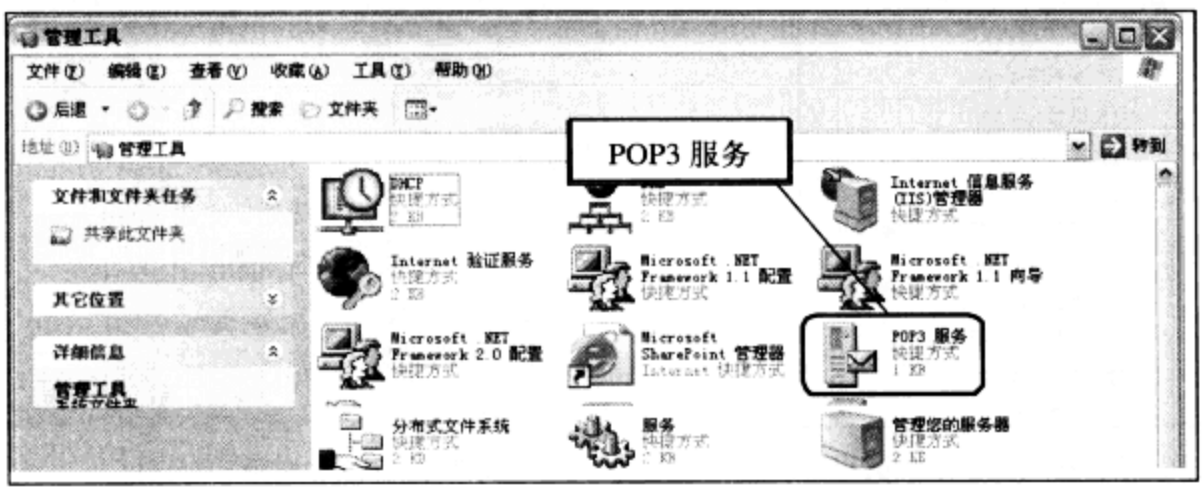


图 7.8 “管理工具”窗体

7.2.3 在 POP3 服务中添加域

当配置好 POP3 服务后，便可以在 POP3 服务当中添加域，实现的具体步骤如下。

双击打开“POP3”服务，用鼠标右键单击“MRGWH”，弹出“添加域”窗口。添加一个新域（如“mr.com”），单击“确定”按钮，如图 7.9 所示。

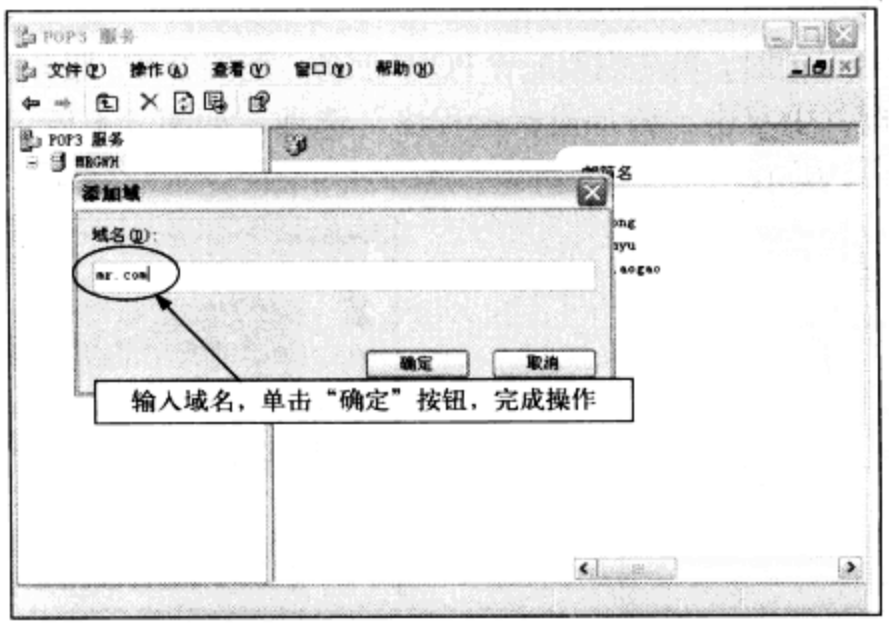


图 7.9 “POP3 服务”窗体

7.2.4 在域中添加新邮箱

当添加好域后，需要向域中添加新邮箱，当添加完“域”后，选中相应的域，单击“添加邮箱”超级链接，弹出“添加邮箱”对话框。在该对话框中添加邮箱名和密码，如图 7.10 所示。单击“确定”按钮完成指定域内邮箱的添加。

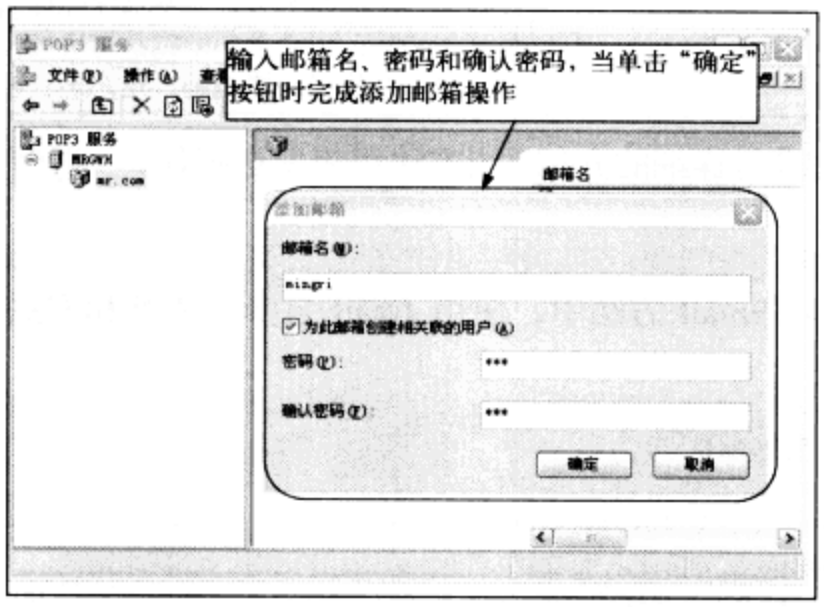


图 7.10 “添加邮箱”对话框

说明 所有域都在本机上分出一定的空间来存放信息，默认位置为 C:\Inetpub\mailroot\Mailbox。

7.2.5 邮件发送核心技术

发送功能主要通过自定义方法 sendEmail 实现。在自定义方法 sendEmail 中利用 Jmail...Message Class 类中的相关属性和方法来发送邮件。Jmail.MessageClass 类中的主要属性如表 7.2 所示。

表 7.2 Jmail...MessageClass 类中的属性及说明

属　性	说　明
Attachments	返回邮件的附件集合
Charset	设置使用的邮件字符集，默认 US-ASCII 中国则为 GB2312
ISOEncodeHeaders	邮件头是否使用 ISO-8859-1 编码 默认值为 True
From	返回或设置发件人的邮件地址
Subject	邮件的主题（标题）
Body	邮件的正文
Prority	返回或设置邮件的优先级
Encoding	设置附件默认编码。有效选项是“base64”或者“quoted-printable”
Date	返回邮件发送时间

其中将对 Prority 属性进行详细的介绍，该属性用来返回或设置邮件的优先级，一共有 5 个级别，1 为最快，5 为最慢。用法如下：

```
Message.Prority=1;        //设置为最快
Response.Write(Messgae.Prority)        //输出优先级
```

Jmail...MessageClass 类中的主要方法如表 7.3 所示。

表 7.3　　Jmail.MessageClass 类中的方法及说明

方　法	说　明
AddRecipient(emaillAddress,recipientName,PGPKey)	为邮件添加一个收件人
AddAttachment(FileName,isInline,ContentType)	为邮件添加一个文件型的附件。如果 Inline 属性被设置为 True，这个附件就是一个可嵌入的附件
Send(mailServer,enque)	发送邮件。邮件服务器是一个描述邮件服务器名称或地址的字符串（包括引号），用户名和密码是可选项。当邮件服务器需要发信认证时可使用，使用的格式是，用户名：密码@邮件服务器

其中将对 Send 方法进行详细的介绍，该方法用来发送邮件，一般情况只使用服务器参数即可。用法如下：

```
Message.Send(server)
```

在用户自定义的 SendEmail 方法中，使用 Jmail 自身的属性和方法实现邮件的发送功能，实现的代码如下：

```
/// <summary>
    /// 自定义方法用来发送邮件
    /// </summary>
    /// <param name="Sender">发件人地址</param>
    /// <param name="Receiver">收件人地址</param>
    /// <param name="Subject">邮件标题</param>
    /// <param name="Content">邮件内容</param>
    /// <param name="server">服务器名</param>
    /// <returns>返回一个布尔值，如果返回True表示发布成功！否则为False</returns>
    public bool sendEmail(string Sender, string Receiver, string Subject, string Content, string server)
    {
        int sunEmail = 0;
        jmail.MessageClass myJmail = new jmail.MessageClass();
        myJmail.Charset = "GB2312"; //设置使用的邮件字符集，默认US-ASCII 中国则为GB2312
        myJmail.ISOEncodeHeaders = false; //邮件头是否使用ISO-8859-1编码 默认值为True
        myJmail.From = Sender;   //返回或设置发件人的地址
        myJmail.Subject = Subject;    //邮件的主题（标题）
        myJmail.AddRecipient(Receiver, "", ""); //添加收件人
        if (FileUp.PostedFile.ContentLength != 0)
        {
            string filePath = FileUp.PostedFile.FileName;
            myJmail.AddAttachment(@filePath,false,"");   //添加一个附件
        }
        myJmail.Body = Content;   //邮件的正文
        return myJmail.Send(server, true);//发送邮件
    }
```

7.2.6　邮件接收核心技术

1. 打开同 POP3 服务器连接的方法

打开同 POP3 服务器的连接，可以设置端口，默认端口是 110。

语法格式如下：

```
Connect(Username,Password,Server,Port)
```

参数说明如下。

- Username：连接 POP3 服务器的用户名。
- Password：连接 POP3 服务器的密码。
- Server：连接 POP3 服务器的服务器的名称。
- Port：连接 POP3 服务器的端口号。

例如，打开 POP3 服务器连接，用户名和密码分别为“mr”和“mrsoft”。服务器为“mr.com”默认端口 110。

实现的代码如下：

```
mailbox.Connect "mr","mrsoft","mr.com";
```

2. 从邮件服务器上删除所有邮件

从邮件服务器上删除所有邮件的方法如下：

```
DeleteMessages()
```

例如，删除邮件服务器上的所有邮件。

实现的代码如下：

```
Mailbox.DeleteMessages();
```

3. 从邮件服务器上删除指定邮件

从邮件的服务器上删除指定的邮件的方法如下：

```
DeleteSingleMessage(MessageID)
```

其中 MessageID 指的是邮件的索引。

例如，从邮件的服务器上删除指定的邮件。

实现的代码如下：

```
Mailbox.DeleteSingleMessage(1)
```

4. 关闭同邮件服务器的连接

当不再使用邮箱的时候，可以将邮件的服务器的连接关闭。

关闭同邮件服务器的连接的方法如下：

```
Disconnect()
```

例如，关闭服务器的连接。

实现的代码如下：

```
mailbox.Disconnect();
```

7.3 电子邮件发送与接收的实现过程

7.3.1 单用户发送和群发邮件

电子邮件发送是利用 Jmail 组建进行发送的。在实现发送邮件时，不但实现了单个用户发送邮件的功能，而且还实现了邮件群发的功能，单用户发送和群发邮件页运行结果如图 7.11 所示。

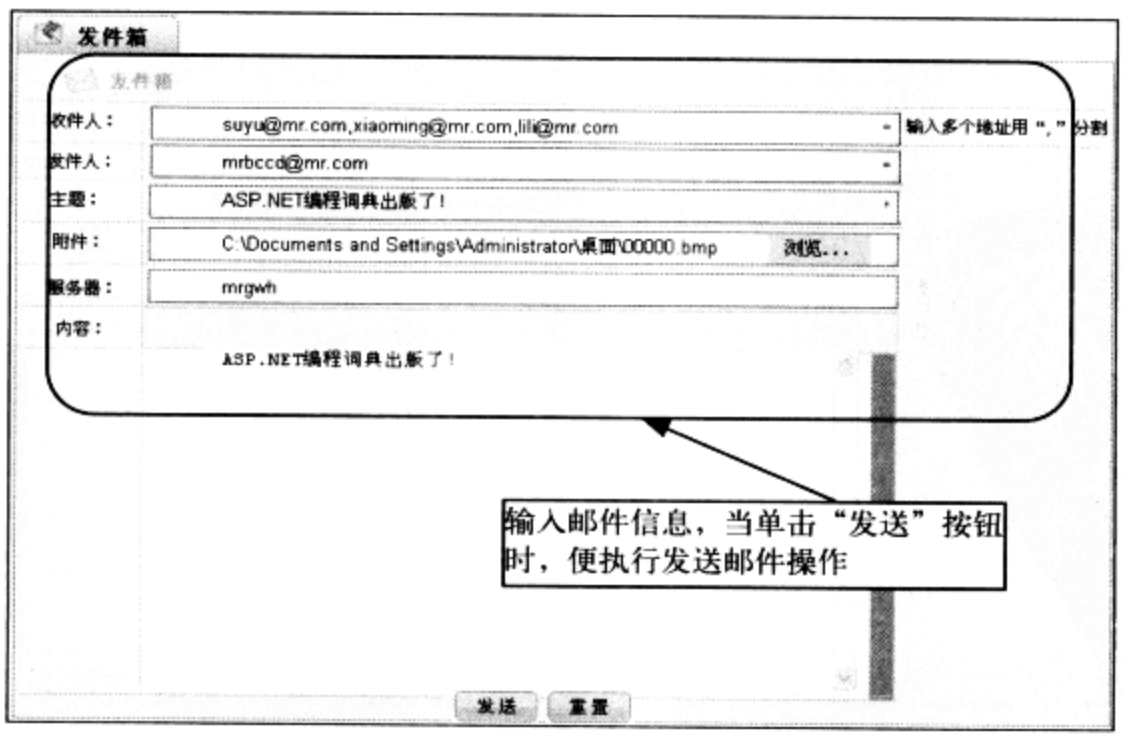


图 7.11 单用户发送和群发邮件

在 sendOutEmail.aspx 页面中，当邮件发送成功后，这时在发件人的邮箱中，便会收到一封回复信，并且这封信是自动回复的。如果发件人群发邮件时，给多少人发送成功，便在发件人的信箱中收到几封回信。打开 C:\Inetpub\mailroot\Mailbox\mr.com\P3_suyu.mbx 路径下的邮箱，如图 7.12 所示。在该邮箱中查看到一封自动回复邮件，如图 7.13 所示。

图 7.12　自动回复邮件运行效果图

图 7.13　查看一封自动回复邮件运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为“sendOutEmail.aspx”，作为单用户发送和群发邮件页。页面 sendOutEmail.aspx 的设计界面如图 7.14 所示。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 5 个 TextBox 控件、1 个 FileUpLoad 控件、2 个 ImageButton 控件和 3 个 RequiredFieldValidator 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 7.4 所示。

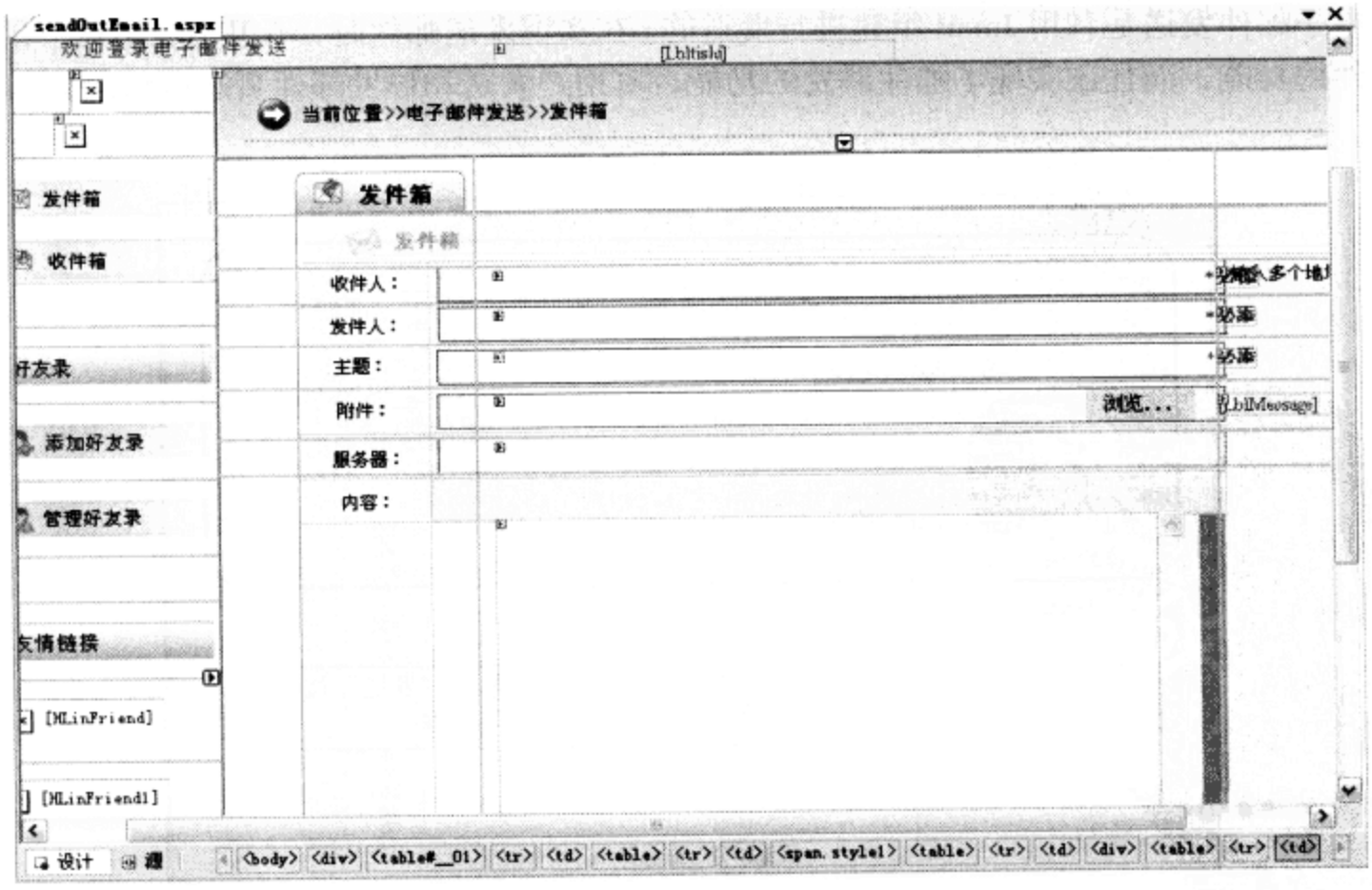


图 7.14　页面 sendOutEmail.aspx 的设计界面图

表 7.4　　单用户发送和群发邮件页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
TextBox	控件的名称分别为 TxtReceive、TextSender、TextSubject、TxtServer 和 TextContent	将 TextContent 控件的 TextMode 属性设置为 MultiLine	用于向邮件中输入信息
FileUpload	控件的名称分别为 FileUp	将控件的 OnLoad 属性设置为 FileUp_Load	用于浏览发送的附件
ImageButton	控件的名称分别设置为 ImgBtnSend 和 ImgBtnchongzhi	将 ImgBtnSend 控件的 ImageUrl 属性设置为～/images/fasong.bmp，将 OnClick 属性设置为 ImgBtnSend_Click 将 ImgBtnchongzhi 控件的 ImageUrl 属性设置为～/images/chongzhi.bmp，将 OnClick 属性设置为 ImgBtnchongzhi_Click	用于执行发送和重置操作
RequiredFieldValidator	将控件的名称分别设置为 rfvSj、rfvfj 和 rfvSubject	例如：将 rfvSj 控件的 ControlToValidate 属性设置为 TxtReceive，将 ErrorMessage 属性设置为必添	验证文本框是否为空

2. 代码实现

自定义 hf()方法，主要是用来判断邮件是否回复成功，成功的话判断是否回复的多条，是多条提示回复多条成功，否则提示回复邮件成功，实现的代码如下：

例程 1　代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
public void hf()
{
    bool issend = false;
    string receiver = this.TextBox1.Text;   //获取收件人地址
    string[] receivers = receiver.Split(',');   // 把多个收件人地址以", "为分隔符拆分为数组
    string receiverAdr = "";      //存储收件人邮件
    string subject="邮件自动回复成功！ ";
    string body = "恭喜你！ 邮件发送成功！ ";
    for (int i = 0; i < receivers.Length; i++)
    {
        receiverAdr = receivers[i];
        //调用自定义方法并传送相应信息（发件人地址，收件人地址，标题，内容，服务器名）
        issend = sendEmail(receiverAdr,TextSender.Text.Trim() , subject, body, TextBoxServer.Text.Trim());
         }
            // 判断是否回复成功，若成功则判断是否回复的多条，是多条提示回复多条成功，否则提示回复邮
件成功
         if (issend)
         {
             if (receivers.Length > 1)
             {
                 this.LblMessage.Text = "自动回复成功，你邮箱里有多条邮件未读！ ";
             }
             else
             {
                 this.LblMessage.Text = "自动回复成功，你邮箱里有1条邮件未读！ ";
             }
         }
     else
     {
         this.LblMessage.Text = "自动回复失败";
     }
 }
```

在 sendEmail 自定义的方法中，将向该方法中传递 5 个参数，即 Sender、Receiver、Subject、Content、server，分别为发件人地址、收件人地址、邮件标题、邮件内容和服务器名。并且返回一个布尔类型的变量，如果返回 True 表示发送成功！否则为 False 说明发送失败。实现的代码如下。

例程 2 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
/// <summary>
/// 自定义方法用来发送邮件
/// </summary>
/// <param name="Sender">发件人地址</param>
/// <param name="Receiver">收件人地址</param>
/// <param name="Subject">邮件标题</param>
/// <param name="Content">邮件内容</param>
/// <param name="server">服务器名</param>
/// <returns>返回一个布尔值，如果返回True表示发布成功！否则为False</returns>
public bool sendEmail(string Sender, string Receiver, string Subject, string Content, string server)
{
    int sunEmail = 0;
    jmail.MessageClass myJmail = new jmail.MessageClass();
    myJmail.Charset = "GB2312"; //设置使用的邮件字符集，默认US-ASCII 中国则为GB2312
    myJmail.ISOEncodeHeaders = false; //邮件头是否使用ISO-8859-1编码 默认值为true;
    myJmail.From = Sender;   //返回或设置发件人的地址
    myJmail.Subject = Subject;    //邮件的主题（标题）
    myJmail.AddRecipient(Receiver, "", ""); //添加收件人
    if (FileUp.PostedFile.ContentLength != 0)
    {
        string filePath = FileUp.PostedFile.FileName;
        myJmail.AddAttachment(@filePath,false,"");   //添加一个附件
    }
    myJmail.Body = Content;   //邮件的正文
    return myJmail.Send(server, true);//发送邮件
}
```

双击“收件箱”按钮，触发其在 ImgBtnReceive_Click 事件，在该事件中使用 Response 的 Redirect 方法实现将页面跳转到 receiveEmail.aspx 页面中，实现的代码如下。

例程 3 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void ImgBtnReceive_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("receiveEmail.aspx");//跳转到receiveEmail.aspx页面中
}
```

双击“发件箱”按钮，触发其 ImgBtnSendOut_Click 事件，在该事件中使用 Response 的 Redirect 方法可实现将页面跳转到 sendOutEmail.aspx 页面中，实现的代码如下。

例程 4 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void ImgBtnSendOut_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("sendOutEmail.aspx");//跳转到sendOutEmail.aspx页面中
}
```

在 ImgBtnDelete_Click 事件中，使用 Response 的 Redirect 方法可实现将页面跳转到 DeleteMail.aspx 页面中，实现的代码如下。

例程 5 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void ImgBtnDelete_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("DeleteMail.aspx");
}
```

双击“发件箱”按钮，触发 ImageButton1_Click 事件，在该事件中使用 Response 的 Redirect

方法实现将页面跳转到电子邮件发送（sendOutEmail.aspx）页面中，实现的代码如下：

例程 6 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
    {
        Response.Redirect("sendOutEmail.aspx");//跳转到sendOutEmail.aspx页面中
    }
```

双击“收件箱”按钮，触发 ImageButton2_Click 事件，在该事件中使用 Response 的 Redirect 方法实现将页面跳转到电子邮件接收（reveiveEmail.aspx）a 页面中，实现的代码如下：

例程 7 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
    {
        Response.Redirect("receiveEmail.aspx");//跳转到receiveEmail页面中
    }
```

在 TextSender_Load 事件中，在 TextSubject 文本框中添加两个属性，实现将鼠标停留或者离开文本框的时候改变文本框的背景颜色，实现的代码如下。

例程 8 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void TextBox1_Load(object sender, EventArgs e)
    {
        this.TextBox1.Attributes.Add("onmouseover", "c=this.style.backgroundColor;this.style.backgroundColor='#f3f8fc'");
        this.TextBox1.Attributes.Add("onmouseout", "this.style.backgroundColor=c");
    }
```

在 TextSender_Load 事件中，在 TextSubject 文本框中添加两个属性，实现将鼠标停留或者离开文本框的时候改变文本框的背景颜色，实现的代码如下。

例程 9 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void TextSender_Load(object sender, EventArgs e)
        {
            this.TextSender.Attributes.Add("onmouseover","c=this.style.backgroundColor;this.style.backgroundColor= '#f3f8fc'");
            this.TextSender.Attributes.Add("onmouseout", "this.style.backgroundColor=c");
        }
```

在 TextSubject _Load 事件中，在 TextSubject 文本框中添加两个属性，实现将鼠标停留或者离开文本框的时候改变文本框的背景颜色，实现的代码如下。

例程 10 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void TextSubject_Load(object sender, EventArgs e)
        {
            this.TextSubject.Attributes.Add("onmouseover", "c=this.style.backgroundColor;this.style.backgroundColor='#f3f8fc'");
            this.TextSubject.Attributes.Add("onmouseout", "this.style.backgroundColor=c");
        }
```

在 TextContent_Load 事件中，在 TextContent 文本框中添加两个属性，实现将鼠标停留或者离开文本框的时候改变文本框的背景颜色，实现的代码如下。

例程 11 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void TextContent_Load(object sender, EventArgs e)
        {
            this.TextContent.Attributes.Add("onmouseover", "c=this.style.backgroundColor;this.style.backgroundColor= '#f3f8fc'");
            this.TextContent.Attributes.Add("onmouseout", "this.style.backgroundColor=c");
        }
```

在 FileUp_Load 事件中，在 FileUp 控件中中添加两个属性，实现将鼠标停留或者离开文本框的时候改变该控件的背景颜色，实现的代码如下。

例程 12　代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void FileUp_Load(object sender, EventArgs e)
    {
        this.FileUp.Attributes.Add("onmouseover", "c=this.style.backgroundColor;this.style.backgroundColor='#f3f8fc'");
        this.FileUp.Attributes.Add("onmouseout", "this.style.backgroundColor=c");
    }
```

在 TextBoxServer_Load 事件中，在 TextBoxServer 文本框中添加两个属性，实现将鼠标停留或者离开文本框的时候改变文本框的背景颜色，实现的代码如下：

例程 13　代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
    protected void TextBoxServer_Load(object sender, EventArgs e)
    {
        this.TextBoxServer.Attributes.Add("onmouseover", "c=this.style.backgroundColor;this.style.backgroundColor='#f3f8fc'");
        this.TextBoxServer.Attributes.Add("onmouseout", "this.style.backgroundColor=c");
    }
```

在 ImgBtnSend_Click 单击事件中，首先调用用户自定义的方法并且传递“发件人地址”、“收件人地址”，“标题”、“内容”和服务器名称的相应信息。然后判断是否发送成功，若成功则判断是否发送的多条，是多条提示发送多条成功，否则提示发送邮件成功，实现的代码如下。

例程 14　代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
 protected void ImgBtnSend_Click(object sender, ImageClickEventArgs e)
{
    bool issend = false;
    string receiver = TextBox1.Text;   //获取收件人地址
    string[] receivers = receiver.Split(','); // 把多个收件人地址以“,”为分隔符拆分为数组
    string receiverAdr = "";      //存储收件人邮件
    for (int i = 0; i < receivers.Length; i++)
    {
        receiverAdr = receivers[i];
        //调用自定义方法并传送相应信息（发件人地址，收件人地址，标题，内容，服务器名）
        issend = sendEmail(TextSender.Text.Trim(), receiverAdr, TextSubject.Text.Trim(), TextContent.Text.Trim(),
TextBoxServer.Text.Trim());
    }
        //判断是否发送成功，若成功则判断是否发送的多条，是多条提示发送多条成功，否则提示发送邮件成功
    if (issend)
    {
        if (receivers.Length > 1)
        {
            Page.RegisterStartupScript("rty", "<script>alert('恭喜您！发送多条邮件成功！');location='sendOutEmail.
aspx'</script>");
        }
        else
        {
            Page.RegisterStartupScript("rty", "<script>alert('恭喜您！发送邮件成功！');location='sendOutEmail.
aspx'</script>");
        }
          this.hf();
        }
    else
    {
          Page.RegisterStartupScript("failing", "<script>alert('很遗憾！发送失败！');location='sendOutEmail.
aspx'</script>");
    }
    }
```

双击页面上的“重置”按钮，触发其 ImgBtnchongzhi_Click 事件，在该事件中实现清空文本框的功能，实现的代码如下。

例程 15 代码位置：光盘\mr\07\Email\ sendOutEmail.aspx.cs

```
protected void ImgBtnchongzhi_Click(object sender, ImageClickEventArgs e)
{
    this.TextBox1.Text = "";//清空收件人文本框中的文本信息
    this.TextSender.Text = "";//清空发件人文本框中的文本信息
    this.TextContent.Text = "";//清空内容文本框中的文本信息
    this.TextSubject.Text = "";//清空主题文本框中的文本信息
    this.TextBoxServer.Text = "";//清空服务器文本框中的文本信息
}
```

7.3.2 电子邮件接收

在电子邮件接收（receiveEmail.aspx）页中主要使用 POP3 协议和 Jmail 组件来接收电子邮件，POP3 协议是利用系统本身自带的功能来实现对电子邮件的接收，虽然使用 POP3 协议接收电子邮件相对简单，但是其功能不是很完善，仍存在许多缺点。而利用 Jmail 组件接收电子邮件相对 POP3 协议更为简洁方便，而且功能强大，大大减少了程序员开发项目时书写代码的数量，现在大多数的电子邮件收发系统都是通过 Jmail 组件来开发的。

本实例不仅可以接收邮件的相应信息，还可以下载附件。所谓附件就是 E-mail 在发送邮件的同时将一些其他格式的文件以附件的方式发送出去，这样当对方接收邮件的时候，邮件中除了主题、发件人和邮件内容外还附加了一个附件。可以将附件下载到本地计算机上进行阅读。这样就相应地出现了带附件的邮件的接收技术。当然接收带附件的邮件的方法有很多，本实例使用 Jmail 组件来接收电子邮件及下载附件。运行效果如图 7.15 图所示。

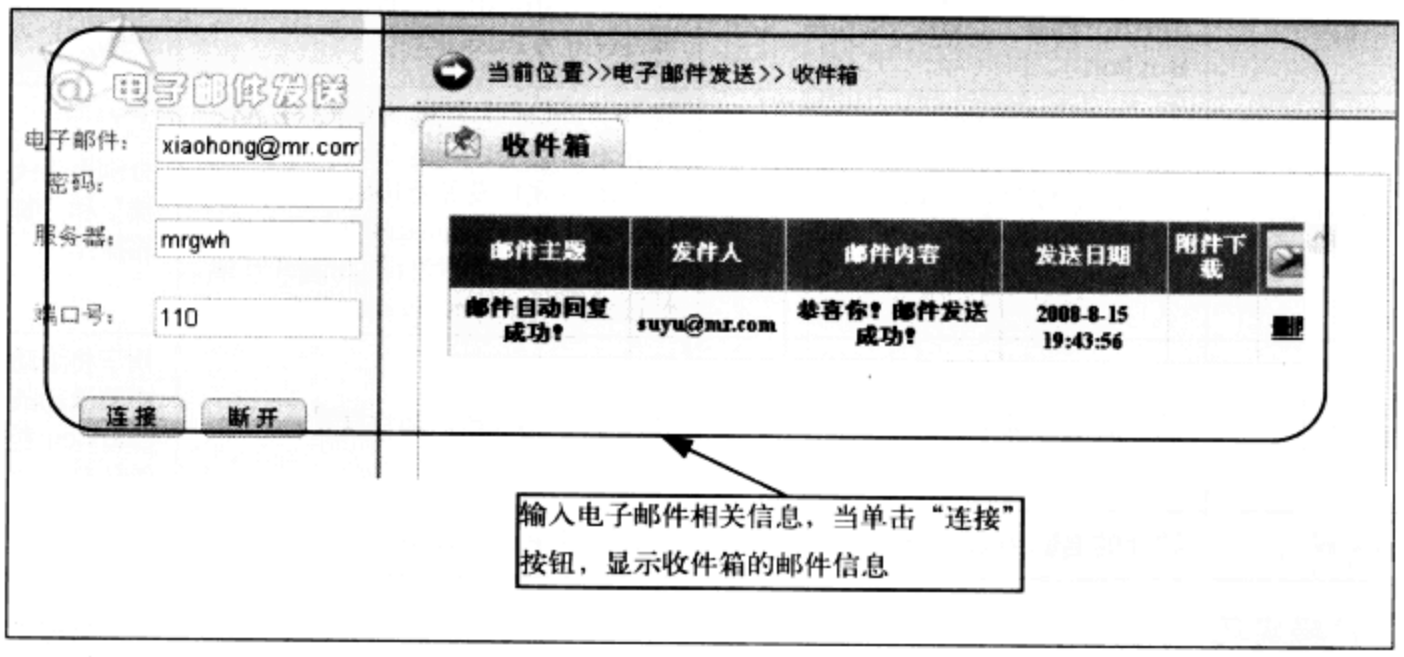


图 7.15 电子邮件接收运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为“receiveEmail.aspx”，作为电子邮件接收页。页面 receiveEmail.aspx 的设计界面如图 7.16 所示。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 4 个 TextBox 控件、2 个 ImageButton 控件和 3 个 RequiredFieldValidator 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 7.5 所示。

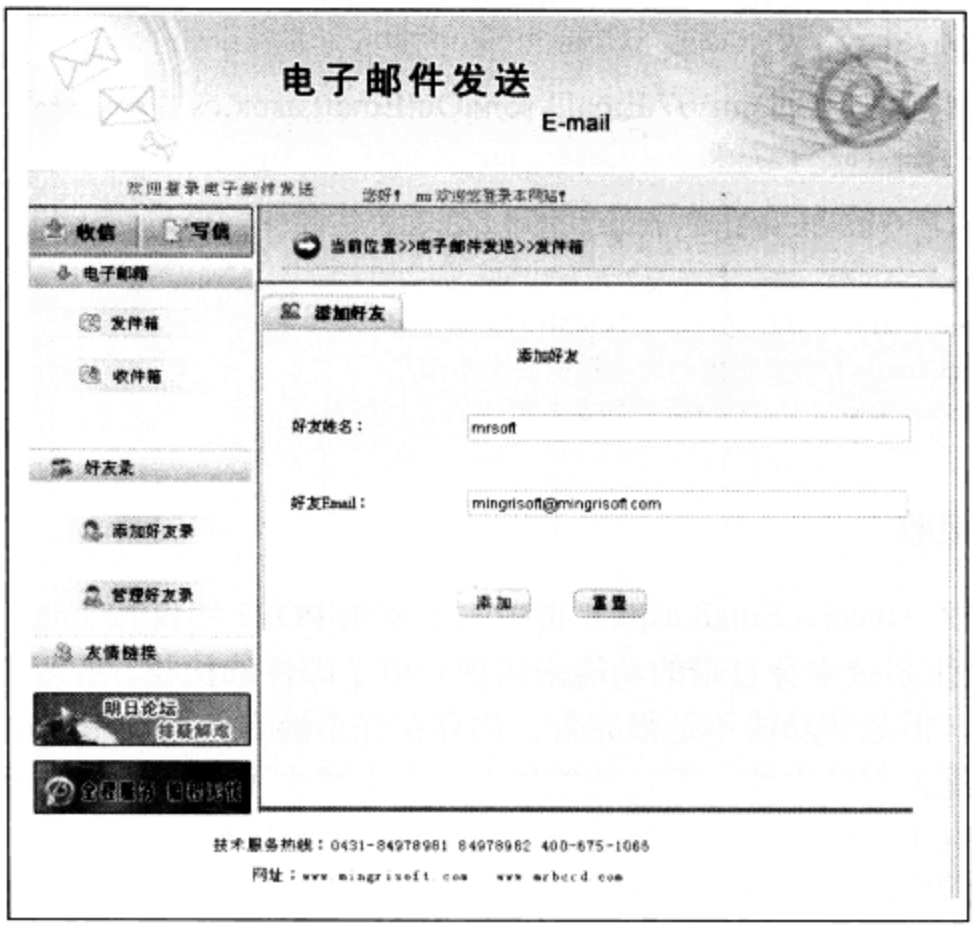


图 7.16　页面 receiveEmail.aspx 的设计界面图

表 7.5　　电子邮件接收页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
TextBox	控件的名称分别为 TextBoxName、TextBoxPass、TextBoxServer、Text BoxPort	其中将 TextBoxPass 控件的 Text Mode 属性设置为 Password	用于输入电子邮件的连接信息
ImageButton	控件的名称分别为 ImgBtn Lianjie 和 ImgBtnDuankai	将 ImgBtnLianjie 控件的 OnClick 属性设置为 ImgBtnLianjie_Click，将 Causes Validation 属性设置为 False，将 Image Url 属性设置为～/images/ lianjie.JPG。将 ImgBtnDuankai 控件的 ImageUrl 属性设置为～/images/ duankai.JPG	分别用于执行“连接”和“断开”邮箱操作
Panel	控件的名称为 Pemail	将控件的 ScrollBars 属性设置为 Both	用于将滚动该控件中的滚动条，显示 GridView 控件中的信息
GridView	控件的名称为 GvEmail	将控件的 PageSize 属性设置为“10”	用于显示电子邮件

2. 代码实现

该页在实现电子邮件接收时，首先需要定义 5 个静态变量，如建立接收邮件对象变量等。实现的代码如下。

例程 16　代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
//定义5个静态变量
public static jmail.POP3Class jpop; //建立接收邮件对象
public static jmail.Attachments atts;//建立附件集接口
public static jmail.POP3Class popMail;          //实例化POP3类对象
public static jmail.Attachment att = new jmail.AttachmentClass();//建立附件对象
public static string path; //用于接收路径
public static jmail.Message mailMessage;        //实例化邮件对象
```

接下来考虑如何查看接收到的邮件并把邮件附件保存在指定的目录下。首先建立一个自定义方法 LookEmail 并接收一个整数，整数用于表示第几封邮件。通过接收到的整数使用 POP3Class 这里主要应用了 DownloadSingleMessage 方法来获取指定的邮件并将其追加到 Message 类中，然后利用 Message 类中的部分属性来获取邮件的相应信息。最后考虑附件如何处理，这需要知道当前的邮件是否存在附件，通过判断附件集合就可知道是否存在附件。存在附件需要把附件的名称显示出来，并创建一个以邮箱地址为名的文件夹，该文件夹中用于暂时存放附件。LookEmail 方法的代码如下。

例程 17 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
// 自定义方法用来查看邮件
protected void lookEmail(int num)
{
  //获取返回的指定邮件，并将其追加到Message类中
   jmail.MessageClass jme = (jmail.MessageClass)jpop.DownloadSingleMessage(num);
   atts = jme.Attachments; //获取附件集合
   att = atts[0]; //获取第1个附件
   jme.Charset = "GB2312";  //设置使用的邮件字符集，默认US-ASCII 中国则为GB2312
   jme.Encoding = "Base64"; //设置附件默认编码
   jme.ISOEncodeHeaders = false;  //邮件头是否使用ISO-8859-1编码 默认值为True
   if (atts.Count >= 1) //判断是否有附件
   {
       //LinkAttachment.Text = att.Name;  //获取附件名
       try
       {
      //判断在存储附件的目录下是否已经存在该附件，如果存在不创建目录，否则创建目录
           if (!File.Exists(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text + "\\" + att.Name))
               {
                   //创建个目录用于存储附件
                   Directory.CreateDirectory(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text);
                   string mailPath = Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text + "\\" + att.Name;
                att.SaveToFile(mailPath); //把附件存储在指定目录下
               }
               //存储该附件的路径
               path = Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text + "\\" + att.Name;
       }
       catch (Exception ex)
       {
           throw new Exception(ex.Message);
       }
   }
   jme.Close();  //关闭邮件信息类
}
```

双击页面中的“连接”按钮，触发其 Button1_Click。在该事件中首先打开 POP3 服务器的连接，打开 POP3 连接需要用户输入正确的电子邮件地址、密码、服务器和端口号。连接成功后，将邮件的中的附件保存在本地磁盘中。实现的代码如下。

例程 18 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
  try
  {
      jpop = new jmail.POP3Class();
      //获取邮件地址、密码、服务器名、端口号
      int number = Convert.ToInt32(TextBoxPort.Text);
```

```
string name = TextBoxName.Text.Trim();
string pass = TextBoxPass.Text.Trim();
string server = TextBoxServer.Text.Trim();
this.Label4.Text = pass;
try
{   //打开POP3服务器的连接
    jpop.Connect(name, pass, server, number);
    ////显示共有几封邮件
    //LabelSum.Text = jpop.Count.ToString();
    if (jpop.Count > 0)
    {
        DataTable dt = new DataTable();
        dt.Columns.Add(new DataColumn("Subject", typeof(string)));
        dt.Columns.Add(new DataColumn("From", typeof(string)));
        dt.Columns.Add(new DataColumn("Body", typeof(string)));
        dt.Columns.Add(new DataColumn("Date", typeof(string)));
        dt.Columns.Add(new DataColumn("Name", typeof(string)));
      for (int i = 1; i <= jpop.Count; i++)
      {
            jmail.MessageClass jme = (jmail.MessageClass)jpop.DownloadSingleMessage(i);
            atts = jme.Attachments; //获取附件集合
                att = atts[0]; //获取第一个附件
                jme.Charset = "GB2312";   //设置使用的邮件字符集，默认US-ASCII 中国则为GB2312
jme.Encoding = "Base64"; //设置附件默认编码
jme.ISOEncodeHeaders = false;   //邮件头是否使用ISO-8859-1编码 默认值为True
DataRow dr = dt.NewRow();
dt.Rows.Add(dr);
dr[0] = jme.Subject; //获取主题
dr[1] = jme.From;      //获取获取发件人邮箱
dr[2] = jme.Body;   //获取内容
dr[3] = jme.Date.ToString();   //获取发送时间
if (atts.Count >= 1) //判断是否有附件
{
    dr[4] = att.Name;   //获取附件名
    try
    {
   //判断在存储附件的目录下是否已经存在该附件，如果存在不创建目录，否则创建目录
    if (!File.Exists(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text + "\\" + att.Name))
      {
       //创建个目录用于存储附件
       Directory.CreateDirectory(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text);
       string mailPath = Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text + "\\" + att.Name;
       att.SaveToFile(mailPath); //把附件存储在指定目录下
       }
        //存储该附件的路径
       path = Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text + "\\" + att.Name;
        }
       catch (Exception ex)
       {
            throw new Exception(ex.Message);
       }
     }
       this.GridView1.DataSource = dt;
       this.GridView1.DataBind();
       jme.Close();   //关闭邮件信息类
```

```
            }
              //lookEmail(1);//显示第1封邮件
            }
              this.Button2.Enabled = true;//让按钮可用
            this.Button1.Enabled = false;//让按钮可用
        }
        catch (Exception error)
          {
            Page.RegisterStartupScript("", "<script>alert('邮件服务器设置错误或未断开连接')</script>");
          }
        }
        catch
        {
        Page.RegisterStartupScript("", "<script>alert('很遗憾，内容填写错误！')</script>");
        }
    }
```

在上述事件中，当查看邮件结束时，需要断开邮件服务器的连接，并在页面中清空所显示邮件的信息和删除用来保存附件所创建的文件夹。但在删除所创建的文件夹前需要先判断一下是否创建了此文件夹。如果创建了此文件夹需要先把文件夹里的内容删除，再删除此文件夹。这些功能需要通过“断开”按钮的 Click 事件中来实现。实现的代码如下。

例程 19 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
protected void Button2_Click(object sender, EventArgs e)
{
    this.Button1.Enabled = true;
    this.Button2.Enabled = false;
    jpop.Disconnect();//关闭邮件服务器的连接
    //判断指定文件夹是否存在
    if (Directory.Exists(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text))
    {
        //获取当前文件夹内的所有文件
        string[] ps = Directory.GetFileSystemEntries(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text);
        //string[] ps = Directory.GetFileSystemEntries(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text);
      //循环删除文件
      if (ps.Length > 0)
      {
          for (int i = 0; i < ps.Length; i++)
          {
              File.Delete(ps[i]);
          }
          //删除指定文件夹
      }
      Directory.Delete(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text);
    }
    //清楚当前邮件的信息（邮件地址、密码、服务器名、端口号）
    TextBoxPort.Text = "";
    TextBoxName.Text = "";
    TextBoxPass.Text = "";
    TextBoxServer.Text = "";
}
```

双击页面中的“收件箱”按钮，触发其 ImgBtnReceive_Click 事件，在该事件中使用 Response 的 Redirect 方法实现将页面跳转到 receiveEmail.aspx 页面中，实现的代码如下。

例程 20 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
protected void ImgBtnReceive_Click(object sender, ImageClickEventArgs e)
{
```

```
        Response.Redirect("receiveEmail.aspx");//跳转到receiveEmail.aspx页面中
    }
```

双击页面中的“发件箱”按钮，触发其 ImgBtnSendOut_Click 事件，在该事件中使用 Response 的 Redirect 方法实现将页面跳转到 SendOutEmail.aspx 页面中，实现的代码如下。

例程 21 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
protected void ImgBtnSendOut_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("sendOutEmail.aspx");//跳转到sendOutEmail.aspx页面中
}
```

在 GridView1_RowCommand 事件中，单击 GridView 控件上的下载超级链接按钮实现将邮件的附件下载下来。实现的代码如下。

例程 22 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
protected void GridView1_RowCommand(object sender, GridViewCommandEventArgs e)
{
    if (e.CommandName == "down")
    {
        string fileName = e.CommandArgument.ToString();
        string filePath = Server.MapPath("AttachFiles\\" + TextBoxName.Text + "\\");
        downFile(filePath + fileName);
    }
}
```

在用户自定义的 downFile 方法中，该方法带一个字符串类型的参数，实现的是将找到附件的路径，实现的代码如下。

例程 23 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
    public void downFile(string path)
    {
        //初始化 FileInfo 类的实例，它作为文件路径的包装
        FileInfo fi = new FileInfo(path);
        Response.Clear();  //清除输出流内容
        //添加输出流头文件的内容
        Response.AddHeader("Content-Disposition", "attachment; filename=" + Server.UrlEncode(fi.Name));
        Response.AddHeader("Content-Length", fi.Length.ToString());
        //添加输出流内容
        Response.ContentType = "application/octet-stream";
        Response.Filter.Close();
        Response.WriteFile(fi.FullName);
        Response.End();    //将当前所有缓冲的输出发送到客户端
    }
```

在 GridView1_RowDeleting 事件中，实现的是删除 GridView 控件指定的邮件的功能，实现的代码如下。

例程 24 代码位置：光盘\mr\07\Email\ receiveEmail.aspx.cs

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    this.Button1.Enabled = true;
    this.Button2.Enabled = false;
    jpop.Disconnect();//关闭邮件服务器的连接
    //判断指定文件夹是否存在
    if (Directory.Exists(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text))
    {
        string[] ps = Directory.GetFileSystemEntries(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text);
        //循环删除文件
        if (ps.Length > 0)
```

```
        {
            for (int i = 0; i < ps.Length; i++)
            {
                File.Delete(ps[i]);
            }
            //删除指定文件夹
        }
        Directory.Delete(Server.MapPath("AttachFiles") + "\\" + TextBoxName.Text);
    }
    popMail = new jmail.POP3Class();
    //获取邮件地址、密码、服务器名、端口号
    int number = Convert.ToInt32(TextBoxPort.Text);
    string name = TextBoxName.Text.Trim();
    string pass = this.Label4.Text;
    string server = TextBoxServer.Text.Trim();
    popMail.Connect(name, pass, server, number);
   int index = Convert.ToInt32(e.RowIndex);
    popMail.DeleteSingleMessage(index + 1);
    popMail.Disconnect();
    RegisterStartupScript("", "<script>alert('成功删除！')</script>");
    GridView1.DataBind();
}
```

7.4 好友录管理

7.4.1 添加好友录

在 AddFriend.aspx 页面中完成添加好友录的操作，在该页面中用户可以添加常用的好友录，也可以添加好友的姓名和好友的电子邮箱地址，以便在发送邮件时，方便快捷地找到该好友的信息，添加好友录页运行结果如图 7.17 所示。

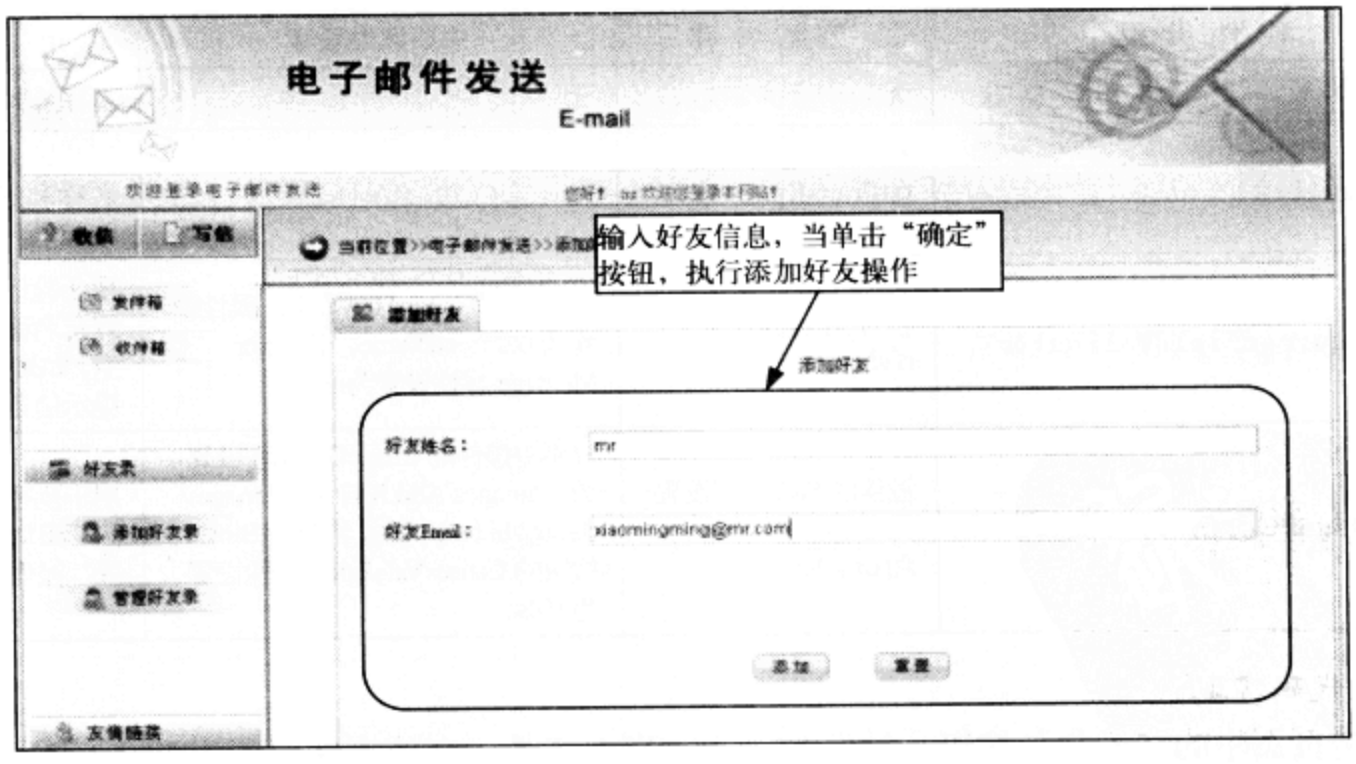


图 7.17 运行效果图

1. 页面设计

(1) 在应用程序中创建 1 个 Web 窗体，命名为 AddFriend.aspx，作为添加好友录页。页面 AddFriend.aspx 的设计界面如图 7.18 所示。

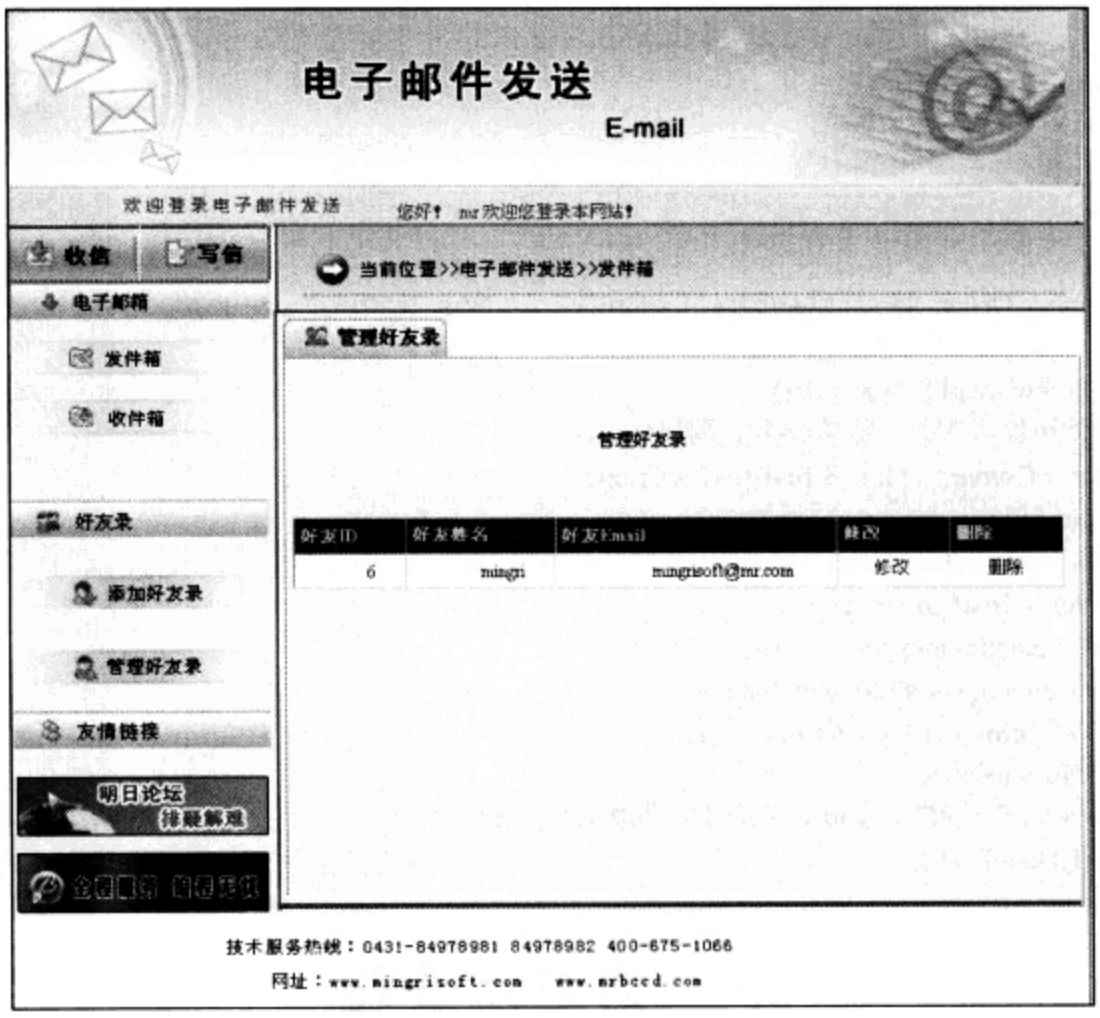


图 7.18 AddFriend.aspx 页面设计界面图

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 TextBox 控件、1 个 RequiredFieldValidator 控件和 2 个 ImageButton 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 7.6 所示。

表 7.6 添加好友录页控件列表

控件类型	控件名称	主要属性设置	用途
Table	无	无	页面布局
TextBox	控件的名称分别为 txtFriendName 和 txtFriEmail	无	用户输入好友的名称和好友的电子邮箱
RequiredFieldValidator	控件的名称为 rfvName	将控件的 ControlToValidate 属性设置为 txtFriendName，将 Error Message 属性设置为必添	用于验证输入的好友名称是否为空，如果为空显示提示信息
ImageButton	控件的名称分别设置为 ImgBtn Friendadd 和 ImgBtnCZ	分别将控件的 ImageUrl 属性设置为～/images/Add.JPG 和～/images/chong zhi.bmp，并且将“ImgBtnCZ”按钮的 CausesValidation 属性设置为 False	用于执行好友录的“添加”和“重置”操作

2. 代码实现

双击页面中的“添加”按钮，触发 btnAdd_Click 事件，在该事件中使用 Insert 语句并且向“tb_Friend”数据表中添加好友姓名和好友邮箱信息，实现的代码如下。

例程 25 代码位置：光盘\mr\07\Email\AddFriend.aspx.cs

```
/// <summary>
/// 添加好友
/// </summary>
```

```
    /// <param name="sender"></param>
    /// <param name="e"></param>
        protected void btnAdd_Click(object sender, EventArgs e)
    {
        if (txtFriendName.Text != "" && txtFriEmail.Text != "")
        {
            string sqlstr = "insert into tb_Friend (FriendName,FriendEmail)" + " values('" + txtFriendName.Text + "','" +
txtFriEmail.Text + "')";
            sqlBind.DataCom(sqlstr);
            Response.Write("<script lanuage=javascript>alert('好友添加成功！');location='javascript:history.go (-1)'</script>");
            }
            if(txtFriendName.Text==""&&txtFriEmail.Text!="")
                Response.Write("<script lanuage=javascript>alert('请输入有效的好友名字！');location='javascript:history.go(-1)'
</script>");
            if(txtFriEmail.Text==""&&txtFriendName.Text!="")
                Response.Write("<script lanuage=javascript>alert('请输入正确的邮箱地址！');location='javascript:history.go(-1)'
</script>");
            else
                Response.Write("<script lanuage=javascript>alert('请确定输入完整后再添加！');location='javascript:history.go(-1)'
</script>");
        }
```

双击页面中的“重置”按钮，触发 ImgBtnCZ_Click 事件，在该事件中实现清空文本框的功能，并且将焦点定位在 txtFriendName 文本框中，实现的代码如下。

例程 26 代码位置：光盘\mr\07\Email\AddFriend.aspx.cs

```
    protected void ImgBtnCZ_Click(object sender, ImageClickEventArgs e)
        {
            txtFriendName.Text = "";
            txtFriEmail.Text = "";
            txtFriendName.Focus();
        }
```

7.4.2 管理好友录

在管理好友录中实现的是管理好友的信息中，在该页面中当单击“删除”超级链接能够删除指定的好友信息操作。实现在“Friend.aspx”页面中，该页面实现管理好友信息，页面运行结果如图 7.19 所示。

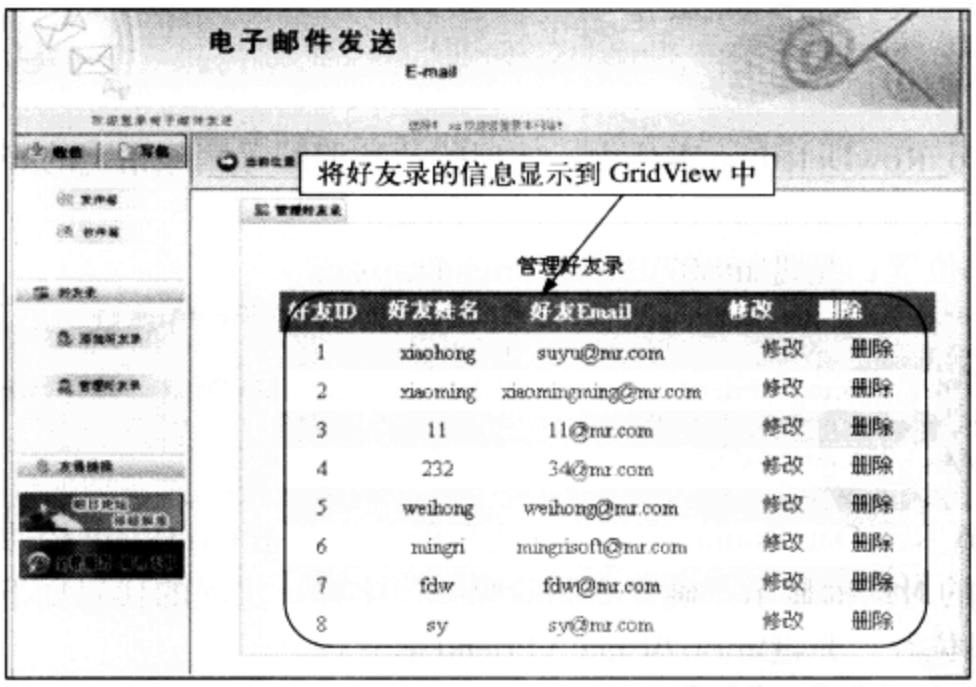


图 7.19 运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为“Friend.aspx”，作为管理好友录页。页面 Friend.aspx 的设计界面如图 7.20 所示。

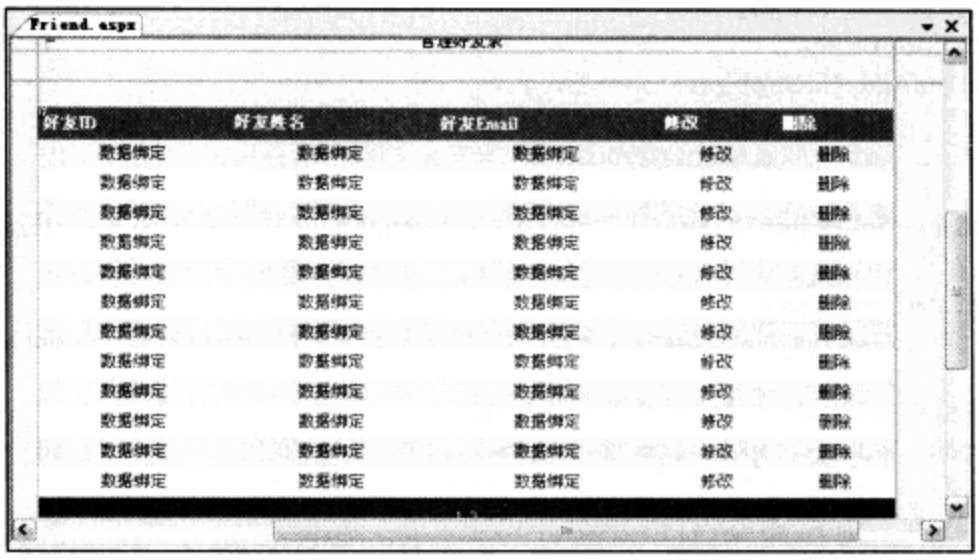


图 7.20　Friend.aspx 页面的设计界面

(2) 在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 GridView 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 7.7 所示。

表 7.7　管理好友录页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
GridView	将控件的名称设置为 gvFriendInfo	将该控件的 AllowPaging 属性设置为 True，将 PageSize 属性设置为 12	用于将好友的信息显示到该控件中

2. 代码实现

声明 1 个 SqlOperate 类对象 sqlBind，目的是调用该类里面的方法，实现的代码如下。

例程 27　代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
SqlOperate sqlBind = new SqlOperate();
```

在 Page_Load 事件中，实现的是将好友信息绑定到 GridView 控件当中，实现的代码如下。

例程 28　代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    tring sqlstr = "select * from tb_Friend";
    gvFriendInfo.DataKeyNames = new string[] { "FriendID" };
    sqlBind.gvBind(gvFriendInfo, sqlstr);
}
```

在 gvFriendInfo_RowDeleting 事件中，实现的是按照编号删除指定的好友的信息，实现的代码如下。

例程 29　代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void gvFriendInfo_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    string sqlstr = "delete from tb_Friend where FriendID='" + gvFriendInfo.DataKeys[e.RowIndex].Value + "'";
    sqlBind.DataCom(sqlstr);
    Response.Redirect("Friend.aspx");
}
```

在 gvFriendInfo_RowDataBound 事件中，实现的是当单击 GridView 控件中的删除按钮，将弹出一个提示信息的对话框显示“确定要删除吗？”字样，实现的代码如下。

例程 30　代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void gvFriendInfo_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        ((LinkButton)(e.Row.Cells[4].Controls[0])).Attributes.Add("onclick", "return confirm('确定删除吗？')");
    }
}
```

在 gvFriendInfo_PageIndexChanging 事件中，实现 GridView 控件的分页功能，实现的代码如下。

例程 31 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void gvFriendInfo_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    gvFriendInfo.PageIndex = e.NewPageIndex;
    gvFriendInfo.DataBind();
}
```

双击“发件箱”按钮，触发 ImageButton1_Click 事件，实现将页面跳转到发送邮件（sendOutEmail.aspx）页面中，实现的代码如下。

例程 32 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("sendOutEmail.aspx");
}
```

双击“收件箱”按钮，触发 ImageButton2_Click 事件，实现将页面跳转到邮件接收的（receiveEmail.aspx）页面中，实现的代码如下。

例程 33 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("receiveEmail.aspx");
}
```

双击“添加好友录”按钮，触发 ImgBtnAdd_Click 事件，实现将页面跳转到添加好友录（AddFriend.aspx）页面中，实现的代码如下。

例程 34 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void ImgBtnAdd_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("AddFriend.aspx");//跳转到AddFriend.aspx页面中
}
```

双击“管理好友录”按钮，触发 Imgguanli_Click 事件，实现将页面跳转到管理好友录（Friend.aspx）页面中，实现的代码如下。

例程 35 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void Imgguanli_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Friend.aspx ");//跳转到Friend.aspx页面中
}
```

7.4.3 好友信息修改

好友录信息修改页面实现在“EditFriInfo.aspx”页面中，页面运行结果如图 7.21 所示。

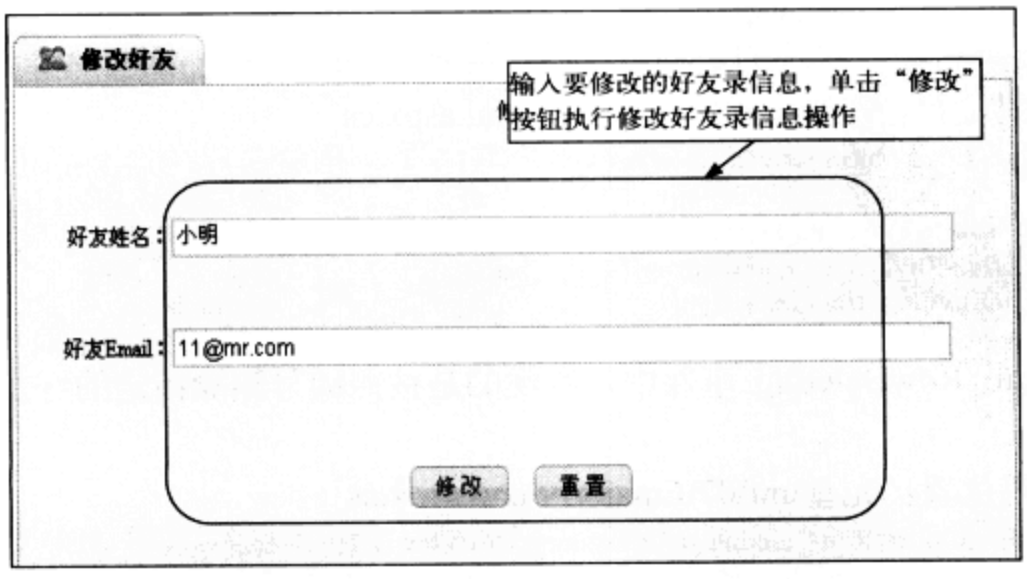


图 7.21 好友信息修改页运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，命名为“EditFriInfo.aspx”，作为管理好友录页。页面 EditFriInfo.aspx 的设计界面如图 7.22 所示。

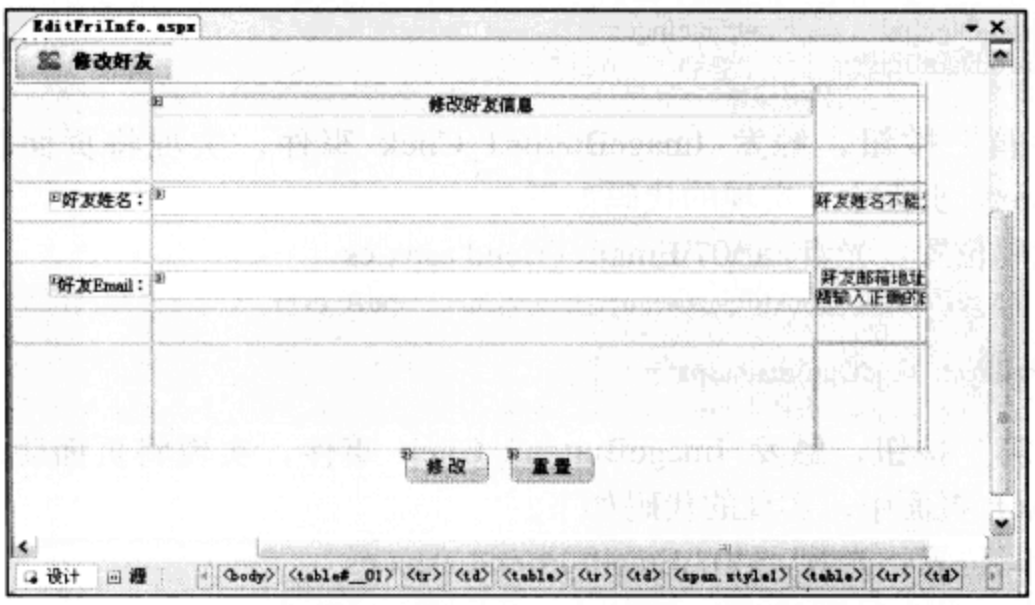


图 7.22　EditFriInfo.aspx 页面的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 Label 控件、2 个 TextBox 控件、2 个 ImageButton 和 1 个控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 7.8 所示。

表 7.8　　管理好友录页控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
Label	控件的名称分别为 LblName 和 LblEmail	将这两个控件的 Text 属性分别设置成“"好友姓名："”和“好友 Email：”	用于显示信息
TextBox	控件的名称分别为 txtFriendName 和 txtFriEmail	无	用于显示用户输入修改好友录的信息
ImageButton	控件的名称分别为 ImgBtnUpdate 和 ImgBtnCZ	将这 2 个控件的 ImageUrl 属性分别设置为～/images/update1.jpg 和～/images/chongzhi.bmp	分别用于执行“修改好友录”操作和“清空文本框操作”

2. 代码实现

声明 1 个 SqlOperate 类对象 sqlBind，目的是调用该类里面的方法，实现的代码如下。

例程 36　代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
SqlOperate sqlBind = new SqlOperate();
```

在 Page_Load 事件中，实现的是将好友信息绑定到 GridView 控件当中，实现的代码如下。

例程 37　代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    tring sqlstr = "select * from tb_Friend";
    gvFriendInfo.DataKeyNames = new string[] { "FriendID" };
    sqlBind.gvBind(gvFriendInfo, sqlstr);
}
```

在 gvFriendInfo_RowDeleting 事件中，实现的是按照编号删除指定的好友的信息，实现的代码如下。

例程 38　代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void gvFriendInfo_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    //定义一个删除的SQL语句
```

```
        string sqlstr = "delete from tb_Friend where FriendID='" + gvFriendInfo.DataKeys[e.RowIndex].Value + "'";
        //调用公共类中的DataCom执行删除SQL语句
        sqlBind.DataCom(sqlstr);
        //跳转到好友信息页
        Response.Redirect("Friend.aspx");
    }
```

在 gvFriendInfo_RowDataBound 事件中，实现的是当单击 GridView 控件中的“删除”按钮，将弹出一个提示信息的对话框显示“确定要删除吗？”字样，实现的代码如下。

例程 39 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void gvFriendInfo_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
      //弹出确认删除对话框
      ((LinkButton)(e.Row.Cells[4].Controls[0])).Attributes.Add("onclick", "return confirm('确定删除吗？')");
    }
}
```

在 gvFriendInfo_PageIndexChanging 事件中，实现 GridView 控件的分页功能，实现的代码如下。

例程 40 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void gvFriendInfo_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    gvFriendInfo.PageIndex = e.NewPageIndex;
    //从数据库中绑定数据
    gvFriendInfo.DataBind();
}
```

双击“发件箱”按钮，触发 ImageButton1_Click 事件，实现将页面跳转到发送邮件（sendOutEmail.aspx）页面中，实现的代码如下。

例程 41 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("sendOutEmail.aspx");
}
```

双击“收件箱”按钮，触发 ImageButton2_Click 事件，实现将页面跳转到邮件接收的（receiveEmail.aspx）页面中，实现的代码如下。

例程 42 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("receiveEmail.aspx");
}
```

双击“添加好友录”按钮，触发 ImgBtnAdd_Click 事件，实现将页面跳转到添加好友录（AddFriend.aspx）页面中，实现的代码如下。

例程 43 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void ImgBtnAdd_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("AddFriend.aspx");//跳转到AddFriend.aspx页面中
}
```

双击“管理好友录”按钮，触发 Imgguanli_Click 事件，实现将页面跳转到管理好友录（Friend.aspx）页面中，实现的代码如下。

例程 44 代码位置：光盘\mr\07\Email\ Friend.aspx.cs

```
protected void Imgguanli_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("Friend.aspx ");//跳转到Friend.aspx页面中
}
```

7.5 网站的打包与发布

网站开发完成后最终的目地是将其发布到 Internet 上，以提供用户浏览访问。实现的网站发布可以使用两种方法。第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍下使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

(1) 首先打开要发布的网站，在网站的项目名称上，单击鼠标右键，在弹出的快捷方式菜单中选择“发布网站”选项，如图 7.23 所示。

(2) 在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮，网站将被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择“...”路径按钮，如图 7.24 所示。

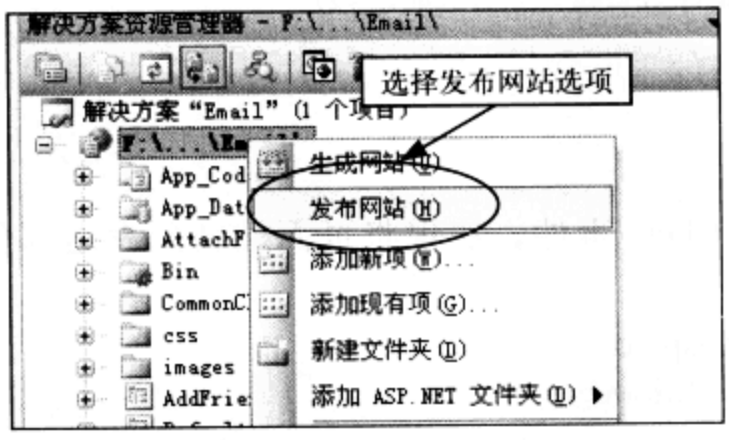


图 7.23 选择发布网站

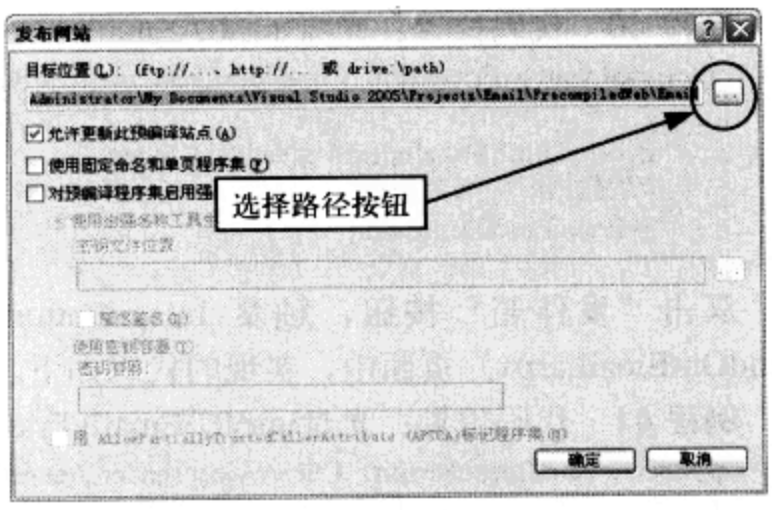


图 7.24 选择路径按钮

(3) 在弹出的窗口中，选择“FTP 站点”选项，在该选择中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码如图 7.25 所示。填写完毕后选择“打开”按钮将会返回“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

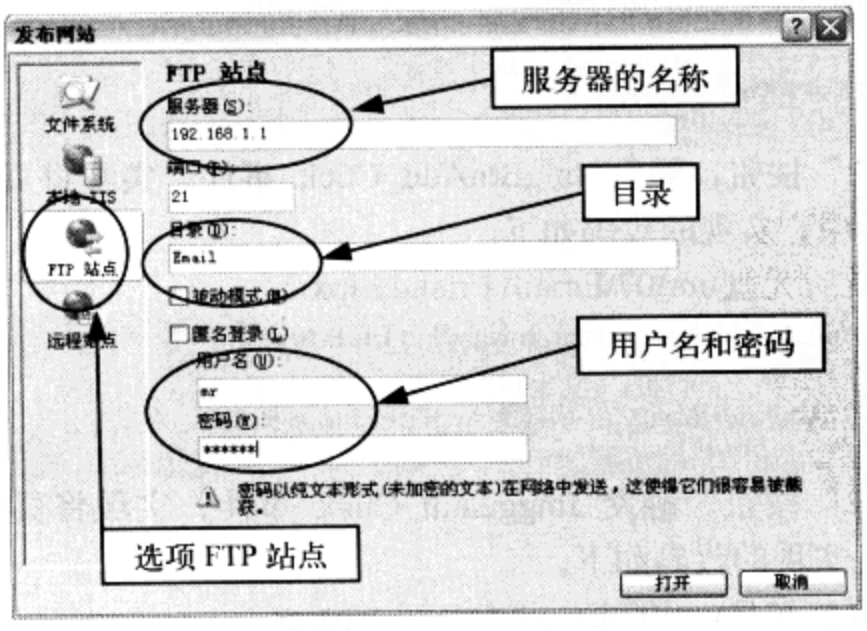


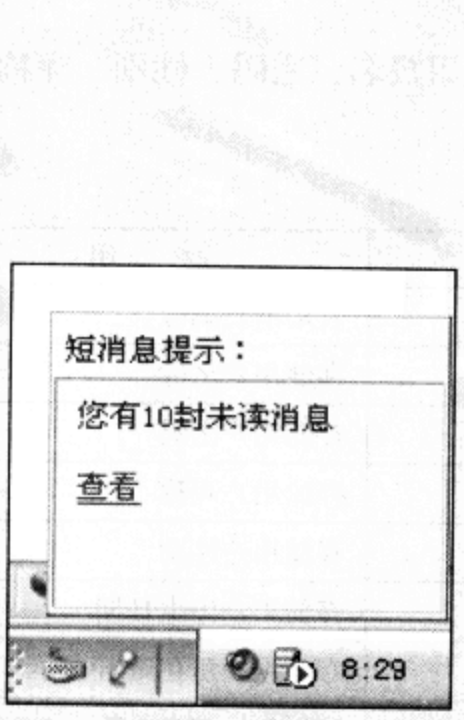
图 7.25 FTP 站点选项

第8章 在线短消息模块

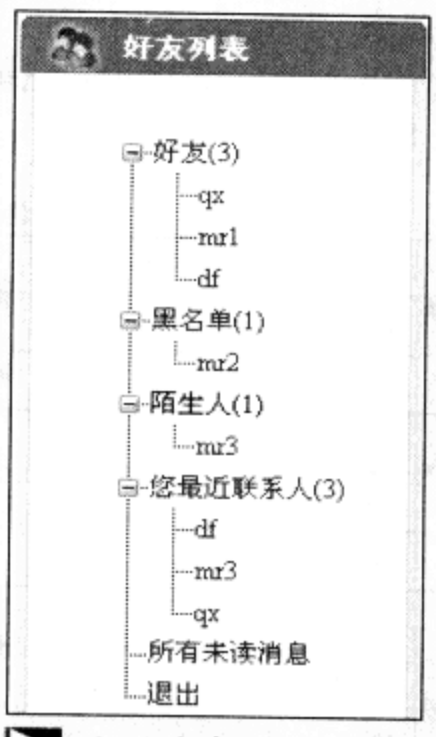
实例位置：光盘\mr\08\

现在网络论坛的种类很多，很多的网友对论坛也是比较热衷的。网友们可以在论坛中娱乐、学习。但是论坛这种形式有一点缺点，用户之间只能通过发帖和回帖进行交流。而不能在线聊天。为了方便网友之间的交流，现在很多论坛都添加了在线短消息模块。用户之间可以根据在线短消息来进行之间的沟通或交流。本章将会介绍如何去开发一个在线短消息模块。通过本章，读者可以学到以下内容。

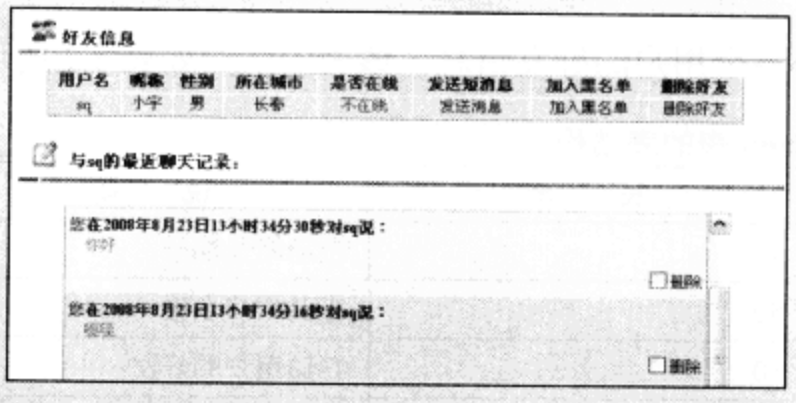
消息提示框



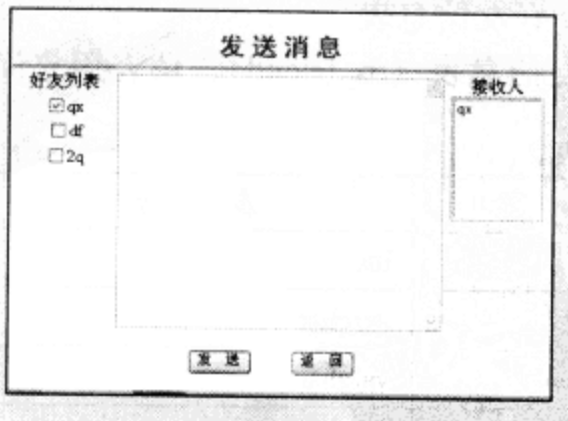
动态树状菜单栏



好友信息及聊天记录查看



发送消息



8.1 在线短消息概述

8.1.1 功能概述

在线短消息模块是在论坛中为了方便网友之间的交流而产生的。使用在线短消息进行交流，必须将对方加为好友或对方加您为好友才能进行。在线短消息有 3 种分组，即好友组、黑名单组和陌生人组。好友组是您加对方为好友的用户，您可以查看到对方的信息和发送消息，陌生人组是对方加您为好友并给您发送了消息，您不可以看到陌生人的信息但可以发送消息，黑名单组是您在好友组中添加过来的，在黑名单组中的用户您不可以给对方发送消息，对方发送的消息您也接收不到。在短消息模块中您还可以在“您最近联系人”中查看到最近几次的联系人和聊天记录。

8.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建一个名为 db_message 的数据库，在该数据库中创建 3 张表。下面将详细介绍这 3 张表的数据结构及用途。

- 用户信息表

用户信息表（tb_user），该表用来记录用户的详细信息包括用户名、密码、性别、昵称等。该表的结构如表 8.1 所示。

表 8.1　　tb_user 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长编号
name	varchar	20	存储用户名称
pass	varchar	20	存储用户密码
nickname	varchar	50	存储用户昵称
sex	char	10	存储用户性别
birthday	datetime	8	存储用户出生日期
city	varchar	50	存储用户所在城市
msgCount	int	4	存储未读消息总数

- 好友信息表

好友信息表（tb_friend），该表用来保存每个用户的好友信息。该表的结构如表 8.2 所示。

表 8.2　　tb_friend 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长编号
selfName	varchar	50	存储用户名称
friendName	varchar	50	存储用户的好友
isBlack	bit	1	存储是否为黑名单

- 短消息信息表

短消息信息表（tb_msg），该表用来保存短消息信息包括发送人名称、接收人名称、发送消息、发送时间等。该表的结构如表 8.3 所示。

表 8.3　　tb_msg 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长编号
nameAnnal	varchar	50	存储用户名称
sender	varchar	50	存储发送消息人名称
accepter	varchar	50	存储接收消息人名称
msg	text	16	存储消息内容
msgDate	datetime	8	存储发送消息时间
isRead	bit	1	存储是否阅读消息

8.2　在线短消息关键技术

8.2.1　防止用户的重复登录（单点登录）

当某个用户登录后，如果其他人使用该用户的登录名和登录密码在其他的计算机上也进行了登录。这样就会查看到两人的聊天记录及一些隐私问题。所以禁止用户的重复登录是非常有必要的。在本程序中如果用户使用已登录的账号进行登录将会给出提示，如图 8.1 所示。

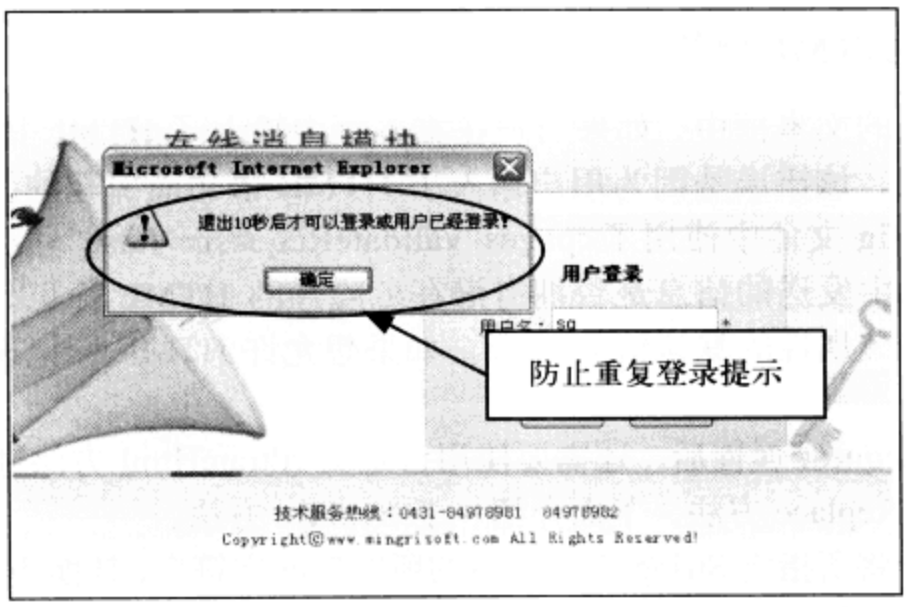


图 8.1　防止重复登录提示

禁止用户的重复登录是使用 Application 对象来实现的。Application 对象是 HttpApplicationState 类的一个实例。在第一个用户请求 ASP.NET 文件时，将启动应用程序并创建 Application 对象。一旦 Application 对象被创建，它就可以共享和管理整个应用程序的信息。在应用程序关闭之前，Application 对象将一直存在。在用户登录成功后将用户的登录名和当前的时间使用 Add 方法添加到 Application 对象中。在程序中的主页面中每隔 5s 将修改一次 Application 对象中的值将该值改为当前的时间。判断用户是否登录和用户是否在线将使用当前的时间减去 Application 对象中的时间来计算差值是否大于 10s。如果大于 10s 说明用户已经不在线或退出。如果小于 10s 说明用户还在程序中并没有退出或掉线。

8.2.2　设计动态树状菜单栏

由于不同的用户会有不同的好友信息，所以每个用户的树状菜单栏是不同的。这就需要动态显示每个人的树状菜单栏。树状菜单栏使用的是 xml 文件来绑定的。在 xml 文件中用来保存

用户的好友信息或陌生人信息等。每一个用户都会有一个以用户名命名的 xml 文件，而每一个 xml 文件都会在一个 xml 文件模板的基础上进行修改成为自己的 xml 文件。在 xml 文件模板中存储着树状菜单的大体形状。xml 文件模板的代码如下。

例程 1 代码位置：光盘\mr\08\Message\xml\ XMLFile.xml

```
<?xml version="1.0" encoding="utf-8"?>
<menuList>
  <friendList value="好友">
  </friendList>
  <blackList value="黑名单">
  </blackList>
  <strangeness value="陌生人">
  </strangeness>
  <linkmanList value="您最近联系人" >
  </linkmanList>
  <notMessage value="所有未读消息" url="notReaderMessage.aspx">
  </notMessage>
  <exitMess value="退出" url="Default.aspx">
  </exitMess>
</menuList>
```

当用户登录时，将会使用文件对象中的 Delete 方法来删除上一次以自己用户命名的 xml 文件，在使用文件对象中的 CopyTo 方法，将 xml 文件模板复制出一个副本并以自己的用户名命名。再将 SQL 语句查询出的好友信息或陌生人信息等添加到 xml 文件中。这样就实现了每一个用户拥有不同的 xml 文件，最后将 xml 文件绑定到树状菜单中就会在页面显示出用户的树状菜单。

8.2.3 过滤和还原 HTML 字符

在输入发送消息的文本框中，如果用户在文本框中输入了 HTML 语句后，单击“发送”按钮时就会提示错误，该错误是因为用户输入了 HTML 语句而引起的。为了避免这种错误的发生，在 Web.config 文件中使用了<pages validateRequest="false">配置。ValidateRequest 属性是用来检查表单中发送的信息是否拥有潜在危险性的 HTML 标记。如果检查到将会引发异常中止该请求，该属性的默认值为 True。如果想允许 HTML 标记通过，将该属性设置为 False。

设置完 ValidateRequest 属性后，还需要使用自定义 filtrateHtml 方法过滤 HTML 字符。在该方法中主要使用了 Replace 方法，下面介绍一下 Replace 方法。

Replace 方法用于将所指定 String 字符串中的所有匹配字符串，替换为其他指定的 String 字符串。该方法的语法如下：

```
public string Replace (
    string oldValue,
    string newValue
)
```

参数说明如下。

- oldValue：要替换的字符串。
- NewValue：要替换 oldValue 的所有匹配项的字符串。
- 返回值：oldValue 字符串已被替换为 newValue 的字符串。

使用 Replace 方法将字符串替换为转义字符后保存到数据库中。实现代码如下。

例程 2 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
public static string filtrateHtml(string str)
{
    str = str.Trim();
    str = str.Replace("'", "&quot;");
    str = str.Replace("<", "&lt;");
    str = str.Replace(">", "&gt;");
```

```
        str = str.Replace(" ", " ");
        str = str.Replace("\n", "<br>");
        return str;
    }
```

当显示用户所发送的消息时，还需要使用自定义 resumeHtml 方法将 HTML 字符还原。使 HTML 字符显示在文本框中。调用该方法需要传入一个字符串变量，该变量表示需要还原的字符。该方法将返回还原后的字符串。实现代码如下。

例程 3 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
public static string resumeHtml(string str)
    {
        str = str.Trim();
        str = str.Replace("&quot;", """);
        str = str.Replace("&lt;", "<");
        str = str.Replace("&gt;", ">");
        str = str.Replace(" ", " ");
        str = str.Replace("<br>", "\n");
        return str;
    }
```

8.2.4 未读消息提示

在本程序中登录用户的好友如果发送了消息，在页面的右下角弹出一个提示框。该提示框是以动态滚动形式弹出的。提示框的运行效果如图 8.2 所示。

图 8.2 未读消息提示框

该提示框是在 JavaScript 中利用 window 对象中的 createPopup 对象来实现的。CreatePopup 对象是一种特殊的顶层窗口，该窗口是出现在应用程序窗口之外的。在该对象中有两个常用的方法，这两个方法的详细说明如下。

- show 方法

该方法用来弹出一个窗口。调用该方法需要传入 4 个参数。4 个参数说明如下。

Left：表示显示区域的左坐标，默认值为 0。

Top：表示显示区域的上坐标，默认值为 0。

Width：表示弹出窗口的宽。

Height：表示弹出窗口的高。

- hide 方法

该方法用来关闭弹出的窗口，该方法没有参数。

在 JavaScript 中首先在构造函数中初始化相应的变量，如窗口大小、提示框内容、屏幕大小等属性。构造函数中的代码如下。

例程 4 代码位置：光盘\mr\08\Message\ blebHint.js

```
function PopBubble(caption,content,see){

    this.content    = content;              // 提示内容
    this.caption= caption;                  //提示标题
    this.see= see;                          //链接文字
    this.width= 150;                        //设置弹出窗口的宽度
    this.height = 120;                      //设置弹出窗口的高度
    this.timeout= 150;                      //设置窗口的停留时间
    this.speed      = 15;                   //设置窗口的弹出速度
    this.step       = 2;                    //设置窗口的弹出步幅
    this.right      = screen.width -1;      //设置屏幕的宽度
    this.bottom = screen.height;            //设置屏幕的高度
    this.left = this.right - this.width;    //设置窗口的左边到屏幕左边的距离
    this.top= this.bottom - this.height;    //设置窗口上边到屏幕上边的距离
    this.timer      = 0;
```

```
        //用来记录窗口是否退缩
        this.pause      = false;
        this.close      = false;
        this.autoHide       = true;
    }
```

在 JavaScript 中的 show 方法中用来实现提示框的弹出效果，在该方法中首先创建 createPopup 对象，再将要显示的提示信息编写在 DIV 中，并将 DIV 添加到窗口对象中，最后通过循环将窗口一步一步的显示出来。实现代码如下。

例程 5 代码位置：光盘\mr\08\Message\ blebHint.js

```
PopBubble.prototype.show = function(){
        //创建一个顶层窗口对象，该对象只有在IE5.5+以上的浏览器才能使用
        var oPopup = window.createPopup();
        this.Pop = oPopup;
        var w = this.width;
        var h = this.height;
        var str = "<DIV style='BORDER-RIGHT: #455690 1px solid; BORDER-TOP: #a6b4cf 1px solid; Z-INDEX:
99999; LEFT: 0px; BORDER-LEFT: #a6b4cf 1px solid; WIDTH: " + w + "px; BORDER-BOTTOM: #455690 1px solid;
POSITION: absolute; TOP: 0px; HEIGHT: " + h + "px; BACKGROUND-COLOR: #DAED9B'>"
                str += "<TABLE style='BORDER-TOP: #ffffff 1px solid; BORDER-LEFT: #ffffff 1px solid' cellSpacing=0
cellPadding=0 width='100%' bgColor=#EEF7CC border=0>"
                str += "<TR>"
                str += "<TD style='FONT-SIZE: 12px;COLOR: #0f2c8c' width=30 height=24></TD>"
                  str += "<TD style='PADDING-LEFT: 4px; FONT-WEIGHT: normal; FONT-SIZE: 12px; COLOR: #1f336b;
PADDING- TOP: 4px' vAlign=center width='100%'>" + this.caption + "</TD>"
                str += "</TR>"
                str += "<TR>"
                str += "<TD style='PADDING-RIGHT: 1px;PADDING-BOTTOM: 1px' colSpan=2 height=" + (h-28) + ">"
                  str += "<DIV style='BORDER-RIGHT: #b9c9ef 1px solid; PADDING-RIGHT: 8px; BORDER-TOP:
#7AA14E 1px solid; PADDING-LEFT: 8px; FONT-SIZE: 12px; PADDING-BOTTOM: 8px; BORDER-LEFT: #7AA14E 1px
solid; WIDTH: 100%; COLOR: #1f336b; PADDING-TOP: 8px; BORDER-BOTTOM: #b9c9ef 1px solid; HEIGHT: 100%'>" +
this.content + "<BR><BR>"
                str += "<DIV style='WORD-BREAK: break-all' align=left><A href='javascript:void(0)' hidefocus=true
id='btCommand'><FONT color=#ff0000>" + this.see+"<embed id='soundControl' src='Windows.wav' mastersound
hidden='true' loop='false' autostart='true'></embed>" + "</FONT></A></DIV>"
                str += "</DIV>"
                str += "</TD>"
                str += "</TR>"
                str += "</TABLE>"
                str += "</DIV>"
            //将设置好的div 添加到顶层窗口的body中
            oPopup.document.body.innerHTML = str;
            this.offset    = 0;
            var obj    = this;
            //添加顶层窗口的鼠标悬停事件，在事件中设置窗口不退缩
            oPopup.document.body.onmouseover = function(){obj.pause=true;}
            //添加顶层窗口的鼠标移除事件，在事件中设置窗口退缩
            oPopup.document.body.onmouseout = function(){obj.pause=false;}
            var fun = function(){
                var x    = obj.left; //获取窗口的左边到屏幕左边的距离
                var y    = 0;
                var width       = obj.width;    //获取窗口的宽度
                var height       = obj.height;  //获取窗口的高度
                    if(obj.offset>obj.height){
                        height = obj.height; //获取窗口的高度
                    } else {
                        height = obj.offset;        //设置窗口的弹出高度
                    }
                y    = obj.bottom - obj.offset; //设置屏幕的高度
                //判断窗口是否弹出到最顶端
                if(y<=obj.top){
                        //递减timeout停留时间
```

```
                obj.timeout--;
                  //判断timeout是否等于0
                if(obj.timeout==0){
                    //
                  window.clearInterval(obj.timer);
                  //判断是否关闭
                  if(obj.autoHide){
                      //调用函数实现窗口的收缩
                      obj.hide();
                  }
              }
          }
           else {
              //设置窗口弹出的步幅
              obj.offset = obj.offset + obj.step;
          }
          //调用窗口对象中的show对象弹出窗口
          obj.Pop.show(x,y,width,height);
      }
      //间隔speed时间调用fun函数弹出窗口
      this.timer = window.setInterval(fun,this.speed)
      //获取超链接对象
      var btCommand = oPopup.document.getElementById("btCommand");
      //设置超链接的onclick事件调用oncommand函数
      btCommand.onclick = function(){
          obj.oncommand();
      }
  }
```

在 JavaScript 中的 hide 方法中用来实现提示框的退缩效果，在该方法中首先将获取弹出窗口的高度。并通过循环将高度递减来实现弹出窗口的退缩效果。实现代码如下。

例程 6 代码位置：光盘\mr\08\Message\ blebHint.js

```
PopBubble.prototype.hide = function(){
        //设置弹出窗口的高度
      var offset   = this.height>this.bottom-this.top?this.height:this.bottom-this.top;
      var obj   = this;
      if(this.timer>0){
          window.clearInterval(obj.timer);
      }
      var fun = function(){
          //判断是否退缩弹出来的窗口
          if(obj.pause= =falsellobj.close){
              //获取窗口的左边到屏幕左边的距离
              var x   = obj.left;
              //
              var y   = 0;
              //获取弹出窗口的宽度
              var width = obj.width;
              //设置弹出窗口的高度
              var height = 0;
              if(obj.offset>0){
              //获取窗口当前的高度
                  height = obj.offset;
              }
              //设置屏幕的高度
              y   = obj.bottom - height;
              if(y>=obj.bottom){
                  window.clearInterval(obj.timer);
                  obj.Pop.hide();
              } else {
                  //设置退缩的步幅
```

```
                obj.offset = obj.offset - obj.step;
            }
            //调用窗口对象弹出窗口
            obj.Pop.show(x,y,width,height);
        }
    }
    this.timer = window.setInterval(fun,this.speed)
}
```

调用提示框是在后台代码中实现的。在后台代码中需要先判断登录用户是否有未读消息。判断是否有未读消息，首先将获取用户信息表中使用 msgCount 字段来存储所有未读消息的总数。再使用 SQL 语句查询出短消息信息表中，所有未读消息的总数。如果用户信息表中存储的未读消息总数小于短消息信息表中的未读消息的总数，说明登录用户有未读的消息，并调用 JavaScript 中的函数来显示提示框。

8.3 公共类的封装与设计

设计公共类，可以提高开发效率，还可以方便以后对程序的维护。对于一个好的程序来说，公共类是不可缺少的部分。在本程序中编写了一个公共类 dataOperate，该类主要用来实现对数据库的操作。例如，执行 SQL 语句，返回多条记录等操作。下面将详细介绍公共类中的方法。

8.3.1 实现判断数据是否存在

自定义 isName 方法用来判断所指定的数据是否存在。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个布尔值，所查询的数据如果存在将返回布尔值 True，否则返回 False。实现代码如下。

例程 7 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
public static bool isName(string sql)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sql, con);
    //判断数据是否存在并返回相应的布尔值
    if (Convert.ToInt32(com.ExecuteScalar()) > 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

8.3.2 实现用户登录操作

自定义 enter 方法用于实现用户登录。调用该方法需要传入 3 个字符串变量，第 1 变量为需要执行的 SQL 语句，第 2 变量为用户的登录名，第 3 个变量为用户的登录密码。该方法将返回一个布尔值变量，如登录成功将返回一个布尔值 True，否则将返回 False。实现代码如下。

例程 8 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
public static bool enter(string sql, string name, string pass)
{
    //创建数据库连接
    SqlConnection con = createCon();
```

```
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sql, con);
        //设置参数的类型
        com.Parameters.Add(new SqlParameter("@name", SqlDbType.VarChar, 20));
        //设置参数值
        com.Parameters["@name"].Value = name;
        com.Parameters.Add(new SqlParameter("@pass", SqlDbType.VarChar, 20));
        com.Parameters["@pass"].Value = pass;
        //判断是否执行成功
        if (Convert.ToInt32(com.ExecuteScalar()) > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

8.3.3 实现更新、插入、删除操作

自定义 execSQL 方法用来实现对数据库的更新、插入和删除操作。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个布尔值变量，当执行成功时将返回 True，否则返回 False。实现代码如下。

例程 9 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
    public static bool execSql(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sql, con);
        //判断SQL语句是否执行成功
        if (com.ExecuteNonQuery() > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

8.3.4 实现查询数据并返回 DataSet

自定义 getRows 方法用来实现查询数据并返回 DataSet 对象。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回 1 个 DataSet 对象，该对象中存储所有查询的数据。实现代码如下。

例程 10 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
    public static DataSet getRows(string sql)
    {
```

```
        //创建DataSet对象
        DataSet ds;
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlDataAdapter对象
        SqlDataAdapter sda = new SqlDataAdapter(sql, con);
        //实例DataSet对象
        ds = new DataSet();
        //填充DataSet对象
        sda.Fill(ds);
        //关闭数据库连接
        con.Close();
        return ds;
    }
```

8.3.5 实现查询数据并返回 SqlDataReader

自定义 getRow 方法用来实现查询数据并返回 SqlDataReader 对象。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回 1 个 SqlDataReader 对象。实现代码如下。

例程 11 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
    public static SqlDataReader getRow(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sql, con);
        //获取ExecuteReader返回的SqlDataReader对象
        SqlDataReader sdr = com.ExecuteReader();
        return sdr;
    }
```

8.3.6 实现返回统计数据的结果

自定义 countData 方法用来返回在 SQL 语句中使用 count 统计数据的结果。调用该方法需要传入一个字符串变量，该字符串变量表示需要执行的 SQL 语句。该方法将返回一个整型变量，该变量是使用 SQL 语句统计后的结果。实现代码如下。

例程 12 代码位置：光盘\mr\08\Message\ dataOperate.cs

```
    public static int countData(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sql, con);
        //返回查询的结果
        return Convert.ToInt32(com.ExecuteScalar());
    }
```

8.4 在线短消息实现过程

8.4.1 用户登录设计

用户登录页面用来实现用户登录功能。在该页面中用户通过输入用户名和密码后单击“登录”按钮来实现用户的登录功能。用户登录页面如图 8.3 所示。

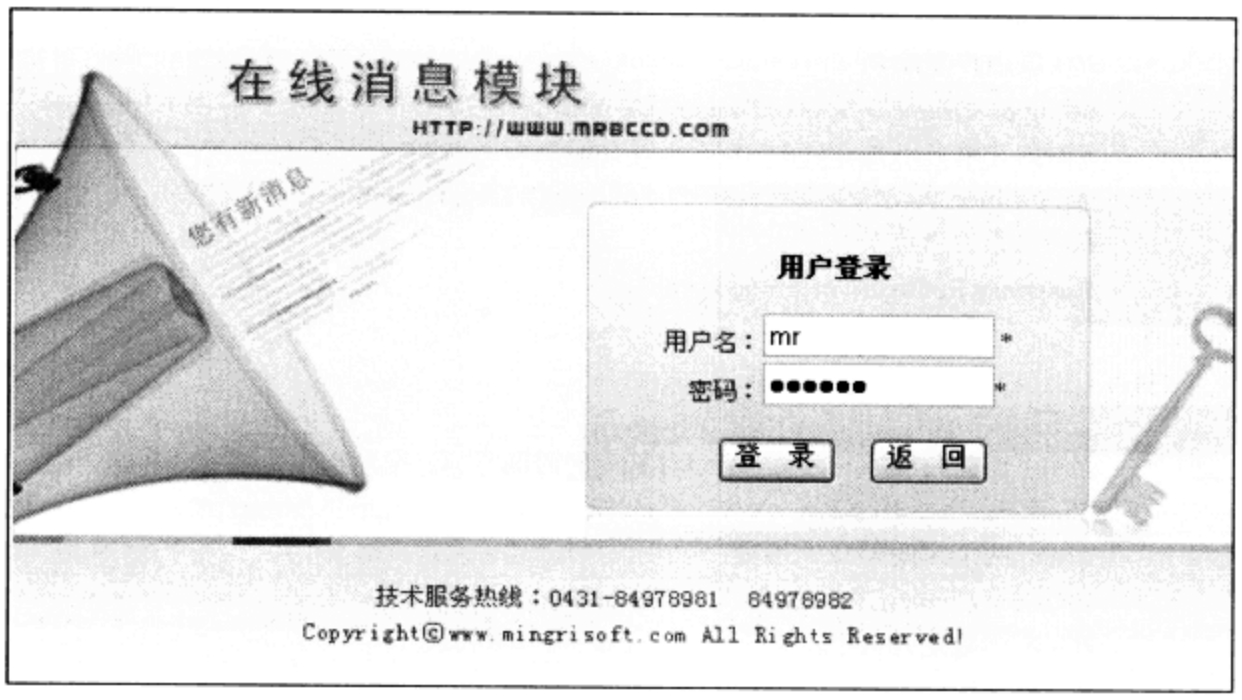


图 8.3 用户登录页面

1. 前台页面设计

(1) 创建 1 个 Web 用户控件页面，命名为 index.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 8.4 所示。

表 8.4 控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说 明
标准/TextBox 控件	txtName	均为默认值	输入用户登录名
	txtPass	TextMode 属性设置为 Password	输入用户登录密码
标准/Button 控件	btnEnter	均为默认值	实现用户登录操作

2. 后台代码编写

单击“登录”按钮可实现用户登录。在该事件中通过获取用户输入的登录名和密码调用公共类中的登录方法来判断用户是否登录成功。还需要判断登录的用户名是否已经登录，来防止用户重复登录发送的错误。实现代码如下。

例程 13 代码位置：光盘\mr\08\Message\ index.aspx.cs

```
//登录按钮的单击事件
    protected void btnEnter_Click(object sender, EventArgs e)
    {
        //创建SQL语句，该语句用来查询用户输入的登录名和密码是否正确
        string sql = "select count(*) from tb_user where name=@name and pass=@pass";
        //调用公共类中的enter方法实现用户登录操作
        if (dataOperate.enter(sql, txtName.Text, txtPass.Text))
```

```
{
    //使用Session保存用户的登录名
    Session["name"] = txtName.Text;
    //判断用户是否已经登录，当application为空时表示未登录
    if (Application[txtName.Text] == null)
    {
        //创建SQL语句，该语句用来设置消息总数
        string sqlUp = "update tb_user set msgCount=0 where name='" + txtName.Text + "'";
        //执行SQL语句操作
        dataOperate.execSql(sqlUp);
        //获取用户登录名
        string userName = Session["name"].ToString();
        //根据登录名创建application对象并存储当前的时间
        Application.Add(Session["name"].ToString(), DateTime.Now);
        //跳转的消息首页
        Response.Redirect("msgIndex.aspx");
    }
    else
    {
        //使用当前时间减去application对象中的时间
        TimeSpan ts = DateTime.Now - Convert.ToDateTime(Application[txtName.Text]);
        //获取两个时间相减的秒数
        int sec = Convert.ToInt32(ts.TotalSeconds);
        //判断秒数是否大于，大于说用户已经离开或掉线
        if (sec > 10)
        {
            //创建SQL语句，该语句用来设置消息总数为
            string sqlUp = "update tb_user set msgCount=0 where name='" + txtName.Text + "'";
            //执行SQL语句
            dataOperate.execSql(sqlUp);
            //获取用户登录名
            string userName = Session["name"].ToString();
            //创建application对象并保存当前时间
            Application.Add(Session["name"].ToString(), DateTime.Now);
            Response.Redirect("msgIndex.aspx");
        }
        else
        {
            RegisterStartupScript("", "<script>alert('退出秒后才可以登录或用户已经登录！')</script>");
        }
    }
}
else
{
    RegisterStartupScript("", "<script>alert('登录失败！')</script>");
}
}
```


8.4.2 在线短消息首页设计

在线短消息首页用来显示所有注册用户的信息，在该页面中用户可以选择某一个用户添加为自己的好友。在线短消息首页如图 8.4 所示。

图 8.4　在线短消息首页

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 msgIndex.aspx。

(2) 在该窗体中添加 1 个表格该表格用于布局，在表格的右侧使用了框架，在该框架中用来显示左侧导航栏所选择的页面。在该窗体中还需要添加控件，所添加的控件类型、控件名称及说明如表 8.5 所示。

表 8.5　　控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说　明
导航/TreeView 控件	TreeView1	设置 DataSourceID 属性为 XmlDataSource1 设置 ShowLines 属性为 True	显示导航栏信息
数据/XmlDataSource 控件	XmlDataSource1	设置 DataFile 属性为～/XMLFile.xml 设置 Xpath 属性为/*/*	设置 XML 文件的数据源
AJAX/ Timer 控件	Timer1	Interval 属性设置为 5000	实现保存用户用户状态

2. 后台代码编写

在页面加载事件中首先调用自定义 setXmlChild 方法修改 xml，再调用自定义 bindTree 方法绑定 TreeView 控件显示菜单栏。最后判断用户是否有未阅读的消息，如果有将使用 JavaScript 提示用户。实现代码如下。

例程 14　代码位置：光盘\mr\08\Message\ msgIndex.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
```

```
                    //调用自定义setXmlChild方法修改xml文件
                    setXmlChild();
                    //调用自定义方法bindTree方法绑定TreeView控件
                    bindTree();
                    //创建SQL语句，查询短消息信息表中所有未读消息的总数
                    string sqlRead = "select count(*) from tb_msg where sender not in(select friendName from tb_friend
where isBlack=1 and selfName='" + Session["name"] + "' ) and accepter='" + Session["name"] + "' and nameAnnal='" +
Session["name"] + "' and isRead=0";
                    //未读消息的个数保存到数据库中
                    int isRead = dataOperate.countData(sqlRead);
                    //创建SQL语句，查询用户信息表中所保存的未读消息总数
                    string sqlMsgCount = "select msgCount from tb_user where name='" + Session["name"] + "'";
                    //调用公共类中的getRow方法并接收该方法所返回的值
                    SqlDataReader sdr = dataOperate.getRow(sqlMsgCount);
                    //读取一条记录
                    sdr.Read();
                    //获取未读消息总数
                    int msgCount = Convert.ToInt32(sdr["msgCount"]);
                    //判断短消息信息表中的未读消息总数是否大于用户信息表中的未读消息总数
                    if (isRead > msgCount)
                    {
                        //创建SQL语句，更新用户信息表中未读消息的总数
                        string sqlUpMesCount = "update tb_user set msgCount=" + isRead + " where name='" + Session["name"] + "'";
                        //调用公共类中的方法执行SQL语句
                        dataOperate.execSql(sqlUpMesCount);
                        ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var msg = new PopBubble('
短消息提示：','您有" + isRead + "封未读消息','查看'); msg.show();", true);
                    }
                }
            }
```

自定义 setXmlChild 方法，该方法设置 xml 文件中的子节点并将 xml 文件显示在 TreeView 控件上。在该方法中通过使用 SQL 语句来获取各个节点的详细信息，并将所获取的信息添加到子节点上。实现代码如下。

例程 15 代码位置：光盘\mr\08\Message\ msgIndex.aspx.cs

```
public void setXmlChild()
    {
        //调用自定义方法设置xml父节点
        setXml();
        //设置xml中好友和黑名单子节点
        //创建SQL语句，查询登录用户的所有好友信息
        string sql = "select * from tb_friend where selfName='" + Session["name"] + "'";
        //调用公共类中的getRows方法并接收该方法返回的值
        DataSet ds = dataOperate.getRows(sql);
        //循环DataSet中的所有行
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            //判断用户的当前好友是否以加入黑名单
            if (!Convert.ToBoolean(ds.Tables[0].Rows[i]["isBlack"]))
            {
                //获取好友名称
                string friendName = ds.Tables[0].Rows[i]["friendName"].ToString();
                //创建XmlDocument对象
                XmlDocument doc = new XmlDocument();
                //加载xml文件
                doc.Load(Server.MapPath("xml//" + Session["name"] + ".xml"));
                //创建XmlNode节点对象
                XmlNode no1;
                //创建XmlElement元素对象
                XmlElement no2;
```

```
            //获取节点
            no1 = doc.SelectSingleNode("menuList/friendList");
            //获取元素
            no2 = doc.CreateElement("friendName");
            //设置元素的属性
            no2.SetAttribute("url", "friendInfo.aspx?id=" + friendName + "&nexus=friend");
            //设置文本
            no2.InnerText = friendName;
            //添加到节点中
            no1.AppendChild(no2);
            //保存xml文件
            doc.Save(Server.MapPath("xml//" + Session["name"] + ".xml"));
        }
        else
        {
            string blackName = ds.Tables[0].Rows[i]["friendName"].ToString();
            XmlDocument doc = new XmlDocument();
            doc.Load(Server.MapPath("xml//" + Session["name"] + ".xml"));
            XmlNode no1;
            XmlElement no2;
            no1 = doc.SelectSingleNode("menuList/blackList");
            no2 = doc.CreateElement("blackName");
            no2.SetAttribute("url", "blackInfo.aspx?id=" + blackName + "");
            no2.InnerText = blackName;
            no1.AppendChild(no2);
            doc.Save(Server.MapPath("xml//" + Session["name"] + ".xml"));
        }
    }
    //设置陌生人子节点
    string sqlStrange = "select distinct sender from tb_msg where accepter='" + Session["name"] + "'";
    SqlDataReader sdrStrange = dataOperate.getRow(sqlStrange);
    while (sdrStrange.Read())
    {
        string s = "select * from tb_friend    where selfName = '" + Session["name"] + "' and friendName = '" +
sdrStrange["sender"] + "'    ";
        if (!dataOperate.isName(s))
        {
            XmlDocument doc = new XmlDocument();
            doc.Load(Server.MapPath("xml//" + Session["name"] + ".xml"));
            XmlNode no1;
            XmlElement no2;
            no1 = doc.SelectSingleNode("menuList/strangeness");
            no2 = doc.CreateElement("StrangeName");
            no2.SetAttribute("url", "friendInfo.aspx?id=" + sdrStrange["sender"] + "&nexus=strange");
            no2.InnerText = sdrStrange["sender"].ToString();
            no1.AppendChild(no2);
            doc.Save(Server.MapPath("xml//" + Session["name"] + ".xml"));
        }
    }
    sdrStrange.Close();
    //设置您最近联系人子节点
    string sqlLinkman = "select top 5 max(id),accepter from tb_msg where accepter not in(select friendName from
tb_friend where isBlack=1 and selfName='" + Session["name"] + "' ) and sender='" + Session["name"] + "'    group by accepter ";
    SqlDataReader sdrLinkman = dataOperate.getRow(sqlLinkman);
    while (sdrLinkman.Read())
    {
        string s = "select count(*) from tb_friend    where selfName = '" + Session["name"] + "' and friendName =
'" + sdrLinkman["accepter"] + "'    ";
        string nexus;
        if (dataOperate.isName(s))
        {
            nexus = "friend";
```

```
            }
            else
            {
                nexus = "strange";
            }
            XmlDocument doc = new XmlDocument();
            doc.Load(Server.MapPath("xml//" + Session["name"] + ".xml"));
            XmlNode no1;
            XmlElement no2;
            no1 = doc.SelectSingleNode("menuList/linkmanList");
            no2 = doc.CreateElement("LinkmanName");
            no2.SetAttribute("url", "friendInfo.aspx?id=" + sdrLinkman["accepter"] + "&nexus=" + nexus);
            no2.InnerText = sdrLinkman["accepter"].ToString();
            no1.AppendChild(no2);
            doc.Save(Server.MapPath("xml//" + Session["name"] + ".xml"));
        }
        sdrLinkman.Close();
    }
```

自定义 setXml 方法用来实现设置 xml 文件中的父节点文本和属性，在该方法中首先将删除 xml 文件再将 xml 模板文件复制为指定的文件名。使用 SQL 语句查询出各个节点的总人数，并将总人数添加到相应的节点上。实现代码如下。

例程 16 代码位置：光盘\mr\08\Message\ msgIndex.aspx.cs

```
public void setXml()
    {
        File.Delete(Server.MapPath("xml//" + Session["name"] + ".xml"));
        //创建FileInfo对象用来复制XML文件
        FileInfo fl = new FileInfo(Server.MapPath("xml//XMLFile.xml"));
        //复制一个XML文件的副本
        fl.CopyTo(Server.MapPath("xml//" + Session["name"] + ".xml"), true);
        //创建SQL语句，该语句用来查询好友的总人数
        string sqlFriend = "select count(*) from tb_friend where selfName='" + Session["name"] + "' and isBlack=0";
        //调用
        int friendList = dataOperate.countData(sqlFriend);
        //创建SQL语句，该语句用来查询黑名单的总人数
        string sqlBlack = "select count(*) from tb_friend where selfName='" + Session["name"] + "' and isBlack=1";
        int blackFriend = dataOperate.countData(sqlBlack);
        //创建SQL语句，该语句用来查询陌生人总数
        string sqlStrange = "select distinct sender from tb_msg where accepter='" + Session["name"] + "'";
        SqlDataReader sdr = dataOperate.getRow(sqlStrange);
        int strange = 0;
        while (sdr.Read())
        {
            string s = "select * from tb_friend    where selfName = '" + Session["name"] + "' and friendName = '" +
sdr["sender"] + "'";
            if (!dataOperate.isName(s))
            {
                strange++;
            }
        }
        sdr.Close();
        //判断最近联系人的总人数
        string sqlLinkman = "select count(*)    from tb_msg where id in(select max(id) from tb_msg where accepter
not in(select friendName from tb_friend where isBlack=1 and selfName='" + Session["name"] + "' ) and sender='" +
Session["name"] + "'    group by accepter ) ";
        int Linkman = dataOperate.countData(sqlLinkman);
        if (Linkman > 5)
        {
            Linkman = 5;
        }
        //创建SQL语句，该语句用来查询未读消息的总数
        string sqlRead = "select count(*) from tb_msg where sender in(select friendName from tb_friend where isBlack=0
and selfName='" + Session["name"] + "' ) and accepter='" + Session["name"] + "' and isRead=0";
```

```
        int isRead = dataOperate.countData(sqlRead);
        //创建XmlDocumnet对象
        XmlDocument doc = new XmlDocument();
        //加载指定的XML文件
        doc.Load(Server.MapPath("xml//" + Session["name"] + ".xml"));
        //创建XmlNode对象并获取好友节点
        XmlNode nodeFriendList = doc.SelectSingleNode("menuList/friendList");
        //创建XmlElemnet对象
        XmlElement eleFriendList = (XmlElement)nodeFriendList;
        //通过使用SetAttribute方法来修改属性值
        eleFriendList.SetAttribute("value", "好友(" + friendList + ")");
        //修改xml中黑名单节点
        XmlNode nodeBlack = doc.SelectSingleNode("menuList/blackList");
        XmlElement eleBlack = (XmlElement)nodeBlack;
        eleBlack.SetAttribute("value", "黑名单(" + blackFriend + ")");
        //修改xml中陌生人节点
        XmlNode nodeStrange = doc.SelectSingleNode("menuList/strangeness");
        XmlElement eleStrange = (XmlElement)nodeStrange;
        eleStrange.SetAttribute("value", "陌生人(" + strange + ")");
        //修改xml修改最近联系人节点
        XmlNode nodeLinkman = doc.SelectSingleNode("menuList/linkmanList");
        XmlElement eleLinkman = (XmlElement)nodeLinkman;
        eleLinkman.SetAttribute("value", "您最近联系人(" + Linkman + ")");
        doc.Save(Server.MapPath("xml//" + Session["name"] + ".xml"));
    }
```

自定义 bindTree 方法用来实现将 xml 文件绑定到 TreeView 控件上并显示出来。在该方法中获取 xml 中各个节点并绑定到 TreeView 控件上。再判断每个节点是否拥有子节点，并调用自定义 setTreeChild 方法设置子节点。最后将 TreeNode 对象添加到 TreeView 控件上显示出来。实现代码如下。

例程 17 代码位置：光盘\mr\08\Message\ msgIndex.aspx.cs

```
protected void bindTree()
{
    //创建XmlDocument对象
    XmlDocument doc = new XmlDocument();
    //加载xml文件
    doc.Load(Server.MapPath("xml//" + Session["name"] + ".xml"));
    //设置好友菜单
    //获取节点
    XmlNode nodeFriendList = doc.SelectSingleNode("menuList/friendList");
    //获取元素
    XmlElement eleFriendList = (XmlElement)nodeFriendList;
    //创建TreeNode对象
    TreeNode tnFriendList = new TreeNode();
    //设置显示文本
    tnFriendList.Text = eleFriendList.GetAttribute("value");
    //设置选择节点时引发的事件
    tnFriendList.SelectAction = TreeNodeSelectAction.Expand;
    //判断是否还有子节点
    if (eleFriendList.HasChildNodes)
    {
        //调用自定义方法获取所有子节点
        TreeViewMenu.Nodes.Add( setTreeChild(tnFriendList, eleFriendList));
    }
    else
    {
        //将TreeNode对象添加到TreeView控件上显示
        TreeViewMenu.Nodes.Add(tnFriendList);
    }
    //设置黑名单菜单
    XmlNode nodeBlack = doc.SelectSingleNode("menuList/blackList");
```

```
XmlElement eleBlack = (XmlElement)nodeBlack;
TreeNode tnBlack = new TreeNode();
tnBlack.Text = eleBlack.GetAttribute("value");
tnBlack.SelectAction = TreeNodeSelectAction.Expand;
if (eleBlack.HasChildNodes)
{
    TreeViewMenu.Nodes.Add(setTreeChild(tnBlack, eleBlack));
}
else
{
    TreeViewMenu.Nodes.Add(tnBlack);
}
//设置陌生人菜单
XmlNode nodeStrange = doc.SelectSingleNode("menuList/strangeness");
XmlElement eleStrange = (XmlElement)nodeStrange;
TreeNode tnStrange = new TreeNode();
tnStrange.Text = eleStrange.GetAttribute("value");
tnStrange.SelectAction = TreeNodeSelectAction.Expand;
if (eleStrange.HasChildNodes)
{
    TreeViewMenu.Nodes.Add(setTreeChild(tnStrange, eleStrange));
}
else
{
    TreeViewMenu.Nodes.Add(tnStrange);
}
//设置最近联系人菜单
XmlNode nodeLinkman = doc.SelectSingleNode("menuList/linkmanList");
XmlElement eleLinkman = (XmlElement)nodeLinkman;
TreeNode tnLinkman = new TreeNode();
tnLinkman.Text = eleLinkman.GetAttribute("value");
tnLinkman.SelectAction = TreeNodeSelectAction.Expand;
if (eleLinkman.HasChildNodes)
{
    TreeViewMenu.Nodes.Add(setTreeChild(tnLinkman, eleLinkman));
}
else
{
    TreeViewMenu.Nodes.Add(tnLinkman);
}
//设置未读消息菜单
XmlNode nodeNotMessage = doc.SelectSingleNode("menuList/notMessage");
XmlElement eleNotMessage = (XmlElement)nodeNotMessage;
TreeNode tnNotMessage = new TreeNode();
tnNotMessage.Text = eleNotMessage.GetAttribute("value");
tnNotMessage.NavigateUrl = eleNotMessage.GetAttribute("url");
tnNotMessage.Target = "mainFrame";
TreeViewMenu.Nodes.Add(tnNotMessage);
//设置退出菜单
XmlNode nodeExitMess = doc.SelectSingleNode("menuList/exitMess");
XmlElement eleExitMess = (XmlElement)nodeExitMess;
TreeNode tnExitMess = new TreeNode();
tnExitMess.Text = eleExitMess.GetAttribute("value");
tnExitMess.NavigateUrl = eleExitMess.GetAttribute("url");
TreeViewMenu.Nodes.Add(tnExitMess);
}
```

自定义 SetTreeChild 方法用来将 xml 文件中的子节点绑定到 TreeView 控件上显示出来。在该方法中主要通过 XmlElement 对象获取所有子节点的数量，通过循环设置 TreeNode 显示文本及跳转路径。最后将 TreeNode 对象返回。实现代码如下。

例程 18 代码位置：光盘\mr\08\Message\ msgIndex.aspx.cs

```
protected TreeNode setTreeChild(TreeNode tn,XmlElement xele)
    {
        //获取子节点的总数
        int j = xele.ChildNodes.Count;
        //创建TreeNode数组
```

```
        TreeNode[] tnChilds = new TreeNode[j];
        for (int i = 0; i < j; i++)
        {
            //实例TreeNode对象
            tnChilds[i] = new TreeNode();
            //设置显示文本
            tnChilds[i].Text = xele.ChildNodes[i].InnerText;
            //设置框架
            tnChilds[i].Target = "mainFrame";
            //设置导航位置
            tnChilds[i].NavigateUrl = xele.ChildNodes[i].Attributes["url"].Value;
            //添加子菜单
            tn.ChildNodes.Add(tnChilds[i]);

        }
        //返回TreeNode
            return tn;
    }
```

在时间控件的 Tick 事件中，将设置 Application 对象的值为当前时间。实现代码如下。

例程 19 代码位置：光盘\mr\08\Message\ msgIndex.aspx.cs

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    //设置application对象中当前的时间
    Application[Session["name"].ToString()] = DateTime.Now;
}
```

8.4.3 好友信息设计

在好友信息页中用户可以查看到好友的详细信息，以及好友的聊天记录等信息。好友信息页面如图 8.5 所示。

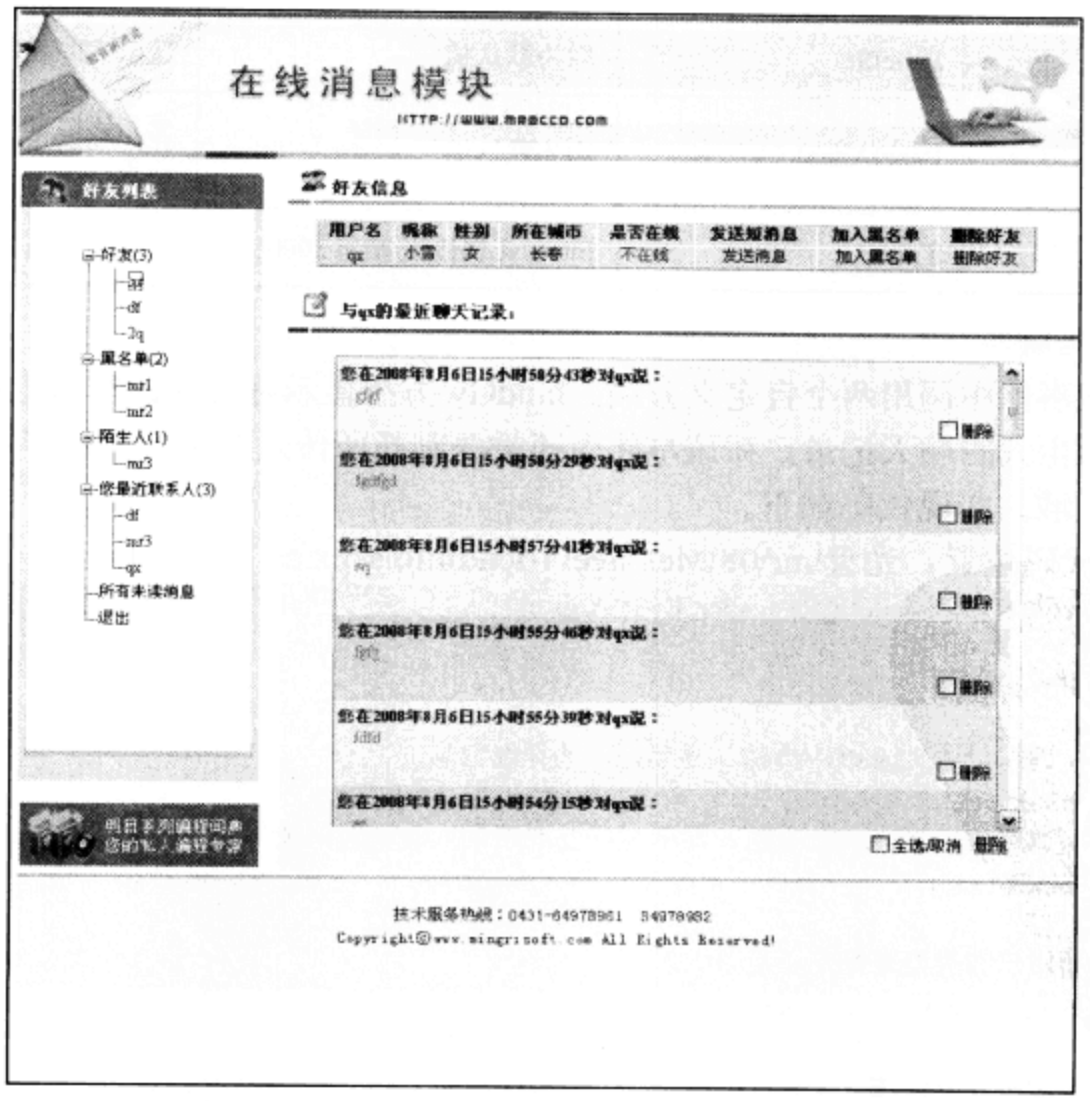


图 8.5 好友信息页

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 friendInfo.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 8.6 所示。

表 8.6　　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
数据/GridView 控件	gvFriendInfo	均为默认值	显示好友的详细信息
标准/Label 控件	labSt	均为默认值	显示陌生人的名称
	labName	均为默认值	显示好友的名称
	labHint	均为默认值	提示用户阅读消息
标准/LinkBtn 控件	linkBtnSend	均为默认值	实现发送消息操作
	linkBtnAddF	均为默认值	实现将陌生人加为好友操作
	linkBtnAddB	均为默认值	实现将陌生人加入黑名单操作
	linkBtnDel	均为默认值	实现删除所选择的聊天记录操作
标准/CheckBox 控件	ckBoxAll	将 AutoPostBack 属性设置为 True	选择或取消全部的聊天记录
数据/ DataList 控件	dlAnnal	均为默认值	显示用户的聊天记录
标准/Panel 控件	panelFriend	均为默认值	显示或隐藏好友区域
	PanelSt	均为默认值	显示或隐藏陌生人区域
	PanelAnnal	均为默认值	显示或隐藏聊天记录区域的滚动条
Ajax/ Timer 控件	Timer	Interval 属性设置为 20000	判断是否有未读消息

2. 后台代码编写

在页面加载事件中调用两个自定义方法。bindGv 方法显示好友的详细信息和 bindDL 方法显示与好友或陌生人的聊天记录。在该事件中还需要判断所传入的用户名是好友还是陌生人，并显示相应的区域。实现代码如下。

例程 20　代码位置：光盘\mr\08\Message\ friendInfo.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //调用自定义bindGV方法，显示好友的详细信息
        bindGv();
        //调用自定义bindDL方法，显示与好友或陌生人的聊天记录
        bindDL();
    }
    //判断用户的类型是好友还是陌生人
    if (Request["nexus"].ToString() == "friend")
    {
        //显示好友区域
        panelFriend.Visible = true;
```

```
            //隐藏陌生人区域
            PanelSt.Visible = false;
            //显示好友名称
            labName.Text = Request["id"].ToString();
        }
        else
        {
            //隐藏好友区域
            panelFriend.Visible = false;
            //显示陌生人区域
            PanelSt.Visible = true;
            //显示陌生人名称
            labSt.Text = Request["id"].ToString();
        }
    }
```

自定义 bindGv 方法用来绑定 GridView 控件显示好友的详细信息，在该方法中获取传入的用户名，并创建 SQL 语句查询用户的详细信息绑定到 GridView 控件上显示出来。实现代码如下。

例程 21 代码位置：光盘\mr\08\Message\ friendInfo.aspx.cs

```
protected void bindGv()
{
    //获取传入的用户名称
    string friendName = Request["id"].ToString();
    //创建SQL语句，查询好友的详细信息
    string sqlFriend = "select * from tb_user where name='" + friendName + "'";
    //设置GridView控件的数据源
    gvFriendInfo.DataSource = dataOperate.getRows(sqlFriend);
    gvFriendInfo.DataBind();
}
```

自定义 bindDL 方法用来绑定 DataList 控件显示聊天记录。在该方法中使用 SQL 语句查询出与好友或陌生人的聊天记录并绑定到 DataList 控件上显示出来。还需要判断聊天记录的总数是否大于 1 如果大于 1 将设置 Panel 控件的滚动条，不大于 1 将使用 Label 控件提示“查看信息后才可看到聊天记录”。实现代码如下。

例程 22 代码位置：光盘\mr\08\Message\ friendInfo.aspx.cs

```
protected void bindDL()
{
    //创建SQL语句查询与好友或陌生人的聊天记录
    string sqlSel = "select * from tb_msg where (sender='" + Session["name"] + "' and accepter='" + Request["id"] + "'
and nameAnnal='" + Session["name"] + "' ) or ( sender='" + Request["id"] + "' and accepter='" + Session["name"] + "' and
nameAnnal='" + Session["name"] + "' and isRead=1 ) order by id desc ";
    //设置数据源
    dlAnnal.DataSource = dataOperate.getRows(sqlSel).Tables[0].DefaultView;
    //设置主键
    dlAnnal.DataKeyField = "id";
    dlAnnal.DataBind();
    //判断是否拥有聊天记录
    if (dlAnnal.Items.Count < 1)
    {
        //使用Label控件提示需要查看聊天记录
        labHint.Text = "查看信息后才可看到聊天记录";
        //设置复选框为不显示状态
        ckBoxAll.Visible = false;
        //设置LinkBtn按钮为不显示状态
        linkBtnDel.Visible = false;
    }
    else
    {
        //设置Panel控件的滚动条
        PanelAnnal.ScrollBars = ScrollBars.Vertical;
```

```
                //设置复选框为显示状态
                ckBoxAll.Visible = true;
                //设置LinkBtn按钮为显示状态
                linkBtnDel.Visible = true;
            }
        }
```

在 GridView 控件的 RowCommand 事件中将实现打开发送消息窗口、将好友加入黑名单和删除好友操作。在该事件中首先将判断要执行的命令并根据相应的命令执行相应的操作。实现代码如下。

例程 23 代码位置：光盘\mr\08\Message\ friendInfo.aspx.cs

```
        protected void gvFriendInfo_RowCommand(object sender, GridViewCommandEventArgs e)
        {
            //判断是否为发送信息操作
            if (e.CommandName == "send")
            {
                //使用JavaScript打开发送消息窗口
                ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "showModalDialog ('sendMessage.aspx?
friendName=" + e.CommandArgument + "&nexus=" + Request["nexus"] + " ',window,'width=340,height=200','alwaysRaised= yes')", true);

            }
                //判断是否为加入黑名单操作
            else if (e.CommandName == "addBlack")
            {
                //创建SQL语句，更新将好友加入到黑名单
                string sql = "update tb_friend set isBlack =1 where selfName='" + Session["name"] + "' and friendName='" +
e.CommandArgument + "'";
                //执行SQL语句
                dataOperate.execSql(sql);
                //使用JavaScript刷新主框架
                ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "window.parent.location.
('msgIndex.aspx?tp=" + DateTime.Now.ToString() + "');", true);
            }
                //判断是否为删除好友操作
            else if (e.CommandName == "delFriend")
            {
                //创建SQL语句，删除好友
                string sqlDel = "delete tb_friend where selfName='" + Session["name"] + "' and friendName='" + e.CommandArgu
ment + "'";
                //执行SQL语句
                dataOperate.execSql(sqlDel);
                //使用JavaScript刷新主框架
                ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "window.parent.location.
reload('msgIndex.aspx?tp=" + DateTime.Now.ToString() + "');", true);
            }
        }
```

在时间控件的 Tick 事件中，首先将调用自定义 bindGV 方法重新绑定 GridView 控件显示用户是否在线，再使用 SQL 语句判断获取消息表中的未读消息总数和用户表中的未读消息总数，将两个总数进行比较。如果消息表中的总数大于用户表中的总数，说明还有未读消息。调用 JavaScript 提示用户有未读消息并更新用户信息表中的未读消息总数。实现代码如下。

例程 24 代码位置：光盘\mr\08\Message\ friendInfo.aspx.cs

```
        protected void Timer1_Tick(object sender, EventArgs e)
        {
            //调用自定义bindGV方法，用来检查用户是否在线
            bindGv();
```

```
            //创建SQL语句，查询出消息表中所有未阅读消息的总数
            string sqlRead = "select count(*) from tb_msg where sender not in(select friendName from tb_friend where
isBlack=1 and selfName='" + Session["name"] + "' ) and accepter='" + Session["name"] + "' and nameAnnal='" + Session["name"] + "'
and isRead=0";
            //保存未读消息的总数
            int isRead = dataOperate.countData(sqlRead);
            //创建SQL语句，该语句用来查询用户表中所保存的未读消息总数
            string sqlMsgCount = "select msgCount from tb_user where name='" + Session["name"] + "'";
            //获取SqlDataReader对象
            SqlDataReader sdr = dataOperate.getRow(sqlMsgCount);
            //读取一条记录
            sdr.Read();
            //获取用户表中未读消息总数
            int msgCount = Convert.ToInt32(sdr["msgCount"]);
            //判断是否提示有未阅读的消息
            if (isRead > Math.Abs(msgCount))
            {
                //创建SQL语句，更新用户信息表中的未读消息总数
                string sqlUpMesCount = "update tb_user set msgCount=" + isRead + " where name='" + Session["name"] + "'";
                //执行操作
                dataOperate.execSql(sqlUpMesCount);
        ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "var msg = new PopBubble('短消息提
示：','您有" + isRead + "封未读消息','查看'); msg.show();", true);MSG1.show();", true);
            }
        }
```

删除聊天记录是在“删除”按钮的单击事件中实现的。在该事件中通过遍例 DataList 控件，判断每一项中的复选框是否选择，如果被选中将通过主键创建 SQL 删除语句将选择的聊天记录删除掉。最后将重新绑定 DataList 对象。实现代码如下。

例程 25　代码位置：光盘\mr\08\Message\ friendInfo.aspx.cs

```
protected void linkBtnDel_Click(object sender, EventArgs e)
{
    //遍例DataList控件中的每一项
    foreach (DataListItem dl in dlAnnal.Items)
    {
        //判断复选框是否为选中状态
        if (((CheckBox)dl.FindControl("CheckBox1")).Checked)
        {
            //获取主键
            string id = dlAnnal.DataKeys[dl.ItemIndex].ToString();
            //创建SQL语句，根据主键删除聊天记录
            string sqlDel = "delete tb_msg where id=" + id;
            //执行SQL语句
            dataOperate.execSql(sqlDel);
        }
    }
    //重新绑定DataList控件
    bindDL();
}
```

8.4.4　发送消息设计

发送消息页中用来实现对好友进行发送消息操作。在该页面中用户还可以选择多个好友进行发送消息操作。发送消息页面如图 8.6 所示。

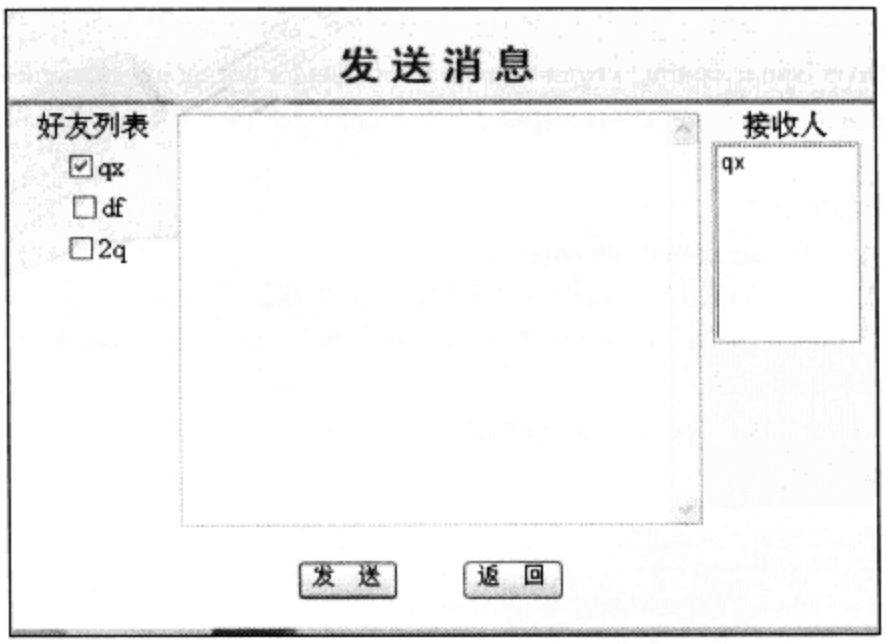


图 8.6　发送消息页

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 sendMessage.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 8.7 所示。

表 8.7　　控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说　　明
标准/TextBox 控件	txtMessage	将 TextMode 属性设置为 MultiLine	输入要发送的短消息
标准/ListBox 控件	lbxFriendList	均为默认值	显示消息的接收人
标准/ CheckBoxList 控件	ckBoxFriendList	将 AutoPostBack 属性设置为 True	选择消息接收人的名称
标准/Button 控件	btnSend	均为默认值	实现发送消息操作
标准/Label 控件	labFriendList	均为默认值	显示复选框列表提示

2. 后台代码编写

在页面加载事件中，首先判断接收消息的人是好友还是陌生人。如果为好友将可以选择多个接收消息人，设置复选框列表的数据源并根据复选框的文本设置所选中状态。将复选框选中的文本添加到接收人的列表框中。如果为陌生人将隐藏复选框列表，并将陌生人的名称添加到列表框中。实现代码如下。

例程 26　代码位置：光盘\mr\08\Message\ sendMessage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //判断是好友还是陌生人
        if (Request["nexus"].ToString() == "friend")
        {
            //创建SQL语句，查询所有的好友信息
            string sqlFriendList = "select * from tb_friend where selfName='" + Session["name"] + "' and isBlack=0 ";
            //设置复选框列表的数据源
            ckBoxFriendList.DataSource = dataOperate.getRows(sqlFriendList);
            //设置复选框列表的显示文本
            ckBoxFriendList.DataTextField = "friendName";
```

```
                //绑定复选框
                ckBoxFriendList.DataBind();
                //获取传入的好友名称
                string s = Request["friendName"];
                //循环复选框列表
                for (int i = 0; i < ckBoxFriendList.Items.Count; i++)
                {
                    //判断复选框列表中的某项文本是否和传入的名称相同
                    if (ckBoxFriendList.Items[i].Text == s)
                    {
                        //设置复选框为选中状态
                        ckBoxFriendList.Items[i].Selected = true;
                    }
                }
                //调用自定义bindListBox方法显示所选中的好友名称
                bindListBox();
            }
            else
            {
                //设置复选框列表为不显示状态
                ckBoxFriendList.Visible = false;
                //设置Label控件为不显示状态
                labFriendList.Visible = false;
                //在列表控件中显示陌生人的名称
                lbxFriendList.Items.Add(Request["friendName"]);
            }
        }
    }
```

定义 bindListBox 方法用来实现将复选框列表中所选中的文本添加到接收人列表框中。在该方法中主要通过循环判断每一个复选框是否为选中状态，如果为选中状态将添加到接收人的列表框中。实现代码如下。

例程 27 代码位置：光盘\mr\08\Message\ sendMessage.aspx.cs

```
protected void bindListBox()
    {
        //清空列表框
        lbxFriendList.Items.Clear();
        //循环复选框列表
        for (int i = 0; i < ckBoxFriendList.Items.Count; i++)
        {
            //判断复选框是否为选中状态
            if (ckBoxFriendList.Items[i].Selected)
            {
                //将选中的复选框文本添加到列表框中
                lbxFriendList.Items.Add(ckBoxFriendList.Items[i].Text);
            }
        }
    }
```

在“发送”按钮的单击事件中将实现发送消息的操作。在该事件中将通过接收人列表中的人数，循环发送消息。发送的消息将保存在消息表中接收人的名下和发送人的名下。如果接收人将发送人添加到黑名单中，那么发送的消息不保存在接收人的名下。实现代码如下。

例程 28 代码位置：光盘\mr\08\Message\ sendMessage.aspx.cs

```
    protected void btnSend_Click(object sender, EventArgs e)
    {
        //判断发送消息是否为空
```

```
            if (txtMessage.Text != "")
            {
                //获取发送消息并使用公共类中的filtrateHtml过滤掉非法字符
                string mess = dataOperate.filtrateHtml(txtMessage.Text);
                //使用循环发送消息
                for (int i = 0; i < lbxFriendList.Items.Count; i++)
                {
                    string sqlInSend = "";
                    string sqlInAcce = "";
                    //获取接收人名称
                    string accepter = lbxFriendList.Items[i].Text;
                    //获取发送人的名称
                    string send = Session["name"].ToString();
                    //创建SQL语句，查询接收人是否将自己加入黑名单
                    string sqlSel = "select * from tb_friend where selfName='" + accepter + "' and friendName='" + send + "'";
                    SqlDataReader sdr = dataOperate.getRow(sqlSel);
                    //读取一条记录，如果不可读取说明自己是在对方的陌生人中
                    if (sdr.Read())
                    {
                        string isBlack = sdr["isBlack"].ToString();
                        //判断对方是否加入自己为黑名单
                        if (!Convert.ToBoolean(isBlack))
                        {
                    //如果对方未将自己加入黑名单，将创建SQL语句在对方的名下保存所发送到消息
                            sqlInAcce = "insert into tb_msg values('" + accepter + "','" + send + "','" + accepter + "','" +
mess + "','" + DateTime.Now.ToString() + "','')";
                        }
                    }
                    else
                    {
                    //如果自己是对方的陌生人，将创建SQL语句在对方的名下保存所发送到消息
                    sqlInAcce = "insert into tb_msg values('" + accepter + "','" + send + "','" + accepter + "','" + mess + "','" +
DateTime.Now.ToString() + "','')";
                    }
                    //创建SQL语句，将发送的消息保存在自己的名下
                    sqlInSend = "insert into tb_msg values('" + send + "','" + send + "','" + accepter + "','" + mess + "','" +
DateTime.Now.ToString() + "','')";
                    //执行SQL语句
                    dataOperate.execSql(sqlInSend);
                    //判断字符串变量是否为空
                    if (sqlInAcce != "")
                    {
                        //执行SQL语句
                        dataOperate.execSql(sqlInAcce);
                    }
                }
                ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "alert('发送成功！');window.
opener.location.replace(opener.location);window.close();", true);
            }
            else
            {
                ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", "alert('发送信息不能为空！
')", true);
            }
        }
```

8.4.5 所有未读消息设计

所有未读消息页面用来显示用户所有未读的消息。在该页面中用户可以查看未读消息，并对消息进行回复，如图 8.7 所示。

图 8.7 所有未读消息页面

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 notReaderMessage.aspx。

（2）在该窗体中添加 1 个 ScriptManager 控件用来管理页面中的 Ajax 控件，添加 1 个 UpdatePanel 控件用来实现局部更新功能，再添加 1 个 Timer 控件用来实现定时显示所有未读消息。

（3）在该窗体中还需要添加 1 个 DataList 控件，该控件用来显示所有的未读消息信息。该控件的前台绑定代码如下。

例程 29 代码位置：光盘\mr\08\Message\ notReaderMessage.aspx

```
<asp:DataList ID="dlListMess" runat="server" BackColor="White" BorderColor="#E7E7FF" BorderStyle="
None" BorderWidth="1px" CellPadding="3" GridLines="Horizontal" OnUpdateCommand="dlListMess_UpdateCommand"
Width="546px" OnEditCommand="dlListMess_EditCommand">
    <ItemTemplate>
        <table border="0" cellpadding="0" cellspacing="0">
            <tr>
                <td style="width: 1932px; height: 19px; text-align: left;">
                <span style=" font-weight:bolder" >  在 <%#dataOperate.strDate(Convert.ToDateTime
(Eval ("msgDate"))) %><%#Eval("sender") %>给您发了信息！ </span>
                </td>
                <td style="width: 18px; height: 19px; text-align: left">
```

```
                                <asp:LinkButton ID="linkBtnReader" runat="server" Font-Underline="False" Width=" 42px"
CommandArgument='<%# Eval("id") %>' CommandName="Update">查看</asp:LinkButton></td>
                                <td style="width: 324px; height: 19px">
                                    <asp:LinkButton ID="LinkButton1" runat="server" CommandArgument='<%#
Eval("sender") %>'
                                        CommandName="Edit">回复</asp:LinkButton></td>
                            </tr>
                        </table>
                    </ItemTemplate>
                    <FooterStyle BackColor="#B5C7DE" ForeColor="#4A3C8C" />
                    <AlternatingItemStyle BackColor="#F7F7F7" />
                    <ItemStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
                    <SelectedItemStyle BackColor="#738A9C" Font-Bold="True" ForeColor="#F7F7F7" />
                    <HeaderStyle BackColor="#4A3C8C" Font-Bold="True" ForeColor="#F7F7F7" />
                </asp:DataList>
```

2. 后台代码编写

在页面的加载事件中将调用自定义 bindDl 方法，该方法用来显示所有未读的消息。在自定义 bindDl 方法中，首先将创建 SQL 语句查询出当前用户所有的未读消息，并将查询的数据绑定到 DataList 控件上显示出来。实现代码如下。

例程 30 代码位置：光盘\mr\08\Message\ notReaderMessage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            //调用自定义方法显示所有未读消息
            bindDl();
        }
    }
    protected void bindDl()
    {
        //创建SQL语句，查询所有未读消息，其中不包括黑名单用户所发的消息
        string sqlMess = "select * from tb_msg where sender not in(select friendName from tb_friend where isBlack=1
and selfName='" + Session["name"] + "' ) and accepter='" + Session["name"] + "'   and nameAnnal='" + Session["name"] + "'
and isRead=0 order by id desc ";
        //调用公共类中的getRows方法并将该方法返回的数据设置为DataList控件的数据源
        dlListMess.DataSource = dataOperate.getRows(sqlMess);
        //绑定DataList控件
        dlListMess.DataBind();
    }
```

在 DataList 控件的 UpdateCommand 事件中将实现打开查看消息窗口功能，在该事件中首先通过 SQL 语句将要阅读的消息修改为已阅读消息，再将用户消息表中的未阅读消息总数减一。最后通过 JavaScript 代码来实现打开查看消息的窗口。实现代码如下。

例程 31 代码位置：光盘\mr\08\Message\ notReaderMessage.aspx.cs

```
protected void dlListMess_UpdateCommand(object source, DataListCommandEventArgs e)
    {
        //创建SQL语句，更新当前的消息为已读消息
        string sqlUpdate = "update tb_msg set isRead=1 where id=" + e.CommandArgument;
        //执行SQL语句
        dataOperate.execSql(sqlUpdate);
        //创建SQL语句，将用户信息表未读消息总数减一
        string sqlUpMesCount = "update tb_user set msgCount=msgCount-1 where name='" + Session["name"] + "'";
        //执行SQL语句
```

```
            dataOperate.execSql(sqlUpMesCount);
            //打开查看消息窗口
             ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", " window.showModalDialog
('readMessage.aspx?id=" + e.CommandArgument + " ', window, 'dialogHeight:400px; dialogWidth: 500px;dialogTop:px; dialogLeft:px; edge:
Raised; center: Yes; help: No; resizable: No; status: No;scroll:No');", true);
        }
```

在 DataList 控件的 EditCommand 事件中实现打开回复消息窗口。在该事件中首先通过 SQL 语句来判断给自己发送信息的用户是否为自己的好友，如果是好友将设置字符串变量 nexus 的值为 friend，否则设置为 strange。最后通过 JavaScript 代码打开回复消息窗口并传入字符串变量 nexus。实现代码如下。

例程 32 代码位置：光盘\mr\08\Message\ notReaderMessage.aspx.cs

```
protected void dlListMess_EditCommand(object source, DataListCommandEventArgs e)
    {
        //创建SQL语句，查询给自己发送消息的用户是否为自己的好友
         string sqlSel = "select count(*) from tb_friend   where selfName = '" + Session["name"]   + "' and friendName ='" +
e.CommandArgument + "'";
        //创建字符串变量保存要传送的参数
        string nexus;
        //判断是否为好友
        if (dataOperate.isName(sqlSel))
        {
            //设置传送的参数为好友
            nexus = "friend";
        }
        else
        {
            //设置传送的参数为陌生人
            nexus = "strange";
        }
        //打开发送消息窗口
         ScriptManager.RegisterClientScriptBlock(UpdatePanel1, this.GetType(), "", " showModalDialog('sendMessage. aspx?
friendName=" + e.CommandArgument + "&nexus=" + nexus + " ', window, 'dialogHeight:400px; dialogWidth: 500px;dialogTop:px;
dialogLeft:px; edge: Raised; center: Yes; help: No; resizable: No; status: No;scroll:No');", true);
    }
```

8.5 网站打包与发布

网站开发完成后最终的目地是将其发布到 Internet 上，以提供用户浏览访问。实现的网站发布可以使用 2 种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍下使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

（1）首先打开要发布的网站，在菜单栏中选择“生成”选项，在弹出的快捷方式菜单中选择“发布网站”选项，如图 8.8 所示。

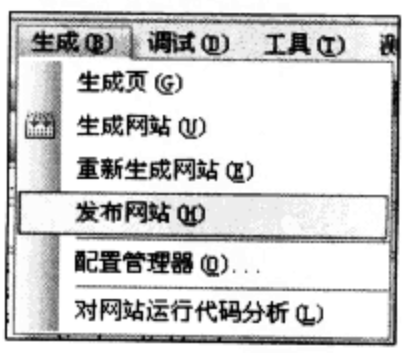


图 8.8 选择“发布网站”

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择“...”路径按钮，如图 8.9 所示。

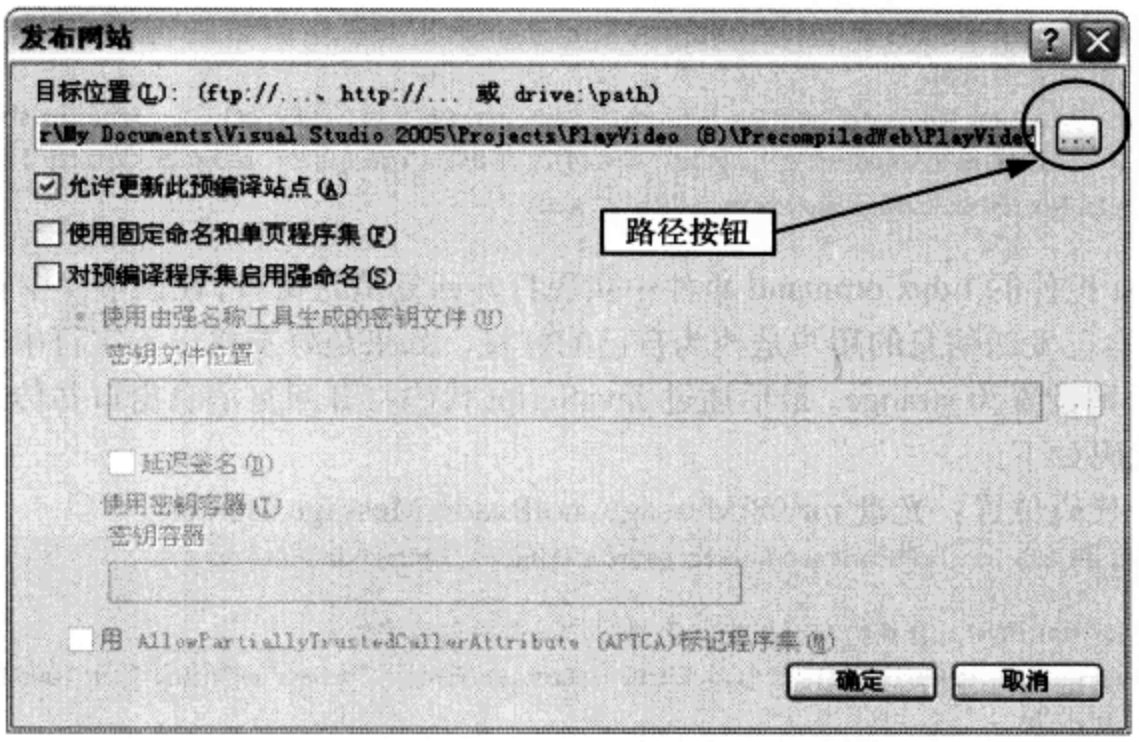


图 8.9　选择路径按钮

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选择中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码，如图 8.10 所示。填写完毕后选择“打开”按钮将会返回“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

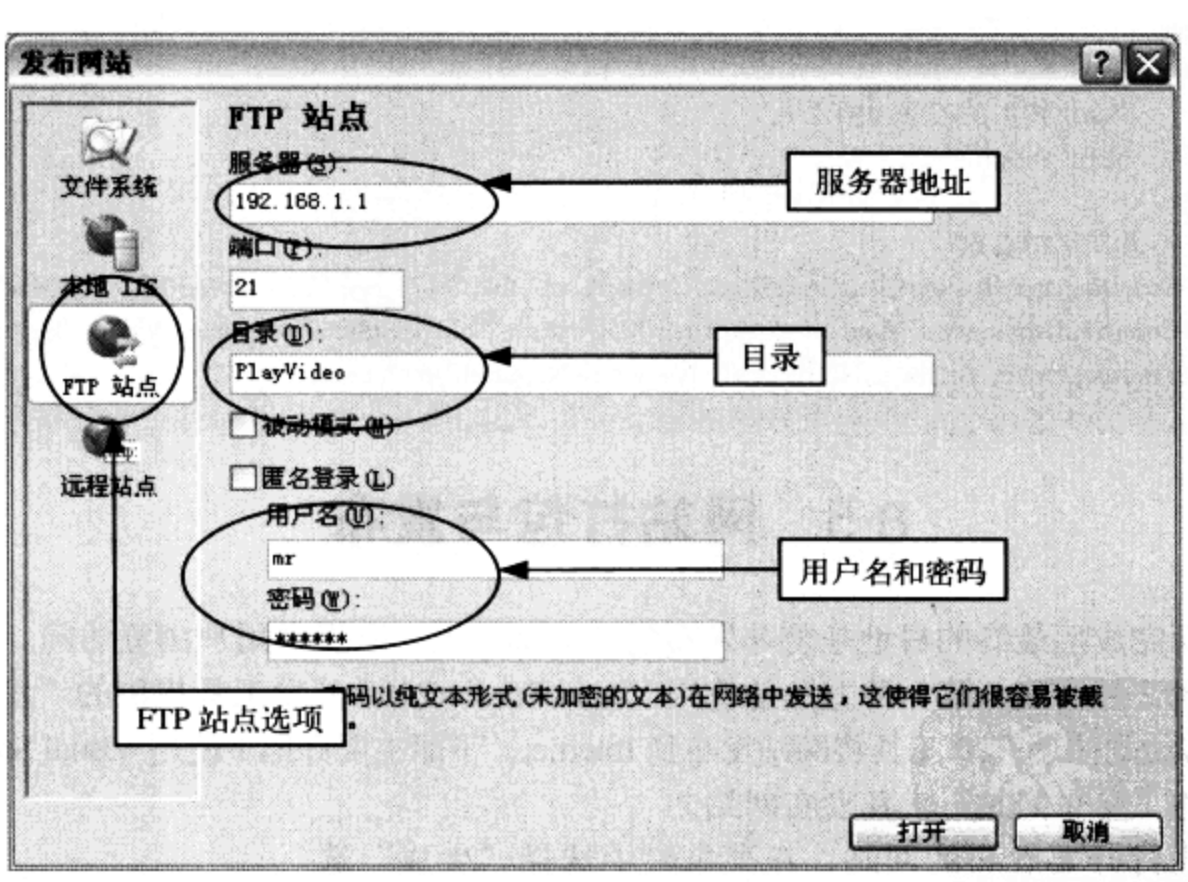


图 8.10　FTP 站点选项

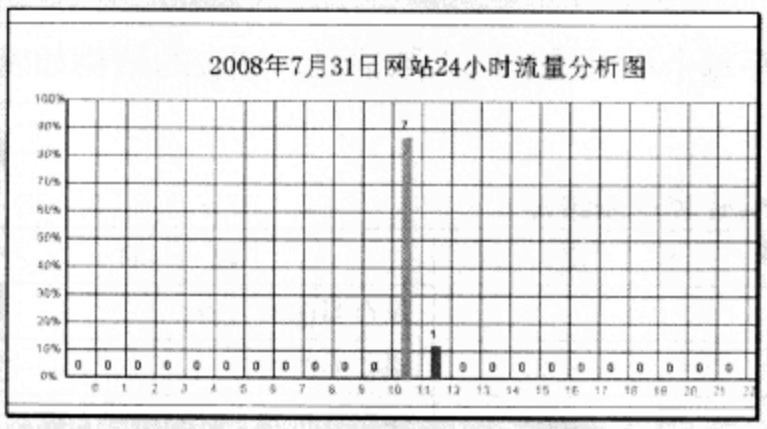
网站统计分析

第9章

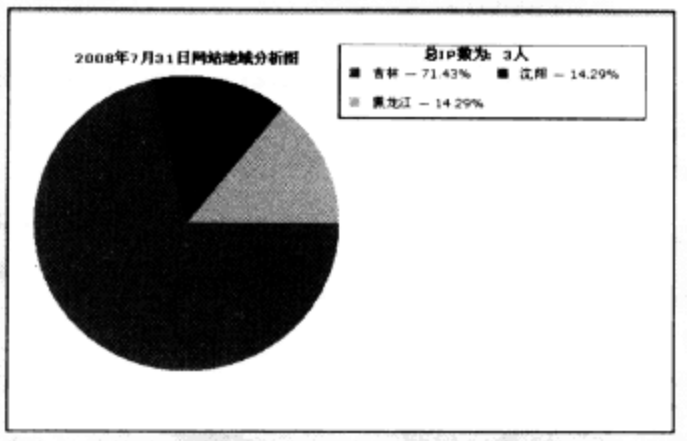
实例位置：光 盘\mr\09\

随着网站建设的日益深入，大型网站的日益增多。对于大型网站来说，网站的统计分析功能是不可缺少的一部分。通过对网站使用统计分析功能，可以非常清楚的反映出网站的人气指数，例如，日网站的访问量，月网站的访问量，地域分析等。当对某个网站进行价值分析时，网站的统计分析是很重要的一个参数，因此网站统计分析模块是不可忽视的。通过本章，读者可以学到以下内容。

柱形图的绘制



饼型图的绘制



日客户端分析

浏览器信息调查信息

浏览器	使用人数	总人数比例
Internet Explorer 6.0	3	100%
Internet Explorer 5.0	0	0%
Firefox 3.0b2	0	0%
其他	0	0%

操作系统信息

操作系统	使用人数	总人数比例
Windows 2003	3	100%
Windows XP	0	0%
Windows 2000	0	0%
UNIX	0	0%
Linux	0	0%
其他	0	0%

日回访率分析

2008年7月31日 回访率的统计

回访率	访问人数	总人数比例
第一次访问	2	66.67%
第一次回访	0	0.00%
第二次回访	0	0.00%
第三次回访	0	0.00%
第四次回访	1	33.33%
第五次回访	0	0.00%
第六次回访	0	0.00%
第七次回访	0	0.00%
第八次回访	0	0.00%
第九次回访	0	0.00%
第十次回访(或多余十次)	0	0.00%

9.1 网站统计分析概述

9.1.1 功能概述

网站统计分析模块是比较重要的模块，通过该模块可以查看网站的人气指数等信息。例如，通过日时段分析或月时段分析，可以查看用户访问时间的分布情况和访问量。通过日地域分析或月地域分析，可以查看各地域访问人数的对比。通过日客户端分析或月客户端分析，可以查看访问用户的浏览器和操作系统的信息。通过日回访率或月回访率，可以查看出网站受欢迎的程度。通过对以上信息的掌握，可以对网站进行实际有效的修改。在本程序中是以图形的形式表示分析结果的，管理人员会更清晰地查看出分析结果。

9.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建一个名为 db_statInfo 的数据库，在该数据库中创建 5 张表，这 5 张表分别用来记录，各时段数据、每日用户信息数据、各地域 IP 数据、浏览器的数据、客户端的数据。下面将详细介绍这 5 张表的数据结构。

- 时段信息表

时段信息表（tb_hourStat），该表用来保存每个小时的访问人数信息。该表的结构如表 9.1 所示。

表 9.1 tb_hourStat 表的表结构

字　段	类　型	长　度	说　明
months	int	4	保存当前的月份
days	int	4	保存当前的日
hour0	int	4	保存 0 时到 1 时的访问人数
hour1	int	4	保存 1 时到 2 时的访问人数
hour2	int	4	保存 2 时到 31 时的访问人数
hour3	int	4	保存 3 时到 4 时的访问人数
hour4	int	4	保存 4 时到 5 时的访问人数
hour5	int	4	保存 5 时到 6 时的访问人数
hour6	int	4	保存 6 时到 7 时的访问人数
hour7	int	4	保存 7 时到 8 时的访问人数
hour8	int	4	保存 8 时到 9 时的访问人数
hour9	int	4	保存 9 时到 10 时的访问人数
hour10	int	4	保存 10 时到 11 时的访问人数
hour11	int	4	保存 11 时到 12 时的访问人数
hour12	int	4	保存 12 时到 13 时的访问人数
hour13	int	4	保存 13 时到 14 时的访问人数
hour14	int	4	保存 14 时到 15 的访问人数
hour15	int	4	保存 15 时到 16 时的访问人数
hour16	int	4	保存 16 时到 17 时的访问人数
hour17	int	4	保存 17 时到 18 时的访问人数
hour18	int	4	保存 18 时到 19 时的访问人数

续表

字　段	类　型	长　度	说　明
hour19	int	4	保存 19 时到 20 时的访问人数
hour20	int	4	保存 20 时到 21 时的访问人数
hour21	int	4	保存 21 时到 22 时的访问人数
hour22	int	4	保存 23 时到 0 时的访问人数
hour23	int	4	保存 0 时到 1 时的访问人数

- 日用户信息表

日用户信息表（tb_dayStat），该表用来保存每日用户信息包括用户 IP、操作系统、浏览器信息等。该表的结构如表 9.2 所示。

表 9.2　　tb_dayStat 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长编号
ip	varchar	30	保存用户的 IP 信息
sumNum	int	4	保存用户的访问次数
days	int	4	保存当前的日
months	int	4	保存当前的月份
systemId	int	4	保存用户的操作系统
browserId	int	4	保存用户的浏览器

- 浏览器信息表

浏览器信息表（tb_browser），该表用来保存浏览器的信息。该表的结构如表 9.3 所示。

表 9.3　　tb_browser 表的表结构

字　段	类　型	长　度	说　明
browserId	int	4	保存浏览器的编号
browserName	varchar	30	保存浏览器的名称

- 操作系统信息表

操作系统信息表（tb_system），该表用来保存操作系统的信息。该表的结构如表 9.4 所示。

表 9.4　　tb_system 表的表结构

字　段	类　型	长　度	说　明
systemId	int	4	保存操作系统的编号
systemName	varchar	30	保存操作系统的名称

- 地域信息表

地域信息表（tb_region），该表用来保存每个地域的 IP 信息。该表的结构如表 9.5 所示。

表 9.5　　tb_region 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长的编号
placename	varchar	50	保存地域的名称
ip	varchar	30	保存地域的 IP

为了方便在程序中进行查询，在该数据库中创建了一个视图和一个存储过程。视图和存储过程的详细介绍如下。

- view_region 视图

视图 view_region 用来保存日用户信息并显示 IP 的对应地域名称。该视图是使用日用户信息表（tb_dayStat）和地域信息表（tb_region）进行连接和获得的。创建视图的代码如下：

```
create view view_region
as
select a.placename, a.Ip, b.sumNum, b.days, b.months from dbo.tb_region a inner join dbo.tb_dayStat b on a.Ip = b.ip
go
```

- P_monthRegion 存储过程

存储过程 P_monthRegion 用来创建一个视图，在存储过程中首先判断视图是否存在，如果视图存在将删除该视图。再通过设置 SQL 语句创建一个视图。所创建的视图用于保存 view_region 视图中月用户的信息。实现代码如下：

```
create   procedure P_monthRegion
@months varchar(10)      --设置一个参数，该参数表示月份
as
begin
if exists ( select * from sysobjects where name = 'view_monthRegion' )   --判断是视图是否存在
begin
drop view view_monthRegion     --如果存在删除视图
end
declare @sql varchar (200)   --创建一个变量该变量用来保存创建视图的语句
set @sql='create view view_monthRegion as select placename,sum(sumNum) as sumNum from view_region where
months='+@months+' group by placename'
exec (@sql)   --执行创建视图的SQL语句
end
go
```

执行该存储过程需要传入一个参数该参数表示月份。例如，需要创建 5 月份的用户信息视图。执行代码如下：

```
exec P_monthRegion @months='5'
```

9.2 网站统计分析关键技术

9.2.1 GDI+绘制图形

为了使管理员更直观地查看统计分析的结果，在时段分析和地域分析中使用了柱形图和饼型图来显示分析结果。柱形图和饼型图是通过使用 GDI+绘图技术来实现的。GDI+是图形设备接口（GDI）的高级版本，它提供了各种丰富的图形图像处理功能。GDI+主要有二维矢量图形、图像处理和版式 3 部分组成。下面将介绍如何使用 GDI+绘制图形。

GDI+存在于 System.Drawing.dll 程序集中。图形图像处理中常常调用的名称空间及说明如表 9.6 所示。

表 9.6　　GDI+基类的主要命名空间及说明

命 名 空 间	说　　明
System.Drawing	提供了对 GDI+基本图形功能的访问
System.Drawing.Drawing2D	提供了高级的二维和矢量图形功能
System.Drawing.Imaging	提供了高级 GDI+图像处理功能
System.Drawing.Printing	把打印机或打印预览窗口作为输出设备时使用的类
System.Drawing.Design	一些预定义的对话框、属性表和其他用户界面元素，与在设计期间扩展用户界面相关
System.Drawing.Text	提供了高级 GDI+字体和文本排版功能

要进行图形的绘制，首先必须创建 Graphics 对象，然后才能利用该对象的方法进行图形的绘制或文字的绘制。Graphics 对象的常用方法及说明如表 9.7 所示。

表 9.7　　Graphics 对象的常用方法及说明

命 名 空 间	说　　明
Clear	清除整个绘图面并以指定背景色填充
DrawArc	已重载。绘制一段弧线，它表示由一对坐标、宽度和高度指定的椭圆部分
DrawBezier	绘制由 4 个 Point 结构定义的贝塞尔样条。已重载
DrawCurve	绘制经过一组指定的 Point 结构的基数样条。已重载
DrawEllipse	绘制一个由边框（该边框由一对坐标、高度和宽度指定）定义的椭圆。已重载
DrawIcon	在指定坐标处绘制由指定的 Icon 表示的图像。已重载
DrawLine	绘制一条连接由坐标对指定的两个点的线条。已重载
DrawPie	绘制一个扇形，该形状由一个坐标对、宽度、高度以及两条射线所指定的椭圆定义。已重载
DrawPolygon	已重载。绘制由一组 Point 结构定义的多边形
DrawRectangle	绘制由坐标对、宽度和高度指定的矩形。已重载
DrawString	在指定位置并且用指定的 Brush 和 Font 对象绘制指定的文本字符串。已重载
FillEllipse	填充边框所定义的椭圆的内部，该边框由一对坐标、一个宽度和一个高度指定。已重载
FillPie	填充由一对坐标、一个宽度、一个高度以及两条射线指定的椭圆所定义的扇形区的内部。已重载
FillRectangle	填充由一对坐标、一个宽度和一个高度指定的矩形的内部。已重载

通过上面的表格已经了解 Graphics 比较常用的方法了，下面介绍如何创建 Graphics 对象。在 ASP.NET 中可以从任何由 Image 类派生的对象创建 Graphics 对象。例如，使用 System.Drawing.Graphics.FromImage(System.Drawing.Image)方法来创建 Graphics 对象并需要传入一个的 Image 对象名，代码如下：

```
Bitmap bitmap = new Bitmap(80, 80);
Graphics g = Graphics.FromImage(bitmap);
Pen RedPen=new Pen(Color.Red,3);                    //创建画笔
g.DrawLine(RedPen,50,20,500,20);                    //绘制直线
```

注意　获得图形对象引用之后，即可绘制对象、给对象着色等。由于图像对象非常占资源，所以在不用这些对象时要用 Dispose 方法及时释放资源。

9.2.2　柱型图的绘制

柱形图主要应用在日或月的时段分析页面中。柱形图的绘制效果如图 9.1 所示。

柱形图的绘制主要使用了两个方法。Graphics 类中的 DrawLine 方法和 FillRectangle 方法。Graphics 方法用于连接两个点的一条直线。使用 DrawLine 方法主要用来实现通过循环来绘制图像的背景网格，该网格用来区分各个时段和流量百分比。DrawLine 方法的语法格式如下：

```
public void DrawLine(Pen pen,int x1,int y1,int x2,int y2)
```

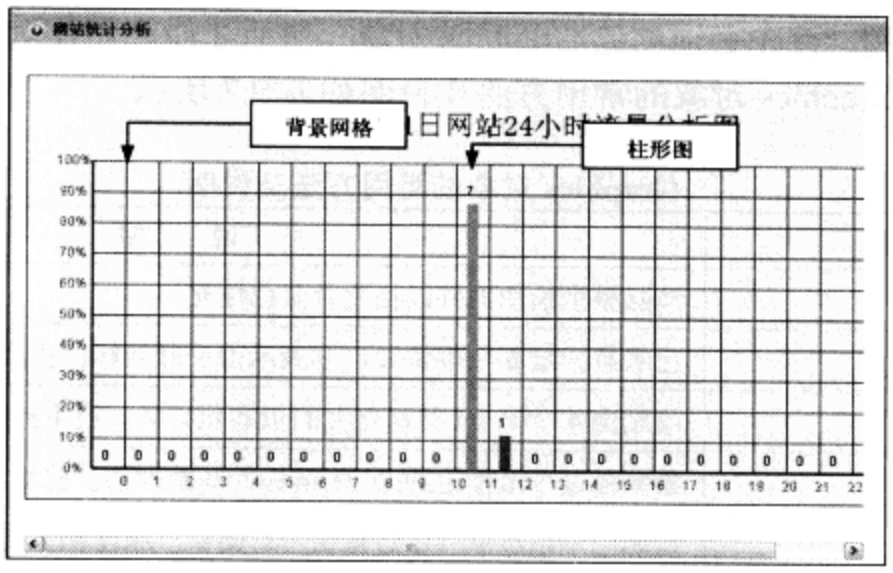


图 9.1　柱形图

pen：确定线条的颜色、宽度和样式。

x1：第 1 个点的 *x* 坐标。

y1：第 1 个点的 *y* 坐标。

x2：第 2 个点的 *x* 坐标。

y2：第 2 个点的 *y* 坐标。

FillRectangle 方法用于填充由一对坐标、一个宽度和一个高度所指定的矩形。使用 FillRectangle 方法主要用来实现表示各个时段访问人数的矩形。该矩形的高度是根据各时段的访问人数所占总访问人数的百分比而获得的。FillRectangle 方法的语法格式如下。

```
public void FillRectangle(Brush brush,int x,int y,int width,int height)
```

参数说明如下。

- brush：确定填充特性的 Brush。
- x：要填充的矩形的左上角的 *x* 坐标。
- y：要填充的矩形的左上角的 *y* 坐标。
- width：要填充的矩形的宽度。
- height：要填充的矩形的高度。

9.2.3　饼型图的绘制

饼型图主要应用在日或月的地域分析页面中。饼型图的绘制效果如图 9.2 所示。

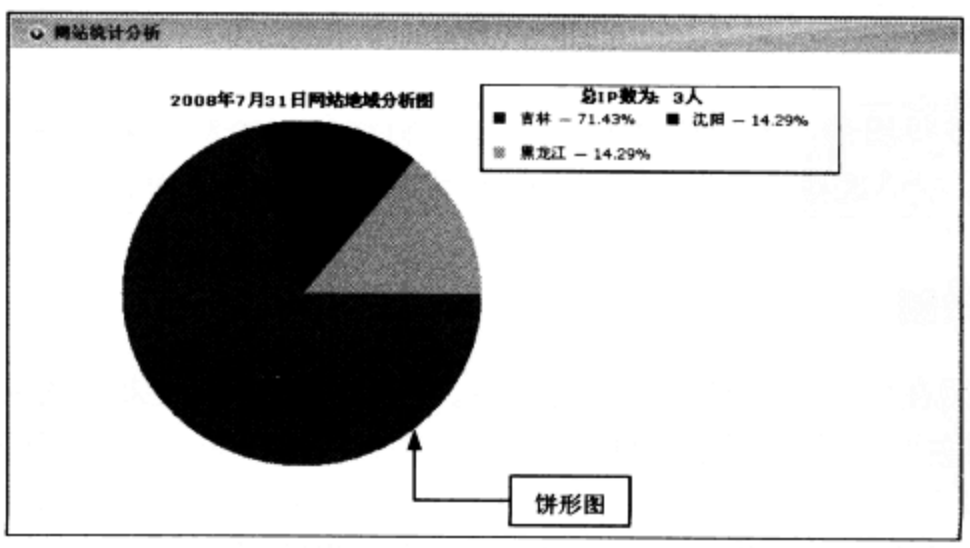


图 9.2　柱形图

在绘制图形时，主要使用了 2 个方法，即 Graphics 类中 FillPie 方法和 Graphics 类中

DrawString 方法。FillPie 方法用于填充由一对坐标、一个高度、一个宽度以及两条射线所指定的扇形区域。FilllPie 方法主要实现表示各地域的访问人数的饼型图。通过对各地域访问人数所占总访问人数的百分比而绘制各个扇形的大小。FillPie 方法的语法格式如下：

```
FillPie(Brush brush,float x,float y,float width,float height,float startAngle,float sweepAngle)
```

参数说明如下。

- brush：确定填充特性的 Brush。
- x：边框左上角的 x 坐标，该边框定义扇形区所属的椭圆。
- y：边框左上角的 y 坐标，该边框定义扇形区所属的椭圆。
- width：边框的宽度，该边框定义扇形区所属的椭圆。
- height：边框的高度，该边框定义扇形区所属的椭圆。
- startAngle：从 x 轴沿顺时针方向旋转到扇形区第 1 个边所测得的角度（以度为单位）。
- sweepAngle：从 startAngle 参数沿顺时针方向旋转到扇形区第 2 个边所测得的角度（以度为单位）。

DrawString 方法用于绘制指定位置的文字字符串。DrawString 方法主要实现提示各个扇形所表示的地域名称以及所占的百分比。DrawString 方法的语法格式如下：

```
public void DrawString (string s,Font font,Brush brush,float x,float y)
```

参数说明如下。

- s：要绘制的字符串。
- font：它定义字符串的文本格式。
- brush：它确定所绘制文本的颜色和纹理。
- x：所绘制文本的左上角的 x 坐标。
- y：所绘制文本的左上角的 y 坐标。

9.2.4 Global.asax 类统计访问人数

统计网站的访问人数主要是通过 Global.asax 类文件中的 session_Start 事件实现的。在 Global.asax 类文件中包含了对程序开始、程序结束、开始会话和结束会话的一些事件，在该文件中可以实现统计访问人数、统计在线人数等信息。若需添加 Global 文件，可以在“添加新项”窗口中选择“全局应用程序类”文件，该文件的默认名为 Global.asax，如图 9.3 所示。

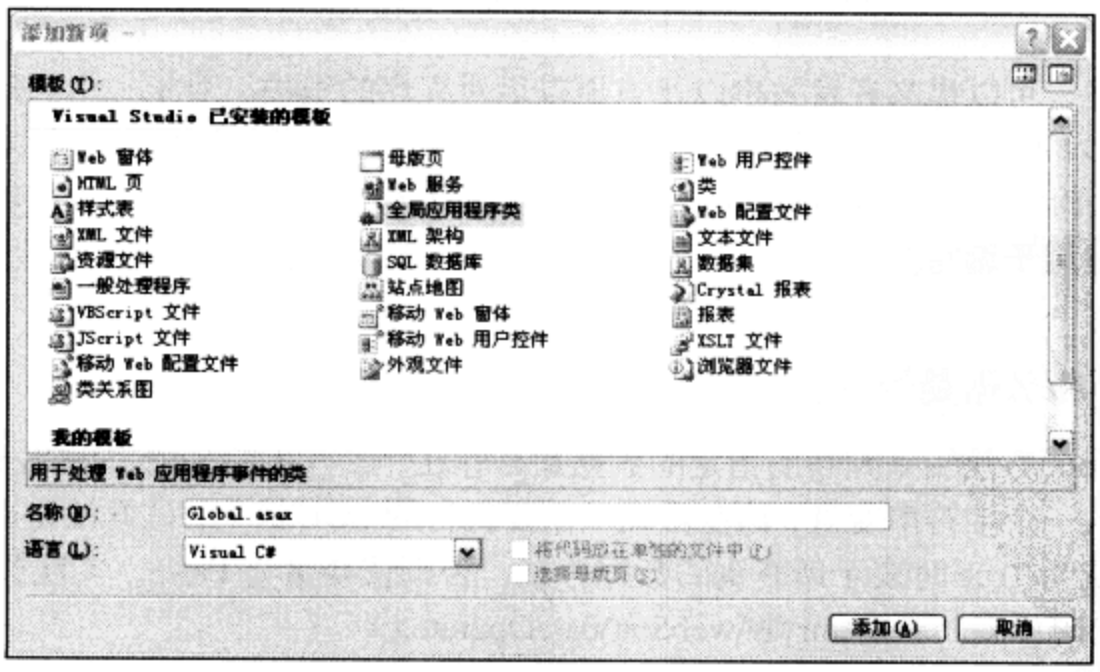


图 9.3　添加全局应用程序类

添加完成 Global 类文件后，在该文件中的 Session_Start 事件中将访问者的相应信息添加到

数据库中。Session_Start 事件将会在一个新用户访问应用程序 Web 站点时，而被触发。Session_Start 事件中的代码如下。

例程 1　代码位置：光盘\mr\09\webStat\ Global.asax

```
        void Session_Start(object sender, EventArgs e)
        {
            //在新会话启动时运行的代码
            //加锁
            Application.Lock();
            //创建SQL语句，查询当前访问的IP是否已被记录过，如果不存在说明未被记录过需要使用插入语句
            string sqlIsIP = "select count(*) from tb_dayStat where ip='" + Request.UserHostAddress + "' and days=" +
DateTime.Now.Day + " and   months= " + DateTime.Now.Month;
            if (!dataOperate.isData(sqlIsIP))
            {
                //获取操作系统
                int systemId = methodOperate.getSystem(Request.ServerVariables["HTTP_USER_AGENT"]);
                //获取浏览器
                int browserId = methodOperate.getBrowser(Request.ServerVariables["HTTP_USER_AGENT"]);
                //创建SQL语句，插入登录用户的信息
                string sqlIns = "insert into tb_dayStat(IP,sumNum,days,months,systemId,browserId) values('" +
Request.UserHostAddress + "',1," + DateTime.Now.Day + "," + DateTime.Now.Month + "," + systemId + "," + browserId + ")";
                //执行SQL语句
                dataOperate.exSql(sqlIns);
                //增加当前时段的访问人数
                dataOperate.insertHour();
            }
            else
            {
                //创建SQL语句，更新指定用户的访问量加一。
                string sqlUp = "update tb_dayStat set sumNum=sumNum+1 where ip='" + Request.UserHostAddress + "'";
                //执行SQL语句
                dataOperate.exSql(sqlUp);
                //增加当前时段的访问人数
                dataOperate.insertHour();
            }
            Application.UnLock();
        }
```

9.3　公共类的封装与设计

设计公共类，可以提高开发效率以及方便以后对程序的维护。对于一个好的程序来说，公共类是不可缺少的部分。在本程序中编写了两个公共类，这两个公共类分别为数据库操作类 dataOperate 和公共方法类 methodOperate。数据库操作类主要用于编写对数据库常用的一些操作。公共方法类用于编写在程序中比较常用的方法或日后需要修改的方法。下面将详细介绍公共类中的方法。

9.3.1　实现判断数据是否存在

自定义 isData 方法用来判断所指定的数据是否存在，该方法编写在 dataOperate 类中。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个布尔值变量，当该变量为 True 时表示所查询的数据存在，否则该变量为 False。实现代码如下。

例程 2　代码位置：光盘\mr\09\webStat\dataOperate.cs

```
public static bool isData(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
```

```
        //打开数据库连接
        con.Open();
        //创建Command对象
        SqlCommand com = new SqlCommand(sql, con);
        //获取ExecuteScalar方法所返回的值
        int ex = Convert.ToInt32(com.ExecuteScalar());
        //关闭数据库连接
        con.Close();
        //判断整型变量并返回相应的布尔值
        if (ex > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

9.3.2 实现返回指定列值

自定义 getTier 方法用于返回指定的列值，该方法编写在 dataOperate 类中。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个字符串变量，该字符串变量表示查询出的列值。实现代码如下。

例程 3 代码位置：光盘\mr\09\webStat\dataOperate.cs

```
    public static string getTier(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建DataAdapter对象
        SqlDataAdapter sda = new SqlDataAdapter(sql, con);
        //创建DataSet对象
        DataSet ds = new DataSet();
        //使用Fill方法填充DataSet对象
        sda.Fill(ds);
        //获取表中首行首列的数据
        string str = ds.Tables[0].Rows[0][0].ToString();
        //关闭数据库连接
        con.Close();
        return str;
    }
```

9.3.3 实现更新、插入、删除操作

自定义 exSql 方法用来实现对数据库的更新、插入和删除操作。该方法编写在 dataOperate 类中。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个布尔值变量，当执行成功是将返回 True，否则返回 False。实现代码如下。

例程 4 代码位置：光盘\mr\09\webStat\dataOperate.cs

```
    public static bool exSql(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开连接
        con.Open();
        //创建Command对象
        SqlCommand com = new SqlCommand(sql, con);
```

```
        //获取ExecuteNonQuery方法所返回的值
        int ex = com.ExecuteNonQuery();
        //关闭数据库连接
        con.Close();
        //判断所返回的值是否大于0，并返回相应的布尔值
        if (ex > 0)
        {
            return true;
        }
        else
            return false;
    }
```

9.3.4 实现返回表中所有数据

自定义 getRows 方法用来实现所查询表中的所有数据，该方法编写在 dataOperate 类中。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个 DataSet 对象，该对象中存储表中所有数据。实现代码如下。

例程 5 代码位置：光盘\mr\09\webStat\dataOperate.cs

```
    public static DataSet getRows(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据连接
        con.Open();
        //创建DataAdapter对象
        SqlDataAdapter sda = new SqlDataAdapter(sql, con);
        //创建DataSet对象
        DataSet ds = new DataSet();
        //通过Fill方法
        sda.Fill(ds);
        //关闭数据库连接
        con.Close();
        //返回DataSet对象
        return ds;
    }
```

9.3.5 实现更新或插入时段数据

自定义 insertHour 方法用来实现更新或插入时段信息表中的数据。该方法编写在 dataOperate 类中。实现代码如下。

例程 6 代码位置：光盘\mr\09\webStat\dataOperate.cs

```
    public static void insertHour()
        {
            //获取系统当前的月份
            string months = DateTime.Now.Month.ToString();
            //获取系统当前的日
            string days = DateTime.Now.Day.ToString();
            //创建SQL语句，用来查询指定的月日是否存在
            string sqlSel = "select count(*) from tb_hourStat where months=" + months + " and days=" + days;
            //判断是否存在
            if (!isData(sqlSel))
            {
                //不存在，使用insert语句插入新的时段计数数据
                string sqlInsert = "insert into tb_hourStat(months,days," + methodOperate.getTime() + ") values("+
months + "," + days + ",1) ";
                exSql(sqlInsert);
            }
            else
```

```
            {
                //存在，使用update语句更新时段计数数据
                string sqlUp = "update tb_hourStat set " + methodOperate.getTime() + "=" + methodOperate.getTime() +
"+1   where months = " + months + " and days=" + days;
                exSql(sqlUp);
            }
        }
```

9.3.6 实现执行存储过程

自定义 execProcedure 方法用来执行存储过程。该方法编写在 dataOperate 类中。调用该方法需要传入一个字符串变量，该变量表示需要查询的月份。实现代码如下。

例程 7 代码位置：光盘\mr\09\webStat\ataOperate.cs

```
public static void execProcedure(string months)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //打开数据库
    con.Open();
    //创建Command对象
    SqlCommand com = new SqlCommand("P_monthRegion", con);
    //设置存储过程
    com.CommandType = CommandType.StoredProcedure;
    //设置参数类型及大小
    SqlParameter parm = new SqlParameter("@months", SqlDbType.VarChar, 10);
    //设置参数值
    parm.Value = months;
    //添加参数
    com.Parameters.Add(parm);
    //执行存储过程
    com.ExecuteNonQuery();
    //关闭数据库连接
    con.Close();
}
```

9.3.7 实现返回当前时间字段

自定义 getTime 方法用来根据当前的时间返回对应表中的时间字段名。该方法编写在 methodOperate 类中。该方法将返回一个字符串变量，该变量表示时间字段名。实现代码如下。

例程 8 代码位置：光盘\mr\09\webStat\dataOperate.cs

```
public static string getTime()
    {
        string h = " ";
        //获取当前的时间进行判断
        switch (DateTime.Now.Hour)
        {
            case 0: h = "hour0"; break;
            case 1: h = "hour1"; break;
            case 2: h = "hour2"; break;
            case 3: h = "hour3"; break;
            case 4: h = "hour4"; break;
            case 5: h = "hour5"; break;
            case 6: h = "hour6"; break;
            case 7: h = "hour7"; break;
            case 8: h = "hour8"; break;
            case 9: h = "hour9"; break;
            case 10: h = "hour10"; break;
            case 11: h = "hour11"; break;
            case 12: h = "hour12"; break;
            case 13: h = "hour13"; break;
```

```
            case 14: h = "hour14"; break;
            case 15: h = "hour15"; break;
            case 16: h = "hour16"; break;
            case 17: h = "hour17"; break;
            case 18: h = "hour18"; break;
            case 19: h = "hour19"; break;
            case 20: h = "hour20"; break;
            case 21: h = "hour21"; break;
            case 22: h = "hour22"; break;
            case 23: h = "hour23"; break;
        }
        return h;
    }
```

9.3.8 实现返回操作系统类型

自定义 getSystem 方法用来返回操作系统的类型。该方法编写在 methodOperate 类中。调用该方法需要传入一个字符串变量，该变量表示所获取客户端用户的操作系统和浏览器的信息。该方法返回一个整型变量，该变量表示操作系统的类型。实现代码如下。

例程 9 代码位置：光盘\mr\09\webStat\dataOperate.cs

```
public static int getSystem(string strAgentInfo)
{
    if (strAgentInfo.Contains("NT 5.2"))
    {
        //表示Windows 2003系统
        return 1;
    }
    else if (strAgentInfo.Contains("NT 5.1"))
    {
        // 表示Windows XP系统
        return 2;
    }
    else if (strAgentInfo.Contains("NT 5"))
    {
        // 表示Windows 2000系统
        return 3;
    }
    else if (strAgentInfo.Contains("unix"))
    {
        //表示UNIX系统
        return 4;
    }
    else if (strAgentInfo.Contains("linux"))
    {
        //表示Linux系统
        return 5;
    }
    return 6;
}
```

9.3.9 实现返回浏览器类型

自定义 getBrowser 方法用来返回浏览器的类型。该方法编写在 methodOperate 类中。调用该方法需要传入一个字符串变量，该变量表示所获取客户端用户的操作系统和浏览器的信息。该方法返回一个整型变量，该变量表示浏览器的类型。实现代码如下。

例程 10 代码位置：光盘\mr\09\webStat\dataOperate.cs

```
public static int getBrowser(string strAgentInfo)
    {
        if (strAgentInfo.Contains("MSIE 6.0"))
```

```
        {
            //表示Internet Explorer 6.0浏览器
            return 1;
        }
        else if (strAgentInfo.Contains("MSIE 5.0"))
        {
            //表示Internet Explorer 5.0浏览器
            return 2;
        }
        else if (strAgentInfo.Contains("Firefox/3.0b2"))
        {
            //表示Firefox 3.0B2浏览器
            return 3;
        }
        return 4;
    }
```

9.4 网站统计的实现过程

9.4.1 统计概述设计

网站统计概述页面用来显示网站的基本信息和统计基本信息，其中包括网站名称、站点类型、站点地区、pv 访问量等信息。网站统计概述页面如图 9.4 所示。

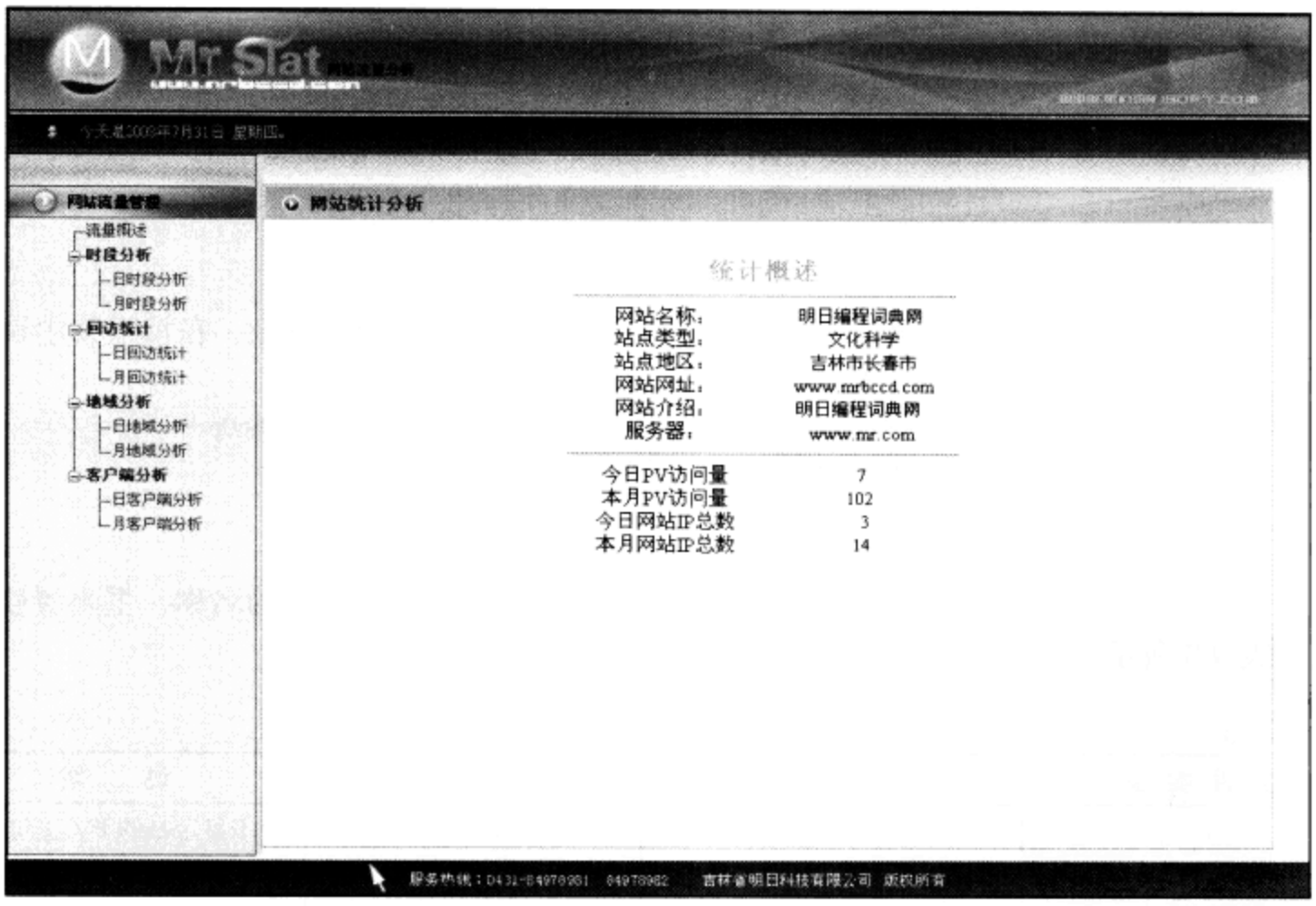


图 9.4 网站统计概述页面

1. 前台页面设计

网站统计分析的显示分为两个框架，左框架用来显示导航，右框架用来显示相应的信息。首先介绍一下框架的设计。

● 框架设计

（1）创建 1 个 Web 窗体，命名为 Default.aspx。

（2）在页面中添加 1 个表格，该表格用于布局。表格分为左右两列，在左侧列中添加 1 个 TreeView 控件该用于导航。该控件的前台代码如下。

例程 11 代码位置：光盘\mr\09\webStat\ Default.aspx

```
<asp:TreeView ID="TreeView1" runat="server" Style="text-align: left" ShowLines="True">
<Nodes>
<asp:TreeNode Text="流量概述" Value="流量概述" Target="mainFrame" NavigateUrl="summarizeStat.aspx">
</asp:TreeNode>
<asp:TreeNode Text="时段分析" Value="时段分析" SelectAction="Expand">
<asp:TreeNode Text="日时段分析" Value="日时段分析" Target="mainFrame" NavigateUrl="dayStat.aspx?id=day">
</asp:TreeNode>
<asp:TreeNode Text="月时段分析" Value="月时段分析" Target="mainFrame" NavigateUrl="dayStat.aspx?id=month">
</asp:TreeNode>
</asp:TreeNode>
<asp:TreeNode Text="回访统计" Value="回访统计" SelectAction="Expand">
<asp:TreeNode Text="日回访统计" Value="日回访统计" Target="mainFrame" NavigateUrl="IpStat.aspx?id=day">
</asp:TreeNode>
<asp:TreeNode Text="月回访统计" Value="月回访统计" Target="mainFrame" NavigateUrl="IpStat.aspx?id=month">
</asp:TreeNode>
</asp:TreeNode>
<asp:TreeNode Text="地域分析" Value="地域分析" SelectAction="Expand">
<asp:TreeNode Text="日地域分析" Value="日地域分析" Target="mainFrame" NavigateUrl="regionStat.aspx?id=day">
</asp:TreeNode>
<asp:TreeNode Text="月地域分析" Value="月地域分析" Target="mainFrame" NavigateUrl="regionStat.aspx?id=month">
</asp:TreeNode>
</asp:TreeNode>
<asp:TreeNode Text="客户端分析" Value="客户端分析" SelectAction="Expand">
<asp:TreeNode Text="日客户端分析" Value="日客户端分析" Target="mainFrame" NavigateUrl="clientInfo.aspx?id=day">
</asp:TreeNode>
<asp:TreeNode Text="月客户端分析" Value="月客户端分析" Target="mainFrame" NavigateUrl="clientInfo.aspx?id=month">
</asp:TreeNode>
</asp:TreeNode>
</Nodes>
</asp:TreeView>
```

（3）在表格的右侧列中，通过使用 iframe 标记创建一个内嵌浮动框架。在该框架中显示相应的信息。设置该框架的 name 参数为 mainFrame。框架代码如下：

```
<iframe id="Iframe1" name="mainFrame" src="summarizeStat.aspx" style="width: 860px;
    height: 450px"></iframe>
```

● 统计概述设计

（1）创建 1 个 Web 窗体，命名为 summarizeStat.aspx。

（2）在该页面中添加 1 个表格，该表格用于布局。页面中主要控件的名称、主要属性及说明，如表 9.8 所示。

表 9.8 控件的名称、主要属性及说明

控件类型	控件名称	主要属性	说明
标准/Label 控件	labDaySum	均为默认值	用于显示日的 PV 访问量
	labMonthSum	均为默认值	用于显示月的 PV 访问量
	labDayIp	均为默认值	用于显示日的 IP 总数
	labMonthIp	均为默认值	用于显示月的 IP 总数

2. 后台代码编写

在该页的页面加载事件中调用自定义 bind 方法。在自定义 bind 方法通过使用 SQL 语句，

查询相应的信息，并将查询到的信息通过 Label 控件显示在页面中。实现代码如下。

例程 12 代码位置：光盘\mr\09\webStat\ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        bind();      //调用自定义bind方法
    }
protected void bind()
    {
        //创建SQL语句，查询查询今日的总访问人数（PV）
        string sqlDaySum = "select sum(" + methodOperate.strDaySum() + ") as sumDay from tb_hourStat where
months=" + DateTime.Now.Month.ToString() + " and days=" + DateTime.Now.Day.ToString();
        //显示今日总访问人数（PV）
        labDaySum.Text = dataOperate.getTier(sqlDaySum);
        //创建SQL语句，查询查询本月的总访问人数（PV）
        string sqlMonthSum = "select " + methodOperate.strMonthSum() + " as sumMonth from tb_hourStat where
months= " + DateTime.Now.Month.ToString();
        //显示本月总访问人数（PV）
        labMonthSum.Text = dataOperate.getTier(sqlMonthSum);
        //创建SQL语句，查询今日总IP数
        string sqlDayIp = "select count(*) from tb_dayStat where   months= " + DateTime.Now.Month + " and days=" +
DateTime.Now.Day;
        //显示今日总IP数
        labDayIp.Text = dataOperate.getTier(sqlDayIp);
        //创建SQL语句，查询本月总IP数
        string sqlMonthIp = "select count(*) from tb_dayStat where   months= " + DateTime.Now.Month;
        //显示本月总IP数
        labMonthIp.Text = dataOperate.getTier(sqlMonthIp);
    }
```

9.4.2 日或月时段分析设计

日或月时段分析页面用来显示，本日或本月 24 小时访问量的分布情况。管理员可以在该页面中了解每小时的访问人数。日或月时段分析页面如图 9.5 和图 9.6 所示。

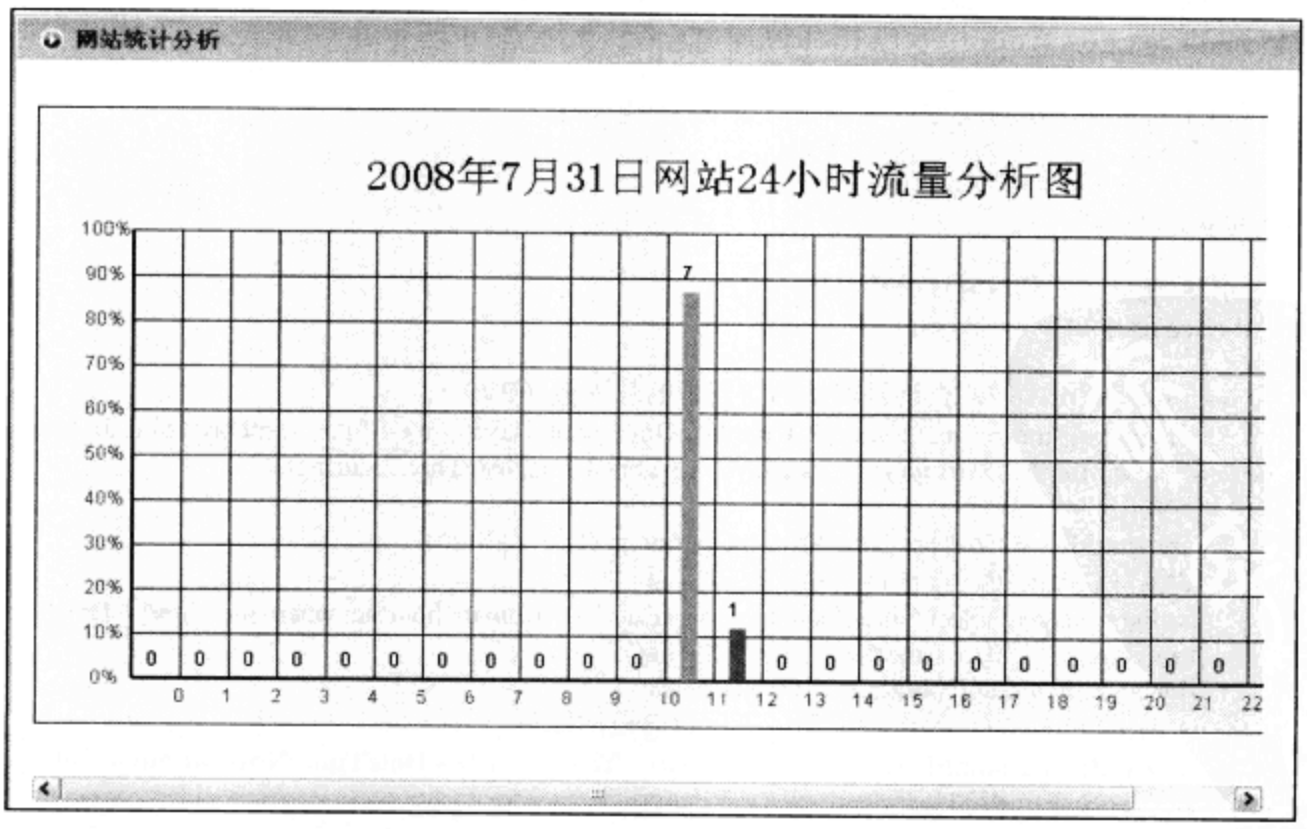


图 9.5 日时段分析页面

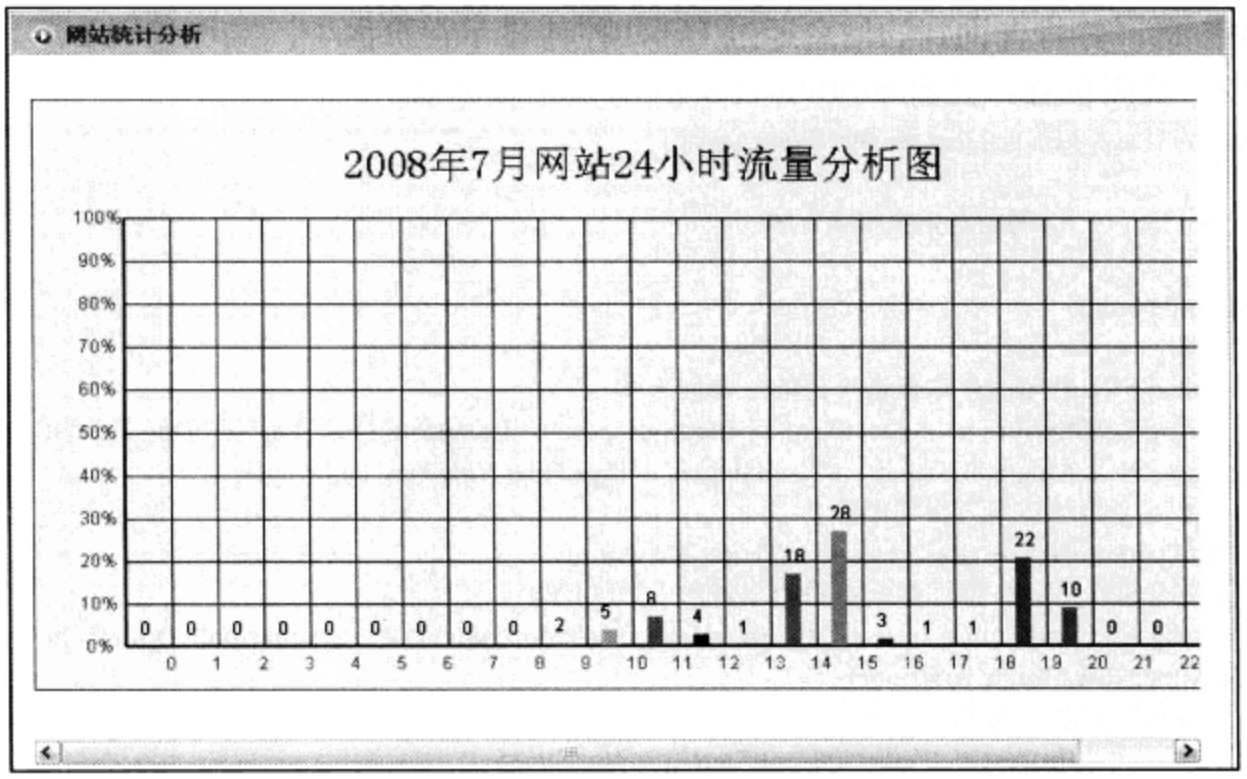


图 9.6　月时段分析页面

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 dayStat.aspx。

（2）在页面中添加 1 个 Image 控件，该控件用来显示以柱形图表示的各时段访问信息。

2. 后台代码编写

在页面加载事件中调用自定义 statFulx 方法，该方法用来显示日时段分析或月时段分析图形。在 statFulx 方法中，首先通过传入的参数判断用户是想显示日时段分析还是月时段分析。如果选择的是日时段分析，首先将通过 SQL 语句计算出今日访问的总人数并保存到变量中，在通过 SQL 语句查询各时段访问的人数并保存到 DataSet 对象中，最后将调用自定义 CreateImage 方法生成图片显示在页面中。如果选择的是月时段分析，代码实现的步骤和日时段分析一样只是对 SQL 的查询条件进行了修改。实现代码如下。

例程 13　代码位置：光盘\mr\09\webStat\dayStat.aspx

```
protected void Page_Load(object sender, EventArgs e)
    {
        //调用自定义statFulx方法，显示日时段或月时段信息
        statFulx();
    }
protected void statFulx()
    {
        //判断查看日还是月时段分析
        if (Request["id"] == "day")
        {
            //编写SQL语句，查询日网站的访问量的总人数（PV）
            string sqlSum = "select sum(" + methodOperate.strDaySum() + ") as sumDay from tb_hourStat where
months=" + DateTime.Now.Month.ToString() + " and days=" + DateTime.Now.Day.ToString();
            //调用
            int sumFlux = Convert.ToInt32(dataOperate.getTier(sqlSum));
            //编写SQL语句，查询日网站的时段信息
            string sqlday = "select " + methodOperate.strDay() + " from tb_hourStat where months=" + DateTime.Now.
Month.ToString() + " and days=" + DateTime.Now.Day.ToString();
            DataSet ds = dataOperate.getRows(sqlday);
            //调用自定义方法绘制柱型图，表示日时段的流量
            CreateImage(sumFlux, ds, DateTime.Now.Year + "年" + DateTime.Now.Month + "月" + DateTime.
Now.Day + "日网站24小时流量分析图");
        }
        else
```

```
            {
                //编写SQL语句，查询月网站的访问量的总人数（PV）
                string sqlMonthSum = "select " + methodOperate.strMonthSum() + " as sumMonth from tb_hourStat
where months= " + DateTime.Now.Month.ToString();
                int sumFlux = Convert.ToInt32(dataOperate.getTier(sqlMonthSum));
                //编写SQL语句，查询月网站的时段信息
                string sqlMonth = "select " + methodOperate.strMonth() + " from tb_hourStat where months= " +
DateTime.Now.Month.ToString();
                DataSet ds = dataOperate.getRows(sqlMonth);
                //调用自定义方法绘制柱形图，表示月时段的流量
                CreateImage(sumFlux, ds, DateTime.Now.Year + "年" + DateTime.Now.Month + "月网站24小时流量分析图");
            }
        }
```

自定义 CreateImage 方法用绘制柱形图并将图形显示在页面中。调用该方法需要传入 3 个参数，这 3 个参数分别为整型变量表示日或月的总访问人数、DataSet 对象表示日或月的各时段的信息、字符串变量表示显示的标题。在该方法中主要使用 Graphics 对象来绘制柱形图，柱形图的高度是根据每个时段访问人数占总人数的百分比而绘制的。实现代码如下。

例程 14 代码位置：光盘\mr\09\webStat\ dayStat.aspx

```
    /// <summary>
        /// 自定义方法绘制柱形图，表示日或月时段的流量
        /// </summary>
        /// <param name="sumFlux">int类型的参数，表示日或月的总访问人数</param>
        /// <param name="ds">DataSet对象，存储日或月的时段信息</param>
        /// <param name="title">标题文字</param>
        private void CreateImage(int sumFiux, DataSet ds, string title)
        {
            //随机生成24个颜色
            ArrayList colors = new ArrayList();
            Random rnd = new Random();
            for (int len = 0; len < ds.Tables[0].Columns.Count; len++)
            {
                colors.Add(new SolidBrush(Color.FromArgb(rnd.Next(255), rnd.Next(255), rnd.Next(255))));
            }
            //设置宽和高
            int height = 400, width = 830;
            //初始化并指定图像的大小
            Bitmap image = new System.Drawing.Bitmap(width, height);
            //创建Graphics类对象
            Graphics g = Graphics.FromImage(image);
            try
            {
                //清空图片背景色
                g.Clear(Color.White);
                //设置字体
                Font font = new System.Drawing.Font("Arial", 9, FontStyle.Regular);
                Font font1 = new System.Drawing.Font("宋体", 20, FontStyle.Regular);
                //
                System.Drawing.Drawing2D.LinearGradientBrush brush = new System.Drawing.Drawing2D.
LinearGradientBrush(new Rectangle(0, 0, image.Width, image.Height), Color.Blue, Color.Blue, 1.2f, true);
                //绘制一个矩形
                g.FillRectangle(Brushes.WhiteSmoke, 0, 0, width, height);
                //设置标题文字的颜色
                Brush brush1 = new SolidBrush(Color.Blue);
                //绘制标题文字
                g.DrawString(title, font1, brush1, new PointF(200, 30));
                //画图片的边框线
                g.DrawRectangle(new Pen(Color.Blue), 0, 0, image.Width - 1, image.Height - 1);
                Pen mypen = new Pen(brush, 1);
                //绘制10个横向线条
                int x = 90;
                for (int i = 0; i < 24; i++)
                {
```

```
        g.DrawLine(mypen, x, 80, x, 370);
        x = x + 30;
    }
    Pen mypen1 = new Pen(Color.Blue, 2);
    g.DrawLine(mypen1, x - 750, 80, x - 750, 370);
    //绘制10个纵向线条
    int y = 80;
    for (int i = 0; i < 10; i++)
    {
        g.DrawLine(mypen, 60, y, 780, y);
        y = y + 29;
    }
    g.DrawLine(mypen1, 60, y, 780, y);
    //设置x轴的小时刻度
    string[] n ={"0","1","2","3","4","5","6","7","8","9",
    "10","11","12","13","14","15","16","17","18","19",
    "20","21","22","23"};
    x = 84;
    for (int i = 0; i < 24; i++)
    {
        //将小时刻度绘制到指定的位置上
        g.DrawString(n[i].ToString(), font, Brushes.Red, x, 374); //设置文字内容及输出位置
        x = x + 30;
    }
    //设置y轴的百分比
    String[] m = {"100%", " 90%", " 80%", " 70%", " 60%", " 50%", " 40%", " 30%",
                  " 20%", " 10%", "   0%"};
    y = 72;
    for (int i = 0; i < 11; i++)
    {
        //将百分比绘制到指定的位置上
        g.DrawString(m[i].ToString(), font, Brushes.Red, 25, y); //设置文字内容及输出位置
        y = y + 29;
    }
    //创建一个数组用来存放，没小时访问人数的百分比
    int[] Count = new int[24];
    int j = 0;
    int number = sumFlux;
    for (j = 0; j < 23; j++)
    {
        string st = ds.Tables[0].Rows[0][j].ToString();
        Count[j + 1] = Convert.ToInt32(ds.Tables[0].Rows[0][j].ToString()) * 100 / number;
    }
    //将数组第1个位，存放23点到0点的访问人数百分比
    Count[0] = Convert.ToInt32(ds.Tables[0].Rows[0][j].ToString()) * 100 / number;
    //显示柱状效果
    x = 70;
    //设置表示23点到0点访问人数百分比柱形的颜色
    SolidBrush mbrush = (SolidBrush)colors[23];
    //绘制23点到0点的访问人数提示文字
    g.DrawString(ds.Tables[0].Rows[0][23].ToString(), font, Brushes.Black, x - 3, (370 - Count[0] * 29 / 10) - 20);
    //绘制表示23点到0点访问人数百分比柱形
    g.FillRectangle(mbrush, x, 370 - Count[0] * 29 / 10, 10, Count[0] * 29 / 10);
    for (int i = 0; i < 23; i++)
    {
        //设置两个柱形之间的颜色
        x = x + 30;
        //设置柱形的颜色
        SolidBrush mybrush = (SolidBrush)colors[i];
        //绘制柱形
        g.FillRectangle(mybrush, x, 370 - Count[i + 1] * 29 / 10, 10, Count[i + 1] * 29 / 10);
        //绘制访问人数提示文字
        g.DrawString(ds.Tables[0].Rows[0][i].ToString(), font, Brushes.Black, x - 3, (370 - Count[i + 1] * 29 / 10) - 20);
    }
    //将图片保存到指定路径下
```

```
            image.Save(Server.MapPath("Images") + "\\Z.jpeg");
            //使用Image控件显示图片
            Image1.ImageUrl = "Images\\Z.jpeg?id=" + rnd.Next(9999);
        }
        finally
        {
            g.Dispose();
            image.Dispose();
        }
    }
```

9.4.3 日或月回访统计设计

日或月回访统计页面用来显示本日或本月的用户对本网站回访的情况，管理员可以在该页面中了解本网站受欢迎的程度。日或月回访统计页面如图 9.7 和图 9.8 所示。

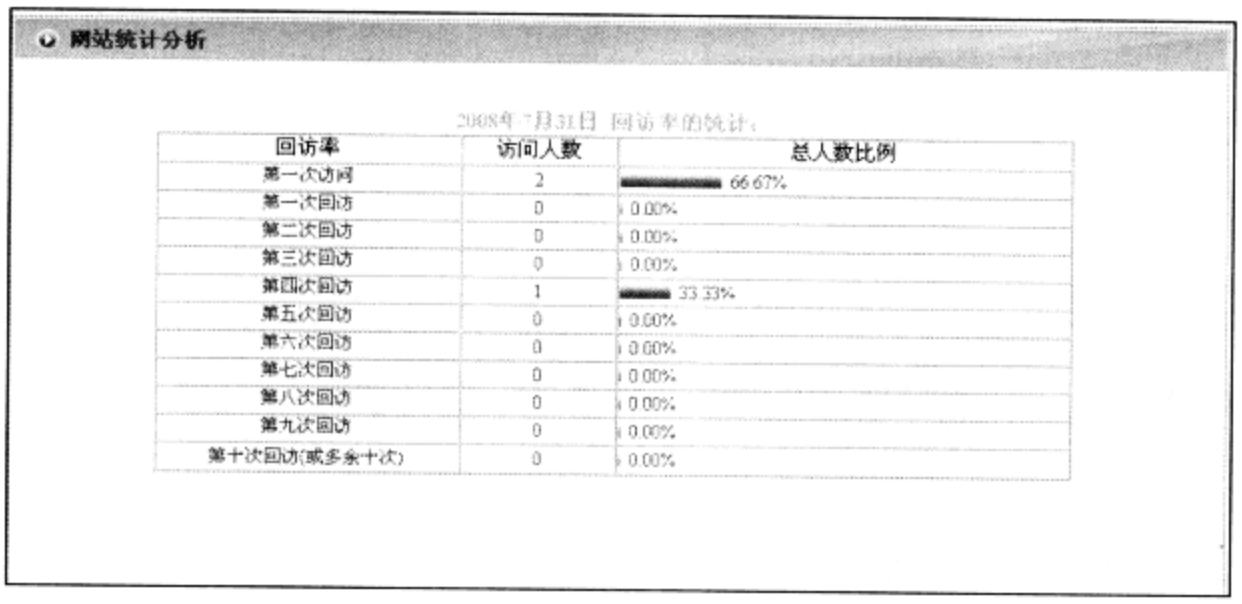


图 9.7 日回访统计页面

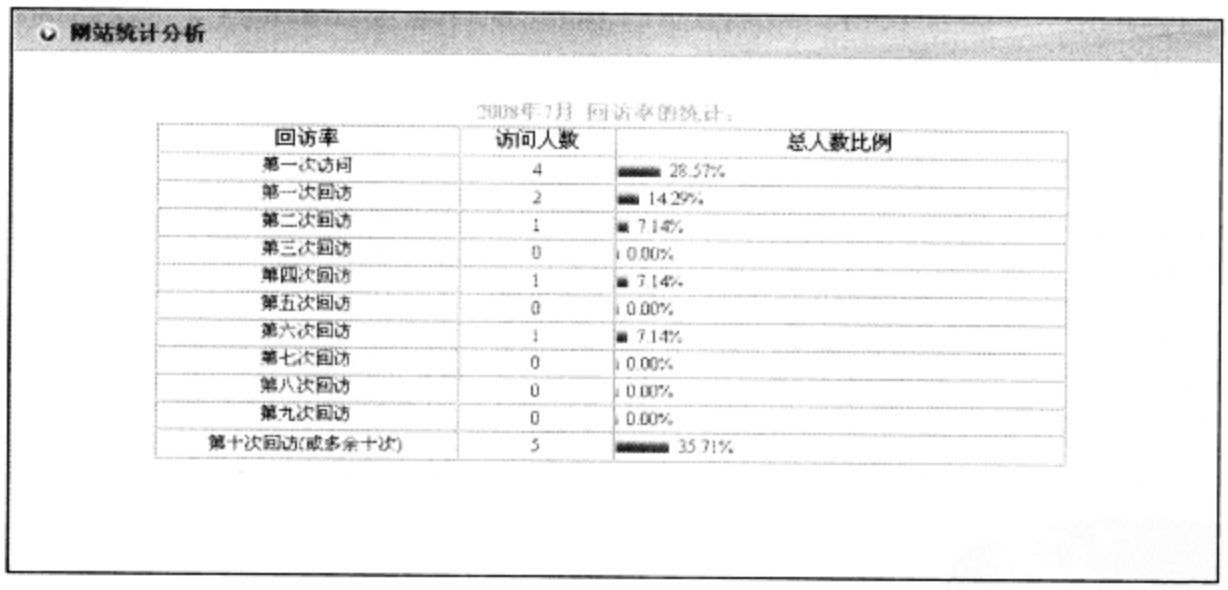


图 9.8 月回访统计页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 IpStat.aspx。

(2) 在该页面中添加 2 个控件。第 1 个控件为 Label 控件，该控件用来显示日期。第 2 个控件为 Literal 控件，该控件用来显示在后台编写的 HTML 代码。

2. 后台代码编写

在页面的加载事件中调用自定义 statIp 方法，自定义 statIp 方法用来显示日回访率或月回访率。在该方法中，首先判断用户是要显示日回访率还是月回访率。如果是日回访率将根据当前日和月设置 SQL 语句的查询条件，并设置 Label 控件显示当前的年月日。如果要显示的是月回

访率将根据当前的月来设置 SQL 语句的查询条件，并设置 Label 控件显示当前的年月。最后都将调用自定义 createTable 方法将回访率以表格的形式显示。实现代码如下。

例程 15 代码位置：光盘\mr\09\webStat\ IpStat.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        statIp();
    }
protected void statIp()
    {
        //判断查看的是日还是月的回访率
        if (Request["id"] == "day")
        {
            //设置查询日的条件
            string sqlSumNum = "and months=" + DateTime.Now.Month + " and days=" + DateTime.Now.Day;
            string sqlCount = " months= " + DateTime.Now.Month + " and days=" + DateTime.Now.Day;
            Label1.Text = DateTime.Now.Year + "年" + DateTime.Now.Month + "月"+DateTime.Now.Day+"日";
            //调用自定义方法，用来创建表格并以百分比显示日回访率
            createTable(sqlSumNum, sqlCount);
        }
        else
        {
            //设置查询月的条件
            string sqlSumNum = "and months=" + DateTime.Now.Month;
            string sqlCount = " months= " + DateTime.Now.Month;
            Label1.Text = DateTime.Now.Year+"年"+DateTime.Now.Month+"月";
            //调用自定义方法，用来创建表格并以百分比显示月回访率
            createTable(sqlSumNum, sqlCount);
        }
    }
```

自定义 createTable 方法以表格的形式显示日回访率或月回访率。调用该方法需要传入 2 个字符串型参数。第 1 个参数表示查询访问人数的条件。第 2 个参数表示查询总访问人数的条件。在该方法中通过使用循环添加表格。最后使用 Literal 控件显示在页面中。实现代码如下。

例程 16 代码位置：光盘\mr\09\webStat\ IpStat.aspx.cs

```
    /// <summary>
        /// 以表格的形式显示日或月的回访率
        /// </summary>
        /// <param name="sqlSumNum">字符串变量，表示查询指定访问人数的条件</param>
        /// <param name="sqlCount">字符串变量，表示查询总访问人数的条件</param>
        protected void createTable(string sqlSumNum, string sqlCount)
        {
            //设置字符串，使用html代码创建表格，该表格为3列分别显示回访率、访问人数、总人数比例
            string str = "<body style=text-align:center><table border=1 width='600' cellpandding=0 cellspacing=0
text-align:center > ";
            str = str + "<tr><td align='center' width='200'>回访率</td><td align='center' width='100'>访问人数</td><td
width='300'>总人数比例</td></tr>";
            //通过for循环，循环添加10行数据
            for (int i = 0; i < 11; i++)
            {
                str = str + "<tr>";
                //for循环，循环添加每列数据
                for (int j = 0; j < 4; j++)
                {
                    //编写SQL语句，查询日或月的访问总人数
                    string sqlc = "select count(sumNum) from tb_dayStat where" + sqlCount;
                    double count = Convert.ToDouble(dataOperate.getTier(sqlc));
                    //判断是否是第10行
                    if (i != 10)
                    {
                        //编写SQL语句，查询指定访问次数的总人数
                        string sql = "select count(sumNum) from tb_dayStat where sumNum=" + (i + 1) + sqlSumNum;
                        double sumNum = Convert.ToDouble(dataOperate.getTier(sql));
```

```
                    //判断是否是第1列，如果是第1列写回访率次数
                    if (j == 0)
                    {
                        //判断是否是第1行，第1行写第1次访问
                        if (i == 0)
                        {
                            str = str + "<td align='center' >第一次访问</td>";
                        }
                        else
                        {
                            str = str + "<td align='center' >第" + number(i - 1) + "次回访</td>";
                        }
                    }
                    else
                        if (j == 1)
                        {
                            str = str + "<td align='center' >" + sumNum + "</td>";
                        }
                        else
                            if (j == 2)
                            {
                                double result = (sumNum / count) * 100;
                                str = str + "<td align='left' ><img src=Images/bar1.gif height=10 width= " +
result.ToString("#0.00") + " %>  " + result.ToString("#0.00") + "%</td>";
                            }
                }
                else
                {
                    //创建SQL语句，查询访问总数大于11的总人数
                    string sql = "select count(sumNum) from tb_dayStat where sumNum >=" + (i) + sqlSumNum;
                    int sumNum = Convert.ToInt32(dataOperate.getTier(sql));
                    if (j == 0)
                    {
                        str = str + "<td align='center' >第十次回访(或多余十次)</td>";
                    }
                    else
                        if (j == 1)
                        {
                            str = str + "<td align='center' >" + sumNum + "</td>";
                        }
                        else
                            if (j == 2)
                            {
                                double result=(sumNum/count)*100;
                                str = str + "<td align='left' ><img src=Images/bar1.gif height=10 width= " +
result.ToString("#0.00") + " %>  " + result.ToString("#0.00") + "%</td>";
                            }
                    }
                }
                str = str + "</tr>";
            }
            str = str + "</table></body>";
            //将表格通过Literal控件显示在页面中
            Literal1.Text = str;
        }
```

自定义 number 方法用来将小写数字转换为大写数字。调用该方法需要传入一个整型变量，该变量表示小写数字。在该方法中定义一个整型数组，在数组中存放大写的数字。根据传入的小写数字作为数组的下标，将下标对应的大写数字返回。实现代码如下。

例程 17 代码位置：光盘\mr\09\webStat\ IpStat.aspx.cs

```
        protected string number(int i)
        {
            string[] num = { "一", "二", "三", "四", "五", "六", "七", "八", "九" };
            return num[i];
        }
```

9.4.4 日或月地域分析设计

日或月地域分析页面用来显示，本日或本月访问网站的用户所分布的地域。管理员可以在该页面中了解到哪里的网友访问本网站最多。日或月地域分析页面如图 9.9 和图 9.10 所示。

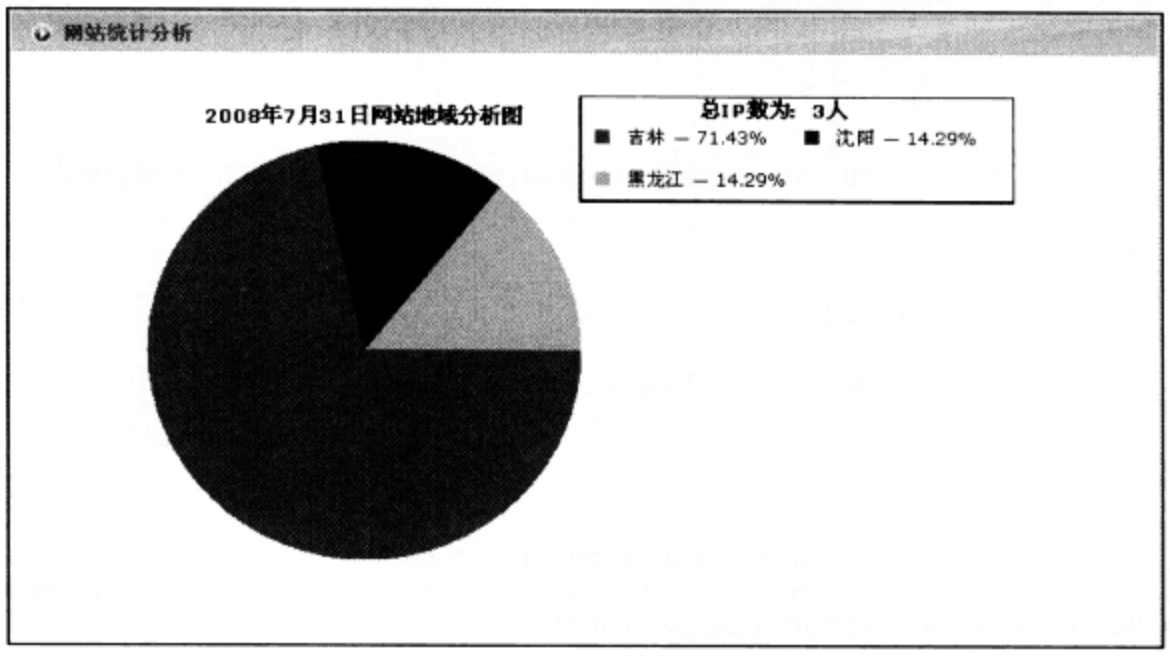


图 9.9 日地域分析页面

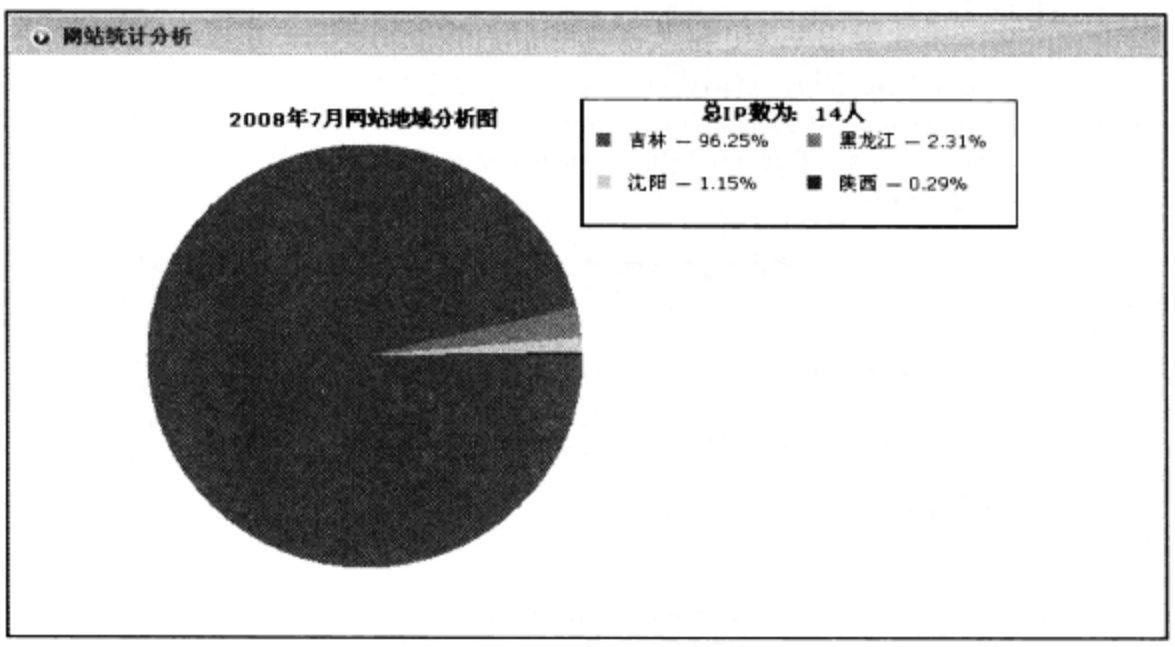


图 9.10 月地域分析页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 regionStat.aspx。

(2) 在该页面中添加 1 个 Image 控件，该控件用来显示日区域分析或月区域分析的饼型图。

2. 后台代码编写

在页面加载的事件中调用自定义 region 方法，该方法用于显示日区域分析或月区域分析的饼型图。在 region 方法中，首先根据传入的参数判断要显示日地域分析图还是月地域分析图。再通过使用 SQL 语句获取相应的参数。最后调用自定义 createImage 方法并传入获取的参数。实现代码如下。

例程 18 代码位置：光盘\mr\09\webStat\ regionStat.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        //调用自定义region方法，显示日区域分析或月区域分析
        region();
    }
```

```
    private void region()
        {
            //判断是显示日区域分析还是月区域分析
            if (Request["id"] == "day")
            {
                //创建SQL语句，该语句用来查询区域视图表中今日的总访问人数
                string sqlt = "select sum(sumNum) from view_region where days=" + DateTime.Now.Day + " and
months=" + DateTime.Now.Month;
                int Total = Convert.ToInt32(dataOperate.getTier(sqlt));
                //创建SQL语句，该语句用来查询区域视图表中今日的IP总数
                string sqlc = "select count(*) from view_region where days=" + DateTime.Now.Day + " and months=" +
DateTime.Now.Month;
                int count = Convert.ToInt32(dataOperate.getTier(sqlc));
                //创建SQL语句，该语句用来查询区域视图表中今日的所有信息
                string sqld = "select * from view_region where days=" + DateTime.Now.Day + " and months=" +
DateTime.Now.Month + " order by sumNum desc ";
                DataSet ds = dataOperate.getRows(sqld);
                //创建SQL语句，该语句用来查询日用户信息表中今日IP总数
                string sqlDayIp = "select count(*) from tb_dayStat where   months= " + DateTime.Now.Month + " and
days=" + DateTime.Now.Day;
                int ipCount = Convert.ToInt32(dataOperate.getTier(sqlDayIp));
                //调用自定义createImage方法，绘制日地域分析饼型图
                createImage(Total, count, ds, DateTime.Now.Year + "年" + DateTime.Now.Month + "月" + DateTime.
Now.Day + "日网站地域分析图", ipCount);
            }
            else
            {
                //调用execPprocedure方法，执行存储过程该存储过程用来创建月地域分析视图
                dataOperate.execPprocedure(DateTime.Now.Month.ToString());
                //创建SQL语句，该语句用来查询月地域分析视图表中总访问人数
                string sqlt = "select sum(sumNum) from view_monthRegion";
                int Total = Convert.ToInt32(dataOperate.getTier(sqlt));
                //创建SQL语句，该语句用来查询月地域分析视图表中IP总数
                string sqlc = "select count(*) from view_monthRegion";
                int count = Convert.ToInt32(dataOperate.getTier(sqlc));
                //创建SQL语句，该语句用来查询月地域分析视图表中所有信息
                string sqld = "select * from view_monthRegion order by sumNum desc ";
                DataSet ds = dataOperate.getRows(sqld);
                //创建SQL语句，该语句用来查询日用户信息表中本月的IP总数
                string sqlMonthIp = "select count(*) from tb_dayStat where   months= " + DateTime.Now.Month;
                int ipCount = Convert.ToInt32(dataOperate.getTier(sqlMonthIp));
                //调用自定义createImage方法，绘制月地域分析饼型图
                createImage(Total, count, ds, DateTime.Now.Year + "年" + DateTime.Now.Month + "月网站地域分析
图", ipCount);
            }
        }
```

自定义 createImage 方法用来绘制日地域分析或月地域分析的饼型图。调用该方法需要传入 5 个参数。在该方法中通过使用 Graphics 对象来绘制饼型图和矩形图。实现代码如下。

例程 19 代码位置：光盘\mr\09\webStat\ regionStat.aspx.cs

```
        /// <summary>
        /// 用于绘制日地域分析或月地域分析的饼型图
        /// </summary>
        /// <param name="Total">整型变量，该变量表示总访问人数</param>
        /// <param name="count">整型变量，该变量表示日地域分析视图或月地域分析视图的IP总数</param>
        /// <param name="ds">DataSet对象，用于存储日第一分析视图或月地域分析视图的所有信息</param>
        /// <param name="title">字符串变量，该变量表示标题</param>
        /// <param name="ipCount">整型变量，该变量表示日或月的IP总数</param>
        private void createImage(int Total, int count, DataSet ds, string title, int ipCount)
        {
            //创建一个整型变量，该变量用于设置颜色提示的换行
```

```
int iloop;
//设置字体，fonttitle为主标题的字体
Font fontlegend = new Font("verdana", 9), fonttitle = new Font("verdana", 10, FontStyle.Bold);
//背景宽
int width = 600;
int bufferspace = 15;
//颜色提示文字的高度
int legendheight = fontlegend.Height * (count + 1) + bufferspace;
//
int titleheight = fonttitle.Height + bufferspace;
//白色背景高
int height = width + legendheight + titleheight + bufferspace;
int pieheight = 300;
Rectangle pierect = new Rectangle(0, titleheight, width - 300, pieheight);
//加上各种随机色
ArrayList colors = new ArrayList();
Random rnd = new Random();
for (iloop = 0; iloop < ds.Tables[0].Rows.Count; iloop++)
    colors.Add(new SolidBrush(Color.FromArgb(rnd.Next(255), rnd.Next(255), rnd.Next(255))));
//创建一个bitmap实例
Bitmap objbitmap = new Bitmap(width, 350);
Graphics objgraphics = Graphics.FromImage(objbitmap);
//画一个白色背景
objgraphics.FillRectangle(new SolidBrush(Color.White), 0, 0, width, height);
//画一个亮黄色背景
objgraphics.FillRectangle(new SolidBrush(Color.LightYellow), pierect);
SolidBrush blackbrush = new SolidBrush(Color.Black);
//以下为画饼图(有几行row画几个)
float currentdegree = 0.0f;
for (iloop = 0; iloop < ds.Tables[0].Rows.Count; iloop++)
{
    //设置饼型的弧度
    float pi = Convert.ToSingle(ds.Tables[0].Rows[iloop]["sumNum"]) / Total * 360;
    //绘制一个饼型
    objgraphics.FillPie((SolidBrush)colors[iloop], pierect, currentdegree, Convert.ToSingle(pi));
    //设置饼型的起始位置
    currentdegree += Convert.ToSingle(ds.Tables[0].Rows[iloop]["sumNum"]) / Total * 360;
}
//以下为生成标题给出颜色地域的提示
StringFormat stringFormat = new StringFormat();
stringFormat.Alignment = StringAlignment.Center;
stringFormat.LineAlignment = StringAlignment.Center;
objgraphics.DrawString(title, fonttitle, blackbrush,
    new Rectangle(0, 0, width - 300, titleheight), stringFormat);
//列出各字段与数目
objgraphics.DrawRectangle(new Pen(Color.Black, 2), 300, 0, width - 300, legendheight);
for (iloop = 0; iloop < count; iloop++)
{
    //判断是否以显示两个标题提示
    if (iloop % 2 == 0)
    {
        //绘制一个矩形
        objgraphics.FillRectangle((SolidBrush)colors[iloop], 310, fontlegend.Height * iloop + 7 + fonttitle.
Height, 10, 10);
        //获取地域的名称
        string st = ds.Tables[0].Rows[iloop]["placeName"].ToString();
        //获取某个IP的访问量
        double sd = Convert.ToDouble(ds.Tables[0].Rows[iloop]["sumNum"]);
        //计算百分比
        float ss = Convert.ToSingle(Math.Round(sd * 100 / Total, 2));
        //绘制文字该问题用于提示颜色表示的地域名称及该颜色所占的百分比
        objgraphics.DrawString(st + " — " + ss + "%", fontlegend, blackbrush,330, fontlegend.Height *
iloop + 5 + fonttitle.Height);
```

```
                    }
                    else
                    {
                        //绘制一个矩形
                        objgraphics.FillRectangle((SolidBrush)colors[iloop], 455, fontlegend.Height * (iloop - 1) + 7 +
fonttitle.Height, 10, 10);
                        string st = ds.Tables[0].Rows[iloop]["placeName"].ToString();
                        double sd = Convert.ToDouble(ds.Tables[0].Rows[iloop]["sumNum"]);
                        float ss = Convert.ToSingle(Math.Round(sd * 100 / Total, 2));
                        //绘制文字该问题用于提示颜色表示地域名称及该颜色所占的百分比
                        objgraphics.DrawString(st + " — " + ss + "%", fontlegend, blackbrush, 475, fontlegend.Height *
(iloop - 1) + 5 + fonttitle.Height);
                    }
                }
                //图像总的高度一行字体的高度，即是最底行的字体高度(height - fontlegend.Height )
                objgraphics.DrawString("总IP数为： " + ipCount + "人", fonttitle, blackbrush, 380, 2);
                objbitmap.Save(Server.MapPath("Images") + "\\r.jpeg");
                Image1.ImageUrl = "Images\\r.jpeg?id=" + rnd.Next(9999);
                objgraphics.Dispose();
                objbitmap.Dispose();
            }
```

9.4.5 日或月客户端分析设计

日或月客户端分析页面用来显示，本日或本月访问网站的用户客户端的情况，包括用户所使用的浏览器信息和用户所使用的操作系统信息。日或月客户端分析页面如图 9.11 和图 9.12 所示。

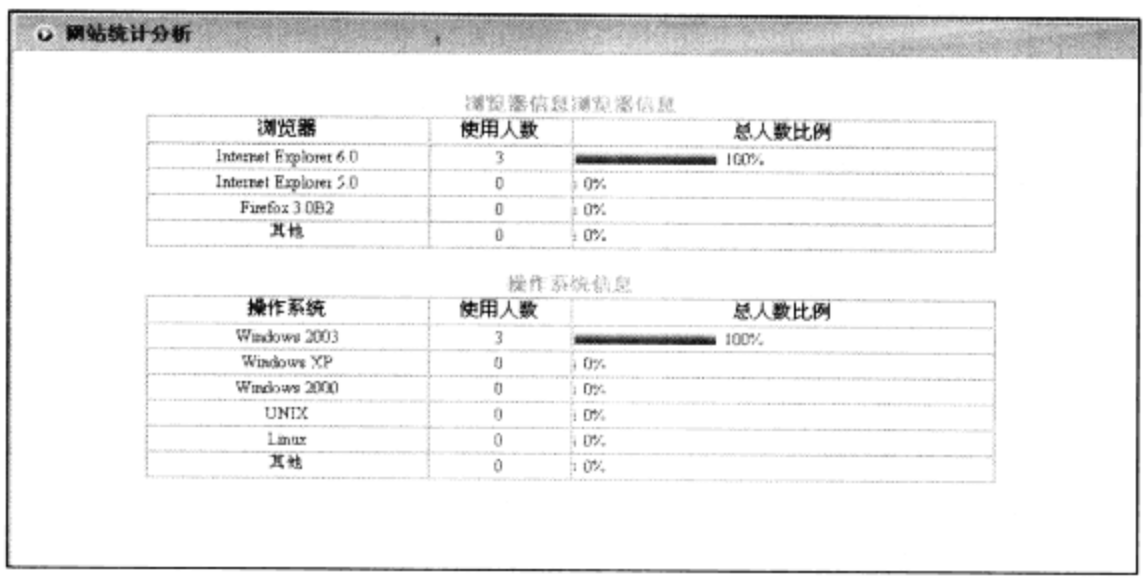


图 9.11 日客户端分析页面

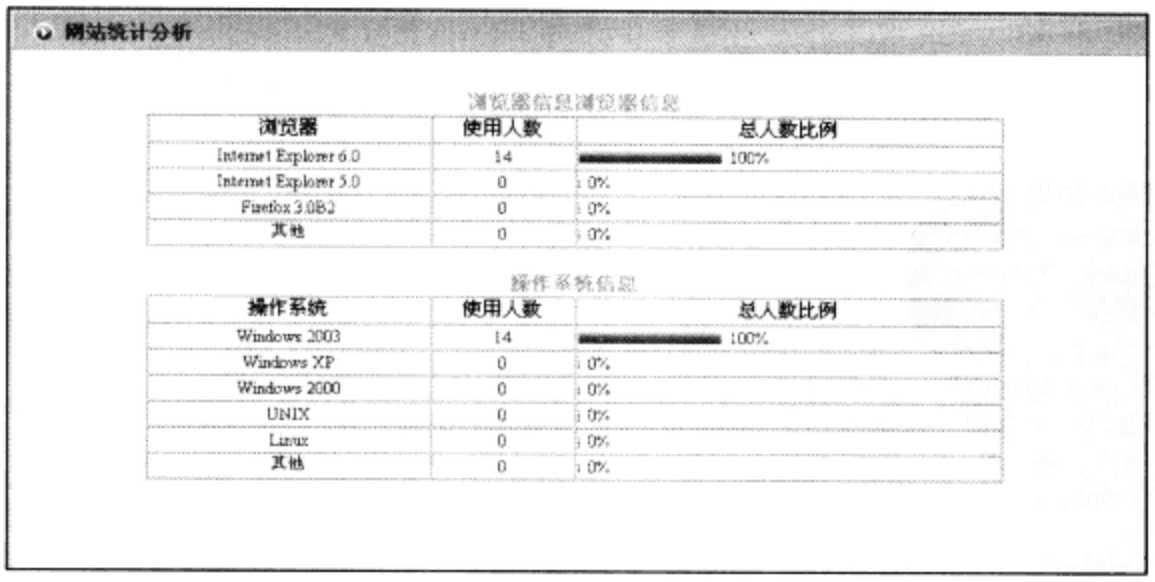


图 9.12 月客户端分析页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 clientInfo.aspx。

(2) 在该页面中添加 2 个 Literal 控件，这 2 个 Literal 控件分别用于显示客户端分析中的，浏览器分析表格和操作系统分析表格。

2. 后台代码编写

在页面加载事件中调用自定义 clientStat 方法，该方法用来编写日客户端分析表格或月客户端分析表格。在自定义 clientStat 方法中，首先判断是需要显示日客户端分析还是月客户端分析，再设置查询条件，最后将调用 createTable 方法编写分析表格并显示在页面中。实现代码如下。

例程 20 代码位置：光盘\mr\09\webStat\clientInfo.aspx

```
protected void Page_Load(object sender, EventArgs e)
    {
        //调用自定义clientStat方法，该方法用来编写日客户端分析或月客户端分析的表格
        clientStat();
    }
    public void clientStat()
    {
        //声明一个string类型变量用来存储SQL条件
        string condition = "";
        //判断查看是日还是月
        if (Request["id"] == "day")
        {
            //设置查询日的条件
            condition = " days= " + DateTime.Now.Day + " and months= " + DateTime.Now.Month;
            //调用自定义方法显示客户端信息
            createTable(condition);
        }
        else
            if (Request["id"] == "month")
            {
                //设置查询月的条件
                condition = " months= " + DateTime.Now.Month;
                //调用自定义方法显示客户端信息
                createTable(condition);
            }
    }
```

自定义 createTable 方法用来编写分析表格，调用该方法需要传入一个字符串变量，该变量表示查询条件。在该方法中将编写 2 个表格这 2 个表格分别显示浏览器分析和客户端分析，最后将 2 个表格分别显示在 2 个 Literal 控件上。实现代码如下。

例程 21 代码位置：光盘\mr\09\webStat\clientInfo.aspx

```
/// <summary>
/// 编写操作系统分析表格和浏览器分析表格
/// </summary>
/// <param name="condition">字符串变量，该变量表示查询语句中的日或月的查询条件</param>
    public void createTable(string condition)
     {
         //获取浏览器数据
         string sqlBrowser = "select * from tb_browser";
         DataSet Browser_ds = dataOperate.getRows(sqlBrowser);
         //获取操作系统数据
         string sqlSystem = "select * from tb_system";
         DataSet System_ds = dataOperate.getRows(sqlSystem);
         //创建一个数组，该数组用来存放，每种操作系统的使用人数
         int[] systems = new int[6];
         for (int i = 0; i < 6; i++)
         {
             string sql = "select count(*) from tb_dayStat where" + condition + " and systemId=" + (i + 1);
             systems[i] = Convert.ToInt32(dataOperate.getTier(sql));
         }
         //创建一个数组，该数组用来存放，每种浏览器的使用人数
         int[] browsers = new int[4];
         for (int j = 0; j < 4; j++)
```

```
            {
                string sql = "select count(*) from tb_dayStat where " + condition + " and browserId=" + (j + 1);
                browsers[j] = Convert.ToInt32(dataOperate.getTier(sql));
            }
            //编写SQL语句查询出今日访问的总IP数
            string sqlCount = "select count(*) from tb_dayStat where " + condition;
            int count = Convert.ToInt32(dataOperate.getTier(sqlCount));
            //创建string类型的变量，该变量用来存储html代码编写的表格。
            string str = "<body style=text-align:center><table border=1 width='600' cellpandding=0 cellspacing=0
text-align:center > ";
            //设置表格有3列和每列的宽
            str = str + "<tr><td align='center' width='200'>浏览器</td><td align='center' width='100'>使用人数</td><td
width='300'>总人数比例</td></tr>";
            //for循环添加每行每列数据
            for (int i = 0; i < Browser_ds.Tables[0].Rows.Count; i++)
            {
                str = str + "<tr>";
                for (int j = 0; j < 3; j++)
                {
                    //判断是否为第1列，并添加数据
                    if (j == 0)
                    {
                        str = str + "<td>" + Browser_ds.Tables[0].Rows[i]["browserName"] + "</td>";
                    }
                    //判断是否为第2列，并添加数据
                    else if (j == 1)
                    {
                        str = str + "<td>" + browsers[i] + "</td>";
                    }
                    //判断是否为第3列，并添加数据
                    else if (j == 2)
                    {
                        //根据计算出来的百分数来显示图片的宽
                        str = str + "<td align='left'> <img src=Images/bar1.gif height=10 width= " + browsers[i] * 100 /
count + " %>  " + browsers[i] * 100 / count + "%</td>";
                    }
                }
                str = str + "</tr>";
            }
            str = str + "</table></body>";
            //显示浏览器统计的表格
            Literal1.Text = str;
            //创建string类型的变量，该变量用来存储html代码编写的表格
            string strSystem = "<body style=text-align:center><table border=1 width='600' cellpandding=0 cellspacing=0
text-align:center > ";
            strSystem = strSystem + "<tr><td align='center' width='200'>操作系统</td><td align='center' width='100'>使
用人数</td><td width='300'>总人数比例</td></tr>";
            for (int i = 0; i < System_ds.Tables[0].Rows.Count; i++)
            {
                strSystem = strSystem + "<tr>";
                for (int j = 0; j < 3; j++)
                {
                    if (j == 0)
                    {
                        strSystem = strSystem + "<td>" + System_ds.Tables[0].Rows[i]["systemName"] + "</td>";
                    }
                    else if (j == 1)
                    {
                        strSystem = strSystem + "<td>" + systems[i] + "</td>";
                    }
                    else if (j == 2)
                    {
                        strSystem = strSystem + "<td align='left'> <img src=Images/bar1.gif height=10 width= " +
systems[i] * 100 / count + " %>  " + systems[i] * 100 / count + "%</td>";
                    }
```

```
            }
            strSystem = strSystem + "</tr>";
        }
        strSystem = strSystem + "</table></body>";
        Literal2.Text = strSystem;
    }
```

9.5 网站打包与发布

网站开发完成后最终的目地是将其发布到 Internet 上，以便用户浏览访问。实现的网站发布可以使用两种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍下使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

（1）首先打开要发布的网站，在菜单栏中选择“生成”选项，在弹出的快捷方式菜单中选择“发布网站”选项，如图 9.13 所示。

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，单击“确定”按钮网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具需要选择“...”路径按钮，如图 9.14 所示。

图 9.13 选择“发布网站”选项

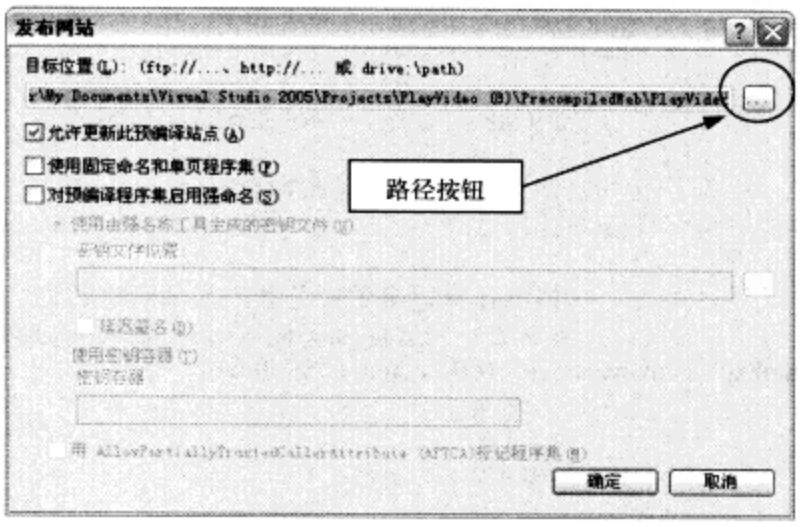


图 9.14 选择路径按钮

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选项中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码如图 9.15 所示。填写完毕后选择“打开”按钮将会返回“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

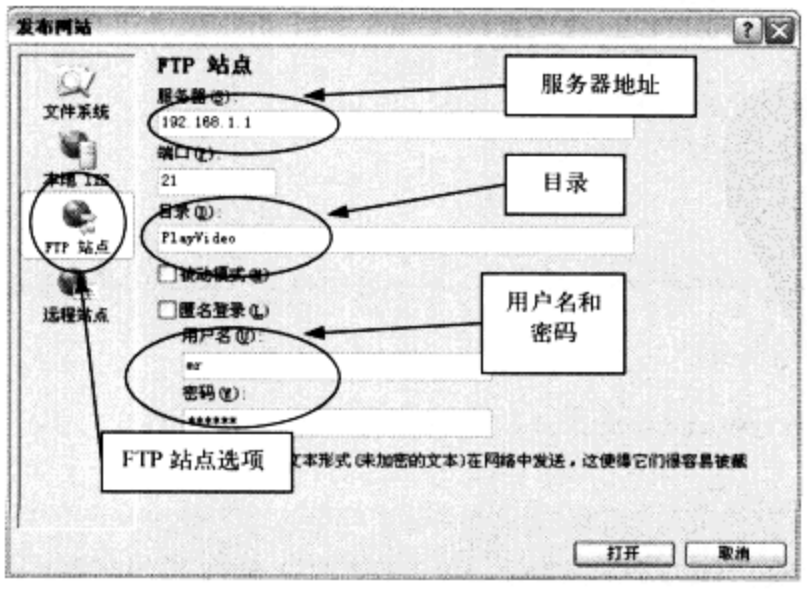


图 9.15 FTP 站点选项

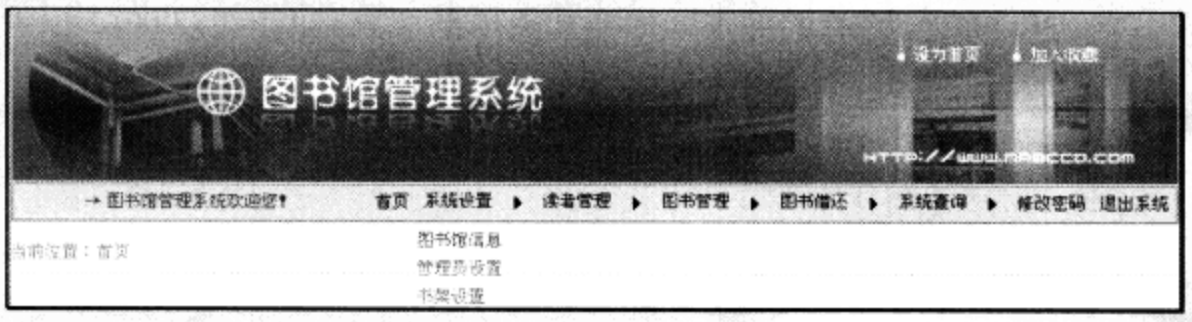
图书馆管理系统（权限分配）

第 10 章

实例位置：光盘\mr\10\

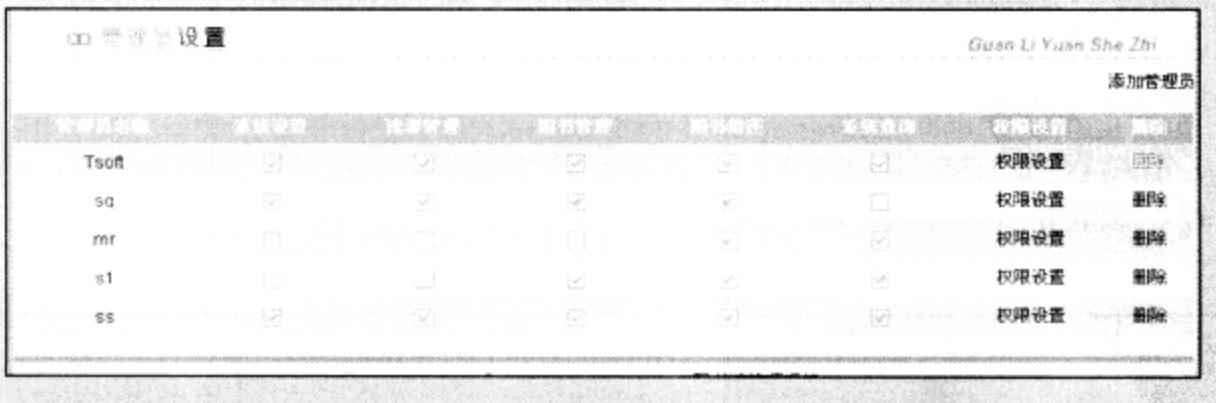
随着计算机应用的普及，计算机已经成为人们生活中不可缺少的一部分。使用计算机对图书馆进行管理也大大减轻了管理员的工作。读者通过使用计算机进行查询图书，可以在很短的时间内查询自己需要的图书。管理员通过使用计算机来对图书进行借阅登记，使用计算机可以准确地计算出图书的归还时间和扣除金额等。在图书馆中使用图书馆管理系统可以解决人工管理带来的许多弊端。通过本章，读者可以学到以下内容。

- Menu 菜单动态编辑



- 图书借阅功能

- 管理员设置



10.1 图书馆管理系统（权限分配模块）概述

10.1.1 功能概述

图书馆管理系统可以解决中小型图书馆借阅、查询、管理等基本功能。在本程序中主要考虑的是权限分配问题。权限分配在程序中是很重要的一部分。在本程序中权限分为 3 种，即读者权限、管理员权限和超级管理员权限。读者的权限只拥有查询功能。管理员的权限是超级管理给予分配的。而超级管理员拥有所有的权限。权限的拥有主要体现在菜单栏中，管理员拥有的权限显示在菜单栏中，不拥有的权限在菜单栏中不显示。在本程序中还有图书借阅、图书续借和图书归还功能。管理员可以通过使用这些功能来实现对图书借阅方面的操作。

10.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建一个名为 db_statInfo 的数据库，在该数据库中创建 10 张表。下面将详细介绍这 10 张表的数据结构及用途。

- 管理员登录表

管理员登录表（tb_user）用来记录管理员的登录信息。该表的结构如表 10.1 所示。

表 10.1　　tb_user 表的表结构

字　段	类　型	长　度	说　明
userId	int	4	自动增长编号
userName	varchar	50	用于记录管理员登录名
userPwd	varchar	50	用于记录管理员登录密码
isSuper	bit	1	用于记录是否为超级管理员

- 权限信息表

权限信息表（tb_admSet）用来保存每个管理员所拥有的权限信息。该表的结构如表 10.2 所示。

表 10.2　　tb_admSet 表的表结构

字　段	类　型	长　度	说　明
userName	varchar	80	用于记录管理员的登录名
systemSet	char	10	用于记录系统设置权限信息
readerManage	char	10	用于记录读者管理权限信息
bookManage	char	10	用于记录图书管理权限信息
bookBorrow	char	10	用于记录图书借阅权限信息
systemSearch	char	10	用于记录图书查询权限信息

- 图书借阅信息表

图书借阅信息表（tb_bookBorrow）用来保存图书借阅时间、图书归还时间、图书条形码、读者条形码等。该表的结构如表 10.3 所示。

表 10.3　　tb_ bookBorrow 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长编号
bookBarcode	varchar	50	用于记录图书的条形码
bookName	varchar	50	用于记录图书的名称

续表

字　段	类　型	长　度	说　明
borrowTime	datetime	8	用于记录借阅的时间
returnTime	datetime	8	用于记录图书归还的时间
readerBarCode	varchar	50	用于记录读者的条形码
readerName	varchar	50	用于记录读者的名称

- 书架信息表

书架信息表（tb_bookcase）用来保存书架名称及编号信息。该表的结构如表 10.4 所示。

表 10.4　　tb_ bookcase 表的表结构

字　段	类　型	长　度	说　明
bookcaseID	int	4	用于记录书架编号
bookcaseName	varchar	80	用于记录书架名称

- 图书信息表

图书信息表（tb_bookInfo）用来保存图书的条形码、库存、作者、出版社等信息。该表的结构如表 10.5 所示。

表 10.5　　tb_ bookInfo 表的表结构

字　段	类　型	长　度	说　明
bookBarCode	varchar	100	用于记录图书条形码
bookName	varchar	100	用于记录图书名称
bookType	int	4	用于记录图书类型编号
bookcase	int	4	用于记录书架编号
bookConcern	varchar	100	用于记录出版社名称
author	varchar	80	用于记录图书作者
price	money	8	用于记录图书价格
stock	int	4	用于记录图书库存
borrowSum	int	4	用于记录图书借阅次数

- 图书类型信息表

图书类型信息表（tb_bookType）用来保存图书的类型、图书可借阅天数、图书租金等信息。该表的结构如表 10.6 所示。

表 10.6　　tb_ bookType 表的表结构

字　段	类　型	长　度	说　明
typeID	int	4	用于记录图书类型编号
typeName	varchar	50	用于记录图书类型名称
borrowDay	int	4	用于记录图书可以借阅天数
hire	int	4	用于记录图书的租金
lagMoney	int	4	用于记录图书滞纳金

- 图书馆信息表

图书馆信息表（tb_library）用来保存图书馆的详细信息。该表的结构如表 10.7 所示。

表 10.7　　tb_ library 表的表结构

字　段	类　型	长　度	说　明
libraryName	varchar	80	用于记录图书馆名称
curator	varchar	80	用于记录馆长
tel	varchar	100	用于记录图书馆电话
address	varchar	200	用于记录图书馆地址
email	varchar	100	用于记录图书馆电子邮件地址
net	varchar	200	用于记录图书馆网站
upbuildTime	datetime	8	用于记录图书馆建馆时间
remark	varchar	500	用于记录图书馆备注信息

- 菜单栏信息表

菜单栏信息表（tb_menuInfo）用来保存菜单栏的文本、跳转路径、权限名称等信息。该表的结构如表 10.8 所示。

表 10.8　　tb_ menuInfo 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长编号
menuName	varchar	50	用于记录权限的名称
menuText	varchar	50	用于记录菜单栏显示的文本
menuUrl	varchar	50	用于记录菜单栏跳转的路径
menuParent	bit	1	用于记录是否为父菜单栏

- 读者信息表

读者信息表（tb_readerInfo）用来保存读者条形码、读者姓名、读者登录密码等信息。该表的结构如表 10.9 所示。

表 10.9　　tb_ readerInfo 表的表结构

字　段	类　型	长　度	说　明
readerBarCode	varchar	50	用于记录读者的条形码
readerPass	varchar	50	用于记录读者登录密码
readerName	varchar	50	用于记录读者姓名
sex	char	10	用于记录读者性别
readerType	varchar	50	用于记录读者类型编号
certificateType	varchar	50	用于记录读者证件类别
certificate	varchar	50	用于记录读者证件号
tel	varchar	50	用于记录读者联系电话
email	varchar	50	用于记录读者电子邮件地址
remark	varchar	500	用于记录读者备注
money	money	8	用于记录读者拥有金额

- 读者类型信息表

读者类型信息表（tb_readerType）用来保存读者类型名称信息、读者类型可借阅图书数量

信息。该表的结构如表 10.10 所示。

表 10.10　　tb_ readerType 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动增长编号
Type	varchar	50	用于记录读者类型名称
Num	varchar	50	用于记录读者类型可借阅图书数量

为了方便在程序中进行查询和保证数据的统一性，在该数据库中创建了一个视图和一个删除型触发器。视图和触发器的详细介绍如下。

- view_bookInfo 视图

视图 view_bookInfon 用来保存图书的详细信息，其中包括图书类型名称和图书书架名称。该视图是图书信息表（tb_bookInfo）、图书类型信息表（tb_bookType）、书架信息表（tb_bookcase）3 张表通过连接获得的。创建视图的代码如下：

```
create view view_bookInfo
as
select * from (tb_bookInfo as a inner join tb_bookType as b on a.bookType=b.typeID ) inner join tb_bookcase as c on
a.bookcase=c.bookcaseID
go
```

- delUser 删除型触发器

delUser 删除型触发器是用来保证数据的统一性。例如，需要删除某一个管理员的信息，就必须保证删除管理员登录表中的管理员信息和权限信息表中的管理员权限信息。实现这个功能也可以通过两条 SQL 语句来实现，但是为了提高执行效率建议使用删除型触发器。触发器可以看做一个存储过程。它与存储过程的区别在于，存储过程通过存储过程名称被调用，而触发器通过触发事件将被调用。删除型触发器是在删除某表中的数据而被触发的。DelUser 删除型触发器编写在权限信息表中（tb_admSet）。该触发器的实现代码如下：

```
create trigger delUser on tb_admSet
for delete
as
delete from tb_user where userName=(select userName from deleted)
```

10.2　图书馆管理系统（权限分配模块）关键技术

10.2.1　Menu 菜单动态编辑

在该程序中是使用 Menu 控件来显示权限信息的，Menu 控件样式与 Windows 应用程序中菜单栏类似。在 Windows 程序中比较常见。Menu 控件具有两种显示模式：静态模式和动态模式。静态显示意味着 Menu 控件始终是完全展开的。整个结构都是可视的，用户可以单击任何部位。而在动态显示的菜单中，只有用户将鼠标指定在父节点上时才会显示其子菜单项。在本程序中 Menu 菜单的效果如图 10.1 所示。

管理员设有的权限在 Menu 控件中将不会显示，而 Menu 控件是根据数据库中所保存的权限信息动态创建的。动态创建 Menu 菜单主要是通过使用 MenuItem 对象来实现。MenuItem 对象可以编辑 Menu 控件中的所有菜单项。MenuItem 对象中有 3 个比较常用的属性。3 个属性分别如下。

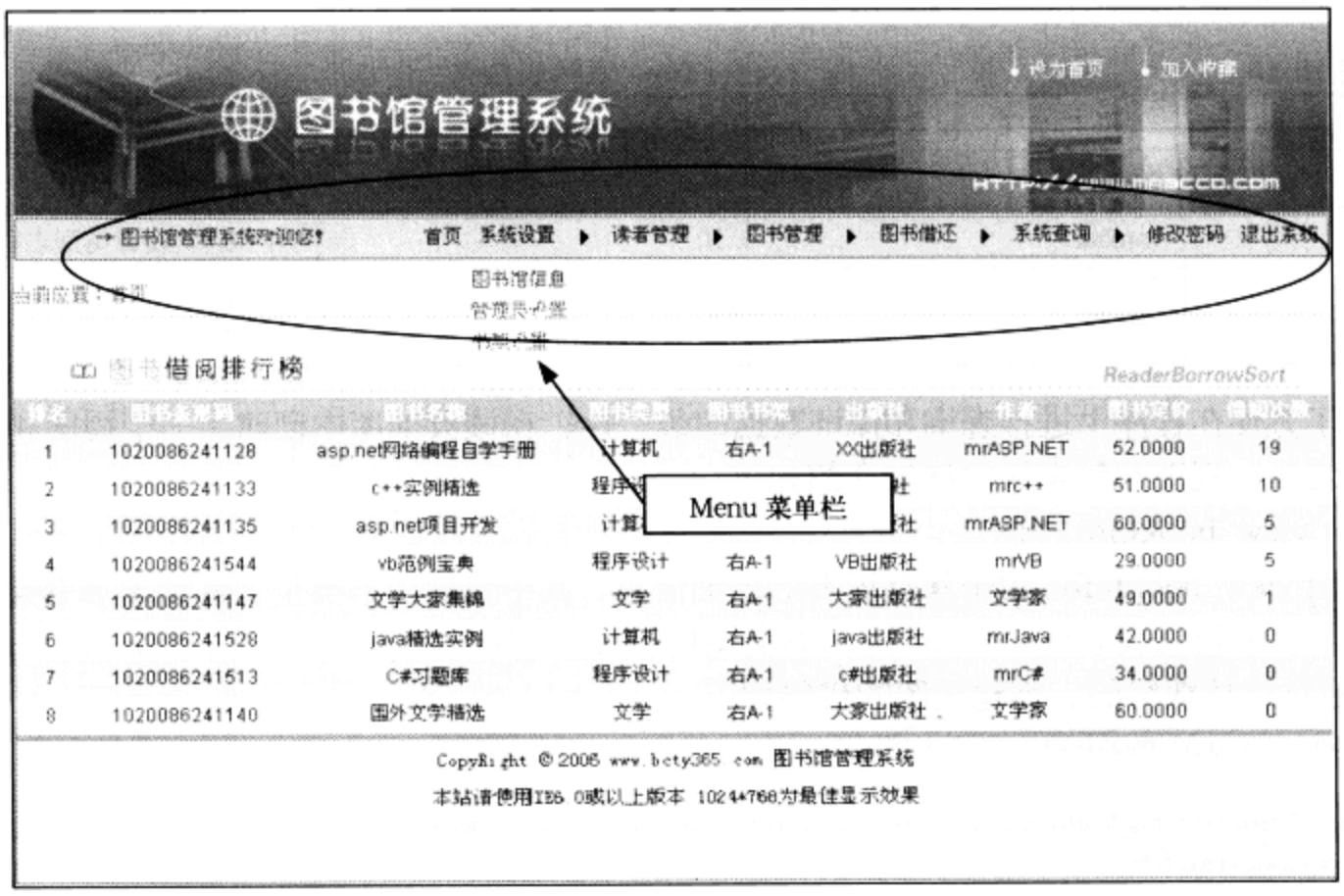


图 10.1　Menu 菜单栏

- Text 属性

该属性用来获取或设置 Menu 控件中显示菜单项的文本。例如，设置 Menu 控件的菜单项文本为“系统设置”代码如下：

```
MenuItem miSystemSet= new MenuItem();
miSystemSet.Text = "系统设置";
```

- NavigateUrl 属性

该属性用来获取或设置单击菜单项时要导航到的 URL。例如，设置 Menu 控件中“图书档案查询”菜单项的导航路径代码如下：

```
MenuItem miBookInfo = new MenuItem();
miBookInfo.Text = "图书档案查询";
miBookInfo.NavigateUrl = "~/systemSearch/bookInfoSearch.aspx";
```

- ChildItems 属性

该属性用来获取 1 个 MenuItemCollection 对象，该对象包含当前菜单项的子菜单项。使用该属性中的 Add 方法可以将子菜单项添加到 ChildItems 属性中。例如，将“图书档案查询”子菜单项添加到“系统查询”菜单项中代码如下：

```
MenuItem miSearch = new MenuItem();
miSearch.Text = "系统查询";
MenuItem miBookInfo = new MenuItem();
miBookInfo.Text = "图书档案查询";
miBookInfo.NavigateUrl = "~/systemSearch/bookInfoSearch.aspx";
miSearch.ChildItems.Add(miBookInfo);
```

10.2.2　借阅业务操作失败使用事务回滚

在图书借阅的业务操作中使用了事务回滚功能，事务回滚可以保证数据的完整性。由于在图书借阅操作中需要执行多步操作，例如，将读者信息表中的金额减去租金，将图书信息表中的图书图书借阅次数加 1 和图书的库存减 1。如果这几步操作中有一步操作出现了错误将会导

致比较严重的错误。为了避免错误的发生在图书借阅业务操作中使用事务回滚操作。

在 ADO.NET 中使用 SqlConnection 对象中的 BeginTransaction 方法来创建一个本地的事务并将该事务添加到 SqlTransaction 类中。SqlTransaction 类表示要在 SQL Server 数据库中处理的 Transact-SQL 事务，该类无法继承。在该类中有两个比较重要的方法，下面具体说明。

- Commit 方法

该方法用来提交数据库的事务，语法的格式如下：

```
Public override void Commit()
```

- Rollback 方法

该方法用来从挂起状态回滚事务，语法的格式如下：

```
Public override void Rollback()
```

10.2.3 权限存储设计思路

由于不同的管理员会拥有不同的管理权限，所以每个管理员的管理权限都保存在数据库中。在数据库中使用了两张表来保存权限信息。一张是权限信息表（tb_admSet）该表中用来存储是否拥有权限，每个权限字段会表示 2 个或 3 个子权限。例如，在图书借阅（bookBorrow）字段中记录着 3 个子权限，如图书借阅、图书续借和图书归还，这 3 个子权限分别用 0 或 1 来表示，并使用逗号将 3 个表示是否拥有权限的 0 或 1 隔开。如果 3 个权限都拥有在图书借阅（bookBorrow）字段下会使用“1，1，1，”的形式记录。另一张菜单栏信息表（tb_menuInfo）用来保存各菜单项在 Menu 控件中显示的文本和跳转路径等信息。可以通过判断权限信息表中的某个字段是否拥有权限，如果拥有再根据菜单栏信息表中的相应文本和跳转路径添加到 Menu 控件上。

10.3 公共类的封装与设计

设计公共类，可以提高开发效率以及方便以后对程序的维护。在本程序中编写了一个公共类 dataOperate，该类主要用来实现对数据库的操作。例如，执行 SQL 语句、返回多条记录等操作。下面将详细介绍公共类中的方法。

10.3.1 实现判断数据是否存在

自定义 seleSQL 方法用来判断所指定的数据是否存在。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个整型变量，当该变量大于 1 时表示查询的数据已经存在。实现代码如下。

例程 1　代码位置：光盘\mr\10\bookManage\ dataOperate.cs

```
// <summary>
// 查询数据是否存在
// </summary>
// <param name="sql">用来执行查询的SQL语句</param>
// <returns>返回一个整型变量，大于0表示存在</returns>
public static int seleSQL(string sql)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sql, con);
    try
    {
        //返回ExecuteScalar方法返回的值
```

```
            return Convert.ToInt32(com.ExecuteScalar());
        }
        catch (Exception e)
        {
            //返回整型值0
            return 0;
        }
        finally
        {
            //关闭数据库连接
            con.Close();
        }
    }
```

10.3.2 实现用户登录操作

自定义 entrySql 方法用于实现用户登录。调用该方法需要传入 3 个字符串变量，第 1 变量为需要执行的 SQL 语句，第 2 变量为用户的登录名，第 3 个变量为用户的登录密码。该方法将返回一个布尔值变量，如登录成功将返回一个布尔值 True，否则将返回 False。实现代码如下。

例程 2 代码位置：光盘\mr\10\bookManage\ dataOperate.cs

```
    // <summary>
    // 该方法用来执行用户登录操作，使用参数传递
    // </summary>
    // <param name="sql">需要执行的sql语句</param>
    // <param name="name">用户登录名</param>
    // <param name="pass">用户登录密码</param>
    // <returns>返回一个布尔值，表示是否登录成功</returns>
    public static bool entrySql(string sql, string name, string pass)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sql, con);
        //设置参数的类型
        com.Parameters.Add(new SqlParameter("name", SqlDbType.VarChar, 50));
        //设置参数的值
        com.Parameters["name"].Value = name;
        //设置参数的类型
        com.Parameters.Add(new SqlParameter("pass", SqlDbType.VarChar, 50));
        //设置参数的值
        com.Parameters["pass"].Value = pass;
        //判断是否登录成功并返回布尔值
        if (Convert.ToInt32(com.ExecuteScalar()) > 0)
        {
            con.Close();
            return true;
        }
        else
        {
            con.Close();
            return false;

        }
    }
```

10.3.3 实现更新、插入、删除操作

自定义 execSQL 方法用来实现对数据库的更新、插入和删除操作。调用该方法需要传入一

个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个布尔值变量，当执行成功将返回 True，否则返回 False。实现代码如下。

例程 3 代码位置：光盘\mr\10\bookManage\ dataOperate.cs

```
// <summary>
// 执行数据库添加、删除、更新等操作
// </summary>
// <param name="sql">需要执行的sql语句</param>
// <returns>返回一个布尔值，表示是否操作成功</returns>
public static bool execSQL(string sql)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sql, con);
    try
    {
        //执行SQL语句
        com.ExecuteNonQuery();
    }
    catch (Exception e)
    {
        //返回布尔值False
        return false;
    }
    finally
    {
        //关闭数据库连接
        con.Close();
    }
    //返回布尔值True
    return true;
}
```

10.3.4 实现查询数据并返回 DataSet

自定义 getDataset 方法用来实现查询数据并返回 DataSet 对象。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回 1 个 DataSet 对象，该对象中存储所有查询的数据。实现代码如下。

例程 4 代码位置：光盘\mr\10\bookManage\ dataOperate.cs

```
// <summary>
// 用来查询记录，以DataSet对象形式返回
// </summary>
// <param name="sql">用来查询的sql语句</param>
// <returns>返回一个DataSet对象</returns>
public static DataSet getDataset(string sql)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //打开数据库连接
    con.Open();
    //创建DataSet对象
    DataSet ds = new DataSet();
    //创建DataAdapter对象
    SqlDataAdapter sda = new SqlDataAdapter(sql, con);
    //填充DataSet数据集
    sda.Fill(ds);
```

```
        //返回DataSet对象
        return ds;
    }
```

10.3.5 实现查询数据并返回 SqlDataReader

自定义 getRow 方法用来实现查询数据并返回 SqlDataReader 对象。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回 1 个 SqlDataReader 对象。实现代码如下。

例程 5 代码位置：光盘\mr\10\bookManage\ dataOperate.cs

```
// <summary>
// 用来查询记录，以SqlDataReader对象形式返回
// </summary>
// <param name="sql">用来查询的sql语句</param>
// <returns>返回一个SqlDataReader对象</returns>
 public static SqlDataReader getRow(string sql)
 {
     //创建数据库连接
     SqlConnection con = createCon();
     //打开数据库连接
     con.Open();
     //创建SqlCommand对象
     SqlCommand com = new SqlCommand(sql, con);
     //返回ExecuteReader方法返回的DataReader对象
     return com.ExecuteReader();
 }
```

10.3.6 实现执行事务处理

自定义 execTransaction 方法用来执行事务处理。调用该方法需要传入一个字符串数组，该数组存储着多条需要执行的 SQL 语句。该方法将返回一个布尔值，该布尔值表示事务执行的是否成功，如果成功将返回一个布尔值 True，否则返回 False。实现代码如下。

例程 6 代码位置：光盘\mr\10\bookManage\ dataOperate.cs

```
// <summary>
// 执行事务操作
// </summary>
// <param name="sql">字符串数组用来存储需要执行的SQL语句</param>
// <returns>返回一个布尔值，表示是否操作成功</returns>
public static bool execTransaction(string[] sql)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //创建SqlTransaction对象
    SqlTransaction sTransaction=null;
    try
    {
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = con.CreateCommand();
        //设置开始事务
        sTransaction = con.BeginTransaction();
        //设置需要执行事务
        com.Transaction = sTransaction;
        foreach (string sqlT in sql)
        {
            //设置SQL语句
            com.CommandText = sqlT;
            //判断是否执行成功
```

```
                if (com.ExecuteNonQuery() <= 0)
                {
                    //设置事务回滚
                    sTransaction.Rollback();
                    //返回布尔值False
                    return false;
                }
            }
            //提交事务
            sTransaction.Commit();
            //返回布尔值True
            return true;
        }
        catch (Exception ex)
        {
            //设置事务回滚
            sTransaction.Rollback();
            //返回布尔值False
            return false;
        }
        finally
        {
            //关闭数据库连接
            con.Close();
        }
    }
```

10.4 图书馆管理系统实现过程

10.4.1 权限菜单栏设计

权限菜单栏显示了管理员所拥有的权限信息。由于每个管理员的管理权限是不同的，所以菜单栏是根据管理员的权限信息而动态创建的。菜单栏如图 10.2 所示。

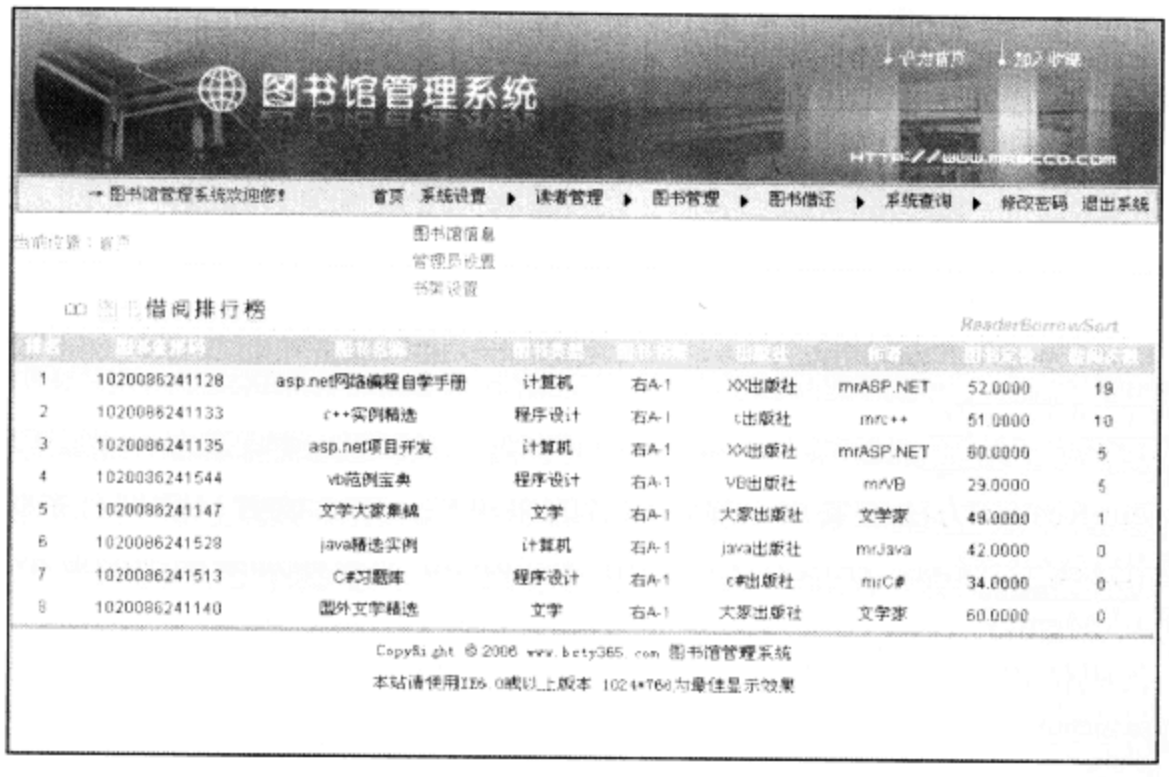


图 10.2 菜单栏

1. 前台页面设计

（1）创建 1 个 Web 用户控件页面，命名为 header.ascx。

（2）在该页面中添加 1 个 Menu 控件，该控件用于实现菜单栏导航。当不同的管理员进行系统后将显示不同的菜单栏。该页面的代码如下。

例程 7 代码位置：光盘\mr\10\bookManage\ header.ascx

```
<%@ Control Language="C#" AutoEventWireup="true" CodeFile="header.ascx.cs" Inherits="header" %>
<table border="0" cellpadding="0" cellspacing="0" style="width: 815px; height: 149px">
    <tr>
        <td colspan="2" style="height: 109px">
            <asp:Image ID="Image1" runat="server" ImageUrl="~/Images/top_bg.gif" Width="815px" /></td>
    </tr>
    <tr>
        <td align="center" style="width: 383px; background-color: #b5b755;">
            图书馆管理系统欢迎您！ </td>
        <td align="center" style="width: 163px;background-color: #b5b755;   text-align: left;">
            <asp:Menu ID="Menu1" runat="server" Orientation="Horizontal" DynamicEnableDefaultPopOutImage=
"False" DynamicVerticalOffset="10" Height="24px" Width="126px">
                <StaticSelectedStyle HorizontalPadding="0px" />
                <StaticMenuItemStyle HorizontalPadding="5px" />
                <DynamicMenuStyle HorizontalPadding="0px" />
                <DynamicMenuItemStyle HorizontalPadding="0px" ItemSpacing="3px" />
                <StaticMenuStyle HorizontalPadding="0px" />
            </asp:Menu>
        </td>
    </tr>
</table>
```

2. 后台代码编写

在页面的加载事件中，首先判断用户的登录类型，根据登录类型得知用户是以读者还是以管理员的身份登录。如果用户是以读者身份登录将调用自定义 menuReader 方法显示读者的 Menu 菜单栏，如果用户是以管理员身份登录将调用自定义 menuManage 方法显示管理员的菜单栏。实现代码如下。

例程 8 代码位置：光盘\mr\10\bookManage\ header.ascx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            //判断用户登录类型
            if (Session["entryType"] == "reader")
            {
                //调用自定义方法，绑定读者的Menu菜单
                menuReader();
            }
            else
            {
                //调用自定义方法，绑定管理员的Menu菜单
                menuManage();
            }
        }
    }
```

自定义 menuReader 方法主要用来显示读者的菜单栏。由于读者只能拥有系统查询的权限、修改密码和退出系统的权限，在该方法中使用 MenuItem 对象添加菜单项的显示文本和跳转路径。最后将使用 Menu 控件的 Add 方法添加到控件中显示出来。实现代码如下。

例程 9 代码位置：光盘\mr\10\bookManage\ header.ascx.cs

```
public void menuReader()
    {
        //创建MenuItem对象
        MenuItem miIndex = new MenuItem();
        //设置显示文本
        miIndex.Text = "首页";
        //设置跳转路径
```

```
        miIndex.NavigateUrl = "../index.aspx";
        //添加到Menu菜单栏中
        this.Menu1.Items.Add(miIndex);
        //设置系统查询权限菜单栏
        MenuItem miSearch = new MenuItem();
        miSearch.Text = "系统查询";
        MenuItem miBookInfo = new MenuItem();
        miBookInfo.Text = "图书档案查询";
        miBookInfo.NavigateUrl = "~/systemSearch/bookInfoSearch.aspx";
        MenuItem miBookBorrow = new MenuItem();
        miBookBorrow.Text = "图书借阅查询";
        miBookBorrow.NavigateUrl = "~/systemSearch/bookBorrowSearch.aspx";
        miSearch.ChildItems.Add(miBookInfo);
        miSearch.ChildItems.Add(miBookBorrow);
        this.Menu1.Items.Add(miSearch);
        //设置密码修改菜单栏
        MenuItem miPass = new MenuItem();
        miPass.Text = "修改密码";
        miPass.NavigateUrl = "../newPass.aspx";
        this.Menu1.Items.Add(miPass);
        //设置退出系统菜单栏
        MenuItem miExit = new MenuItem();
        miExit.Text = "退出系统";
        miExit.NavigateUrl = "../entry.aspx";
        this.Menu1.Items.Add(miExit);
    }
```

自定义 menuManage 方法主要用来显示管理员的菜单栏。由于不同的管理员拥有不同的权限，所以菜单栏中显示的内容也会不同。在该方法中首先将使用 SQL 语句查询当前管理员的权限信息，在判断每个权限字段中是否有 1。如果有 1 代表拥有权限信息，将调用自定义 bindMenu 方法显示菜单栏，否则将不会显示在菜单栏中。实现代码如下。

例程 10 代码位置：光盘\mr\10\bookManage\ header.ascx.cs

```
protected void menuManage()
    {
        //创建SQL语句，该语句用来查询管理员所拥有的权限
        string sqlSel = "select * from tb_admSet where userName='" + Session["userName"] + "'";
        //调用公共类中getRow方法，并接收该方法返回的对象
        SqlDataReader sdr = dataOperate.getRow(sqlSel);
        //读取一条记录
        sdr.Read();
        //创建MenuItem对象
        MenuItem miIndex = new MenuItem();
        //设置菜单项显示文本
        miIndex.Text = "首页";
        //设置跳转路径
        miIndex.NavigateUrl = "../index.aspx";
        //添加到Menu菜单栏中
        this.Menu1.Items.Add(miIndex);
        //判断是否拥有系统设置中的权限
        if (dataOperate.isAdm(sdr["systemSet"].ToString()))
        {
            //创建SQL语句，查询指定的权限信息
            string sqlSelName = "select * from tb_menuInfo where menuName='systemSet' ";
            //调用自定义方法添加权限信息到Menu菜单栏中
            biandMenu(sqlSelName, "systemSet");
        }
        if (dataOperate.isAdm(sdr["readerManage"].ToString()))
        {
            string sqlSelName = "select * from tb_menuInfo where menuName='readerManage' ";
            biandMenu(sqlSelName, "readerManage");
        }
```

```
if (dataOperate.isAdm(sdr["bookManage"].ToString()))
{
    string sqlSelName = "select * from tb_menuInfo where menuName='bookManage' ";
    biandMenu(sqlSelName, "bookManage");
}
if (dataOperate.isAdm(sdr["bookBorrow"].ToString()))
{
    string sqlSelName = "select * from tb_menuInfo where menuName='bookBorrow' ";
    biandMenu(sqlSelName, "bookBorrow");
}
if (dataOperate.isAdm(sdr["systemSearch"].ToString()))
{
    string sqlSelName = "select * from tb_menuInfo where menuName='systemSearch' ";
    biandMenu(sqlSelName, "systemSearch");
}
//创建MenuItem对象
MenuItem miPass = new MenuItem();
//设置菜单项显示文本
miPass.Text = "修改密码";
//设置跳转路径
miPass.NavigateUrl = "../newPass.aspx";
//添加到Menu菜单栏中
this.Menu1.Items.Add(miPass);
//创建MenuItem对象
MenuItem miExit = new MenuItem();
//设置菜单项显示文本
miExit.Text = "退出系统";
//设置跳转路径
miExit.NavigateUrl = "../entry.aspx";
//添加到Menu菜单栏中
this.Menu1.Items.Add(miExit);
}
```

自定义 bindMenu 方法主要用来创建权限菜单栏。在该方法中先判断管理员所拥有的权限再使用 MenuItem 对象添加菜单栏的父菜单项和子菜单项。调用该方法需要传入 2 个字符串变量，第 1 个变量表示 SQL 语句，第 2 个表示字段名。实现代码如下。

例程 11 代码位置：光盘\mr\10\bookManage\ header.ascx.cs

```
// <summary>
// 自定义方法，创建权限菜单栏
// </summary>
// <param name="strSql">字符串变量，表示SQL语句</param>
// <param name="strField">字符串变量，表示字段名</param>
public void biandMenu(string strSql, string strField)
{
    //创建SQL语句，该语句用来查询指定用户的权限信息
    string sqlSel = "select * from tb_admSet where userName='" + Session["userName"] + "'";
    //调用公共类中的getRow方法并接收该方法返回的对象
    SqlDataReader sdr = dataOperate.getRow(sqlSel);
    //读取一条记录
    sdr.Read();
    //获取指定字段的记录，并将该记录通过使用“,”分隔填充到字符串数组中
    string[] strAdm = sdr[strField].ToString().Split(',');
    //创建一个整型变量
    int count = 0;
    //遍历字符串数组，使用变量记录数组中有几个“1”
    foreach (string admS in strAdm)
    {
        if (admS == "1")
        {
            count += 1;
        }
    }
```

```
        //调用公共类中的getDataset方法并接收该方法返回的对象
        DataSet ds = dataOperate.getDataset(strSql);
        //创建MenuItem对象数组
        MenuItem[] mi = new MenuItem[count + 1];
        //使用for循环实例MenuItem对象
        for (int i = 0; i < count + 1; i++)
        {
            mi[i] = new MenuItem();
        }
        //设置级菜单项的文本
        mi[0].Text = ds.Tables[0].Rows[0][2].ToString();
        int miTag = 0;
        //循环数组
        for (int j = 0; j < strAdm.Length; j++)
        {
            if (strAdm[j] == "1")
            {
                //设置变量自增
                miTag += 1;
                //设置级菜单项的文本
                mi[miTag].Text = ds.Tables[0].Rows[j + 1][2].ToString();
                //设置级菜单项的跳转路径
                mi[miTag].NavigateUrl = ds.Tables[0].Rows[j + 1][3].ToString();
                //添加到父菜单项中
                mi[0].ChildItems.Add(mi[miTag]);
            }
        }
        //添加到Menu菜单栏中
        this.Menu1.Items.Add(mi[0]);
    }
```

10.4.2 管理员设置设计

管理员设置页面用于对管理员信息的查看和管理，包括查看管理员的权限信息和添加管理员、删除管理员等操作。管理员设置页面如图 10.3 所示。

图 10.3 管理员设置页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 userManage.aspx

(2)在页面中添加 1 个 Table 表格用于布局，添加 1 个<a>标记可实现打开添加管理员页面，添加 1 个 GridView 控件可显示管理员的显示信息并对管理员进行管理。GridView 控件的前台代码如下。

例程 12 代码位置：光盘\mr\10\bookManage\ userManage.aspx

```
<asp:GridView ID="gvAdmSet" runat="server" AutoGenerateColumns="False" CellPadding="4" ForeColor="#333333"
GridLines="None" Width="815px" OnRowDeleting="gvAdmSet_RowDeleting"
OnRowDataBound="gvAdmSet_RowDataBound">
<FooterStyle BackColor="#507CD1" Font-Bold="True" ForeColor="White" />
<Columns>
<asp:BoundField DataField="userName" HeaderText="管理员名称" />
<asp:TemplateField HeaderText="系统设置">
<ItemTemplate>
<asp:CheckBox ID="ckboxSystemSet" runat="server" Enabled="False" />
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="读者管理">
<ItemTemplate>
<asp:CheckBox ID="ckboxReaderManage" runat="server" Enabled="False" />
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="图书管理">
<ItemTemplate>
<asp:CheckBox ID="ckboxBookManage" runat="server" Enabled="False" />
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="图书借还">
<ItemTemplate>
<asp:CheckBox ID="ckboxBookBorrow" runat="server" Enabled="False" />
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="系统查询">
<ItemTemplate>
<asp:CheckBox ID="ckboxSystemSearch" runat="server" Enabled="False" />
</ItemTemplate>
</asp:TemplateField>
<asp:TemplateField HeaderText="权限设置">
<ItemTemplate>
<a href="#" onclick="window.open('admSet.aspx?userName=<%#Eval("userName")%>',",
'width=340,height=400');" >
权限设置</a>
</ItemTemplate>
</asp:TemplateField>
<asp:CommandField ShowDeleteButton="True" HeaderText="删除" />
</Columns>
<SelectedRowStyle BackColor="#D1DDF1" ForeColor="#333333" Font-Bold="True" />
<PagerStyle BackColor="#2461BF" ForeColor="White" HorizontalAlign="Center" />
<HeaderStyle BackColor="#99C89D" Font-Bold="True" ForeColor="White" />
<AlternatingRowStyle BackColor="White" />
<RowStyle BackColor="#EFF3FB" />
<EditRowStyle BackColor="#2461BF" />
</asp:GridView>
```


2. 后台代码编写

在页面加载事件中，首先判断用户是以读者还是以管理员身份登录。如果用户是以管理员身份登录将判断用户是否登录，如未登录将跳转到登录页面，登录将调用自定义 bindUerManage 方法显示所有管理员的信息。实现代码如下。

例程 13　代码位置：光盘\mr\10\bookManage\ userManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        //判断用户的登录类型，是读者身份还是管理员身份
        if (Session["entryType"] != "reader")
        {
            //判断管理员是否登录
            if (Session["userName"] != null)
            {
                    bindUerManage();
            }
            else
            {
                //返回到登录页面
                Response.Redirect("../entry.aspx"); .
            }
        }
        else
        {
            Response.Write("<script>alert('您没有此权限');location='../index.aspx';</script>");
        }
    }
```

自定义 bindUerMange 方法主要用来显示所有管理员的信息。在该方法中通过使用 SQL 语句查询出所有管理员信息并绑定到 GridView 控件中显示出来。实现代码如下。

例程 14　代码位置：光盘\mr\10\bookManage\ userManage.aspx.cs

```
public void bindUerManage()
    {
        //创建SQL语句，该语句查询管理员的权限信息
        string sql = "select * from tb_admSet";
        //调用公共类中的getDataset方法，并将该方法返回的对象绑定到GridView控件上
        gvAdmSet.DataSource = dataOperate.getDataset(sql);
        //设置主键字段
        gvAdmSet.DataKeyNames=new string[] {"userName"};
        gvAdmSet.DataBind();
    }
```

删除管理员是在 GridView 控件的 RowDeleting 事件中实现的。在该事件中先获取管理员的名称，在创建 SQL 删除语句最后调用公共类中的 execSQL 方法执行 SQL 语句。但对管理员删除后还需要重新绑定 GridView 控件显示管理员信息。实现代码如下。

例程 15　代码位置：光盘\mr\10\bookManage\ userManage.aspx.cs

```
protected void gvAdmSet_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        //获取主键字段
        string userName = this.gvAdmSet.DataKeys[e.RowIndex].Value.ToString();
        //创建SQL语句，该语句用来删除指定的管理员
        string sql = "delete from tb_admSet where userName='" + userName + "'";
        //调用公共类中的execSQL方法执行SQL语句
        dataOperate.execSQL(sql);
        //重新绑定管理员信息
        bindUerManage();
    }
```

在 GridView 控件的 RowDataBound 事件中将设置表示权限的复选框和“删除”按钮是否可用。为了防止登录的管理员删除自己的信息，将登录管理员的“删除”按钮设置为不可用。在 GridView 控件中管理员的每个权限是使用复选框表示的，在 RowDataBound 事件中判断管理员是否拥有某个权限，如果将设置权限的复选框设置为选中状态，实现的代码如下。

例程 16　代码位置：光盘\mr\10\bookManage\ userManage.aspx.cs

```
protected void gvAdmSet_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        //判断是否是数据行
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            //获取管理员名称
            string userName = e.Row.Cells[0].Text;
            //判断管理员名称是否和管理员登录名相同
            if (userName.ToLower() == Session["userName"].ToString())
            {
                //设置删除按钮不可用
                e.Row.Cells[7].Enabled = false;
            }
            //创建SQL语句，该语句查询管理员权限信息
            string sqlSel = "select * from tb_admSet where userName='" + userName + "'";
            //调用公共类中的getRow方法并接收返回值
            SqlDataReader sdr=dataOperate.getRow(sqlSel);
            //读取一条记录
            sdr.Read();
            //判断管理员是否拥有系统设置权限
            if (dataOperate.isAdm(sdr["systemSet"].ToString()))
            {
                //获取系统设置复选框
                CheckBox ck = (CheckBox)e.Row.FindControl("ckboxSystemSet");
                //设置复选框为选中状态
                ck.Checked = true;
            }
            //判断管理员是否拥有读者管理权限
            if (dataOperate.isAdm(sdr["readerManage"].ToString()))
            {
                //获取读者管理复选框
                CheckBox ck = (CheckBox)e.Row.FindControl("ckboxReaderManage");
                //设置复选框为选中状态
                ck.Checked = true;
            }
            //判断管理员是否拥有图书管理权限
            if (dataOperate.isAdm(sdr["bookManage"].ToString()))
            {
                //获取图书管理复选框
                CheckBox ck = (CheckBox)e.Row.FindControl("ckboxBookManage");
                //设置复选框为选中状态
                ck.Checked = true;
            }
            //判断管理员是否拥有图书借阅权限
            if (dataOperate.isAdm(sdr["bookBorrow"].ToString()))
            {
                //获取图书借阅复选框
                CheckBox ck = (CheckBox)e.Row.FindControl("ckboxBookBorrow");
                //设置复选框为选中状态
                ck.Checked = true;
            }
            //判断管理员是否拥有图书查询权限
            if (dataOperate.isAdm(sdr["systemSearch"].ToString()))
            {
                //获取图书查询复选框
                CheckBox ck = (CheckBox)e.Row.FindControl("ckboxSystemSearch");
                //设置复选框为选中状态
                ck.Checked = true;
            }
        }
    }
```

10.4.3 添加管理员设计

添加管理员页用来增加新的管理员，在该页面中还可以设置管理员为超级管理员。添加管理员页面，如图 10.4 所示。

图 10.4 添加管理员页

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 addUser.aspx.

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 10.11 所示。

表 10.11　　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/TextBox 控件	txtUserName	均为默认值	输入管理员登录名
	txtPass	TextMode 属性设置为 Password	输入管理员密码
	txtQpass	TextMode 属性设置为 Password	输入确认管理员密码
标准/CheckBox 控件	cbSuper	均为默认值	选择是否为超级管理员
标准/Button 控件	btnAdd	均为默认值	实现添加管理员操作
HTML/Button 控件	btnClose	在 onclick 事件中编写关闭窗口方法 window.close()	实现关闭页面操作
验证/RequiredFieldValidator 控件	RequiredFieldValidatorName	ControlToValidate 属性设置为 txtUserName	验证管理员登录名是否为空
	RequiredFieldValidatorPass	ControlToValidate 属性设置为 txtPass	验证管理员密码是否为空
验证/ CompareValidator 控件	CompareValidatorPass	ControlToCompare 属性设置为 txtPass ControlToValidate 属性设置为 txtQpass	验证输入的两次密码是否一致

2. 后台代码编写

在“添加”按钮的单击事件中，将完成添加新管理员的操作和添加管理员的权限信息操作。在该事件中首先将获取管理员的登录名及密码。使用 SQL 语句查询输入的管理员登录名是否存在，若存在则不可以添加。如果不存在将执行插入新的管理员操作。实现代码如下。

例程 17　代码位置：光盘\mr\10\bookManage\ addUser.aspx.cs

```
protected void btnAdd_Click(object sender, EventArgs e)
    {
        //获取输入的管理员登录名
        string userName = txtUserName.Text;
        //获取输入的管理员密码
        string pass = txtPass.Text;
        //整型变量用来记录是否为超级管理员
        int isSuper = 0;
        //判断复选框是否被选中
        if (cbSuper.Checked)
        {
            isSuper = 1;
        }
        //创建SQL语句，该语句用来查询填写的管理员登录名是否存在
        string sqlSel = "select count(*) from tb_user where userName='" + userName + "'";
```

```
            //调用公共类中的seleSQL方法执行SQL语句并判断返回的值
            if (dataOperate.seleSQL(sqlSel) <= 0)
            {
                //创建SQL语句，该语句用来实现添加新管理员操作
                string sql = "insert into tb_user values('" + userName + "','" + pass + "','"+ isSuper+"')";
                //执行SQL语句并判断是否执行成功
                if (dataOperate.execSQL(sql))
                {
                    //创建字符串变量用来保存SQL语句
                    string sqlIns = "";
                    //判断是否是超级管理员
                    if (isSuper == 1)
                    {
                        //是超级管理员使用SQL语句添加超级权限
                        sqlIns = "insert into tb_admSet values('" + userName + "','1,1,1,','1,1,','1,1,','1,1,1,','1,1,')";
                    }
                    else
                    {
                        //不是超级管理员使用SQL语句添加默认权限
                        sqlIns = "insert into tb_admSet values('" + userName + "','0,0,0,','0,0,','0,0,','1,1,1,','1,1,')";
                    }
                    dataOperate.execSQL(sqlIns);
                    RegisterStartupScript("", "<script>alert('添加成功！');window.opener.location.href= window.opener.
location='userManage.aspx';window.close();</script>");
                }
                else
                {
                    RegisterStartupScript("", "<script>alert('设置失败！')</script>");
                }
            }
            else
            {
                RegisterStartupScript("", "<script>alert('该管理员已经存在！')</script>");
            }
        }
```

10.4.4 管理员权限设置设计

管理员权限设置页用来，设置管理员的每个权限信息。管理员权限设置页如图 10.5 所示。

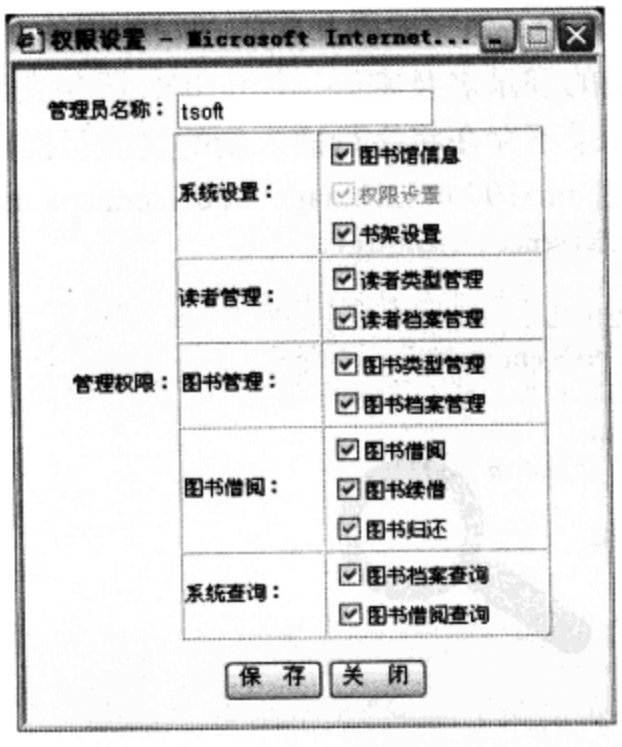


图 10.5 管理员权限设置页

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 admSet.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 10.12 所示。

表 10.12　　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/TextBox 控件	txtUserName	ReadOnly="True"	显示管理员名称
标准/Button 控件	btnSave	均为默认值	实现保存权限设置操作
HTML/Button 控件	btnClose	在 onclick 事件中编写关闭窗口方法 window.close()	实现关闭页面操作
标准/ CheckBoxList 控件	cklistSystemSet	均为默认值	设置系统设置权限
	cklistReaderManage	均为默认值	设置读者管理权限
	cklistBookManage	均为默认值	设置图书管理权限
	cklistBookBorrow	均为默认值	设置图书借阅权限
	cklistSystemSearch	均为默认值	设置图书查询权限

2. 后台代码编写

在页面的加载事件中调用自定义 bindUser 方法显示管理员的权限信息。在自定义 bindUser 方法中获取传入的管理员名称，根据管理员的名称查询该管理员所拥有的权限。并通过循环选中 CheckBoxList 中所对应的权限。为了防止登录的管理员修改自己的权限设置，将通过判断传入的管理员名称是否和登录管理员的名称相同，如果相同将权限设置复选框为不可用状态。实现代码如下。

例程 18　代码位置：光盘\mr\10\bookManage\ admSet.aspx.cs

```
static string userName;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            //调用自定义方法显示该管理员当前拥有的权限
            bindUser();
        }
    }
    //用来绑定管理员当前拥有的权限
    public void bindUser()
    {
        //获取传入的管理员名称
        userName = Request.QueryString["userName"].ToString().ToLower();
        //获取登录管理员的名称
        string strSName = Session["userName"].ToString().ToLower();
        //显示传入的管理员名称
        txtUserName.Text = userName;
        //创建SQL语句，用来查询管理员的权限信息
        string sql = "select * from tb_admSet where userName='" + userName + "'";
        //调用公共类中的getRow方法并接收返回值
        SqlDataReader sdr = dataOperate.getRow(sql);
        //读取一条记录
        sdr.Read();
        //显示系统设置下的所有权限
        string[] systemSet = sdr["systemSet"].ToString().Split(',');
        for (int i = 0; i < systemSet.Length; i++)
```

```
{
    if (systemSet[i] == "1")
    {
        cklistSystemSet.Items[i].Selected = true;
        if (i == 1)
        {
            if (strSName == userName)
            {
                cklistSystemSet.Items[i].Enabled = false;
            }
        }
    }
}
//显示读者管理下的所有权限
string[] readerManage = sdr["readerManage"].ToString().Split(',');
for (int i = 0; i < readerManage.Length; i++)
{
    if (readerManage[i] == "1")
    {
        cklistReaderManage.Items[i].Selected = true;
    }
}
//显示图书管理下的所有权限
string[] bookManage = sdr["bookManage"].ToString().Split(',');
for (int i = 0; i < bookManage.Length; i++)
{
    if (bookManage[i] == "1")
    {
        cklistBookManage.Items[i].Selected = true;
    }
}
//显示图书借阅下的所有权限
string[] bookBorrow = sdr["bookBorrow"].ToString().Split(',');
for (int i = 0; i < bookBorrow.Length; i++)
{
    if (bookBorrow[i] == "1")
    {
        cklistBookBorrow.Items[i].Selected = true;
    }
}
//显示系统查询下的所有权限
string[] systemSearch = sdr["systemSearch"].ToString().Split(',');
for (int i = 0; i < systemSearch.Length; i++)
{
    if (systemSearch[i] == "1")
    {
        cklistSystemSearch.Items[i].Selected = true;
    }
}
```

在“保存”按钮的单击事件中，通过循环获取表示各个权限的复选框状态，最后通过使用SQL 语句将跟新管理员的权限信息。实现代码如下。

例程 19 代码位置：光盘\mr\10\bookManage\ admSet.aspx.cs

```
protected void btnSave_Click(object sender, EventArgs e)
    {
        //获取系统设置下的所选权限
```

```
string systemSet="";
for (int i = 0; i < cklistSystemSet.Items.Count; i++)
{
    if (cklistSystemSet.Items[i].Selected)
    {
        systemSet += "1,";
    }
    else
    {
        systemSet += "0,";
    }
}
//获取读者管理下的所选权限
string readerManage = "";
for (int i = 0; i < cklistReaderManage.Items.Count; i++)
{
    if (cklistReaderManage.Items[i].Selected)
    {
        readerManage += "1,";
    }
    else
    {
        readerManage += "0,";
    }
}
//获取图书管理下的所选权限
string bookManage = "";
for (int i = 0; i < cklistBookManage.Items.Count; i++)
{
    if (cklistBookManage.Items[i].Selected)
    {
        bookManage += "1,";
    }
    else
    {
        bookManage += "0,";
    }
}
//获取图书借阅下的所选权限
string bookBorrow = "";
for (int i = 0; i < cklistBookBorrow.Items.Count; i++)
{
    if (cklistBookBorrow.Items[i].Selected)
    {
        bookBorrow += "1,";
    }
    else
    {
        bookBorrow += "0,";
    }
}
//获取系统查询下的所选权限
string systemSearch = "";
for (int i = 0; i < cklistSystemSearch.Items.Count; i++)
{
    if (cklistSystemSearch.Items[i].Selected)
    {
        systemSearch += "1,";
    }
    else
    {
        systemSearch += "0,";
```

```
                }
            }
            //创建SQL语句，用来更新管理员的权限信息
            string sql = "update tb_admSet set systemSet='" + systemSet + "',systemSearch='" + systemSearch +
"',readerManage='" + readerManage + "',bookManage='" + bookManage + "' ,bookBorrow='" + bookBorrow + "' where
userName='" + userName + "'";
            //调用公共类中的execSQL方法执行SQL语句更新操作
            if (dataOperate.execSQL(sql))
            {
                //提示设置成功
                RegisterStartupScript("", "<script>alert('设置成功！');opener.location.reload();window.close();</script>");
            }
            else
            {
                RegisterStartupScript("", "<script>alert('设置失败！')</script>");
            }
        }
```

10.4.5 图书借阅设计

图书借阅页面用来实现图书的借阅功能。在该页面中管理员通过输入读者的条形码和图书的条形码来实现图书的借阅功能。图书借阅页面如图 10.6 所示。

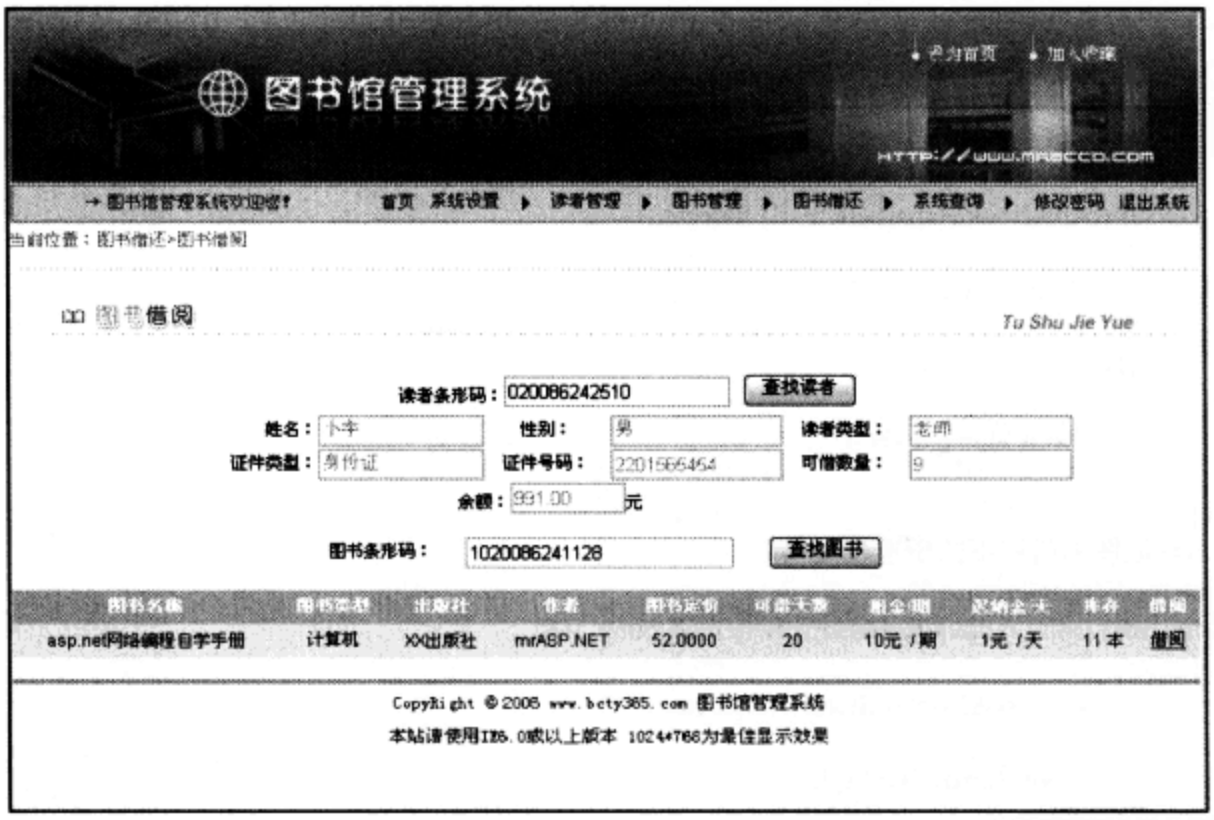


图 10.6 图书借阅页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 bookBorrow.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 10.13 所示。

表 10.13 控件类型、控件名称及说明

控件类型	控件名称	主要属性	说明
标准/TextBox 控件	txtReaderBarCode	均为默认值	输入读者的条形码
	txtReaderName	Enabled 属性设置为 False	显示读者的姓名
	txtReaderSex	Enabled 属性设置为 False	显示读者的性别

续表

控件类型	控件名称	主要属性	说明
标准/TextBox 控件	txtReaderType	Enabled 属性设置为 False	显示读者的类型
	txtCertificateType	Enabled 属性设置为 False	显示读者的证件类别
	txtCertificate	Enabled 属性设置为 False	显示读者的证件号
	txtNum	Enabled 属性设置为 False	显示读者可借图书数量
	txtMoney	Enabled 属性设置为 False	显示读者剩余金额
	txtBookBarCode	均为默认值	输入图书的条形码
标准/Button 控件	btnReaderSearch	均为默认值	查询读者信息按钮
	btnBookSearch	Enabled 属性设置为 False	查询图书信息按钮
标准/GridViewt 控件	gvBookBorrow	均为默认值	显示图书信息

2. 后台代码编写

在“查找读者”按钮的单击事件中将调用自定义 bindReaderInfo 方法。在自定义 bindReaderInfo 方法中通过输入的读者条形码，设置 SQL 语句查询出该读者的详细信息，并显示在文本框中。实现代码如下。

例程 20 代码位置：光盘\mr\10\bookManage\ bookBorrow.aspx.cs

```
protected void btnReaderSearch_Click(object sender, EventArgs e)
    {
        bindReaderInfo();       //调用自定义方法显示读者信息
    }
    //自定义方法显示读者信息
public void bindReaderInfo()
    {
        string readerBarCode = txtReaderBarCode.Text;                                   //获取读者条形码
        //创建SQL语句在读者信息表中查询符合读者条形码条件的记录
        string readerSql = "select * from tb_readerInfo where readerBarCode='" + readerBarCode + "'";
        SqlDataReader sdr = dataOperate.getRow(readerSql);                              //获取该读者详细信息
        if (sdr.Read())
        {                                                                               //读取一条记录
            txtReaderName.Text = sdr["readerName"].ToString();                          //显示读者姓名
            txtReaderSex.Text = sdr["Sex"].ToString();                                  //显示读者性别
            txtCertificateType.Text = sdr["certificateType"].ToString();               //显示证件类型
            txtCertificate.Text = sdr["certificate"].ToString();                        //显示证件号
            string money = sdr["money"].ToString();
            txtMoney.Text = money.Substring(0, money.Length - 2);
            //创建SQL语句在读者类型表中查询符合读者类型编号的记录
            string readerTypeSql = "select * from tb_readerType where id=" + sdr["readerType"].ToString();
            SqlDataReader typeSdr = dataOperate.getRow(readerTypeSql);                  //获取读者类型信息
            typeSdr.Read();                                                             //读取一条记录
            txtReaderType.Text = typeSdr["type"].ToString();                            //显示读者类型
            int borrowNum = Convert.ToInt32(typeSdr["num"]);                            //获取可借图书总数
            //创建SQL语句在图书借阅表中查询符合读者条形码条件的读者借了几本图书（图书未还的）
            string selSql = "select count(*) from tb_bookBorrow where readerBarCode='" + readerBarCode + "' ";
            int alreadyNum = dataOperate.seleSQL(selSql);                               //获取图书已借数
            txtNum.Text = Convert.ToString(borrowNum - alreadyNum);                     //显示可以借阅图书数
            btnBookSearch.Enabled = true;
        }
```

```
        else
            RegisterStartupScript("", "<script>alert('读者条形码输入错误!')</script>");
    }
```

在“查找图书”按钮的单击事件中实现查找图书信息，并将图书信息显示在 GridView 控件中。在该事件中首先将判断读者是否还可以借阅图书，还需要判断是否输入了图书的条形码。最后将通过 SQL 语句查询出图书的信息并显示在 GridView 控件上。实现代码如下。

例程 21 代码位置：光盘\mr\10\bookManage\ bookBorrow.aspx.cs

```
protected void btnBookSearch_Click(object sender, EventArgs e)
{
    if (Convert.ToInt32(txtNum.Text.Trim()) > 0)                    //判断读者是否还可以借书
    {
        if (txtBookBarCode.Text.Trim() != "")                       //判断图书条形码是否为空
        {
            string bookBarCode = txtBookBarCode.Text;               //获取图书条形码
            //创建SQL语句在图书信息表中查询符合图书条形码条件的记录
            string sql = "select * from tb_bookInfo where bookBarCode='" + bookBarCode + "'";
            DataSet ds = dataOperate.getDataset(sql);
            if (ds.Tables[0].Rows.Count > 0)
            {
                gvBookBorrow.DataSource = ds;                       //获取数据源
                gvBookBorrow.DataKeyNames = new string[] { "bookBarCode" }; //设置主键
                gvBookBorrow.DataBind();                            //绑定GridView控件
            }
            else
                RegisterStartupScript("", "<script>alert('图书条形码错误！')</script>");
        }
        else
        {
            RegisterStartupScript("", "<script>alert('图书条形码不能为空')</script>");
        }
    }
    else
    {
        RegisterStartupScript("", "<script>alert('借阅数量已满！不可以再借阅')</script>");
    }
}
```

为了方便管理员的对图书信息的查看，在 GridView 控件的 RowDataBound 事件中将设置信息显示的格式，例如，设置图书的租金显示格式为每期多少元“n 元/期”。实现代码如下。

例程 22 代码位置：光盘\mr\10\bookManage\ bookBorrow.aspx.cs

```
protected void gvBookBorrow_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType == DataControlRowType.DataRow)
        {   //设置图书类型
            string bookType = e.Row.Cells[1].Text.ToString();                   //获取图书类型编号
            //创建SQL语句在图书类型表中查询符合图书类型编号条件的记录
            string typeSql = "select * from tb_bookType where TypeID=" + bookType;
            SqlDataReader typeSdr = dataOperate.getRow(typeSql);
            typeSdr.Read();                                                     //读取一条记录
            e.Row.Cells[1].Text = typeSdr["typeName"].ToString();               //显示图书类型
            e.Row.Cells[5].Text = typeSdr["borrowDay"].ToString();              //显示图书可借天数
            e.Row.Cells[6].Text = typeSdr["hire"].ToString() + "元 / 期";
            e.Row.Cells[7].Text = typeSdr["lagMoney"].ToString() + "元 / 天";
```

```
        }
    }
```

图书借阅功能是在 GridView 控件的 SelectedIndexChanging 事件中实现的，在该事件中通过获取图书的条形码查询该图书的信息并判断该图书是否还有库存。如果图书还有库存将实现图书借阅的功能，图书借阅功能将会完成一系列的 SQL 语句操作。如向图书借阅表添加图书借阅信息，更新图书的库存及借阅次数，更新读者的金额信息。这些操作将通过公共类中的执行事务方法去执行。实现代码如下。

例程 23 代码位置：光盘\mr\10\bookManage\ bookBorrow.aspx.cs

```
protected void gvBookBorrow_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
{
    //获取选中图书条形码
    string bookBarCode = gvBookBorrow.DataKeys[e.NewSelectedIndex].Value.ToString();
    //创建SQL语句使用内联连接条件为图书类型编号，查询条件为符合图书条形码的记录
    string sql = "select * from tb_bookInfo as a inner join tb_bookType as b on a.bookType=typeID where
a.bookBarCode='" + bookBarCode + "'";
    SqlDataReader sdr = dataOperate.getRow(sql);                                    //获取图书信息
    //读取一条记录
    sdr.Read();
    //判断图书是否还有库存
    if (Convert.ToInt32(sdr["stock"]) > 0)
    {
        float   readerMoney =Convert.ToSingle( txtMoney.Text);
        float bookHire = Convert.ToSingle(sdr["hire"].ToString());
        if (readerMoney>bookHire)
        {
            string sqlBookBorrow = "select count(*) from tb_bookBorrow where   bookBarcode='" +
txtBookBarCode.Text + "' and readerBarCode='" + txtReaderBarCode.Text + "' ";
            if (dataOperate.seleSQL(sqlBookBorrow) == 0)
            {
                int borrowDay = Convert.ToInt32(sdr["borrowDay"]);              //获取借阅天数
                string bookName = sdr["bookName"].ToString();                   //获取图书名称
                string borrowTime = DateTime.Now.Date.ToShortDateString();      //获取借阅日期
                string readerBarCode = txtReaderBarCode.Text;                   //获取读者条形码
                string returnTime = DateTime.Now.Date.AddDays(borrowDay).ToShortDateString();
                                                                                //获取应还日期
                string readerName = txtReaderName.Text;                         //获取读者姓名
                //设置SQL语句数组
                string[] sqlT = new string[4];
                //设置SQL语句，将图书借阅信息插入到图书借阅表中
                sqlT[0] = "insert tb_bookBorrow values('" + bookBarCode + "','" + bookName + "','" +
borrowTime + "','" + returnTime + "','" + readerBarCode + "','" + readerName + "')";
                //设置SQL语句，更新图书的借阅次数和图书的库存信息
                sqlT[1] = "update tb_bookInfo set borrowSum=borrowSum+1,stock=stock-1 where
bookBarCode='" + bookBarCode + "'";
                //设置SQL语句，更新读者的金额
                sqlT[2] = "update tb_readerInfo set money=money-" + bookHire + " where readerBarCode='" +
readerBarCode + "'";
                //设置SQL语句
                //sqlT[3] = "update tb_bookInfo   where bookBarCode='" + bookBarCode + "'";
                //调用公共类中的execTransaction方法执行事务
                if (dataOperate.execTransaction(sqlT))
                {
                    bindReaderInfo();                                           //重新绑定读者信息
                    gvBookBorrow.DataSource = null;                             //将数据源设置为空
```

```
                    gvBookBorrow.DataBind();
                    txtBookBarCode.Text = "";                    //将图书条形码文本框清空
                    RegisterStartupScript("", "<script>alert('借阅成功！')</script>");
                }
                else
                {
                    RegisterStartupScript("", "<script>alert('借阅失败！')</script>");
                }
            }
            else
            {
                RegisterStartupScript("", "<script>alert('该读者已经借阅此图书不可以在借阅！')</script>");
            }

        }
        else
        {
            RegisterStartupScript("", "<script>alert('读者金额不足不可以借阅图书！')</script>");
        }
    }
    else
    {
        RegisterStartupScript("", "<script>alert('图书已没有库存不可以借阅！')</script>");
    }
}
```

10.4.6 图书续借设计

图书续借页面用来实现图书的续借功能，如图读者借阅的图书还未到期或快要到期，但读者还想要借阅此书，此时就可以使用图书续借功能来延长图书的归还日期，如图 10.7 所示。

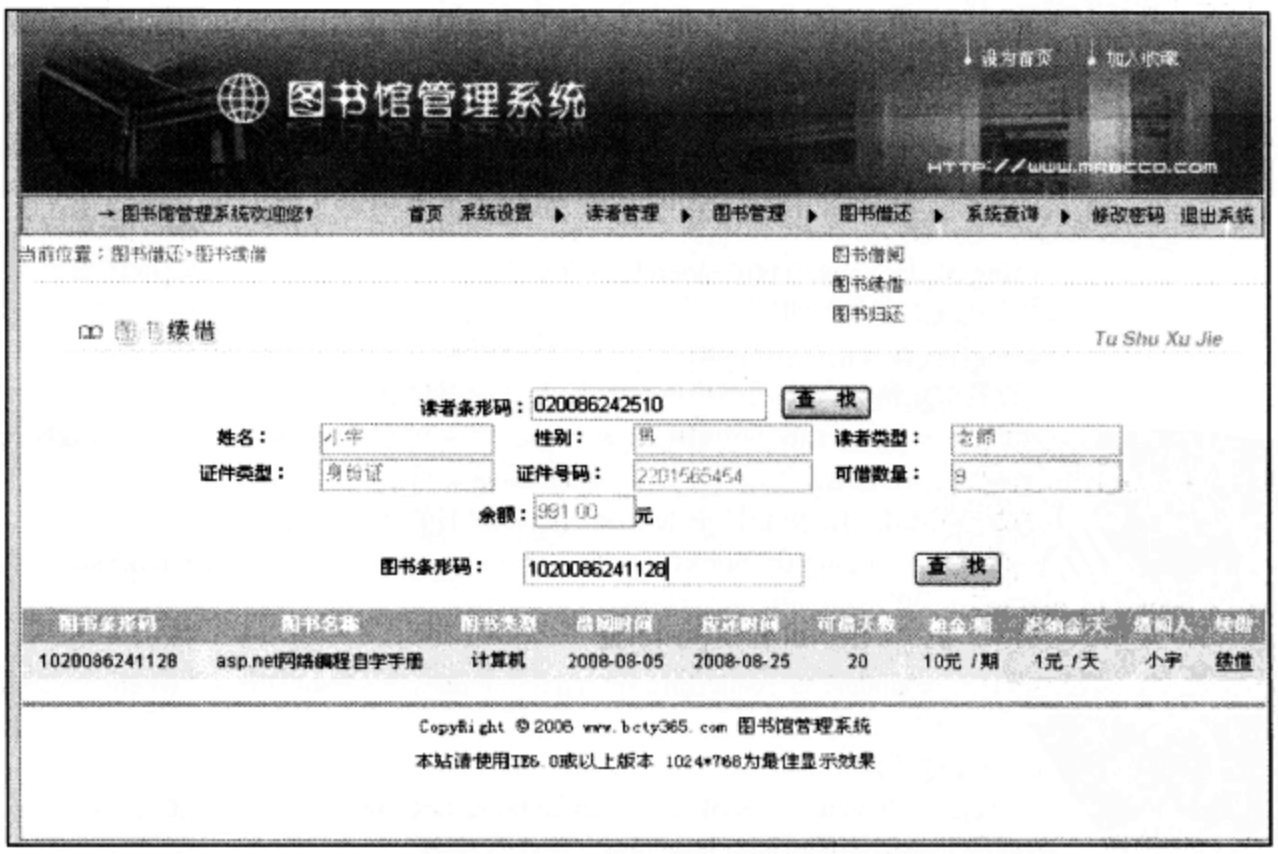


图 10.7 图书续借页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 bookRenewal.aspx。

（2）在该窗体中添加的控件与图书借阅窗体中添加的控件一样，这里就不介绍了。

2. 后台代码编写

在“查找”按钮的单击事件中，将获取到读者的条形码，通过读者条形码创建 SQL 语句，查询出读者的详细信息，并将信息显示文本框中。最后通过读者的条形码查询出读者所有已借阅的图书并显示在 GridView 控件上。实现代码如下。

例程 24 代码位置：光盘\mr\10\bookManage\ bookRenewal.aspx

```
protected void btnReaderSearch_Click(object sender, EventArgs e)
    {
        string readerBarCode = txtReaderBarCode.Text;                                   //获取读者条形码
        //创建SQl语句，在读者信息表中查询符合读者条形码条件的记录
        string readerSql = "select * from tb_readerInfo where readerBarCode='" + readerBarCode + "'";
        SqlDataReader sdr = dataOperate.getRow(readerSql);
        if (sdr.Read())                                                                 //读取一条记录
        {
            txtReaderName.Text = sdr["readerName"].ToString();                          //显示读者姓名
            txtReaderSex.Text = sdr["Sex"].ToString();                                  //显示读者性别
            txtCertificateType.Text = sdr["certificateType"].ToString();               //显示读者证件类型
            txtCertificate.Text = sdr["certificate"].ToString();                        //显示证件号码
            string readerMoney = sdr["money"].ToString();
            txtMoney.Text = readerMoney.Substring(0, readerMoney.Length - 2);          //显示余额
            //创建SQL语句，在读者类型表中查询符合读者类型编号条件的记录
            string readerTypeSql = "select * from tb_readerType where id=" + sdr["readerType"].ToString();
            SqlDataReader typeSdr = dataOperate.getRow(readerTypeSql);
            typeSdr.Read();                                                             //读取一条记录
            txtReaderType.Text = typeSdr["type"].ToString();                            //获取读者类型
            int borrowNum = Convert.ToInt32(typeSdr["num"]);                        //获取读者可以借阅图书的总数
//创建SQL语句，在图书借阅表中查询符合读者条形码条件的读者借了几本图书（未还的图书）
            string selSql = "select count(*) from tb_bookBorrow where readerBarCode='" + readerBarCode + "' ";
            int alreadyNum = dataOperate.seleSQL(selSql);                               //获取图书已借的数目
            txtNum.Text = Convert.ToString(borrowNum - alreadyNum);                     //显示可以借阅数
            //创建SQL语句，在图书借阅表中查询符合读者条形码条件未归还图书的记录
            string sql = "select * from tb_bookBorrow as a join view_bookInfo as b on a.bookBarCode=
b.bookBarCode where   a.readerBarCode='" + txtReaderBarCode.Text + "'";
    //调用自定义方法显示已借阅未归还图书信息
            bindGridView(sql);
        }
        else
            RegisterStartupScript("", "<script>alert('读者条形码输入错误！')</script>");
    }
```

当显示读者的信息和读者已借阅的图书后，如果因读者已借阅的图书过多而不方便查找。可以在图书条形码文本框中输入要续借图书的条形码。可通过单击“查找”按钮查询图书，并显示在 GridView 控件中。在该事件中首先将获取图书的条形码，通过使用读者的条形码和图书的条形码查询图书的信息。实现代码如下。

例程 25 代码位置：光盘\mr\10\bookManage\ bookRenewal.aspx

```
    protected void btnBookSearch_Click(object sender, EventArgs e)
        {
            if (txtBookBarCode.Text.Trim() != "")                                   //判断图书条形码是否为空
            {
                string bookBarCode = txtBookBarCode.Text;                           //获取图书条形码
        //创建SQL语句，在图书借阅表中查询符合读者条形码条件和图书条形码条件未归还图书的记录
                string sqlBorrow = "select count(*) from tb_bookBorrow where bookBarCode='" + bookBarCode + "'and
readerBarCode='" + txtReaderBarCode.Text + "'";
                //判断是否有未归还的图书
```

```
                if (dataOperate.seleSQL(sqlBorrow) > 0)
                {
                    string sql = "select * from tb_bookBorrow as a join view_bookInfo as b on a.bookBarCode=
b.bookBarCode where   a.readerBarCode='" + txtReaderBarCode.Text + "'and a.bookBarCode='" + bookBarCode + "'";
                    bindGridView(sql);                                    //调用自定义方法显示已借阅未归还图书的信息
                }
                else
                {
                    RegisterStartupScript("", "<script>alert('该图书没有借阅过，请到借阅功能中操作！')</script>");
                    gvBookRenewal.DataBind();
                }
            }
            else
            {
                RegisterStartupScript("", "<script>alert('图书条形码不能为空')</script>");
            }
```

图书续借功能是在 GridView 控件的 SelectedIndexChanging 事件中实现的，在该事件中将判断当前的日期是否为已过应还日期的一半时间。如，读者借阅图书的日期为 2008 年 8 月 1 日，应还图书的日期为 2008 年 8 月 21 日，这里需要判断一下当前的日期是否大于 2008 年 8 月 11 日，如果大于该日期才可以使用图书续借功能，否则不可以使用。使用图书续借功能需要执行 3 个 SQL 语句，第 1 个 SQL 语句用来扣除图书的租金，包括迟纳金；第 2 个 SQL 语句用来增加图书的借阅次数；第 3 个 SQL 语句用来更新图书的借阅时间和图书的归还时间。实现代码如下。

例程 26 代码位置：光盘\mr\10\bookManage\ bookRenewal.aspx

```
protected void gvBookRenewal_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
    {
        //获取当前图书条形码
        string bookBarCode = gvBookRenewal.DataKeys[e.NewSelectedIndex].Value.ToString();
        //获取图书应归还的日期
        DateTime returnDate = Convert.ToDateTime(gvBookRenewal.Rows[e.NewSelectedIndex].Cells[4].Text);
        //获取当前日期
        DateTime todayDate = DateTime.Now.Date;
        //将当前日期减去图书应归还的日期
        TimeSpan ts = todayDate - returnDate;
        //获取两个日期相差的天数
        int daysDate = ts.Days;
        //获取图书的滞纳金
        string strLagMoney = gvBookRenewal.Rows[e.NewSelectedIndex].Cells[7].Text;
        //将获取的图书滞纳金的单位去掉，只保留金额
        int lagMoney = Convert.ToInt32(strLagMoney.Substring(0, strLagMoney.Length - 5));
        //获取图书的租金
        string strHire = gvBookRenewal.Rows[e.NewSelectedIndex].Cells[6].Text;
        //将获取的租金单位去掉
        int hire = Convert.ToInt32(strHire.Substring(0, strHire.Length - 5));
        //获取该图书可借阅的天数
        int borrowDay = Convert.ToInt32(gvBookRenewal.Rows[e.NewSelectedIndex].Cells[5].Text);
        string hint = "";
        //获取图书可借阅天数一半的日期
        DateTime isHalf = returnDate.AddDays(-borrowDay / 2);
        //判断图书借阅日期是否超过一半可借天数
        if (isHalf < todayDate)
        {
        //创建字符串数组用来保存SQL语句
```

```
                string[] sqlT = new string[3];
                //判读图书是否以超过规定的借阅时间
                if (daysDate > 0)
                {
                    //创建SQL语句，在读者的金额中减去图书的租金和图书滞纳金
                    sqlT[0] = "update tb_readerInfo set money=money- " + hire + "- " + lagMoney * daysDate + " where
readerBarCode = '" + txtReaderBarCode.Text + "' ";
                    hint = "您的图书归还期已过" + daysDate + "天，将扣除" + lagMoney * daysDate + "元金额";
                }
                else
                {
                    //创建SQL语句，在读者的金额中减去图书的租金
                    sqlT[0] = "update tb_readerInfo set money=money- " + hire + " where readerBarCode = '" +
txtReaderBarCode.Text + "' ";
                }
                //创建SQL语句，将图书的借阅次数加一
                sqlT[1] = "update tb_bookInfo set borrowSum=borrowSum+1 where bookBarCode='" + bookBarCode + "'";
                //创建SQL语句，该语句用于实现更新图书的借阅时间和图书的归还时间
                sqlT[2] = "update tb_bookBorrow set borrowTime='" + todayDate + "', returnTime='" + todayDate.AddDays
(borrowDay) + "' where readerBarCode='" + txtReaderBarCode.Text + "' and bookBarCode='" + bookBarCode + "' ";
                //判断事务是否执行成功
                if (dataOperate.execTransaction(sqlT))
                {
                    //创建SQL语句，查询读者信息
                    string readerSql = "select * from tb_readerInfo where readerBarCode='" + txtReaderBarCode.Text + "'";
                    //调用公共类中的getRow方法，执行SQL语句并接收返回的SqlDataReader对象
                    SqlDataReader sdr = dataOperate.getRow(readerSql);
                    //读取一条记录
                    sdr.Read();
                    //获取读者金额
                    string readerMoney = sdr["money"].ToString();
                    //显示余额
                    txtMoney.Text = readerMoney.Substring(0, readerMoney.Length - 2);
                    //创建SQL语句，查询图书借阅信息
                    string sql = "select * from tb_bookBorrow as a join view_bookInfo as b on a.bookBarCode=
b.bookBarCode where   a.readerBarCode='" + txtReaderBarCode.Text + "'";
                    //调用自定义方法，显示已借阅而未归还的图书信息
                    bindGridView(sql);
                    RegisterStartupScript("", "<script>alert('续借成功！ " + hint + "')</script>");
                }
                else
                {
                    RegisterStartupScript("", "<script>alert('续借失败！ ')</script>");
                }
            }
            else
            {
                RegisterStartupScript("", "<script>alert('借阅天数未过半不可以续借！ ')</script>");
            }
        }
```

10.4.7 图书归还设计

图书归还页面用来实现图书的归还功能，如果读者想要归还已借阅的图书、快要到期或已经过期的图书，都可以在该页面中实现。图书归还页面如图 10.8 所示。

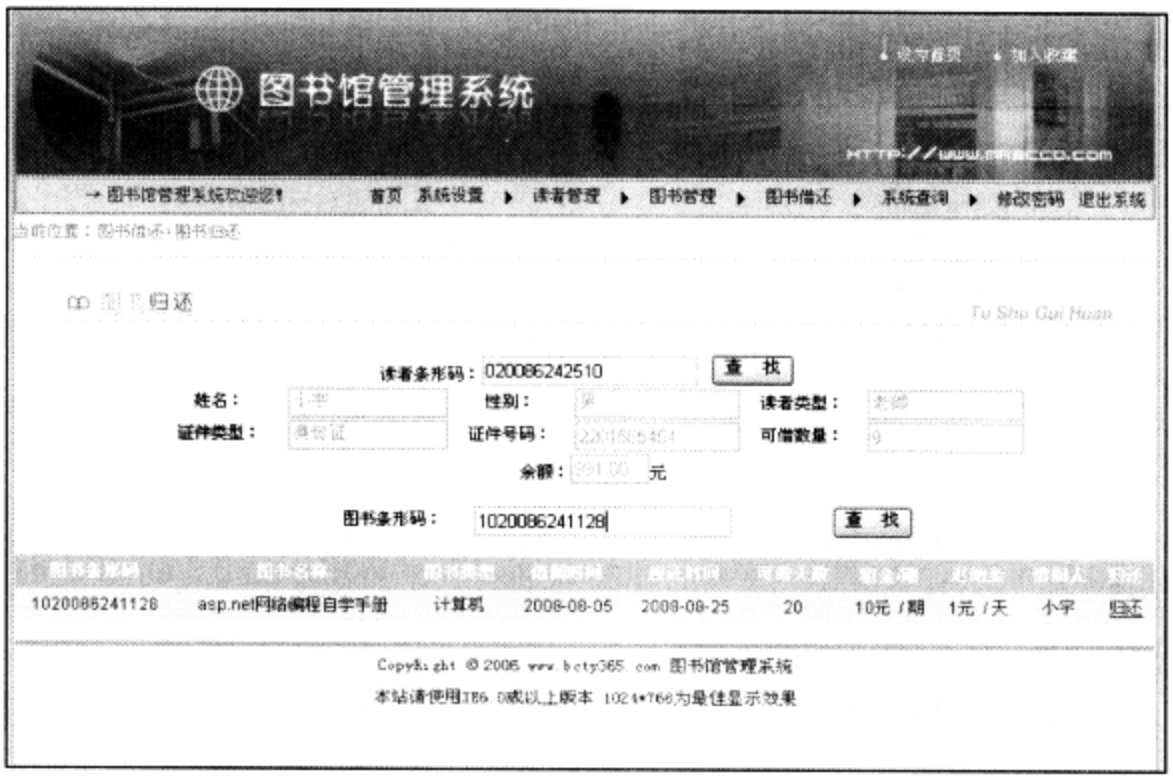


图 10.8　图书归还页面

1. 前台页面的设计。

(1) 创建 1 个 Web 窗体，命名为 bookReturn.aspx。

(2) 在该窗体中添加的控件与图书借阅窗体中添加的控件一样，这里就不再赘述。

2. 后台代码的编写。

由于读者信息的查询和图书信息的查询与图书续借中的类似，这里就不做讲解了。下面主要讲解一下图书归还功能的实现。图书归还功能主要在 GridView 控件中的 SelectedIndexChanging 事件中实现的。在该事件中将获取当前日期减去应还日期的天数差。通过天数差判断图书归还日期是否已过。如果已过将会执行 3 个 SQL 语句，第 1 个 SQL 语句用来扣除滞纳金，第 2 个 SQL 语句将图书的库存加一，第 3 个 SQL 语句用来删除图书的借阅信息。如果未过将不执行第 1 个操作。实现代码如下。

例程 27　代码位置：光盘\mr\10\bookManage\ bookReturn.aspx

```
protected void gvBookReturn_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
{
    //获取当前选择图书条形码
    string bookBarCode = gvBookReturn.DataKeys[e.NewSelectedIndex].Value.ToString();
    //获取图书应还的日期
    DateTime returnDate = Convert.ToDateTime(gvBookReturn.Rows[e.NewSelectedIndex].Cells[4].Text);
    //获取图书当前的日期
    DateTime todayDate = DateTime.Now.Date;
    //将2个日期相减
    TimeSpan ts = todayDate - returnDate;
    //获取2个日期的相差的天数
    int daysDate = ts.Days;
    //获取滞纳金
    string strLagMoney = gvBookReturn.Rows[e.NewSelectedIndex].Cells[7].Text;
    int lagMoney = Convert.ToInt32(strLagMoney.Substring(0, strLagMoney.Length - 5));
    string hint = "";
    //创建字符串数组
    string[] sqlT ;
    int i = 0;
    //判断图书归还日期是否已过
    if (daysDate > 0)
```

```
            {
                sqlT = new string[3];
                //创建SQL语句，扣除图书的滞纳金
                sqlT[i++] = "update tb_readerInfo set money=money - " + lagMoney * daysDate + " where
readerBarCode = '" + txtReaderBarCode.Text + "' ";
                //编写提示信息
                hint = "您的图书归还期已过" + daysDate + "天，将扣除" + lagMoney * daysDate + "元金额";
            }
            else
            {
                sqlT = new string[2];
            }
            //创建SQL语句，将图书的库存加一
            sqlT[i++] = "update tb_bookInfo set stock=stock+1 where bookBarCode='" + bookBarCode + "'";
            //创建SQL语句，删除该图书的借阅信息
            sqlT[i] = "delete tb_bookBorrow where bookBarCode='" + bookBarCode + "' and readerBarCode='" +
txtReaderBarCode.Text + "' ";
            if (dataOperate.execTransaction(sqlT))
            {
                //调用自定义方法显示已借阅未归还的图书信息
                bindReaderInfo();
                RegisterStartupScript("", "<script>alert('图书归还成功！ "+hint+"')</script>");
            }
            else
            {
                RegisterStartupScript("", "<script>alert('图书归还失败！ ')</script>");
            }
        }
```

10.4.8 图书档案查询设计

图书档案查询页面用来实现读者对某个图书的搜索功能。在该页面中读者可以根据图书的条形码、图书名称、作者等条件，进行图书的搜索功能。使用该功能可以方便读者了解图书的库存信息、图书的租金、图书的借阅次数等信息，如图 10.9 所示。

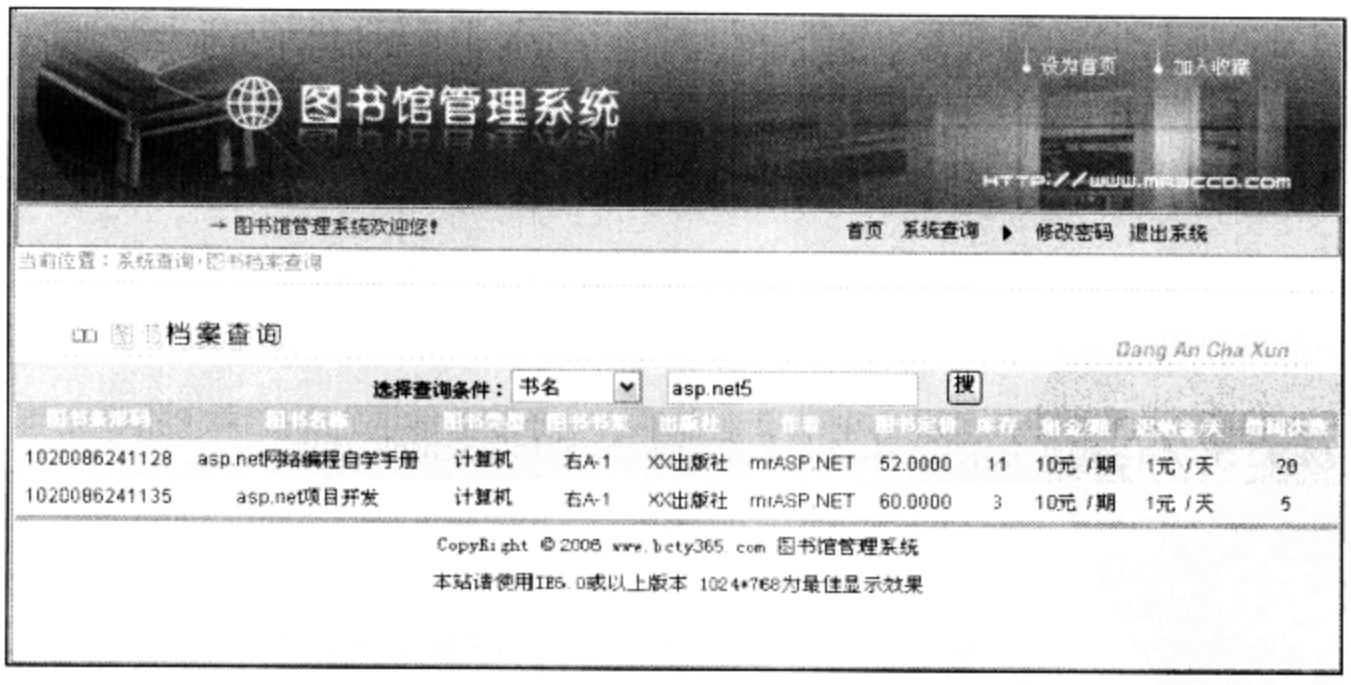


图 10.9 图书档案查询页面

1. 前台页面的设计

（1）创建 1 个 Web 窗体，命名为 bookInfoSearch.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 10.14 所示。

表 10.14　　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/TextBox 控件	txtKey	均为默认值	输入搜索的值
标准/Button 控件	btnSearch	均为默认值	实现搜索操作
标准/GridViewt 控件	gvBookInfo	均为默认值	显示图书的信息
标准/ DropDownList 控件	ddlCondition	均为默认值	选择搜索条件
	ddlBookType	设置 Visible 属性为 False	选择图书类型

2. 后台代码的编写

条件选择下拉列表框的 SelectedIndexChanged 事件，用来显示或隐藏选择图书类型的下拉列表框。该事件用于判断当前选择的条件是否为图书类型。如果是隐藏输入条件值的文本框，将显示选择图书类型的下拉列表框并调用自定义方法 bindBookType 绑定该下拉列表框中的值。否则将隐藏图书类型的下拉列表框，显示输入条件值的文本框。实现代码如下。

例程 28　代码位置：光盘\mr\10\bookManage\bookInfoSearch.aspx

```
protected void ddlCondition_SelectedIndexChanged(object sender, EventArgs e)
{
    //判断当前选择的值是否为图书类型
    if (ddlCondition.SelectedValue.Trim() == "bookType")
    {
        //隐藏文本框
        txtKey.Visible = false;
        //显示选择图书类型的下拉列表框
        ddlBookType.Visible = true;
        //隐藏验证控件
        RequiredFieldValidator1.Visible = false;
        //调用自定义方法绑定图书类型信息
        bindBookType();
    }
    else
    {
        //显示文本框
        txtKey.Visible = true;
        //显示验证控件
        RequiredFieldValidator1.Visible = true;
        //隐藏选择图书类型的下拉列表框
        ddlBookType.Visible = false;
    }
}
```

在自定义 bindBookTye 方法中绑定选择图书类型下拉列表框中的值。在该方法中将通过 SQL 语句查询出图书类型信息。将图书类型信息绑定到下拉列表框中，并设置显示的文本和选择项的值。实现代码如下。

例程 29　代码位置：光盘\mr\10\bookManage\bookInfoSearch.aspx

```
public void bindBookType()
{
    //创建SQL语句，查询图书的类型信息
    string typeSql = "select * from tb_bookType";
    //调用公共类中的getDataset方法执行SQL语句，并将返回值设置为数据源
    ddlBookType.DataSource = dataOperate.getDataset(typeSql);
    //下拉列表中设置显示的文本
    ddlBookType.DataTextField = "typeName";
    //设置下拉列表的值
    ddlBookType.DataValueField = "typeID";
    ddlBookType.DataBind();
}
```

在“搜”按钮的单击事件中实现图书的查询功能，在该事件中首先将获取读者选择的查询条件，再获取用户选择或输入的查询值。通过查询条件和查询值来创建 SQL 语句查询图书信息。实现代码如下。

例程 30 代码位置：光盘\mr\10\bookManage\bookInfoSearch.aspx

```
protected void btnSearch_Click(object sender, EventArgs e)
{
    //获取查询条件
    string Condition = ddlCondition.Text;
    string key = "";
    //判断下拉列表框是否显示
    if (ddlBookType.Visible)
    {
        //获取下拉列表框中选择的值
        key = ddlBookType.SelectedValue;
    }
    else
     //获取文本框中输入的值
     key = txtKey.Text;
    //创建SQL语句，查询图书信息
     sql = "select * from view_bookInfo where " + Condition + " like '%" + key + "%'";
    //调用自定义方法将图书信息显示在GridView控件上
     bindBookInfo(sql);
    txtKey.Text = "";
}
```

10.5 网站打包与发布

网站开发完成后最终的目地是将其发布到 Internet 上，以提供用户浏览访问。实现的网站发布可以使用两种方法，第 1 种方法是使用 Visual Studio 2008 开发工具提供的“发布网站”工具。第 2 种方法是使用 FTP 工具将网站发布到 Internet。下面主要介绍使用 Visual Studio 2008 开发工具提供的“发布网站”工具发布网站。

（1）首先打开要发布的网站，在菜单栏中选择“生成”选项，在弹出的快捷方式菜单中选择“发布网站”选项，如图 10.10 所示。

（2）在“发布网站”窗口的目标位置中用户可以在此处输入一个保存的路径，然后单击“确定”按钮，网站会被编译并保存到所指定的路径下，用户可以使用所下载的 FTP 工具将所编译的文件上传到指定的 Intenet 上，如果要使用开发工具自带的 FTP 工具就需要选择“...”路径按钮，如图 10.11 所示。

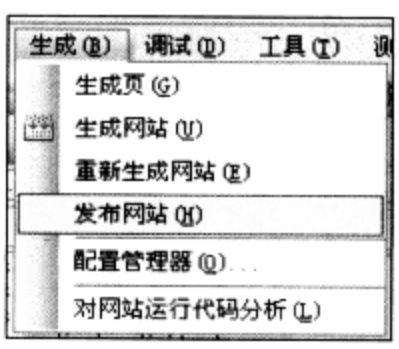


图 10.10 选择“发布网站”

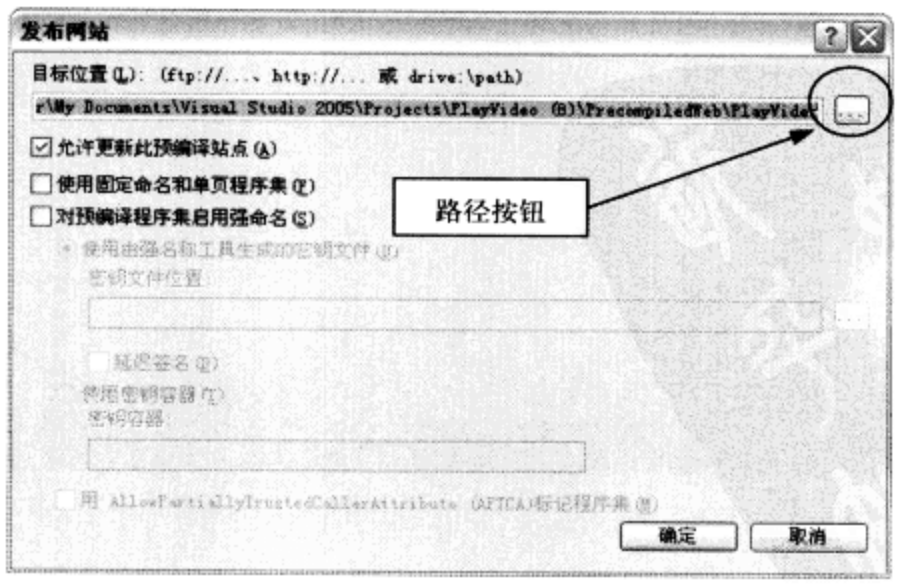


图 10.11 选择路径按钮

（3）在弹出的窗口中，选择“FTP 站点”选项，在该选择中会显示需要填写的相应信息，如服务器地址、目录、用户名及密码如图 10.12 所示。填写完毕后选择“打开”按钮将会返回“发布网站”窗口，在该窗口中选择“确定”按钮，网站就会发布到 Internet 上。

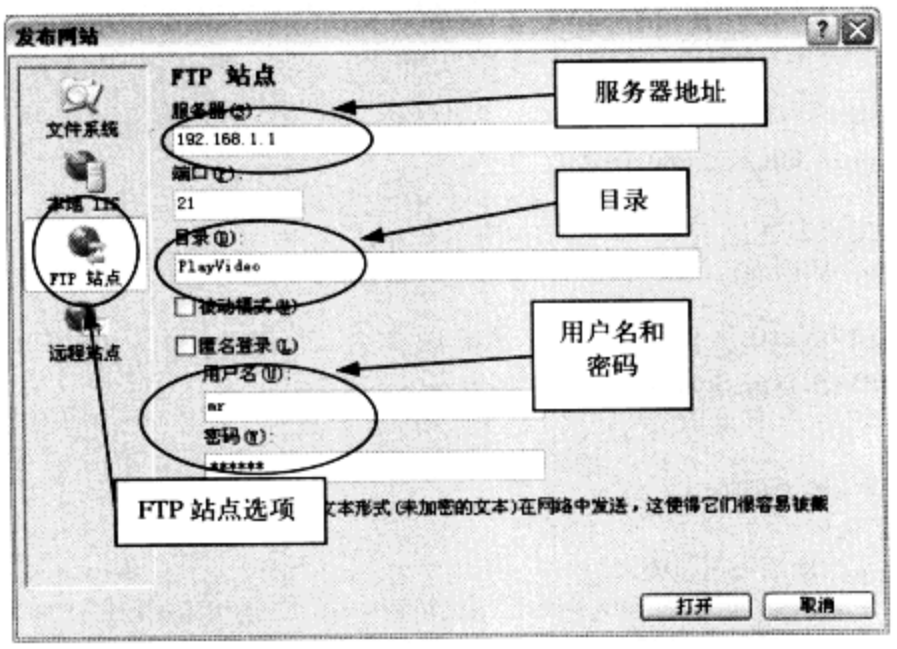


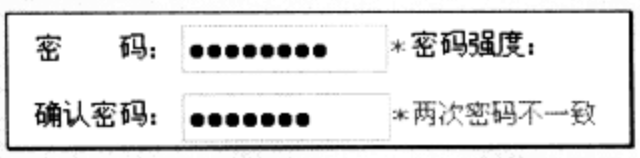
图 10.12　FTP 站点选项

第11章 会员注册登录模块

实例位置：光盘\mr\11\

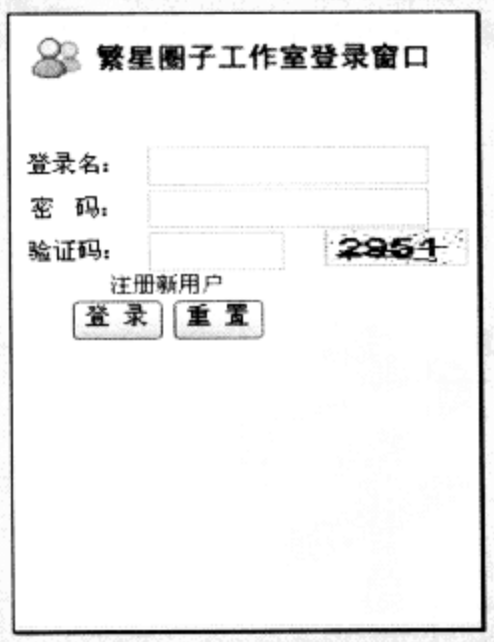
现在的网站功能越来越多，但很多网站都需要用户成为本网站的会员后，才能使用网站中的更多功能。这就需要在网站中提供会员注册和登录功能。在开发会员注册和会员登录时需要考虑到网络的安全问题，例如在登录时使用非法程序破解密码等问题。本章将介绍如何实现会员的登录和会员的注册。通过本章，读者可以学到以下内容。

密码验证

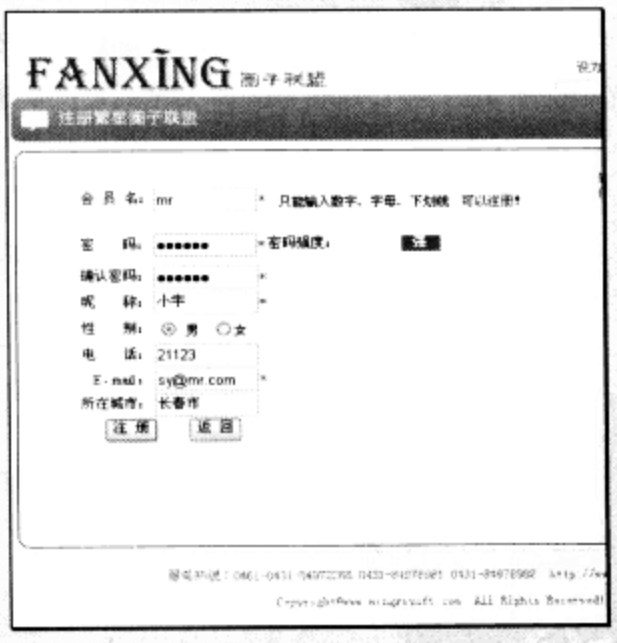


会员名自动验证及密码提示功能

用户登录功能



会员注册功能



11.1 概　　述

11.1.1 功能概述

在程序开发时会员注册登录是比较重要的模块，因为该模块需要考虑的东西比较多。例如，验证会员名、密码的安全性、防止 SQL 注入攻击和验证码等一些功能。在本章中读者将了解到这些功能的开发思路和实现的过程。可以提高读者基础模块的开发能力。

11.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建一个名为 db_GetPass 的数据库，在该数据库中创建一个 tb_User 表，该表用来存储用户注册的会员信息。该表的表结构如表 11.1 所示。

表 11.1　　tb_User 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动编号
userName	varchar	20	用来登录的会员名
userPass	varchar	50	用来登录的密码
nickName	varchar	20	用户的昵称
sex	char	10	用户的性别
phone	varchar	15	联系电话
E-mail	varchar	50	电子邮件地址
city	varchar	50	所在城市

11.1.3 会员注册流程图

在会员注册时用户需要填写注册信息，当填写完某个注册信息时会判断一下用户填写的是否正确，如果不正确将给出相应的提示。最后用户单击“注册”按钮即可实现会员注册功能。会员注册的流程图如图 11.1 所示。

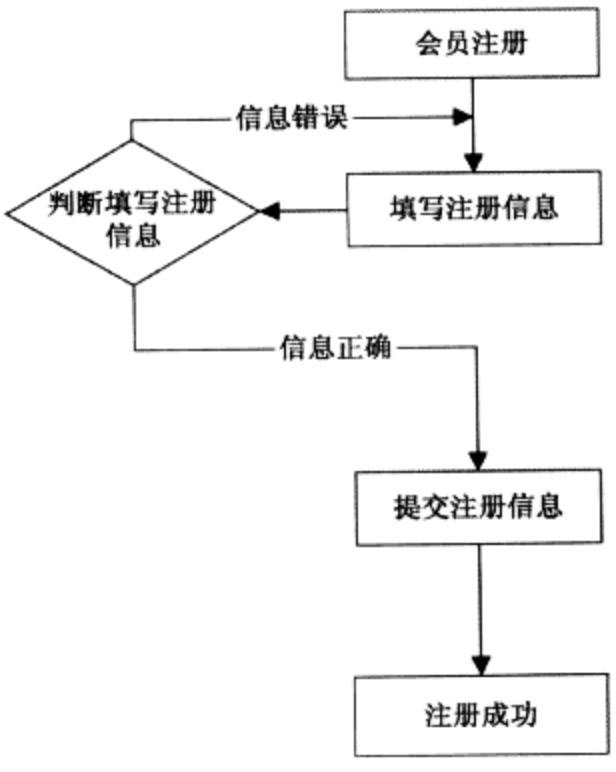


图 11.1　会员注册流程图

11.2 关 键 技 术

11.2.1 防止 SQL 注入式攻击

在判断用户输入的会员名和密码是否和数据库中的相同时，需要注意一下 SQL 注入式攻击，SQL 注入式攻击是指利用设计上的漏洞，在目标服务器上运行 SQL 命令以及进行其他方式的攻击。例如，SQL 注入式攻击，假如在登录页面里添加一个文本框用于输入会员名，一个按钮用来登录。在文本框中输入会员名“mr”，然后用 SQL 语句查找出数据库中符合条件的记录。SQL 语句如下：

```
Select   count(*) from LoginInfo where Name='mr'
```

通过上面的语句可以在数据库中查询出一条 Name 字段为 mr 的用户信息。如果在文本框中输入“mr'or'1'='1”那么 SQL 语句如下：

```
Select   count(*) from LoginInfo where Name='mr' or '1'='1'
```

这样一条 SQL 语句能够查找出 LoginInfo 表中的所有记录，为了防止这样 SQL 语句的攻击，通常使用 SqlCommand.Parameters 属性传参的方法将非法字符过滤掉。首先需要添加参数的名称、类型和大小。最后设置参数的值。下面的代码将设置 name 参数。实现代码如下：

```
com.Parameters.Add(new SqlParameter("@name", SqlDbType.VarChar, 50));
com.Parameters["@name"].Value = TextName.Text;
```

11.2.2 验证码技术

所谓验证码，就是将一串随机产生的数字或字母，生成一幅图片，在图片里加上一些干扰像素，由用户肉眼识别其中的验证码信息，并在文本框中输入正确的验证码，验证成功后才能使用某些功能。验证码能够有效地防止非法用户用特定程序暴力破解方式，进行不断地登录尝试来破解密码。实际上用验证码是现在很多网站通行的方式，虽然登录麻烦一点，但是还是很有必要的。不少网站为了防止用户利用机器人自动注册、登录、灌水，都采用了这个技术。

11.2.3 验证码的绘制

在本程序中的验证码是使用 4 位随机数字和背景噪点组合而成的。随机数的生成主要是通过 Random 对象中的 Next 方法来实现的。Next 方法说明如下。

Next 方法用于返回非负数的随机数。该方法语法如下：

```
public virtual int Next ()
```

返回值：大于或等于零且小于整型最大可能值的 32 位带符号整数。

4 位随机数生成后，就实现了添加噪点的操作，该操作的目的是使随机数不会轻易地被非法程序辨别。噪点主要是使用 Graphics 对象中的 DrawLine 方法添加彩色噪点线和 Bitmap 对象中的 SetPixel 方法添加彩色噪点。这两个方法的详细说明如下。

- DrawLine 方法

该方法用来绘制一条连接由坐标对指定的两个点的线条。语法如下：

```
public void DrawLine (
    Pen pen,
    int x1,
    int y1,
    int x2,
    int y2
)
```

参数说明如下。

- Pen：它确定线条的颜色、宽度和样式。
- x1：第 1 个点的 x 坐标。

- y1：第 1 个点的 *y* 坐标。
- x2：第 2 个点的 *x* 坐标。
- y2：第 2 个点的 *y* 坐标。
- SetPixel 方法

该方法用来获取 Bitmap 中指定像素的颜色。语法如下：

```
public void SetPixel (
    int x,
    int y,
    Color color
)
```

参数说明如下。

- x：要设置的像素的 *x* 坐标。
- y：要设置的像素的 *y* 坐标。
- color：表示要分配给指定像素的颜色。

11.2.4 Ajax 验证会员名是否存在

在验证会员名时使用了 Ajax 局部无刷新功能。要实现局部无刷新，首先应该添加 2 个控件，即 ScriptManager 控件和 UpdatePanel 控件。ScriptManager 控件又称为脚本管理控件，它能够管理 Web 页上的脚本。同时使用 ScriptManager 控件和 UpdatePanel 控件，可实现界面无刷新的 Web 环境。ScriptManager 控件不但能够动态创建与 Web 服务器相关的脚本，而且这些脚本也能支持 Web 页上的局部更新功能。

目前比较流行的无刷新验证会员名的方式是，当用户输入会员名后，再单击其他文本框时，将会自动提示用户输入的会员名是否已注册，效果如图 11.2 所示。

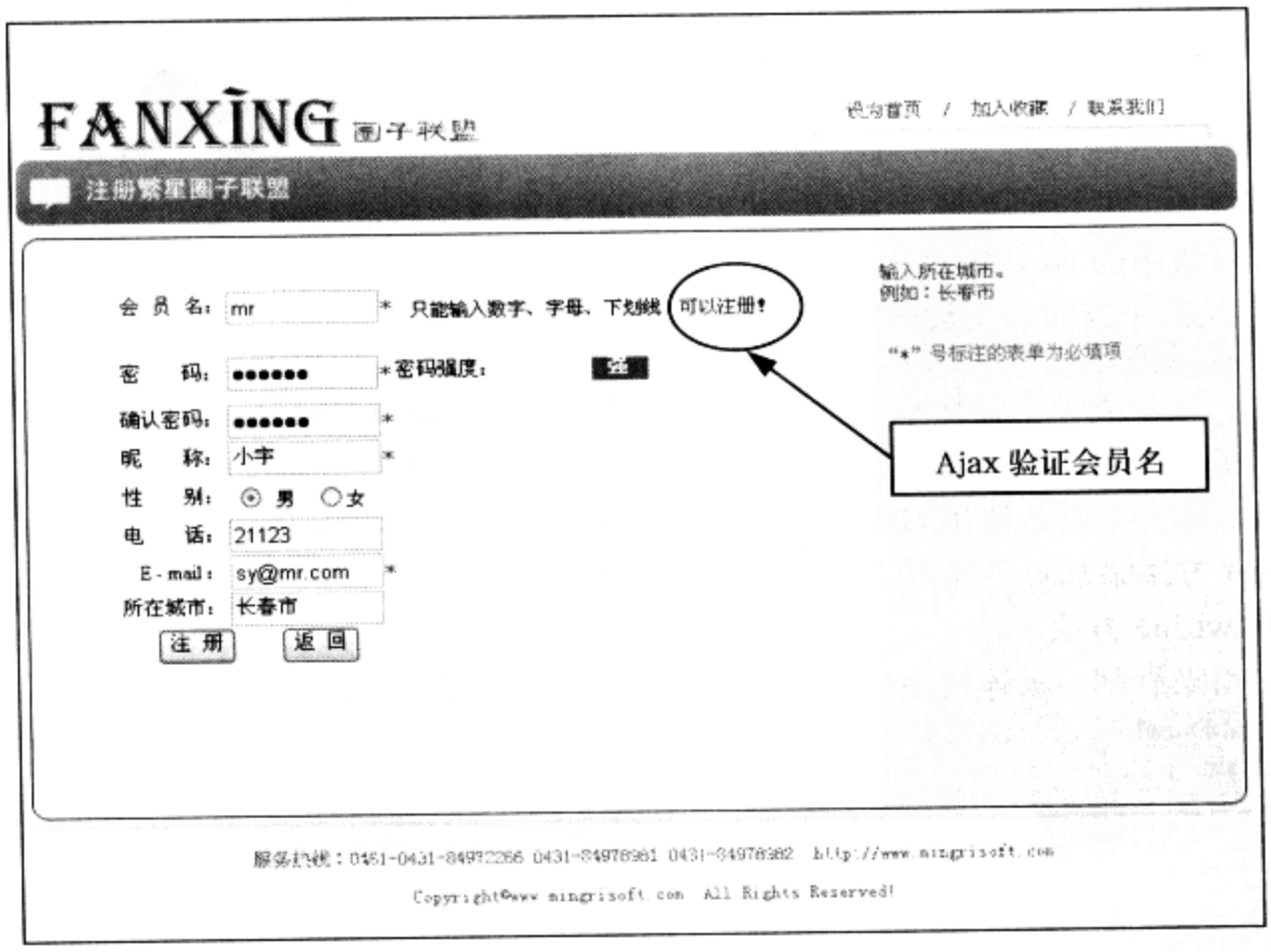


图 11.2 Ajax 验证会员名

实现这种方式首先需要将用来输入会员名的文本框控件和提示是否已注册的 Label 控件添加到 UpdatePanel 控件中，这样就可以实现局部更新功能。Label 控件提示会员名是否已注册，主要

是通过使用文本框中的 textChanged 事件来实现。该事件是在文本框内容发生改变且失去焦点时引发的。必须注意的是，使用该事件还需要将文本框 AutoPostBack 属性设置为 True，使文本修改后自动回发到服务器。在 textChanged 事件中将通过 SQL 语句来查询用户输入的会员名是否存在于数据表中，如果存在，则将设置 Label 控件进行提示，否则将提示会员名可以注册。

11.2.5 密码强弱提示

密码强弱提示是对用户填写登录密码的复杂程度来给出相应的提示。使用该技术可以增强用户对自己密码的保护意识，对如今的网络来说这样做是非常必要的。在本程序中，当用户输入密码后，将会自动提示用户输入密码的强弱，效果如图 11.3 所示。

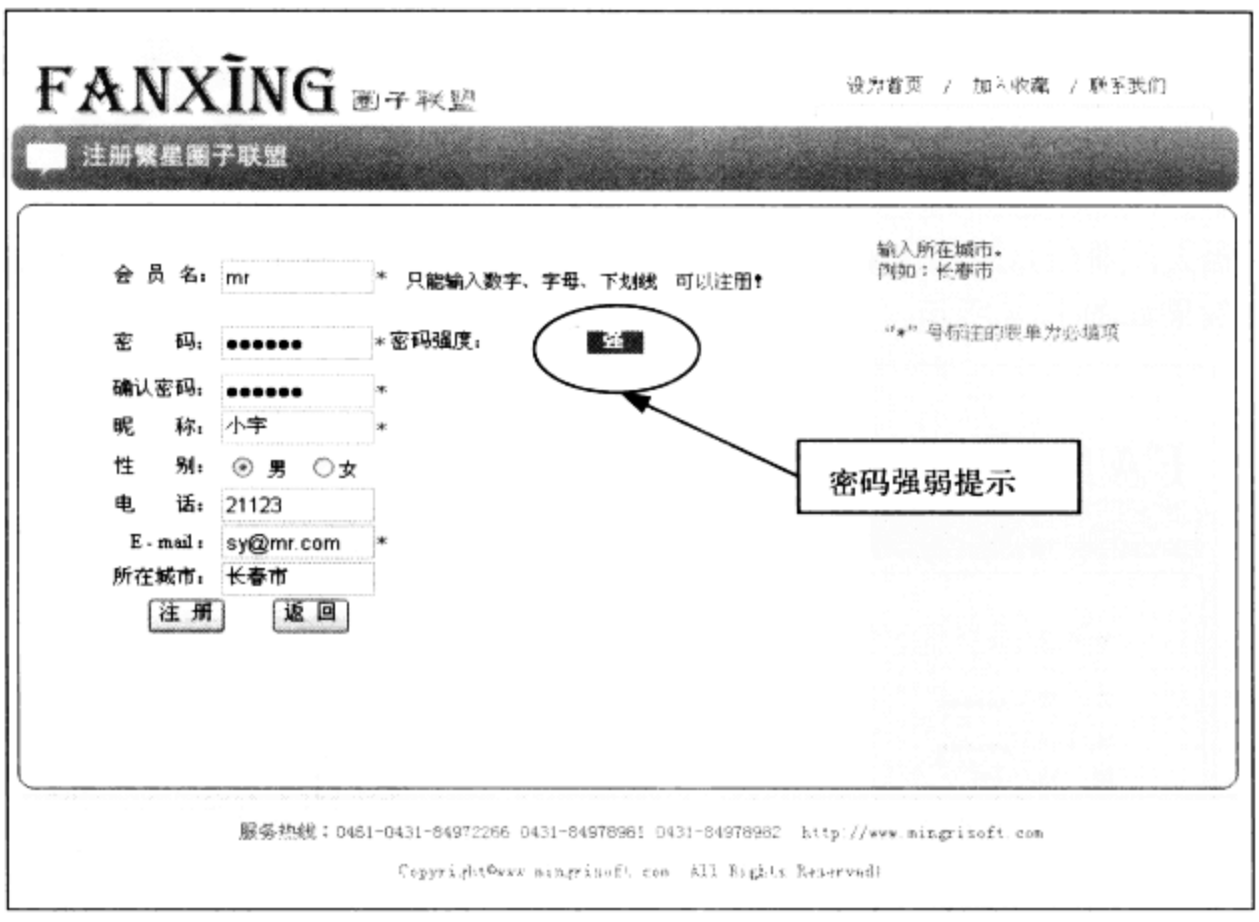


图 11.3 密码强弱提示

该技术是使用 JavaScript 脚本用于判断用户输入的密码位数是否大于 6 位，如果小于 6 将设置单元格“弱”的背景颜色，否则将设置“强”的单元格背景颜色。JavaScript 的实现代码如下。

例程 1 代码位置：光盘\mr\11\RegisterLogin\Register.aspx

```
<script>
    function passHint()
    {
        var txt=document.getElementById('txtPass').value;
        if(txt.length<6)
        {
            document.all("tab").rows[0].cells[1].bgColor="red";
            document.all("tab").rows[0].cells[2].bgColor="";
        }else
        {
        document.all("tab").rows[0].cells[2].bgColor="red";
                document.all("tab").rows[0].cells[1].bgColor="";
        }
    }</script>
```

在输入密码文本框中的 onChange 事件中将调用 JavaScript 中的 passHint 函数，来判断用户填写的密码强度并给出相应的提示。onChange 事件将会在用户填写密码后选择其他控件时引发。密码文本框的前台代码如下：

```
<asp:TextBox ID="txtPass"  runat="server"  onchange="passHint()"  TextMode="Password"
Width="88px"></asp:TextBox>
```

11.2.6 MD5 加密

为了提高密码的安全性。在本实例中将密码存入数据库前使用了 MD5 加密。加密一般有两种，即双向加密和单向加密。双向加密最常用，它既能加密又能解密；单向加密只能对数据进行加密，不能对其解密。MD5 加密就是单向加密。MD5 加密是根据指定的密码和哈希算法生成一个适合于存储在配置文件中的哈希密码。命名空间为 System.Web.Security。语法如下：

```
public static string HashPasswordForStoringInConfigFile( string password,string passwordFormat )
```

参数说明如下。

- password：要进行哈希运算的密码。
- passwordFormat：要使用的哈希算法。PasswordFormat 是一个 String，表示 FormsAuthPasswordFormat 枚举值之一。

11.2.7 智能提示输入信息

在用户输入注册信息时给出提示，可以使用户在注册时更了解所填写信息的约束。智能提示输入信息效果如图 11.4 所示。

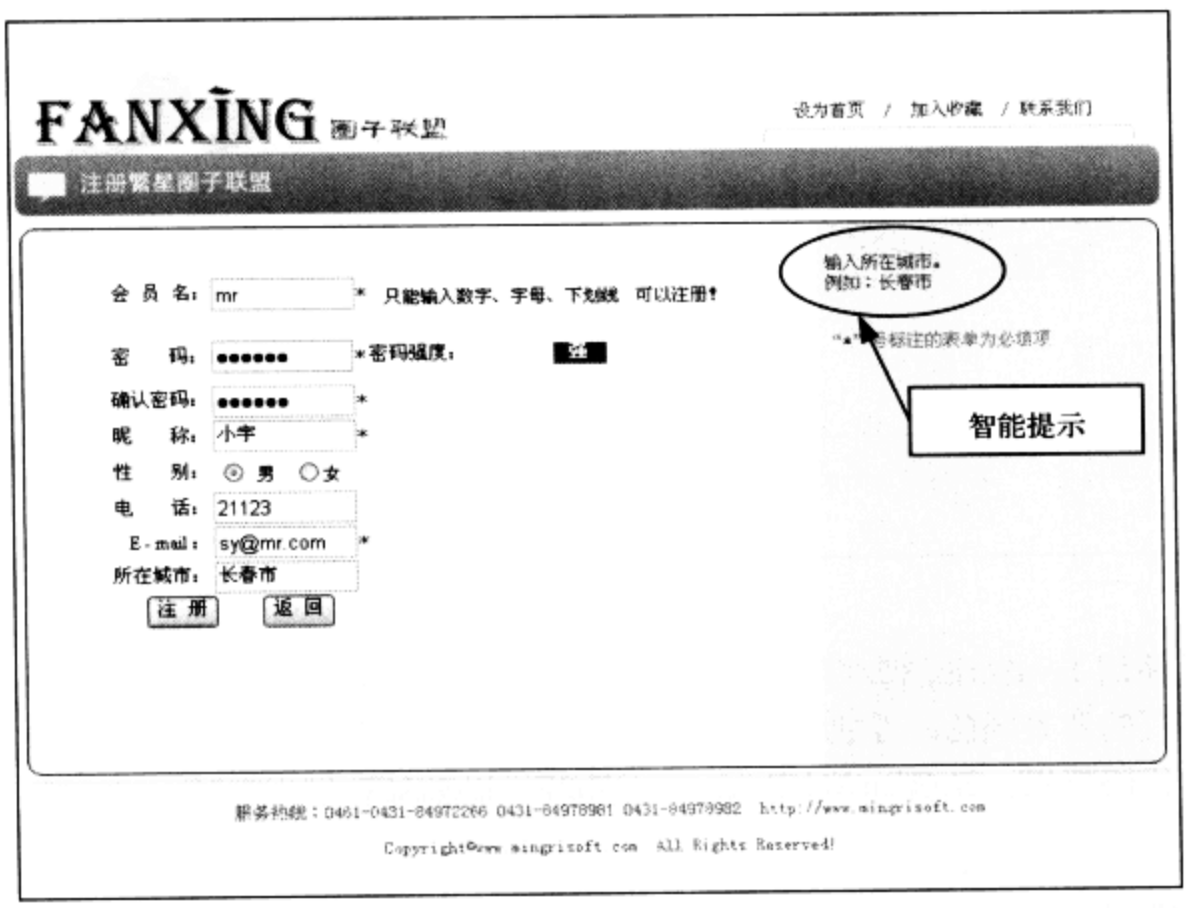


图 11.4 智能提示输入信息

这个功能主要是使用 JavaScript 脚本和文本框中的获取焦点事件 onFocus 来实现。当用户单击某个文本框时将会引发该文本框的 onFocus 事件，在该事件中将会调用 JavaScript 脚本中已经编写好的函数。在每个函数中将会通过 HTML 的 span 标记来显示提示信息。JavaScript 脚本中的各个函数代码如下。

例程 2 代码位置：光盘\mr\11\RegisterLogin\Register.aspx

```
    <script>
//显示会员名输入提示
function tName()
    {
      document.getElementById("sp").innerHTML="只能输入数字、字母下划线,<br>例如：mr_2008";
    }
```

```
//显示密码输入提示
function tPass()
    {
        document.getElementById("sp").innerHTML="为了提供密码的安全性。<br>建议密码在6位以上。";
    }
    //显示昵称输入提示
 function tNickName()
 {
        document.getElementById("sp").innerHTML="在圈子中使用的昵称。<br>例如：宇过天晴";
 }
 //显示电话码输入提示
function tPhone()
{
        document.getElementById("sp").innerHTML="输入手机号，以方便联系您<br>手机号应为11位";
}
//显示电子邮件输入提示
function tEmail()
{
        document.getElementById("sp").innerHTML="请输入正确的电子邮件。<br>例如：mr2008@mr.com";
}
//显示所在城市输入提示
function tCity()
{
    document.getElementById("sp").innerHTML="输入所在城市。<br>例如：长春市";
}
    </script>
```

11.3 实 现 过 程

11.3.1 用户登录设计

用户登录页面用于实现用户的登录功能，在该页面中用户还可以跳转到用户注册页面中。用户登录页面如图 11.5 所示。

图 11.5 用户登录页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，默认名为 Default.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 11.2 所示。

表 11.2　　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/TextBox 控件	txtUserName	均为默认值	用于输入登录名
	txtUserpass	TextMode 属性设置为 Password	用于输入登录密码
	txtCode	均为默认值	用于输入验证码
HTML/Image 控件	imgCode	Alt 属性设置为“看不清，请点击我！”	用于显示验证吗
标准/LinkBtn 控件	likbtnRegister	PostBackUrl 属性设置为～/Register.aspx CausesValidation 属性设置为 False	用于跳转注册新用户页面
标准/Button 控件	btnLog	均为默认值	用于实现登录操作

2. 后台代码编写

在“登录”按钮的单击事件中，先判断用户输入的验证码，如果用户输入的验证码正确才可进行用户登录的操作。用户登录操作主要是通过使用 SQL 语句在数据库中进行查询来实现的。这里需要注意一下用户输入的会员密码，在用户注册时为了增加保密性，对会员密码进行了 MD5 加密并保存到数据库中，在此也应该将会员密码通过 MD5 加密后再使用 SQL 语句查询。为了防止 SQL 注入式攻击，这里使用了 Parameters 属性以传入参数的方法来设置 SQL 语句。实现代码如下。

例程 3　代码位置：光盘\mr\11\RegisterLogin\Default.aspx.cs

```
protected void btnLog_Click(object sender, EventArgs e)
    {
        //获取验证码
        string code = txtCode.Text;
        //判断用户输入的验证码是否正确
        if (Request.Cookies["CheckCode"].Value == code)
        {
            //创建数据库连接
            SqlConnection con = new SqlConnection("server=.;database=db_Register;uid=sa;pwd=;");
            //打开数据库连接
            con.Open();
            //使用MD5加密，将用户输入的密码加密
            string pass = FormsAuthentication.HashPasswordForStoringInConfigFile(txtUserpass.Text, "MD5");
            //创建SQL语句，用来查询用户输入的用户名和密码是否正确
            string sqlSel = "select count(*) from tb_userInfo where userName=@name and userPass=@pass";
            //创建SqlCommand对象
            SqlCommand com = new SqlCommand(sqlSel, con);
            //使用Parameters的add方法添加参数类型
            com.Parameters.Add(new SqlParameter("name", SqlDbType.VarChar, 20));
            //设置Parameters的参数值
            com.Parameters["name"].Value = txtUserName.Text;
            com.Parameters.Add(new SqlParameter("pass", SqlDbType.VarChar, 50));
            com.Parameters["pass"].Value = pass;
            //判断ExecuteScalar方法返回的参数是否大于表示登录成功，并给出提示
            if (Convert.ToInt32(com.ExecuteScalar()) > 0)
            {
                RegisterStartupScript("", "<script>alert('登录成功！')</script>");
                //清空文本框
                txtCode.Text = txtUserName.Text = "";
            }
            else
            {
                RegisterStartupScript("", "<script>alert('用户名或密码错误！')</script>");
            }
        }
        else
        {
            RegisterStartupScript("", "<script>alert('验证码输入错误！')</script>");
        }
    }
```

11.3.2 会员注册设计

会员注册页面用来实现会员的注册功能，在该页面中用户需要填写会员的基本信息，如会员名、密码、昵称等。会员注册页面如图 11.6 所示。

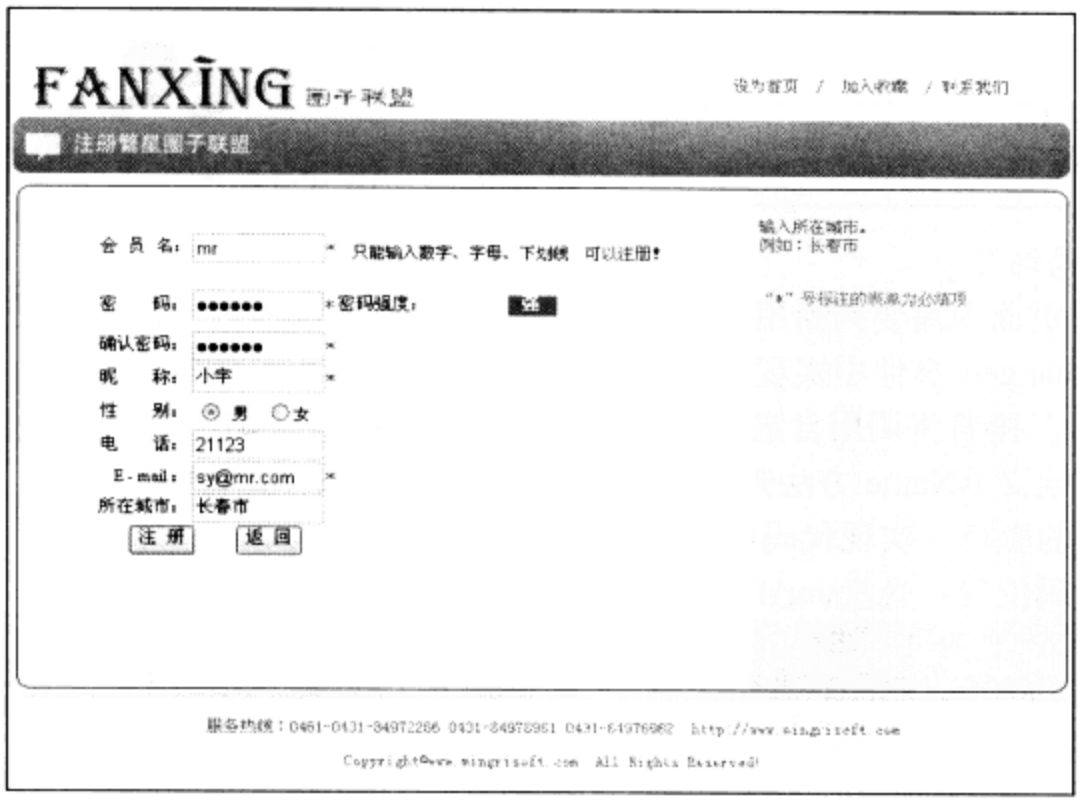


图 11.6 会员注册页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 Register.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 11.3 所示。

表 11.3 控件类型、控件名称及说明

控件类型	控件名称	主要属性	说明
标准/TextBox 控件	txtName	AutoPostBack 属性设置为 True	用于输入登录名
	txtPass	TextMode 属性设置为 Password	用于输入登录密码
	txtQpass	TextMode 属性设置为 Password	用于输入验证码
	txtNickname	均为默认值	用于输入昵称
	txtPhone	均为默认值	用于输入电话号码
	txtEmail	均为默认值	用于输入电子邮件地址
	txtCity	均为默认值	用于输入所在城市
标准/RadioButtonList 控件	radlistSex	均为默认值	用于显示验证码
标准/Label 控件	labUser	均为默认值	提示用户输入的会员名是否满足要求
	labIsName	均为默认值	提示用户名是否已注册
	labEbb	均为默认值	显示密码安全性为“弱”提示
	labStrong	均为默认值	显示密码安全性为“强”提示
标准/Button 控件	btnRegister	均为默认值	用于实现注册操作
	btnReturn	PostBackUrl 属性设置为~/Default.aspx CausesValidation 属性设置为 False	用于返回到登录页面

续表

控件类型	控件名称	主要属性	说明
验证/RequiredFieldValidator 控件	rfvName	ControlToValidate 属性设置为 txtName	用于验证用户名是否为空
验证/CompareValidator 控件	covPass	ControlToCompare 属性设置为 txtPass ControlToValidate 属性设置为 txtQpass	用于验证用户输入的两次密码是否一致
验证/RegularExpressionValidator 控件	revEmail	ControlToValidate 属性设置为 txtEmail ValidationExpression 属性设置为\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*	用于验证用户输入的邮件地址格式

2. 后台代码编写

在用户注册页面中需要判断用户填写的会员名是否已经注册过。该功能在填写“会员名”文本框的 TextChanged 事件中实现。在该事件中首先将判断用户填写的会员名是否为空，如果为空将给出提示。接着将调用自定义 isNameFomrmar 方法判断用户输入的会员名格式是否正确。最后调用自定义 isName 方法判断会员名是否存在。如果存在将使用 Label 控件给出提示并设置 Label 控件的颜色。实现代码如下。

例程 4 代码位置：光盘\mr\11\RegisterLogin\Register.aspx.cs

```
<!-- **********************例程1-4  Register.aspx.cs ************************-->
protected void txtName_TextChanged(object sender, EventArgs e)
    {
        //判断用户名是否为空
        if (txtName.Text == "")
        {
            //使用Label控件给出提示
            labIsName.Text = "用户名不能为空";
            //设置Label控件的颜色
            labIsName.ForeColor = System.Drawing.Color.Red;
        }
        else
        {
            //调用自定义isNameFormar方法判断用户名是否满足格式要求
            if (isNameFormar())
            {
                //调用isName自定义方法判断用户名是否已注册
                if (isName())
                {
                    labIsName.Text = "用户名已存在！";
                    labIsName.ForeColor = System.Drawing.Color.Red;
                }
                else
                {
                    labIsName.Text = "可以注册！";
                    labIsName.ForeColor = System.Drawing.Color.Blue;
                }
            }
            else
            {
                labIsName.Text = "";
            }
        }
    }
```

自定义 isNameFormar 方法用来判断用户填写的会员名格式是否正确。用户填写的会员名只能为字母、数字和下划线。在该方法中主要通过使用 Regex 对象的 IsMatch 方法来判断会员名是否满足设置的正则表达式。该方法将返回一个布尔值，当用户填写的会员名格式正确，将返回布尔值 True，否则将返回 False。实现代码如下。

例程 5 代码位置：光盘\mr\11\RegisterLogin\Register.aspx.cs

```
<!-- **********************例程1-5  Register.aspx.cs ************************-->
    protected bool isNameFormar()
    {
        //创建1个布尔型变量并初始化为False
        bool blNameFormar = false;
        //设置正则表达式
        Regex re = new Regex("^\\w+$");
        //使用Regex对象中的IsMatch方法判断用户名是否满足正则表达式
        if (re.IsMatch(txtName.Text))
        {
            //设置布尔变量为True
            blNameFormar = true;
            //设置Label控件的颜色
            labUser.ForeColor = System.Drawing.Color.Black;
        }
        else
        {
            labUser.ForeColor = System.Drawing.Color.Red;
            blNameFormar = false;
        }
        //返回布尔型变量
        return blNameFormar;
    }
```

自定义 isName 方法用来判断用户输入的会员名是否存在。在该方法中通过使用 SQL 语句查询表中使用已存在的会员名，如果存在该方法将返回一个布尔值 True，否则将返回 False。实现代码如下。

例程 6 代码位置：光盘\mr\11\RegisterLogin\Register.aspx.cs

```
    protected bool isName()
    {
        //创建1个布尔型变量并初始化为False
        bool blIsName = false;
        //创建SQL语句，用来判断用户名是否存在
        string sqlSel = "select count(*) from tb_userInfo where userName='" + txtName.Text + "' ";
        //创建数据库连接
        SqlConnection con = new SqlConnection("server=.;database=db_Register;uid=sa;pwd=;");
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sqlSel, con);
        //判断ExecuteScalar方法返回的参数是否大于0，若大于则表示用户名已存在
        if (Convert.ToInt32(com.ExecuteScalar()) > 0)
        {
            blIsName = true;
        }
        else
        {
            blIsName = false;
        }
        //返回布尔值变量
        return blIsName;
    }
```

在“注册”按钮的单击事件中，先判断会员名的格式是否正确，如果正确将判断会员名是否不存在。当两个条件都满足后将获取用户填写的会员信息，为了提高会员密码的保密性，将使用 MD5 加密后在添加到数据库中。实现代码如下。

例程 7 代码位置：光盘\mr\11\RegisterLogin\Register.aspx.cs

```
    protected void btnRegister_Click(object sender, EventArgs e)
    {
        //调用isNameFormar自定义方法判断用户名输入的是否满足要求
        if (isNameFormar())
        {
```

```
//调用自定义isName方法判断用户名是否存在
if (isName())
{
    //使用Label控件显示提示信息
    labIsName.Text = "用户名已存在！";
    //设置Label控件的颜色
    labIsName.ForeColor = System.Drawing.Color.Red;
    RegisterStartupScript("", "<script>alert('请正确填写信息！')</script>");
}
else
{
    //获取用户填写的会员名
    string userName = txtName.Text;
    //获取用户填写的密码并使用MD5进行加密
    string userPass = FormsAuthentication.HashPasswordForStoringInConfigFile(txtPass.Text, "MD5");
    //获取昵称
    string nickname = txtNickname.Text;
    string sex = "";
    //获取用户选择的性别
    if (radlistSex.SelectedValue.Trim() == "男")
    {
        sex = "男";
    }
    else
    {
        sex = "女";
    }
    //获取电话号
    string phone = txtPhone.Text;
    //获取电子邮件地址
    string email = txtEmail.Text;
    //获取所在城市
    string city = txtCity.Text;
    //创建SQL语句，用来添加用户的详细信息
    string sqlIns = "insert into tb_userInfo values('" + userName + "','" + userPass + "','" + nickname +
"','" + sex + "','" + phone + "','" + email + "','" + city + "')";
    //创建数据库连接
    SqlConnection con = new SqlConnection("server=.;database=db_Register;uid=sa;pwd=;");
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sqlIns, con);
    //判断ExecuteNonQuery方法返回的参数是否大于0，若大于0则表示注册成功
    if (com.ExecuteNonQuery() > 0)
    {
        RegisterStartupScript("", "<script>alert('注册成功！')</script>");
        //清空文本框中的信息
        txtName.Text = txtNickname.Text = txtPhone.Text = txtEmail.Text = txtCity.Text = "";
        labIsName.Text = "";
    }
    else
    {
        RegisterStartupScript("", "<script>alert('请正确填写信息！')</script>");
    }
}
else
{
    RegisterStartupScript("", "<script>alert('请正确填写信息！')</script>");
}

}
```

11.3.3 验证码设计

验证码是由 4 个随机数和一些使用 GDI+技术绘制的噪点而组成的。验证码可以防止非法用户使用暴力破解用户的密码。验证码如图 11.7 所示。

图 11.7 验证码

1. 前台页面设计

创建 1 个 Web 窗体，命名为 CheckCode.aspx。该页面用于在后台实现验证码功能。

2. 后台代码编写

实现验证码主要是通过在页面的加载事件中，调用自定义方法 CreateCheckCodeImage 方法，将 4 位验证码添加噪点并显示在页面上。由于 CreateCheckCodeImage 方法需要传入 1 个参数，该参数表示验证码，所以还需要调用 GenerateCheckCode 方法获取 4 位的数字。实现代码如下。

例程 8 代码位置：光盘\mr\11\RegisterLogin\Register.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        //调用自定义方法绘制验证码
        CreateCheckCodeImage(GenerateCheckCode());
    }
```

自定义 GenerateCheckCode 方法用来生成 1 个 4 位数，在该方法中主要通过使用 Random 对象来生成随机数，对该数进行取余并进行转换为字符型，并将该数保存到 Cookies 中以方便比较用户输入的验证码。最后将给 4 位数返回。实现代码如下。

例程 9 代码位置：光盘\mr\11\RegisterLogin\Register.aspx.cs

```
private string GenerateCheckCode()
{
    //创建整型变量
    int number;
    //创建字符型变量
    char code;
    //创建字符串变量并初始化为空
    string checkCode = String.Empty;
    //创建Random对象
    Random random = new Random();
    //使用For循环生成4个数字
    for (int i = 0; i < 4; i++)
    {
        //生成一个随机数
        number = random.Next();
        //将数字转换成为字符型
        code = (char)('0' + (char)(number % 10));
        checkCode += code.ToString();
    }
    //将生成的随机数添加到Cookies中
    Response.Cookies.Add(new HttpCookie("CheckCode", checkCode));
    //返回字符串
    return checkCode;
}
```

自定义 CreateCheckCodeImage 方法用来将验证码绘制噪点并显示在页面中。调用该方法需要传入 1 个字符串型变量，该变量表示 1 个 4 位的验证码。在该方法中通过使用 Graphics 对象对验证码进行绘制噪点功能，以达到验证码较模糊的目的。实现代码如下。

例程 10 代码位置：光盘\mr\11\RegisterLogin\Register.aspx.cs

```
private void CreateCheckCodeImage(string checkCode)
{
    //判断字符串不等于空和Null
    if (checkCode == null || checkCode.Trim() == String.Empty)
        return;
    //创建1个位图对象
```

```
            System.Drawing.Bitmap image = new System.Drawing.Bitmap((int)Math.Ceiling((checkCode.Length * 12.5)), 22);
            //创建Graphics对象
            Graphics g = Graphics.FromImage(image);
            try
            {
                //生成随机生成器
                Random random = new Random();
                //清空图片背景色
                g.Clear(Color.White);
                //画图片的背景噪音线
                for (int i = 0; i < 2; i++)
                {
                    int x1 = random.Next(image.Width);
                    int x2 = random.Next(image.Width);
                    int y1 = random.Next(image.Height);
                    int y2 = random.Next(image.Height);
                    g.DrawLine(new Pen(Color.Black), x1, y1, x2, y2);
                }
                Font font = new System.Drawing.Font("Arial", 12, (System.Drawing.FontStyle.Bold));
                System.Drawing.Drawing2D.LinearGradientBrush brush = new System.Drawing. Drawing2D.Linear Gradient
Brush (new Rectangle(0, 0, image.Width, image.Height), Color.Blue, Color.DarkRed, 1.2f, true);
                g.DrawString(checkCode, font, brush, 2, 2);
                //画图片的前景噪音点
                for (int i = 0; i < 100; i++)
                {
                    int x = random.Next(image.Width);
                    int y = random.Next(image.Height);
                    image.SetPixel(x, y, Color.FromArgb(random.Next()));
                }
                //画图片的边框线
                g.DrawRectangle(new Pen(Color.Silver), 0, 0, image.Width - 1, image.Height - 1);
                //将图片输出到页面上
                System.IO.MemoryStream ms = new System.IO.MemoryStream();
                image.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                Response.ClearContent();
                Response.ContentType = "image/Gif";
                Response.BinaryWrite(ms.ToArray());
            }
            finally
            {
                g.Dispose();
                image.Dispose();
            }
        }
```

11.4 程序调试与错误处理

验证会员名是否存在的功能，是在 TextBox 控件的 TextChanged 事件中实现的。在该事件中通过编写代码来判断用户输入的会员名是否存在，并给出相应的提示。完成编写 TextChanged 事件中的代码后，还需要设置 TextBox 控件的一个 AutoPostBack 属性。该属性用来设置当在文本框中按<Enter>键或选择其他控件时，是否发生自动回发到服务器的操作。该属性常常会被初学者忽略，而达不到预期的效果。

AutoPostBack 属性默认值为 False，不会引发自动回发到服务器的操作。当开发人员想要在 TextChanged 事件中实现某些操作时，此时必须将 AutoPostBack 属性默认值为 True。当引发 TextChanged 事件后将会自动回发到服务器，在服务器中进行相应的处理，从而实现 TextChanged 事件中的操作。

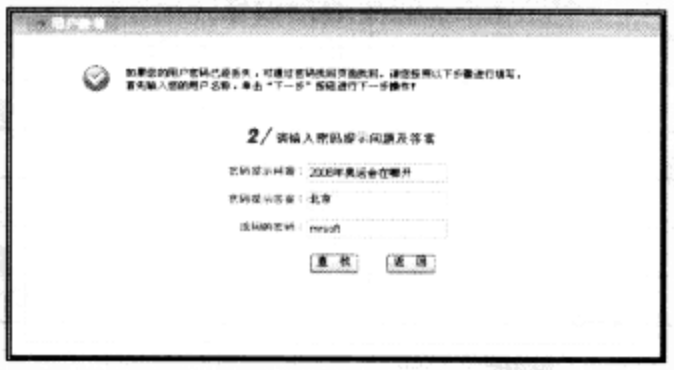
会员密码找回模块

第12章

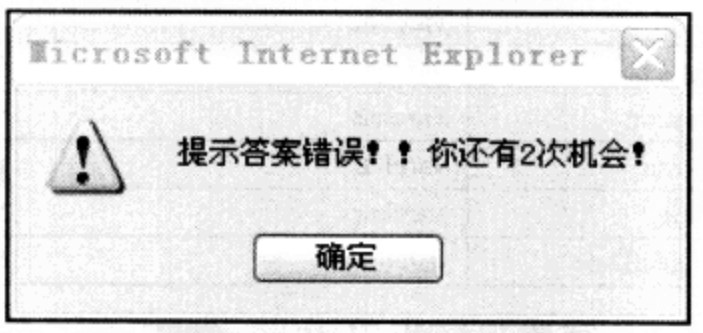
实例位置：光盘\mr\12\

随着网站对会员开放功能的增加，越来越多的网友都会通过网站注册成为会员。但是这样就有可能将某个网站的会员帐号或密码忘记。为了解决这一问题这里将介绍密码找回模块。现在比较流行的密码找回方式有两种，一种方式是通过使用回答密码提示问题来找回会员忘记的密码，另一种方式是在会员注册时将会员的登录名和密码信息，通过电子邮件发送到会员的邮箱中。本模块结合了这两种比较流行的密码找回方式，使读者掌握这两种密码找回方式的使用和设计思路。通过本章，读者能够学到以下内容。

密码找回功能

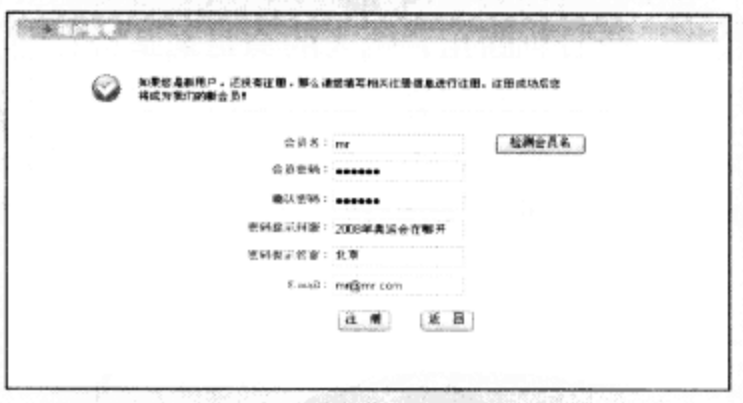


密码找回三次提示



用户登录功能

会员注册功能



12.1 概　　述

12.1.1 功能概述

会员密码找回功能，是提供给已注册会员的一个功能。该功能是为了解决会员忘记密码而带来的不必要的损失。本程序中的密码找回使用了两种方式，第 1 种是在用户添写注册信息时填写密码提示问题和密码提示答案。当会员忘记密码时可以进入密码找回页，在该页中将会显示用户填写的密码提示问题，只要填写正确的答案就会显示会员的密码。另 1 种方式是当用户填写完注册信息后单击“注册”按钮时，将会把用户注册的会员名和密码发送到用户填写的邮箱中，当用户忘记密码时可以直接到自己的邮箱中找回自己的密码。

12.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建一个名为 db_GetPass 的数据库，在该数据库中创建 1 个名为 tb_User 的表，该表用来存储用户注册的会员信息。该表的表结构如表 12.1 所示。

表 12.1　　tb_User 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动编号
Name	varchar	50	用来登录的会员名
Pass	varchar	50	用来登录的密码
Question	varchar	50	密码提示问题
Answer	varchar	50	密码提示答案
Email	varchar	50	电子邮件地址
CongealDate	datetime	8	冻结密码找回功能的日期

12.1.3 密码找回流程图

在会员密码找回过程中，用户首先需要输入要找回密码的会员名，输入后单击“下一步”按钮，此时判断用户输入的会员名是否存在，如果存在才可以进入回答密码提示问题页面，当用户填写完密码回答问题后单击“查找”按钮，判断用户输入的密码提示问题是否正确，如果正确将显示密码，不正确还需要判断用户是第几次回答问题，如果回答问题超过 3 次将会冻结此功能 24 小时。密码找回流程图如图 12.1 所示。

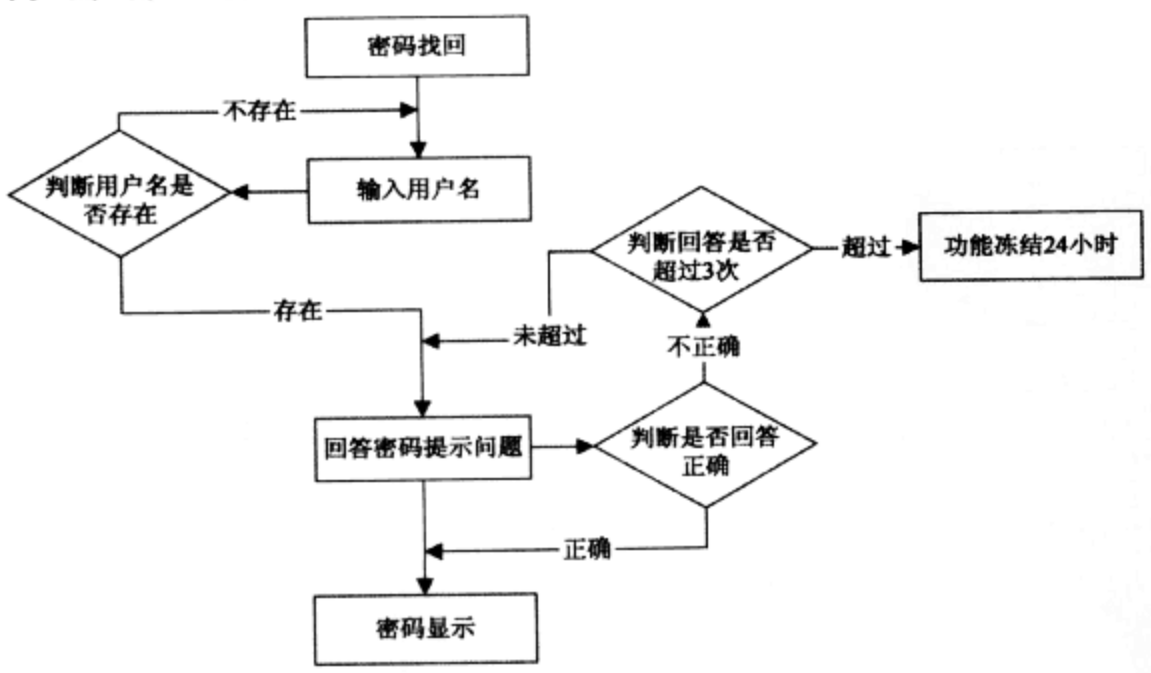


图 12.1　密码找回流程图

12.2 关 键 技 术

12.2.1 会员名验证技术

会员名验证是为了保证会员名的惟一性，目前普遍的会员登录都是通过使用会员名进行登录，如果会员名不惟一就有可能引发登录错误。会员名验证技术首先获取到用户填写的会员名，在通过会员名在数据库的会员注册信息表中进行查询，如果有相同的会员名说明用户填写的会员名已经存在，并给出提示。会员名验证如图 12.2 所示。

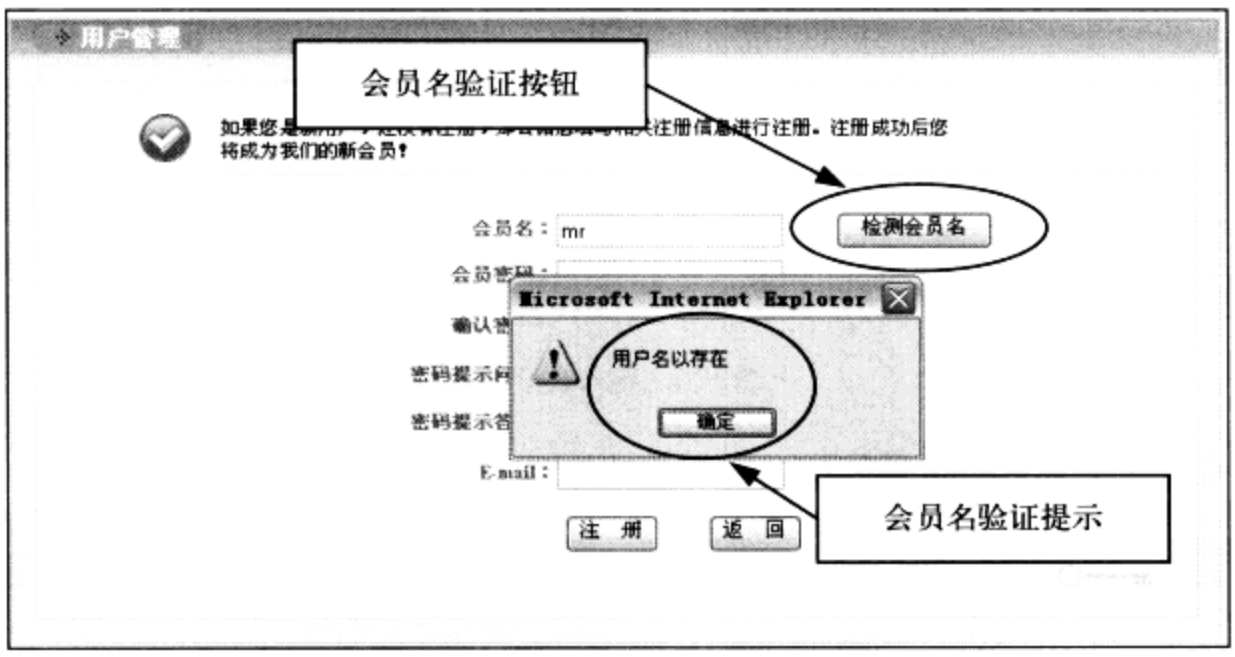


图 12.2 会员名验证

会员名验证是使用“检测会员名”按钮来实现的，在该按钮的单击事件中调用自定义的 isName 方法判断会员名是否存在。调用该方法需要传入一个参数，该参数为字符串类型表示用户填写的会员名。在该方法中通过连接数据库，并使用 SQL 语句判断会员名是否存在，如果存在则将返回一个布尔值 True，不存在将返回 False。实现代码如下。

例程 1 代码位置：光盘\mr\12\getPassword\Register.aspx.cs

```
protected bool isName(string name)
{
    SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["con"]);
    con.Open();
    string sqlSel = "select count(*) from tb_User where Name='" + name + "'";
    SqlCommand com=new SqlCommand(sqlSel,con);
    if (Convert.ToInt32(com.ExecuteScalar()) > 0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

12.2.2 Panel 控件分步显示内容

Panel 控件也可以叫面板控件，该控件中可以包含一组控件或信息。在开发项目中常用该控件来显示或隐藏一组控件或信息。在密码找回过程中，使用该控件是为了将密码找回分为两个

部分进行操作。第 1 个部分就是输入会员名，运行效果如图 12.3 所示。

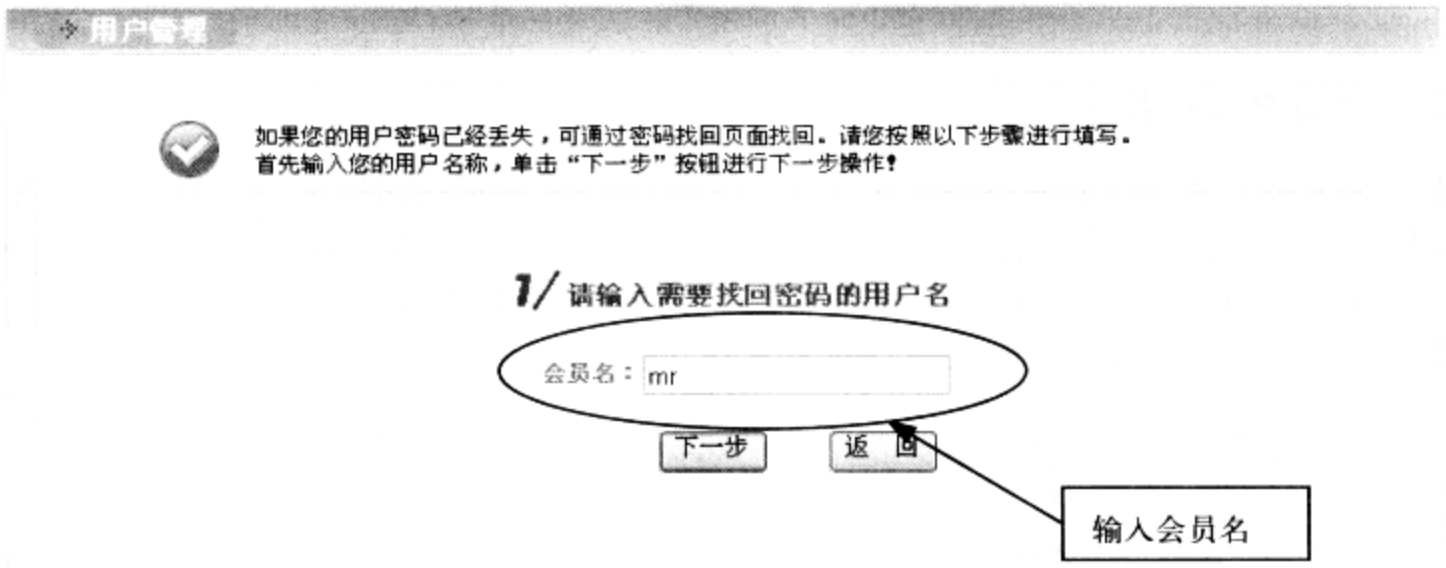


图 12.3　输入会员名

第 2 个部分是回答密码提示问题，运行效果如图 12.4 所示。

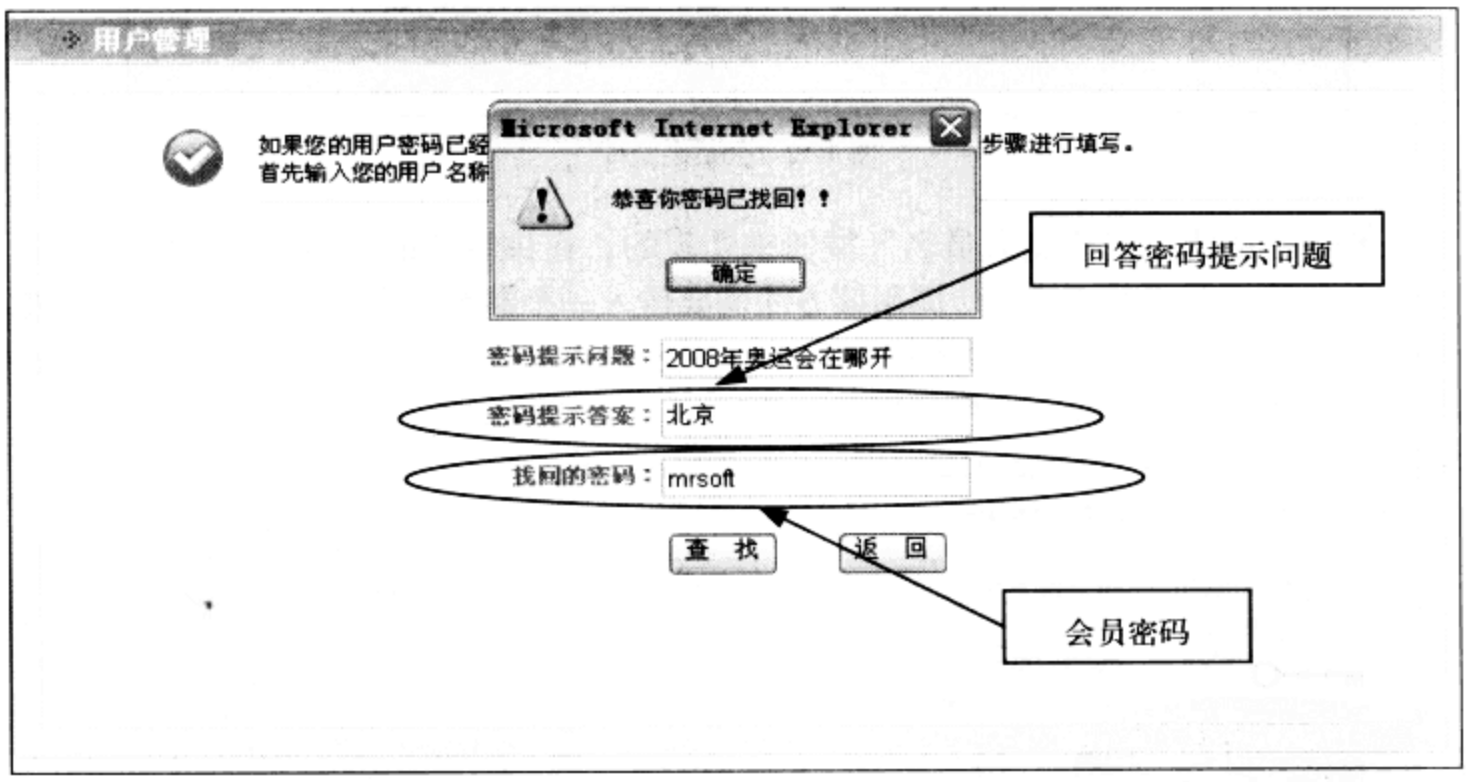


图 12.4　回答密码提示问题

使用 Panel 控件将密码找回分为两个部分操作的优点是，美化页面和判断会员名。美化页面可以给用户页面简洁，步骤清晰的感觉。而判断会员名是先判断用户输入的会员名是否存在，如果存在才可以让用户进行回答密码提示问题的操作。如果会员名不存在就不让用户进行回答密码提示问题的操作。

在本程序中主要使用到了 Panel 控件的 Visible 属性。

Visible 属性用来设置或获取 Panel 控件是否显示或隐藏在页面上。该属性有 True 和 False 两个参数。默认值为 True。

12.2.3 发送邮件技术

在用户填写完注册信息后，单击“注册”按钮时将用户填写的会员名和密码信息发送到用户填写的邮箱中。当用户忘记密码或会员名时可以在自己的邮箱中找回。

本程序中的邮件发送是利用 SMTP 进行发送的。SMTP 是 Windows 系统自带的邮件服务器，用于发送邮件。在使用 SMTP 发送邮件前需要先安装配置 SMTP 服务器。SMTP 的安装和配置将会在后面的章节中介绍。配置完成 SMTP 后，在 ASP.NET 程序中导入 System.Web.Mail 命名空间，通过调用该命名空间中的 MailMessage 对象来创建邮件，并设置邮件信息。发送邮件信息是通过 SmtpMail 类来完成的。

MailMessage 类用于构造电子邮件，该类的常用属性及说明如表 12.2 所示。

表 12.2　MailMessage 类常用属性及说明

属　性　值	说　　明
From	发送邮件人的电子邮件地址
To	接收邮件人的邮件地址列表，多个地址以分号分隔
Subject	电子邮件的主题
Body	电子邮件的正文
BodyFormat	电子邮件正文的内容类型，由 MailFormat 枚举值指定，可以是 Html 或 Text
Priority	电子邮件的优先级，由 MailPriority 枚举值指定，可以是 Low、Normal 或 Hight

当单击“注册”按钮时，在“注册”按钮的单击事件中将会调用自定义 sendMail 方法。SendMail 方法将会完成邮件发送的操作。调用该方法需要传入 3 个参数，这 3 个参数都是字符串类型，分别表示会员注册名、登录密码和用户的邮件地址。实现代码如下。

例程 2　代码位置：光盘\mr\12\getPassword\Register.aspx.cs

```
protected void sendMail(string name, string pass,string email)
{
    try {
        string Email = email;    //电子邮件地址
        //电子邮件的内容
        string body = "您好！您在某某网站的注册已经成功，您注册名为：'"+name+"'密码为：'" + pass+"'";
        string subject = "某某网站提示！";  //邮件的主题
        //创建MailMessage对象
        MailMessage myEmail = new MailMessage();
        //设置发件人地址
        myEmail.From = "xiaoyu@mr.com";
        //设置收件人地址
        myEmail.To = email;
        //设置邮件主题
        myEmail.Subject = subject;
        //设置邮件内容
        myEmail.Body = body;
        //设置邮件正文类型
        myEmail.BodyFormat = MailFormat.Text;
        //设置服务器名
        SmtpMail.SmtpServer = "MRSY";
        //发送电子邮件
        SmtpMail.Send(myEmail);
        RegisterStartupScript("true", "<script>alert('发送成功！')</script>");
    }
    catch (Exception ex)
    {
        Response.Write("邮箱不存在");
    }
}
```

12.2.4 3 次找回密码机会

3 次找回密码的机会是为了防止非法用户多次尝试回答密码问题而获得密码的目的。本程序中的该功能只给用户 3 次回答密码提示问题的机会，当用户操作 3 次以后，该功能将会冻结 24 小时，用户要在 24 小时后才可以使用该功能，效果如图 12.5 所示。

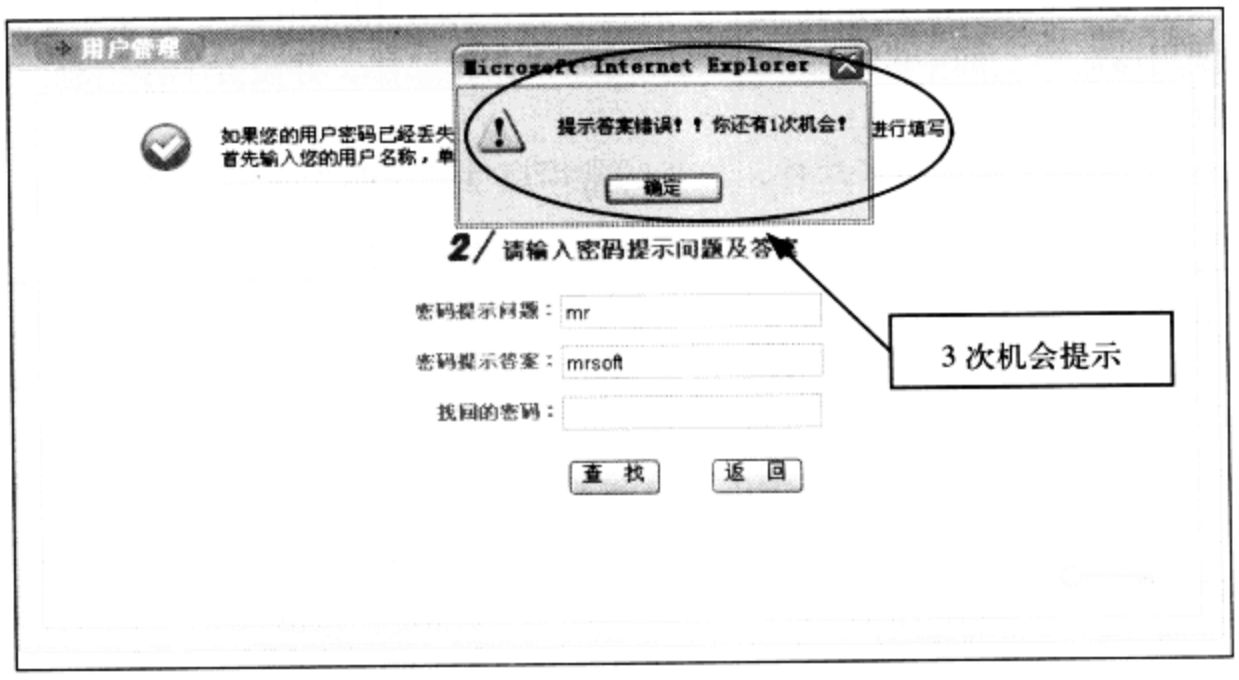


图 12.5 密码找回 3 次机会

实现此功能需要先创建一个静态的整型变量，并给该变量赋值为 0。在“查找”按钮的单击事件中先将整型变量累加。如果用户回答密码提示问题错误，将判断整型变量是否小于 3，如果小于 3 将给出还有几次机会的提示，如果大于 3 将给出用户冻结 24 小时的信息提示。

12.2.5 SMTP 服务的安装与配置

SMTP 是 Windows 系统自带的一种简单的邮件服务器，用于发送邮件，在 ASP.NET 中可以通过 System.Web.Mail 类创建邮件。在网站设计过程中，往往需要用到邮件发送的功能，如果发送量不是很大，SMTP 完全能够胜任。需要注意的是在使用 SMTP 邮件服务器之前，首先需要安装并配置好 SMTP 服务器，然后在 ASP.NET 程序中通过调用 MailMessage 对象来创建邮件，设定好发件人信息、收件人信息就可以发送邮件了。ASP.NET 发送的邮件会被 SMTP 所接受，正常情况下，SMTP 会将接受的邮件加入发送队列中，进行相关的操作。

使用 System.Web.Mail 命名空间中 SmtpMail 类发送电子邮件时要求保存在可用的 SMTP 服务器，最方便的方法就是使用 Windows 2003 内置的 SMTP 服务组件。下面分别来介绍 SMTP 服务组件的安装与配置。

- 安装 SMTP 服务

在默认情况下，IIS 中并没有安装 SMTP 服务，可通过“控制面板”中“添加或删除程序”来安装。安装 SMTP 服务时将创建一个默认的 SMTP 配置，用户随后可以使用 IIS 管理器自定义该配置。SMTP 服务的安装步骤如下。

（1）选择“开始”→“设置”→“控制面板”→“添加或删除程序”命令，打开“添加或删除程序”对话框，如图 12.6 所示。

（2）单击对话框左侧的“添加/删除 Windows 组件”，打开“Windows 组件向导”对话框，如图 12.7 所示。

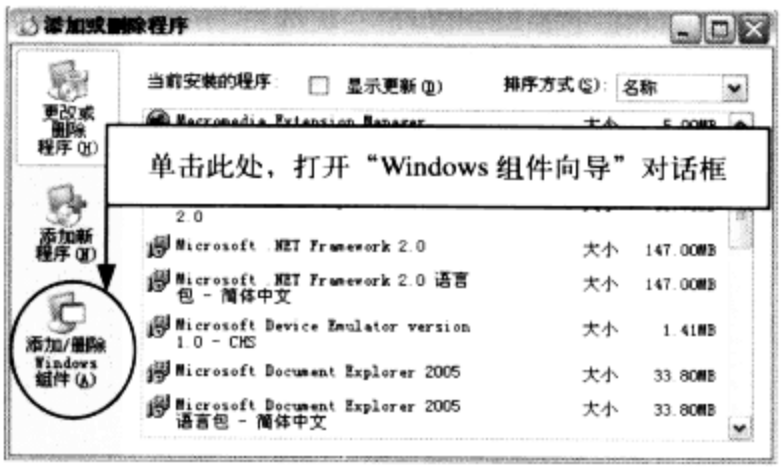


图 12.6　添加或删除程序

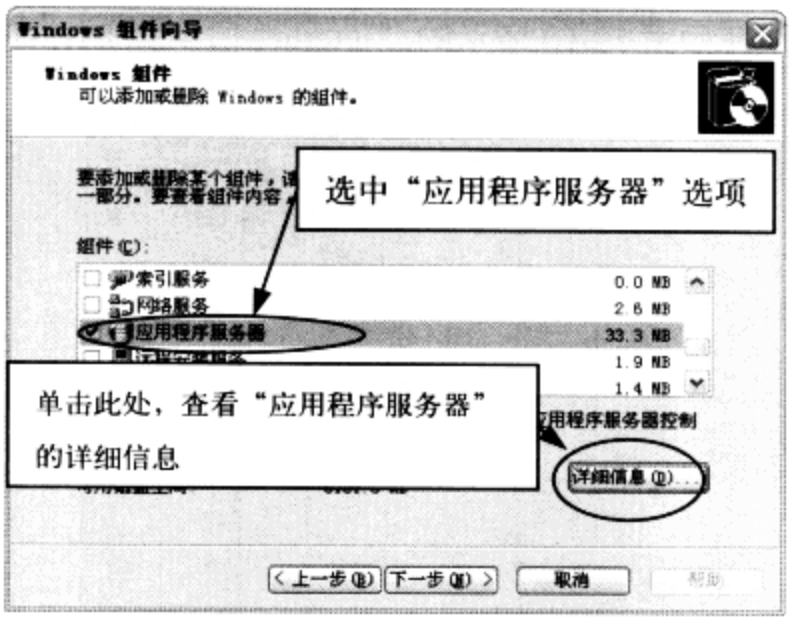


图 12.7　Windows 组件向导

(3) 选中“应用程序服务器”复选框，单击“详细信息”按钮，打开“应用程序服务器”对话框，如图 12.8 所示。

(4) 选中“Internet 信息服务(IIS)”复选框，单击“详细信息”按钮，打开“Internet 信息服务（IIS)”对话框，选中“SMTP Services”复选框，准备安装 SMTP 服务，如图 12.9 所示。

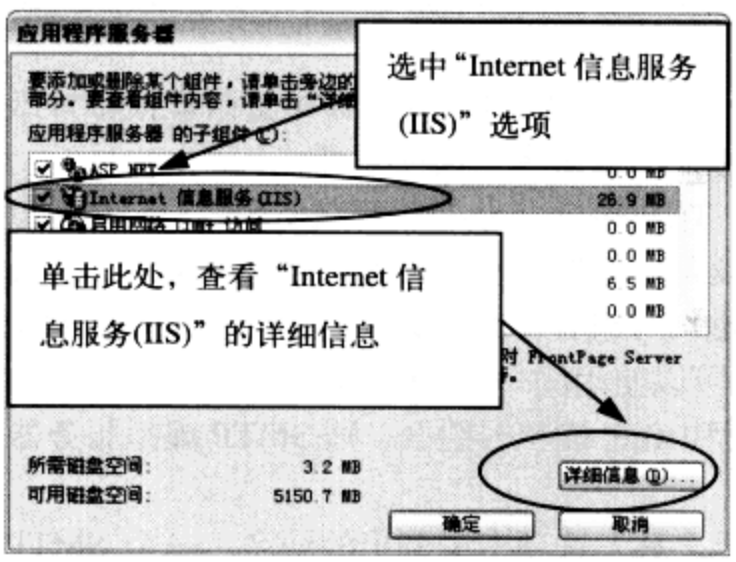


图 12.8　应用程序服务器

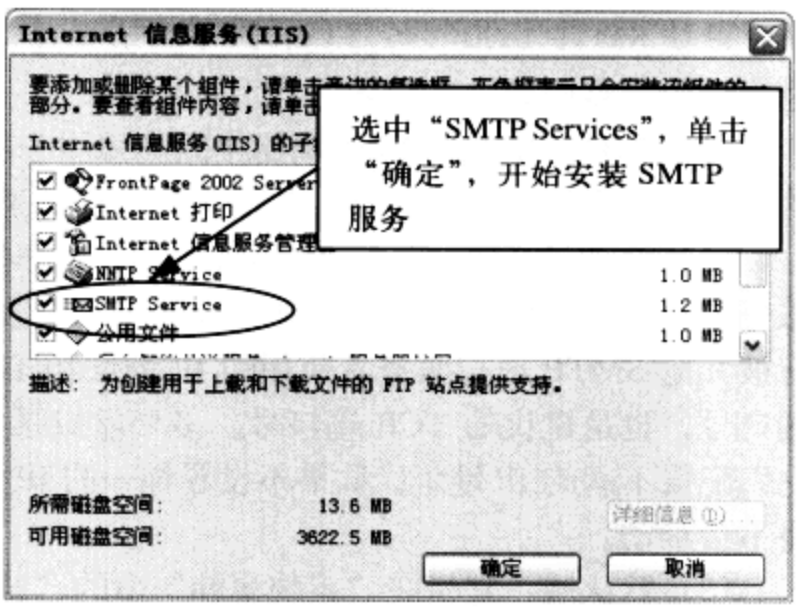


图 12.9　Internet 信息服务

(5) 单击“确定”按钮，然后单击“下一步”按钮，安装完成后，单击“完成”按钮，完成邮件传输协议（SMTP）服务的安装。

注　意

由于安装 SMTP 服务时，将在 C:\Inetpub\Mailroot 中创建一个具有消息存储区的默认 SMTP 服务器配置。设置 SMTP 服务时，可以为 SMTP 服务配置全局设置，还可以为虚拟服务器的单个组件配置设置。IIS SMTP 服务只是一个中继代理，电子邮件将转发到 SMTP 服务器进行传递。

配置 SMTP 虚拟计算机

安装 SMTP 服务后，系统将会在 IIS 管理器中创建一个新节点。若要配置 SMTP 虚拟服务器，必须启动 IIS 管理器。

1．依次单击“开始”→“设置”→“控制面板”→“Internet 信息服务(IIS)管理器”选项，打开如图 12.10 所示的“Internet 信息服务(IIS)管理器”对话框。

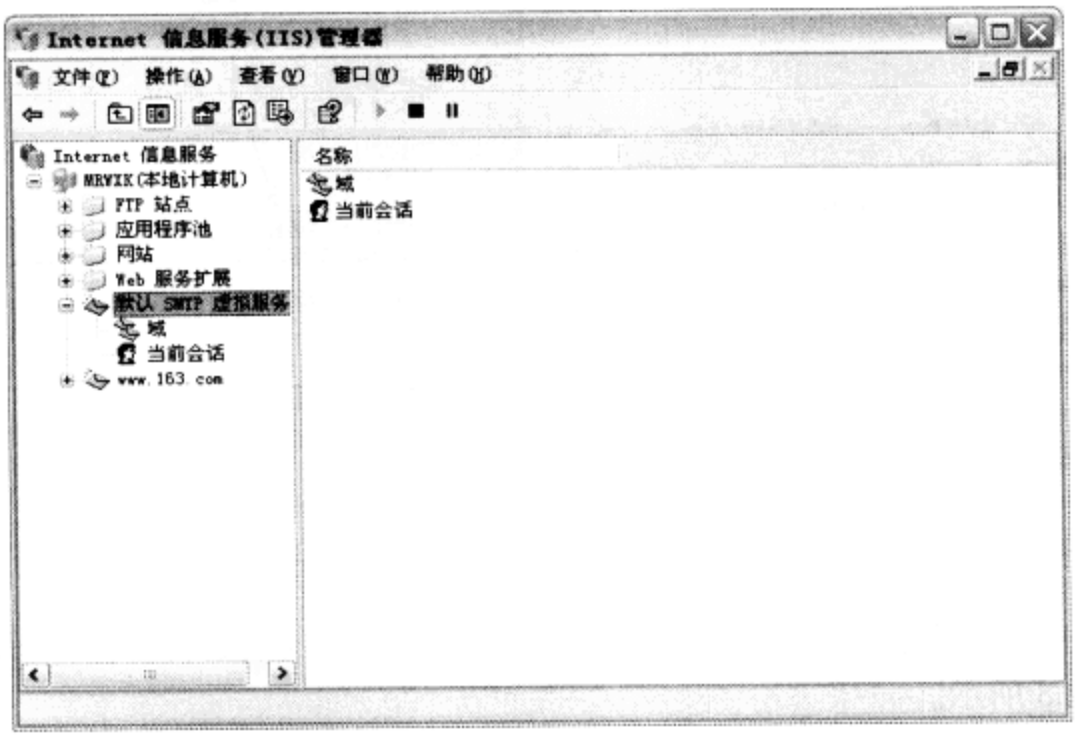


图 12.10　Internet 信息服务管理器

2. 默认设置

默认 SMTP 虚拟服务器具有以下默认设置。如果想创建一个新的虚拟服务器，可以使用“新建虚拟服务器向导”来配置。

(1) 名称：出现在 IIS 管理器中的虚拟服务器的名称。可以在 IIS 管理器中更改虚拟服务器的名称。只需该虚拟服务器上单击鼠标右键，然后选择“重命名”选项即可。

(2) IP 地址/TCP 端口：IP 地址均为“未分配”，TCP 端口号为 25。可以使用“SMTP 虚拟服务器属性”对话框中的“常规”选项卡来更改该设置。如果更改该设置，则必须指定一个没有被其他 SMTP 虚拟服务器使用的 IP 地址和 TCP 端口号组合。TCP 端口号 25 是默认的 TCP 端口号，也是建议的 TCP 端口号。多个虚拟服务器可以使用同一个 TCP 端口号，但是必须为它们配置不同的 IP 地址。如果不设置惟一的 IP 地址和 TCP 端口号组合，则 SMTP 虚拟服务器将不会启动。

(3) 默认域：是指在“系统属性”中的“计算机名称”选项卡中列出的域名。一个 SMTP 虚拟服务器只能有一个默认域，且不能删除该域。若要在 IIS 管理器中更改默认域的名称，请双击“虚拟服务器”选项，然后双击“域”选项，在右侧列表中选择“本地（或默认）域”选项，然后右键单击本地（或默认）域，在弹出的快捷菜单中选择“重命名”选项。

(4) 主目录：C:\Inetpub\Mailroot。主目录是 SMTP 的根目录，必须是运行 SMTP 服务计算机的本地目录。

12.3　会员密码找回实现过程

12.3.1　用户登录设计

用户登录页面用来实现用户的登录，在该页面中还可以跳转到会员注册页面和密码找回页面。用户登录页面如图 12.11 所示。

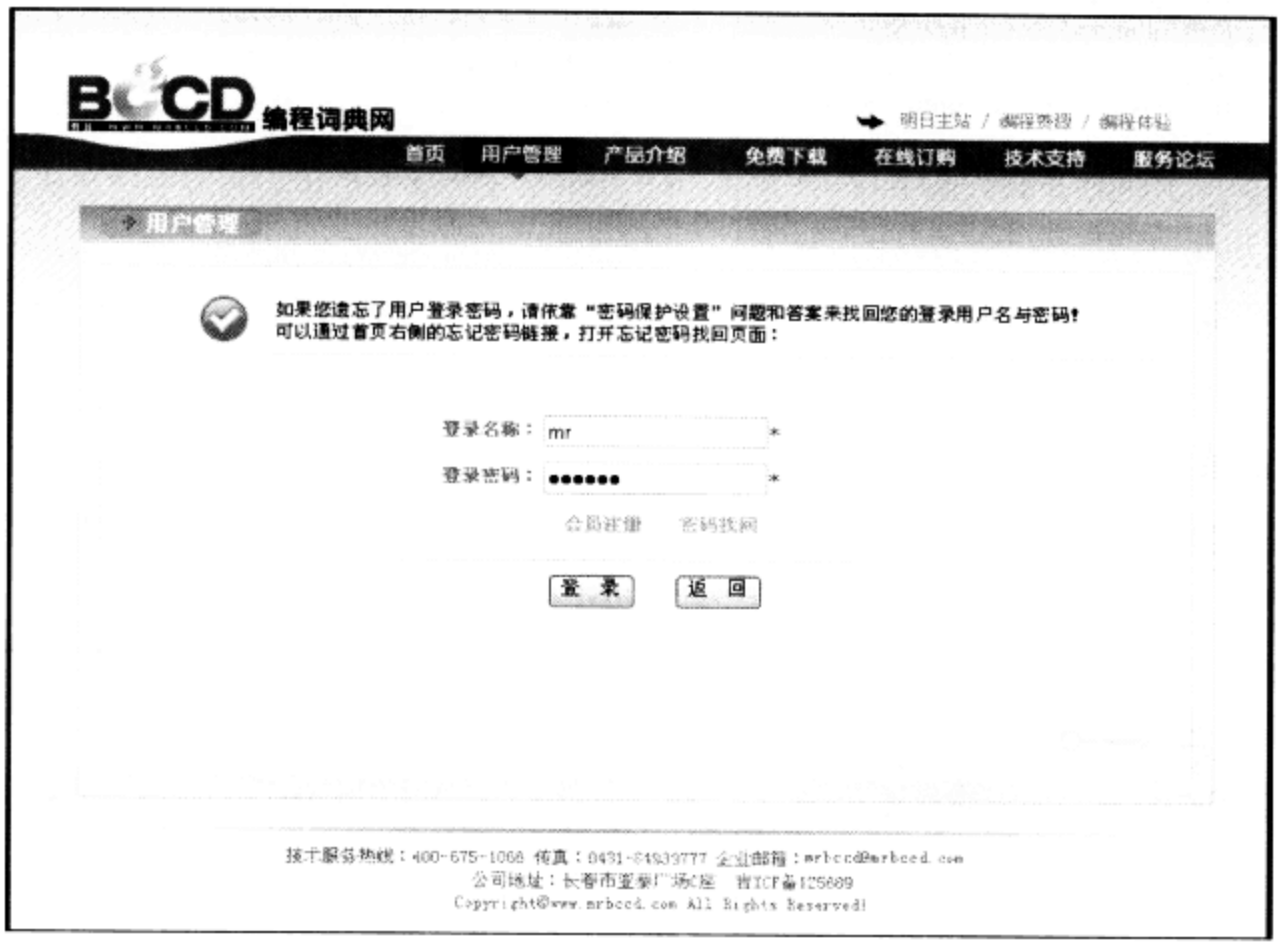


图 12.11　用户登录页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 Default.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 12.3 所示。

表 12.3　　控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说　明
标准/TextBox 控件	txtName	均为默认值	输入会员登录名
	txtPass	设置 TextMode 属性为 Password	输入会员登录密码
标准/Button 控件	btnEnter	均为默认值	实现登录操作
标准/LinkButton 控件	linkBtnRegister	设置 PostBackUrl 属性为～/Register.aspx	跳转会员注册页面
	linkBtnGetPass	设置 PostBackUrl 属性为～/GetPass.aspx	跳转密码找回页面

2. 后台代码编写

在“登录”按钮的单击事件中，将查询用户输入的会员名和密码是否正确。如果正确将给出相应的提示。实现代码如下。

例程 3　代码位置：光盘\mr\12\getPassword\Default.aspx.cs

```
protected void btnEnter_Click(object sender, EventArgs e)
    {
        //获取会员登录名
        string name = txtName.Text;
        //获取登录密码
        string pass = txtPass.Text;
        //编写SQL语句，查询表中是否拥有满足指定条件的记录
        string sqlSel = "select count(*) from tb_User where Name='" + name + "'and Pass='" + pass + "' ";
        //创建数据库连接
        SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["con"]);
        //打开连接
```

```
            con.Open();
            SqlCommand com = new SqlCommand(sqlSel, con);
            //判断返回的参数是否大于0，如果大于0则说明登录成功
            if (Convert.ToInt32(com.ExecuteScalar()) > 0)
            {
                RegisterStartupScript("", "<script>alert('登录成功！')</script>");
            }
            else
            {
                RegisterStartupScript("", "<script>alert('登录失败！')</script>");
            }
        }
```

12.3.2 会员注册设计

会员注册页面用来实现会员的注册功能，在该页面中用户需要填写会员的基本信息，如会员名、密码、密码提示问题等。会员注册页面如图 12.12 所示。

图 12.12 会员注册页面

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 Register.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 12.4 所示。

表 12.4 控件类型、控件名称及说明

控件类型	控件名称	主要属性	说明
标准/TextBox 控件	txtName	均为默认值	填写会员登录名
	txtPass	设置 TextMode 属性为 Password	填写会员登录密码
	txtValidatePass	设置 TextMode 属性为 Password	填写确认密码
	txtPassQuestion	均为默认值	填写密码提示问题
	txtPassAnswer	均为默认值	填写密码提示答案
	txtEmail	均为默认值	填写电子邮箱地址

续表

控件类型	控件名称	主要属性	说明
标准/Button 控件	btnIsName	设置 CausesValidation 属性为 False	检查会员名是否存在
	btnRegister	均为默认值	实现注册操作
	btnReturn	设置 PostBackUrl 属性为～/Default.aspx	跳转到首页面

2. 后台代码编写

在“检测会员名”按钮的单击事件中，先判断用户输入的会员名是否为空，如果不为空将调用自定义 isName 方法来查询用户名是否存在，并给出相应的提示。实现代码如下。

例程 4 代码位置：光盘\mr\12\getPassword\Register.aspx.cs

```
protected void btnIsName_Click(object sender, EventArgs e)
    {
        //判读用户输入的会员名是否为空，为空则给出提示
        if (txtName.Text != "")
        {
            //调用自定义方法判断用户输入的会员名是否存在
            if (isName(txtName.Text))
            {
                RegisterStartupScript("", "<script>alert('用户名以存在')</script>");
            }
            RegisterStartupScript("", "<script>alert('可以注册')</script>");
        }
        else {
            RegisterStartupScript("", "<script>alert('用户名不能为空！')</script>");
        }
    }
```

自定义 isName 方法用来查询用户输入的会员名是否存在。调用该方法需要传入一个参数，该参数为字符串类型，表示用户输入的会员名。该方法将会返回一个布尔值，当该值为 True 时表示用户查询的会员名已经存在，如果为 False 表示会员名不存在。实现代码如下。

例程 5 代码位置：光盘\mr\12\getPassword\Register.aspx.cs

```
protected bool isName(string name)
    {
        //创建数据库连接
        SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["con"]);
        //打开数据库
        con.Open();
        //创建SQL语句，查询指定的条件是否存在
        string sqlSel = "select count(*) from tb_User where Name='" + name + "'";
        //创建sqlCommand对象
        SqlCommand com=new SqlCommand(sqlSel,con);
        //判断sqlCommand对象的ExecuteScalar方法返回的参数是否大于0，大于0说明用户已存在
        if (Convert.ToInt32(com.ExecuteScalar()) > 0)
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

在“注册”按钮的单击事件中，也需要判断一下用户输入的会员名是否存在，这样做的目的是为了防止用户没有单击“检测会员名”按钮。如果会员名不存在，将获取到的相关信息通过 SQL 语句添加到数据中，如果添加成功，可调用自定义 sendMail 方法将用户的会员名及密码发送到用户填写的邮箱中。实现代码如下。

例程 6 代码位置：光盘\mr\12\getPassword\Register.aspx.cs

```
protected void btnRegister_Click(object sender, EventArgs e)
    {
        //获取用户名
        string Name = txtName.Text;
        //获取密码
        string Pass = txtPass.Text;
        //获取密码提示问题
        string PassQuestion = txtPassQuestion.Text;
        //获取密码提示答案
        string PassAnswer = txtPassAnswer.Text;
        //获取E-mail
        string Email = txtEmail.Text;
        if (!isName(txtName.Text))
        {
            //创建数据库连接
            SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["con"]);
            //打开数据库连接
            con.Open();
            //编写SQL语句，插入用户注册信息
            string Sql = "insert into tb_User values('" + Name + "','" + Pass + "','" + PassQuestion + "','" + PassAnswer + "','" +
Email + "','"+DateTime.Now.ToString()+"')";
            SqlCommand com = new SqlCommand(Sql, con);
            //判断SQL语句是否执行成功
            if (Convert.ToInt32(com.ExecuteNonQuery()) > 0)
            {
                RegisterStartupScript("true", "<script>alert('注册成功！')</script>");
                //调用自定义方法，并传入2个参数，将密码保存到邮件中
                sendMail(Name,Pass, Email);
                txtName.Text = txtPass.Text = txtPassQuestion.Text = txtPassQuestion.Text = txtEmail.Text
= txtPassAnswer.Text="";
            }
            else
            {
                RegisterStartupScript("false", "<script>alert('注册失败！')</script>");
            }
            con.Close();
        }
        else
        {
            RegisterStartupScript("", "<script>alert('用户名以存在')</script>");
        }
    }
```


12.3.3 会员密码找回设计

会员密码找回页面用来实现密码的找回功能，在该页面中会员首先输入自己的会员名，如图 12.13 所示。通过回答密码提示问题，即可找回自己的密码，如图 12.14 所示。

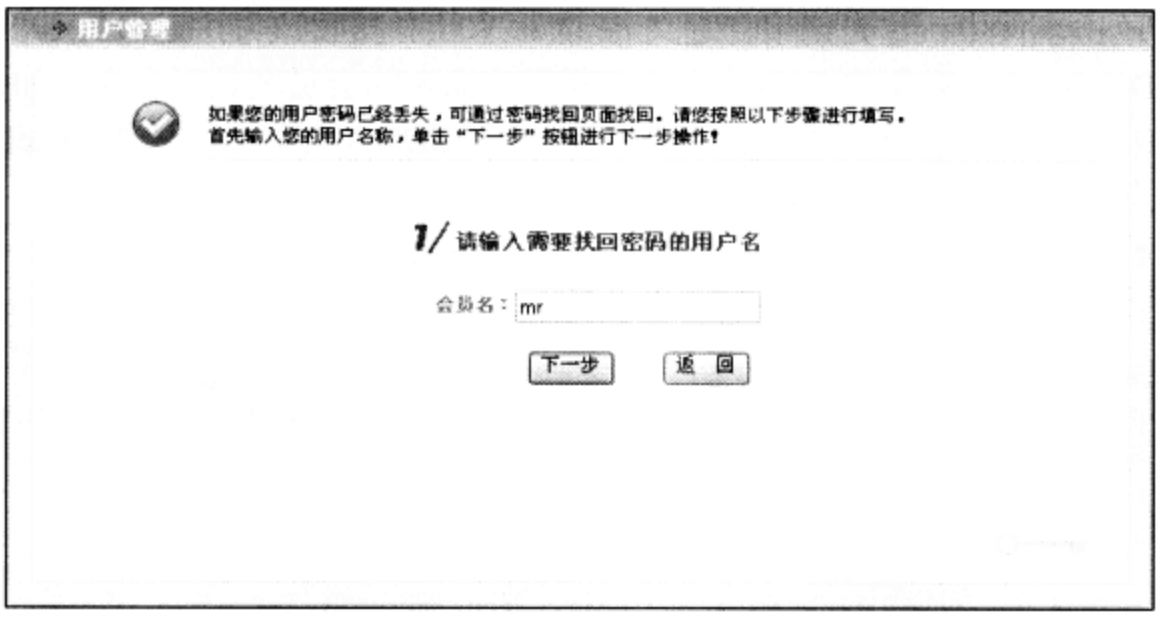


图 12.13　密码找回第 1 步

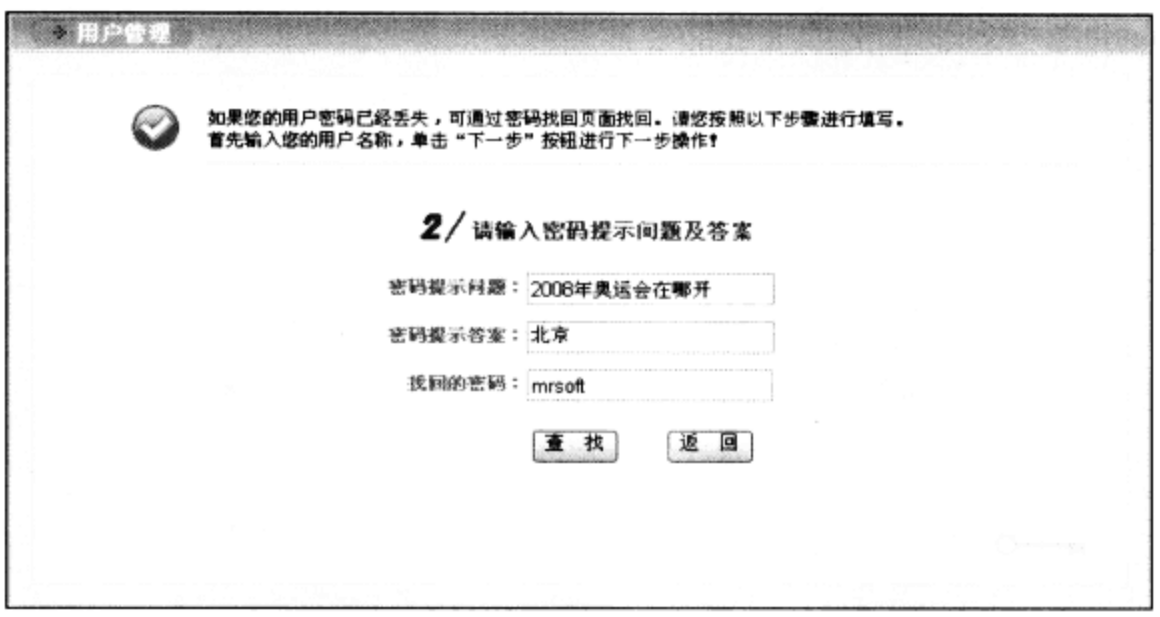


图 12.14　密码找回第 2 步

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 GetPass.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 12.5 所示。

表 12.5　　控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说　明
标准/TextBox 控件	txtName	均为默认值	填写会员登录名
	txtQuestion	均为默认值	填写密码提示问题
	txtAnswer	均为默认值	填写密码提示答案
	txtPass	设置 ReadOnly 属性为 True	显示会员的登录密码
标准/Button 控件	btnNext	设置 CausesValidation 属性为 False	进入回答密码提示问题操作
	btnNreturn	设置 PostBackUrl 属性为～/Default.aspx	跳转到首页面
	btnGet	均为默认值	实现找回密码操作
	btnReturn	设置 PostBackUrl 属性为～/Default.aspx	跳转到首页面
标准/Panel 控件	PanelInputName	均为默认值	显示输入登录用户名区域
	PanelGetPass	设置 Visible 属性为 False	显示回答密码提示问题区域

2. 后台代码编写

密码找回操作是分为两步进行操作的，当用户输入完会员名后单击“下一步”按钮，在“下

一步”按钮的单击事件中，首先调用自定义 getData 方法并接收该方法返回的 SqlDataReader 对象，根据 SqlDataReader 对象的 Reade 方法判断用户输入的会员名是否存在。如果存在，再判断用户是否被冻结了密码找回功能。如果未冻结，将隐藏输入会员名区域的 Panel，显示回答密码提示问题区域的 Panel。实现代码如下。

例程 7 代码位置：光盘\mr\12\getPassword\GetPass.aspx.cs

```
protected void btnNext_Click(object sender, EventArgs e)
{
    //调用自定义方法，并传入2个参数，用来查询用户名是否存在
    SqlDataReader sdr = getData("select * from tb_User where Name=@value", txtName.Text);
    //判断用户名是否存在
    if (sdr.Read())
    {
        //获取冻结密码找回的日期
        DateTime congeal = Convert.ToDateTime(sdr["congealDate"]);
        //创建TimeSpan对象，该对象表示两个时间的间隔
        TimeSpan ts = DateTime.Now - congeal;
        //获取两个时间的间隔以小时表示
        int hours = Convert.ToInt32(ts.TotalHours);
        //判断时间是否大于个24，如果大于将显示回答密码提示问题区域
        if (hours > 24)
        {
            //如果存在隐藏输入用户名区域
            PanelInputName.Visible = false;
            //显示找回密码区域
            PanelGetPass.Visible = true;
            //显示密码提示问题
            txtQuestion.Text = sdr["Question"].ToString();
        }
        else
        {
            RegisterStartupScript("", "<script>alert('还有" + (24 - hours) + "小时后可以使用该功能！！')</script>");
        }
    }
    else
    {
        RegisterStartupScript("false", "<script>alert('用户名不存在！！')</script>");
    }
}
```

自定义 getData 方法用来查询用户输入的会员名是否存在。调用该方法需要传入 2 个参数，这 2 个参数是字符串类型，分别表示 SQL 语句和会员名。该方法将返回 1 个 SqlDataReader 对象。实现代码如下。

例程 8 代码位置：光盘\mr\12\getPassword\GetPass.aspx.cs

```
protected SqlDataReader getData(string sql, string value)
{
    //创建数据库连接
    SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["con"]);
    //打开数据库连接
    con.Open();
    //创建sqlCommand对象
    SqlCommand com = new SqlCommand(sql, con);
    //添加1个参数，并指定参数的类型及大小
    com.Parameters.Add(new SqlParameter("@value", SqlDbType.VarChar, 50));
    //设置参数的值
    com.Parameters["@value"].Value = value;
    //返回DataReader对象
    return com.ExecuteReader();
}
```

在“查找”按钮的单击事件中，首先将递增公共的静态变量 i。接着判断用户输入的密码提示问题是否正确，如果正确将显示会员密码。如果不正确将通过变量 i 判断用户是第几次回答密码提示问题，如果回答的次数超过 3 次，将通过 SQL 语句更新回答密码提示问题的冻结时间并返回首页。实现代码如下。

例程 9 代码位置：光盘\mr\12\getPassword\GetPass.aspx.cs

```
    protected void btnGet_Click(object sender, EventArgs e)
    {
        ++i;  //该变量是公共的静态变量用来保存回答密码提示问题的次数调用自定义方法，并传入2个参数，用来查询密码提示答案是否正确
        SqlDataReader sdr = getData("select * from tb_User where Name='" + txtName.Text + "' and Answer=@value", txtAnswer.Text);
        //判读密码提示答案是否存在
        if (sdr.Read())
        {
            //判断用户回答的密码提示问题是否正确
            if (txtAnswer.Text == sdr["Answer"].ToString())
            {
                //显示会员的密码
                txtPass.Text = sdr["Pass"].ToString();
                RegisterStartupScript("true", "<script>alert('恭喜你密码已找回！！')</script>");
            }
        }
        else
        {
            //判断用户是否已回答了3次问题
            if (i < 3)
            {
                RegisterStartupScript("false", "<script>alert('提示答案错误！！ 你还有" + (3 - i) + "次机会！')</script>");
            }
            else
            {
                i = 0;
                //创建SQL语句，更新会员回答密码提示问题的冻结时间
                string sql = " update tb_User set congealDate='" + DateTime.Now.ToString() + "' where Name='" + txtName.Text + "' ";
                //创建数据库连接
                SqlConnection con = new SqlConnection(ConfigurationManager.AppSettings["con"]);
                //打开数据连接
                con.Open();
                //创建SqlCommand对象
                SqlCommand com = new SqlCommand(sql, con);
                //执行Sql语句
                com.ExecuteNonQuery();
                RegisterStartupScript("false", "<script>alert('您3次回答问题的机会以用完！在24小时后才可以使用此功能！');location='Default.aspx'</script>");
            }
        }
    }
```

12.4 程序调试与错误处理

在编写完程序后需要对程序进行调试，而断点是程序调试中的核心，它是 ASP.NET 的一个指令，能够使代码运行到指定的行，然后停下来等待用户检查应用程序当前的状态。断点模式可以看作为一种超时程序中所有的元素（例如，函数、变量和对象）都保留在内存中。在中断

模式下，您可以检查它们的位置和状态，以查看是否存在冲突或 bug。您可以在中断模式下对程序进行调整。没有这个功能，调试大的程序几乎是不可能的。

12.4.1 断点

断点通知调试器，是应用程序在某点上（暂停执行）或某情况发生时中断。发生中断时，程序和调试器处于中断模式。进入中断模式并不会终止或结束程序的执行，所有元素（如函数、变量和对象）都保留在内存中。可以在任何时候继续执行。

插入断点有 3 种方式，分别如下。

（1）在要设置断点代码行的左边的灰色空白处单击鼠标左键，会出现一个红色圆点并且代码会高亮显示。如图 12.15 所示。

```
protected bool isName()
{
    //创建一个布尔型变量并初始化为false;
    bool blIsName = false;
    //创建SQL语句，该语句用来判断用户名是否存在
    string sqlSel = "select count(*) from tb_userInfo where userName='" + txtName.Text + "' ";
    //创建数据库连接
    SqlConnection con = new SqlConnection("server=.;database=db_Register;uid=sa;pwd=;");
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sqlSel, con);
    //判断ExecuteScalar方法返回的参数是否大于0,大于表示用户名已存在
    if (Convert.ToInt32(com.ExecuteScalar()) > 0)
    {
        blIsName = true;
    }
    else
    {
        blIsName = false;
    }
    //返回布尔值变量
    return blIsName;

}
```

图 12.15　灰色处设置断点

（2）用鼠标右键单击设置断点的代码行，选择“断点”→“插入断点”选项，如图 12.16 所示。

（3）单击要设置断点的代码行，选择菜单中的“调试”→“切换断点(G)”命令，如图 12.17 所示。

删除断点有 4 种方式，分别如下。

（1）单击断点行旁边的灰色空白处的红色圆点，此时红色圆点将会消失表示已删除了断点。

图 12.16　右键设置断点

图 12.17　调试中设置断点

单击断点的行旁边的灰色空白处的红色圆点，选择“删除断点”选项，如图 12.18 所示。

（2）用鼠标右键单击设置断点的代码行，选择“断点”→“删除断点”选项，如图 12.19 所示。

（3）单击要设置断点的代码行，选择菜单中的“调试”→“切换断点(G)”命令，断点将会消失。

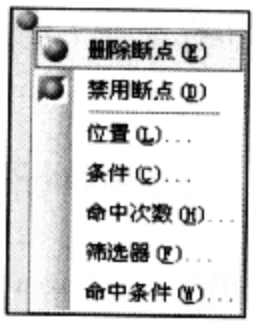


图 12.18　在灰色处删除断点

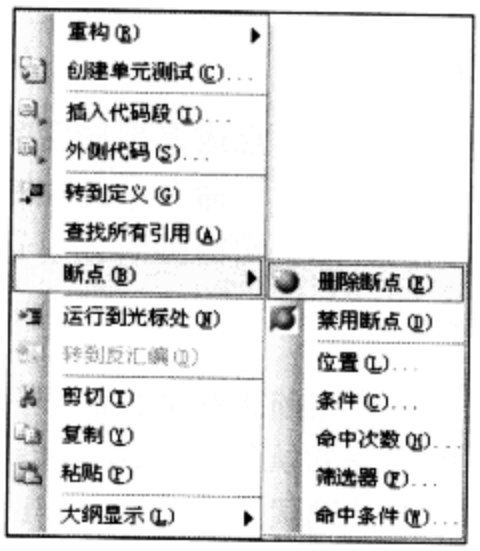


图 12.19　鼠标右键删除断点

12.4.2　开始执行

断点的执行可以通过在“调试”菜单中选择“启动调试”、“逐语句”或“逐过程”选项来执行程序并调试，也可以通过单击鼠标右键执行代码中的某行，然后从快捷菜单中选择“运行到光标处”选项。

如果选择“启动调试”、“逐语句”或“逐过程”选项执行程序，则应用程序启动并一直运行到断点，如图 12.20 所示。此时可以检查变量值，或检查程序状态。

```
92    protected bool isName()
93    {
94        //创建一个布尔型变量并初始化为false.
95        bool blIsName = false;
96        //创建SQL语句，该语句用来判断用户名是否存在
97        string sqlSel = "select count(*) from tb_userInfo where userName='" + txtName.Text + "' ";
98        //创建数据库连接
99        SqlConnection con = new SqlConnection("server=.;database=db_Register;uid=sa;pwd=;");
100       //打开数据库连接
101       con.Open();
102       //创建SqlCommand对象
103       SqlCommand com = new SqlCommand(sqlSel, con);
104       //判断ExecuteScalar方法返回的参数是否大于0,大于表示用户名已存在
105       if (Convert.ToInt32(com.ExecuteScalar()) > 0)
106       {
107           blIsName = true;
108       }
109       else
110       {
111           blIsName = false;
112       }
113       //返回布尔值变量
114       return blIsName;
115
116   }
```

图 12.20　“启动调试”、“逐语句”或“逐过程”运行结果图

程序运行到断点还是光标位置，是由断点和光标的先后位置决定的。如果光标在断点前，程序运行到光标处，如图 12.21 所示。如果光标在断点后则运行到断点处。某些情况下，不出现中断。这意味着执行始终未到达设置光标处的代码。

```
24    protected void btnRegister_Click(object sender, EventArgs e)
25    {
26        //调用isNameFormar自定义方法判断用户名输入的是否满足要求
27        if (isNameFormar())
28        {
29            //调用自定义isName方法判断用户名是否已存在
30            if (isName())
31            {
32                //使用Label控件显示提示信息
33                labIsName.Text = "用户名已存在！";
34                //设置Label控件的颜色
35                labIsName.ForeColor = System.Drawing.Color.Red;
36                RegisterStartupScript("", "<script>alert('请正确填写信息！')</script>");
37            }
38            else
39            {
40                //获取用户填写的会员名
41                string userName = txtName.Text;
```

图 12.21　光标在断点前运行

12.4.3 中断执行

当应用程序执行到一个断点或发生异常时，调试器就会中断程序的执行。也可以通过在“调试”菜单上，单击“全部中断”命令手动中断执行。这时调试器将停止所有在调试器下运行的程序。但程序并不退出，而且可以随时恢复运行。调试器和应用程序处于中断模式。

12.4.4 停止执行

停止调试意味着终止当前正在调试的程序并结束调试会话。与中断执行不同，中断执行意味着暂停正在调试的进程的执行，但调试会话仍处于活动状态。

通过选择菜单中的“调试”→“停止调试”命令或单击“调试”工具栏中的按钮来结束运行和调试。也可以退出正在调试的应用程序，调试将自动停止。

12.4.5 单步执行

单步执行是最常见的调试过程之一，即每次执行一行代码。“调试”菜单中提供了 3 种模式，即逐语句、逐过程和跳出。

“逐语句”和“逐过程”的差异仅在于它们处理函数调用的方式不同。这两个命令都指示调试器执行下一行的代码。如果某一行包含函数调用，“逐语句”仅执行调用本身，然后在函数内的第 1 个代码行处停止。而“逐过程”执行整个函数，然后在函数外的第 1 行处停止。如果要查看函数调用的内容，则使用“逐语句”。若要避免单步执行函数，那么最好使用“逐过程”。

位于函数调用的内部并想返回调用函数时，可以使用“跳出”。“跳出”将一直执行代码，直到函数返回，然后在调用函数中的返回点处中断。

基于 XML 技术的留言本

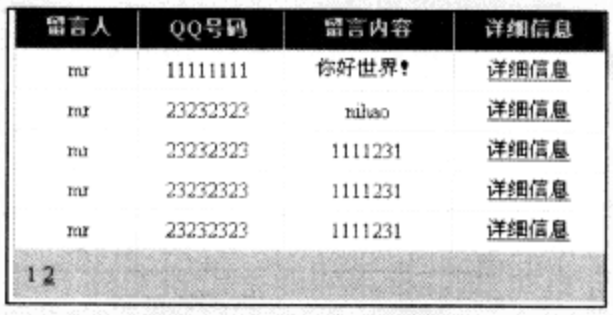
第 13 章

实例位置：光盘\mr\13\

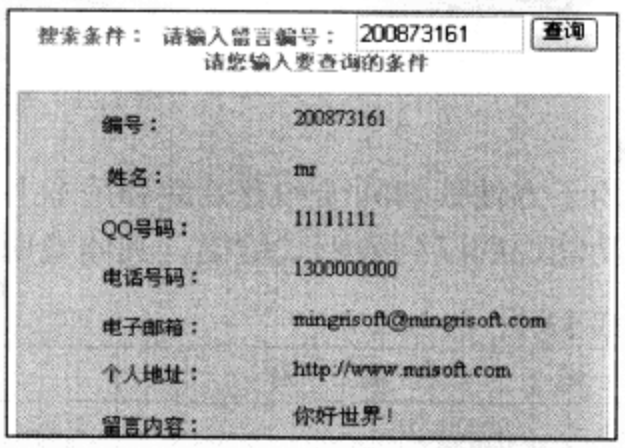
目前许多网站都设置了留言本，它提供了一个与用户交流、沟通的平台。留言本在开发网站中的应用很多，但是基于 XML 和 Ajax 技术的留言本为数不多，应用 XML 技术可有效地提高网站的访问速度，而应用 Ajax 可实现页面的无刷新效果。通过对本章的学习，读者能够学到以下内容。

使用 AJAX 技术删除留言信息时更新父窗体

数据绑定并实现分页功能

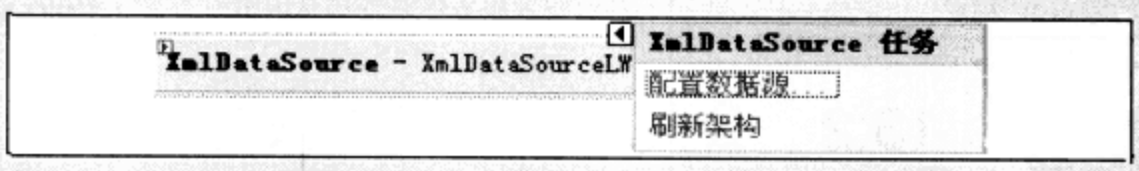


数据绑定并实现分页功能



导航功能

通过向导配置 XmlDataSource 控件



13.1 概　　述

本模块是通过 ASP.NET+XML+Ajax+SQL Server 2000 开发的一个的留言本。游客可在网站首页查看用户的留言内容，而注册的用户通过登录界面并验证成功后，可以发表留言信息、查询留言信息等。管理员登录后，可以对用户的留言信息进行删除操作等功能。在网站首页提供了具体的导航功能，分别为“查看留言”、“发表留言”、“删除留言”、“重新登录”和“用户注册”。

本章主要介绍基于 XML 技术的留言本的具体实现过程，程序运行效果如图 13.1 所示。

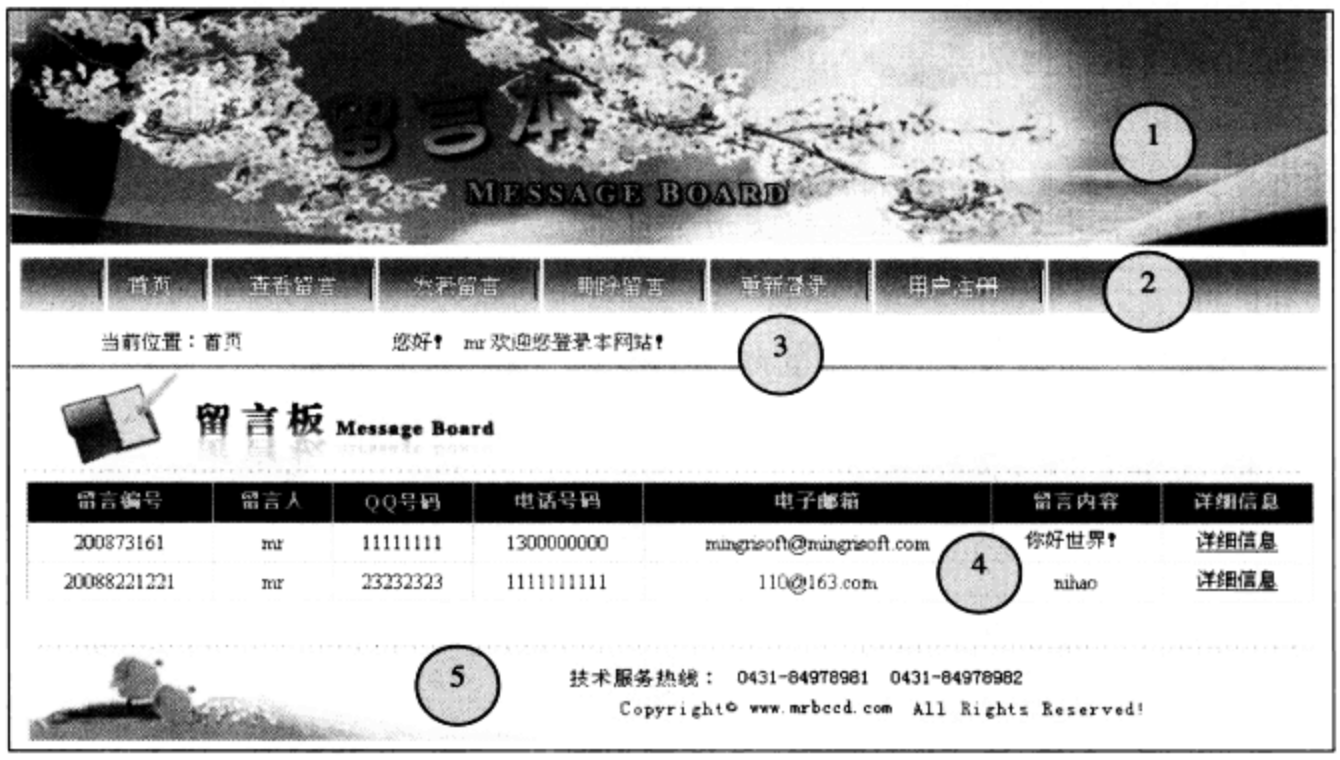


图 13.1　基于 XML 技术的留言本首页

为了方便读者阅读和有效地利用本书附赠光盘中的实例，这里将网站首页面即图 13.1 各部分的功能说明以列表形式给出，具体说明如表 13.1 所示。

表 13.1　　网站首页解析

区　域	名　称	说　明	对 应 文 件
1	网站标头	主要用于网站的旗帜广告	header.ascx
2	网站导航	主要用于网站的导航	Default.aspx
3	显示当前位置登录用户名	主要用于显示当前位置和当前登录用户的用户名称	Default.aspx
4	显示留言板信息	主要用于将 XML 中的留言信息绑定到 GridView 控件中，并且将信息内容显示出来	Default.aspx
5	网站脚标	主要用于显示技术服务热线等信息内容	footer.ascx

13.2 公共类的封装与设计

在网站开发项目中以类的形式组织、封装一些常用的方法和事件，将会在编程过程中起到事半功倍的效果。创建公共类可以减少重复代码的编写，并有利于代码维护。下面以本模块创建的公共类 LeaveWord 为例详细说明下公共类的封装与设计。

13.2.1 公共类的创建

LeaveWord 类是自定义的一个类。在此类中创建了几个常用的方法，如数据库连接类等。创建公共类，在“解决方案资源管理器”中的项目文件名称上单击鼠标右键，在弹出的右键菜单中选择“添加新项”命令，打开“添加新项”对话框。在该对话框中选择“类”图标，并在“名称”文本框中输入 LeaveWord.cs，最后单击“确定”按钮即可，如图 13.2 所示。

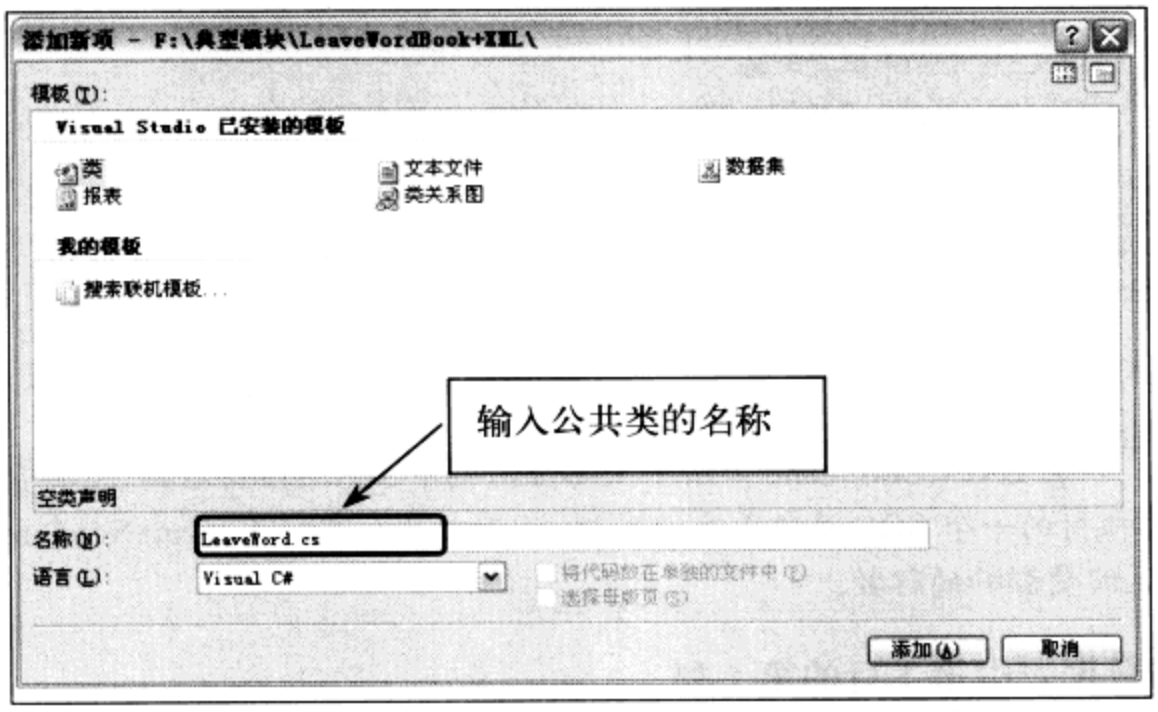


图 13.2 创建类文件

13.2.2 建立数据库连接

在公共类 LeaveWord 中创建 1 个 SqlConnection 类型的 getcon 方法用于连接数据库。此方法中，首先声明一个连接数据库的字符串，然后根据连接字符串创建 1 个 SqlConnection 实例，最后返回创建的 SqlConnection 类的对象 sqlCon 中，代码如下。

例程 1 代码位置：光盘\mr\13\LeaveWordBook+XML\App_Code\LeaveWord.cs

```
#region  连接数据库
 /// <summary>
 /// 创建时间:2008-8-18
 /// 连接数据库
 /// </summary>
 public SqlConnection getcon()
 {
//连接Sql Server 2000的连库字符串
string strCon = "Data Source=(local);DataBase=db_LeaveWordXMl;User ID=sa;PWD=";
//创建并且实例化1个SqlConnection的类对象
SqlConnection sqlCon = new SqlConnection(strCon);
return sqlCon;//返回1个SqlConnection类的类对象sqlCon
 }
 #endregion
```

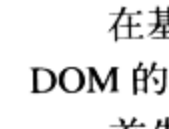
13.2.3 执行数据库的添加、删除和修改操作

在此公共类中创建了 1 个 EXECCommand 方法，主要用于执行 Insert（添加语句）、Update（修改语句）、Delete（删除语句）等 SQL 语句，并返回受影响的行数。

这里将该方法定义成一个整型并且通过返回的值来判断所执行的 SQL 语句是否成功，执行成功则返回 1，否则返回 0。具体实现的代码如下。

例程 2　代码位置：光盘\mr\13\LeaveWordBook+XML\App_Code\LeaveWord.cs

```
#region  执行SQL语句，返回受影响的行数
/// <summary>
/// 创建时间:2008-8-18
/// 执行SQL语句，返回受影响的行数
/// 返回值：返回整数类型
/// 参数：SqlStr 要执行的SQL语句
/// </summary>
public int EXECCommand(string SqlStr)
{
    int i;///定义一个整数类型的变量i
    SqlConnection con =this.getcon();///创建并且实例化1个类对象con
    con.Open();//打开数据库连接
    SqlCommand amd = new SqlCommand(SqlStr, con);
    i = amd.ExecuteNonQuery();
    con.Close();//关闭数据库连接
    return i;///返回1个整数类型的变量i
}
#endregion
```



说　明　在 EXECCommand 方法中主要应用到了 SqlCommand 类，该类表示要对 SQL Server 数据库执行的一个 SQL 语句或存储过程，同时应用该类的 ExecuteNonQuery 方法执行 SQL 语句并返回受影响的行数。

13.2.4　返回数据表中第 1 行的第 1 列

EXECuteScalar 方法用于返回数据表中记录的数目，返回值为 int 类型，执行成功则返回 1，否则返回 0。该方法的完整设计代码如下。

例程 3　代码位置：光盘\mr\13\LeaveWordBook+XML\App_Code\LeaveWord.cs

```
#region 返回数据表中记录的数目
/// <summary>
/// 创建时间：2008-8-18
/// 该方法用于实现返回数据表中记录的数目
/// 参数：SqlStr 要执行的Sql语句
/// </summary>
public int EXECuteScalar(string SqlStr)
{
    int i;///定义1个整数类型
    SqlConnection con = this.getcon();
    con.Open();//打开数据库链接
    SqlCommand com = new SqlCommand(SqlStr, con);
    i = Convert.ToInt32(com.ExecuteScalar());
    con.Close();//关闭数据库链接
    return i;
}
#endregion
```

13.3　关键技术详解

13.3.1　使用 DOM 处理 XML

在基于 XML 技术的留言本中，主要是使用 DOM 处理 XML 数据，下面详细介绍如何通过 DOM 的方式处理 XML 格式的数据。

首先了解一下 DOM（Document Object Model，文档对象模型），DOM 是操作 XML 数据的

国际通用数据模型。由于 XML 格式的数据以元素为基本单位，整个 XML 文档只有一个根元素，并且元素之间可以嵌套，所以 XML 数据非常适合用树的形式来表达。DOM 将一个 XML 文档看成一棵树，它定义了一种将 XML 文档中的元素和属性对应于特定节点的方法。因此，可以认为 DOM 定义了 XML 文档在内存中的表示形式。

一旦将一个 XML 文档转化为内存中的一棵 DOM 树，开发人员就可以通过程序对 XML 文档进行操作。在 DOM 中，文档由节点的集合组成，这些节点具有父/子关系。“节点”是主要对象，并且可以是不同类型，例如，“文档”、“元素”、“属性”、“文本”、“处理指令”、“CDATA 部分”和“注释”。每个文档的逻辑结构都有单个“文档”节点，它没有父节点，可包含零个或多个“元素”子节点。由此可见 DOM 处理 XML 的重要性。下面介绍下几个常见的处理 XML 数据的类。

1. XmlNode 类

.NET Framework 中定义了 1 个抽象类 XmlNode 类来表示 DOM 树中的所有节点。

- XmlNode 类的属性

XmlNode 类的属性及说明如表 13.2 所示。

表 13.2　　XmlNode 类的属性及说明

属　性	说　明
Attributes	获取 1 个 XmlAttributeCollection，它包含该节点的属性
BaseURI	获取当前节点的基 URI
ChildNodes	获取节点的所有子节点
FirstChild	获取节点的第 1 个子级
HasChildNodes	获取一个值，该值指示节点是否有任何子节点
InnerTex	获取或设置节点及其所有子节点的串联值
InnerXml	获取或设置仅代表该节点的子节点的标记
IsReadOnly	获取一个值，该值指示节点是否是只读的
Item	已重载。获取指定的子元素
LastChild	获取节点的最后一个子级
LocalName	当在派生类中被重写时，获取节点的本地名称
Name	当在派生类中被重写时，获取节点的限定名
NamespaceURI	获取该节点的命名空间 URI
NextSibling	获取紧接在该节点之后的节点
NodeType	当在派生类中被重写时，获取当前节点的类型
OuterXml	获取表示此节点及其所有子节点的标
OwnerDocument	获取该节点所属的 XmlDocument
ParentNode	获取该节点（对于可以具有父级的节点）的父级节点
Prefix	获取或设置该节点的命名空间前缀
PreviousSibling	获取紧接在该节点之前的节点
SchemaInfo	获取作为架构验证的结果分配给此节点的架构验证后信息集
Value	获取或设置节点的值

- XmlNode 类的方法

XmlNode 类的方法及说明如表 13.3 所示。

表 13.3 XmlNode 类的方法及说明

方　法	说　明
AppendChild	将指定的节点添加到该节点的子节点列表的末尾
CloneNode	创建该节点的副本
CreateNavigator	创建 XPathNavigator 以浏览此对象
GetNamespaceOfPrefix	查找当前节点范围内离给定的前缀最近的 xmlns 声明，并返回声明中的命名空间 URI
GetPrefixOfNamespace	查找当前节点范围内离给定的命名空间 URI 最近的 xmlns 声明，并返回声明中定义的前缀
InsertAfter	将指定的节点紧接着插入指定的引用节点之后
InsertBefore	将指定的节点紧接着插入指定的引用节点之前
Normalize	将此 XmlNode 下子树完全深度中的所有 XmlText 节点都转换成“正常”形式，在这种形式中只有标记（即标记、注释、处理指令、CDATA 节和实体引用）分隔 XmlText 节点，即没有相邻的 XmlText 节点
PrependChild	将指定的节点添加到该节点的子节点列表的开头
RemoveAll	移除当前节点的所有子节点和/或属性
RemoveChild	移除指定的子节点
ReplaceChild	用 newChild 节点替换子节点 oldChild
SelectNodes	选择匹配 Xpath 表达式的节点列表
SelectSingleNode	选择匹配 XPath 表达式的第 1 个 XmlNode
Supports	测试 DOM 是否实现特定的功能
WriteContentTo	该节点的所有子节点会保存到指定的 XmlWriter 中
WriteTo	当在派生类中被重写时，将当前节点保存到指定的 XmlWriter 中

2. XmlDocument 类

XmlDocument 类扩展了 XmlNode，并代表 XML 文档。它是内存方式的读取器，在内存中将 Xml 数据用树型结构来表示，它允许往返遍历树的各个节点，允许对节点进行读取和修改。

- XmlDocument 类的属性

XmlDocument 类的属性及说明如表 13.4 所示。

表 13.4 XmlDocument 类的属性及说明

属　性	说　明
Attributes	当前节点的属性集合
BaseURI	当前节点的基 URI
ChildNodes	节点的所有子节点
DocumentElement	文档的根
DocumentType	DOCTYPE 声明的节点
FirstChild	节点的第 1 个子节点
HasChildNodes	是否有任何子节点
Implementation	获取当前文档的 XmlImplementation 对象
InnerText	节点包含的所有文本内容
InnerXml	节点所包含的所有 XML 内容
IsReadOnly	当前节点是否是只读的
Item	获取指定的子元素
LastChild	最后一个子节点

续表

属　性	说　明
LocalName	获取节点的本地名称
Name	获取节点的限定名
NamespaceURI	获取该节点的命名空间 URI
NameTable	获取与此实现关联的 XmlNameTable
NextSibling	获取紧接在该节点之后的节点
NodeType	获取当前节点的类型
OuterXml	获取表示此节点及其所有子节点的标记
OwnerDocument	获取当前节点所属的 XmlDocument
ParentNode	获取该节点（对于可以具有父级的节点）的父级
Prefix	获取或设置该节点的命名空间前缀
PreserveWhitespace	获取或设置一个值，该值指示是否在元素内容中保留空白
PreviousSibling	获取紧接在该节点之前的节点
SchemaInfo	返回节点的后架构验证信息集（PSVI）
Schemas	获取或设置与此 XmlDocument 关联的 XmlSchemaSet 对象
Value	获取或设置节点的值
XmlResolver	设置 XmlResolver 以用于解析外部资源

- XmlDocument 类的方法

XmlDocument 类的方法及说明如表 13.5 所示。

表 13.5　　XmlDocument 类的方法及说明

方　法	说　明
AppendChild	将指定的节点添加到该节点的子节点列表的末尾
CreateAttribute	创建具有指定名称的 XmlAttribute
CreateCDataSection	创建包含指定数据的 XmlCDataSection
CreateComment	创建包含指定数据的 XmlComment
CreateDocumentFragment	创建 XmlDocumentFragment
CreateDocumentType	返回新的 XmlDocumentType 对象
CreateElement	创建 XmlElement
CreateEntityReference	创建具有指定名称的 XmlEntityReference
CreateNavigator	创建 1 个用于导航此文档的新 XPathNavigator 对象
CreateNode	创建 XmlNode
CreateProcessingInstruction	创建 1 个具有指定名称和数据的 XmlProcessingInstruction
CreateSignificantWhitespace	创建 1 个 XmlSignificantWhitespace 节点
CreateTextNode	创建具有指定文本的 XmlText
CreateWhitespace	创建 1 个 XmlWhitespace 节点
CreateXmlDeclaration	创建 1 个具有指定值的 XmlDeclaration 节点
GetElementById	获取具有指定 ID 的 XmlElement
GetElementsByTagName	返回一个 XmlNodeList，它包含与指定名称匹配的所有元素的列表
GetNamespaceOfPrefix	查找当前节点范围内离给定的前缀最近的 xmlns 声明，并返回声明中的命名空间 URI
GetPrefixOfNamespace	查找当前节点范围内离给定的命名空间 URI 最近的 xmlns 声明，并返回声明中定义的前缀

续表

方　法	说　明
GetType	获取当前实例的 Type
ImportNode	将节点从另一个文档导入当前文档
InsertAfter	将指定的节点紧接着插入指定的引用节点之后
InsertBefore	将指定的节点紧接着插入指定的引用节点之前
Load	加载指定的 XML 数据
LoadXml	从指定的字符串加载 XML 文档
Normalize	将 XmlText 节点都转换成“正常”形式
PrependChild	将指定的节点添加到该节点的子节点列表的开头
ReadNode	根据 XmlReader 中的信息创建 1 个 XmlNode 对象，读取器必须定位在节点或属性上
RemoveAll	移除当前节点的所有子节点和/或属性
RemoveChild	移除指定的子节点
ReplaceChild	用新节点替换旧节点
Save	将 XML 文档保存到指定的位置
SelectNodes	选择匹配 Xpath 表达式的节点列表
SelectSingleNode	选择匹配 XPath 表达式的第 1 个 XmlNode
Supports	测试 DOM 实现是否实现特定的功能
Validate	验证 XmlDocument 是不是 Schemas 属性中包含的 XML 架构定义语言(XSD)架构
WriteContentTo	将 XmlDocument 节点的所有子级保存到指定的 XmlWriter 中
WriteTo	将 XmlDocument 节点保存到指定的 XmlWriter

下面对以上 XmlDocument 类的方法列表中的几个重用的方法进行介绍。

- Load 方法

该方法可以从 1 个字符串指定的 XML 文件或是 1 个流对象、1 个 TextReader 对象、1 个 XmlReader 对象导入 XML 数据。

- LoadXml 方法

该方法完成从一个特定的 XML 文件导入 XML 数据的功能。默认情况下，LoadXml 方法既不保留空白，也不保留有意义的空白。此方法不执行 DTD 或架构验证。

- Save 方法

该方法将 XML 数据保存到 1 个 XML 文件或是 1 个流对象、1 个 TextReader 对象、1 个 XmlReader 对象导入 XML 数据。

- XmlDocument 类的事件

XmlDocument 类的事件及说明如表 13.6 所示。

表 13.6　　XmlDocument 类的事件及说明

事　件	说　明
NodeChanged	当属于该文档的节点的 Value 已被更改时发生
NodeChanging	当属于该文档的节点的 Value 将被更改时发生
NodeInserted	当属于该文档的节点已被插入另一个节点时发生
NodeInserting	当属于该文档的节点将被插入另一个节点时发生
NodeRemoved	当属于该文档的节点已被从其父级移除时发生
NodeRemoving	当属于该文档的节点将被从文档中移除时发生

13.3.2 ASP.NET 操作 XML 文档

如果需要在程序中处理 XML 数据，首先要创建 1 个 XmlDocument 对象，然后通过 Load 方法从 XML 数据源中加载数据。当 XML 文件被 XmlDocument 对象加载至内存后，DOM 树就自动生成了。这时就可以对 DOM 树中的节点进行操作。

由于 XML 中的数据主要表现为元素和属性两种形式。因此，读取节点数据的方法也就分为读取元素和读取属性两种，下面分别进行介绍。

1. 读取元素

读取元素可以通过某个节点的 ChildNodes 属性获取其子节点集合，然后，根据节点的 NodeType 属性确定此节点是否属于元素节点。元素节点的名称可以通过 Name 属性获取，其所对应的 XML 数据可以由 InnerXML 和 OuterXML 属性获取。

例如，对于 XML 元素：<Name>小聪聪</Name>在 DOM 中对应的节点，它的 InnerXML 的值为“小聪聪”。

2. 读取属性

要访问某个属性的值，必须获取该属性所属的元素。当前节点是元素时，可以使用 HasAttribute 方法查看是否存在任何与此元素关联的属性。从某元素中检索属性，可使用 XMLElement 对象的 GetAttribute 和 GetAttributeNode 方法。GetAttribute 方法以字符串的方式返回属性值，而 GetAttributeNode 方法返回 1 个 XMLAttribute 对象，代表一个属性节点。

首先在页面的 Page_Load 事件中创建 1 个 XmlDocument 类对象，然后使用它的 Load 方法将 Xml 文件内容装载到内存。最后，调用自定义方法 showNode，将 XML 文件中的各节点显示在页面上。实现的代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    XmlDocument doc = new XmlDocument();///创建并实例化一个XmlDocument对象
    doc.Load(Server.MapPath("Student.xml"));///将Student.xml文件内容装载到内存
    showNode(doc);///调用自定义方法showNode
}
```

由于 XmlDocument 在内存中是被解析为树的形式，所以需要使用递归函数遍历 XML 文件的所有节点。实现遍历 XM 的代码如下：

```
void showNode(XmlNode node)
{
    Response.Write("<li>节点类型："+node.NodeType.ToString ()+"节点名称："+node.LocalName );
    if (!node.HasChildNodes)
    {
        Response.Write(" "+node.InnerText);
    }
    if (node.NodeType == XmlNodeType.Element)
    {
        if (node.Attributes.Count > 0)
        {
            for (int i = 0; i < node.Attributes.Count; i++)
            {
                Response.Write("属性："+node.Attributes [i].LocalName +"="+node.Attributes [i].Value);
            }
        }
    }
    if (node.HasChildNodes)
    {
        for (int i = 0; i < node.ChildNodes.Count; i++)
        {
            showNode(node.ChildNodes [i]);
```

```
            }
        }
    }
```

上述代码实现过程中，主要是调用了名为 Student.xml 的 XML 文档，该 XML 文档的主要结构代码编写如下：

```
<?xml version="1.0" encoding="utf-8" ?>
<StuInfo xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <student>
    <ID>1001</ID>
    <Name>小贯</Name>
    <Sex value ="女"></Sex>
    <Birthday age ="23">1985-02-16</Birthday>
  </student>
  <student>
    <ID >1002</ID>
    <Name>小房</Name>
    <Sex/>
    <Birthday age ="27">1982-10-17</Birthday>
  </student>
</StuInfo>
```

13.3.3 创建 DOM 节点

XML 文件中每一项都可以被认为是一个节点。在 XML 文件中 XmlDocument 为所有节点类型提供了一系列的 CreateXXX 方法。具体介绍如下。

- CreateCDataSection：创建包含指定数据的 XmlCDataSection。
- CreateComment：创建包含指定数据的 XmlComment。
- CreateDocumentType：返回新的 XmlDocumentType 对象。
- CreateNode：创建 XmlNode。
- CreateTextNode：创建具有指定文本的 XmlText。
- CreateXmlDeclaration：创建 1 个具有指定值的 XmlDeclaration 节点。

创建新节点后，可应用以下几个方法将节点插入树中，其使用的常见的方法及方法说明如表 13.7 所示。

表 13.7　　将节点插入到树中的方法及说明

方　法	说　明
InsertBefore	插入引用节点之前
InsertAfter	插入引用节点之后
AppendChild	将节点添加到给定节点的子节点列表的末尾。如果所添加的节点是 XmlDocumentFragment，则会将文档片段的全部内容移至该节点的子列表中
PrependChild	将节点添加到给定节点的子节点列表的开头。如果所添加的节点是 XmlDocumentFragment，则会将文档片段的全部内容移至该节点的子列表中
Append	将 XmlAttribute 节点追加到与元素关联的属性集合的末尾

以下代码实现的是如何创建 DOM 节点，并将节点添加在 XML 文件中，程序代码如下：

```
public void createNode()
{
    //创建XmlDocument对象
    XmlDocument doc = new XmlDocument();
    //加载XML文件
    doc.Load(Server.MapPath("Books.xml"));
    //创建1个Book节点
    XmlElement BookNode = doc.CreateElement("Book");
    //将Book节点添加到根节点下
```

```
        doc.DocumentElement.PrependChild(BookNode);
        //创建存储信息的节点
        XmlElement BookNameNode = doc.CreateElement("BookName");
        XmlElement BookPriceNode = doc.CreateElement("Price");
        //将文本信息添加到相应的节点上
        BookNameNode.AppendChild(doc.CreateTextNode("ASP.NET数据库系统开发案例精选"));
        BookPriceNode.AppendChild(doc.CreateTextNode("49.0元"));
        //将BookNameNode和BookPriceNode添加到Book节点下
        BookNode.AppendChild(BookNameNode);
        BookNode.AppendChild(BookPriceNode);
        //保存XML文件
        doc.Save(Server.MapPath("Books.xml"));
    }
```

13.3.4 创建 DOM 节点的属性

在创建 DOM 节点的新属性时，首先需要获取该元素节点并使用 SetAttribute 方法将创建的新属性添加到该元素的属性集合中。

例如，可以使用 CreateAttribute 方法创建 XMLAttribute 节点，接着获取元素节点，然后使用 SetAttributeNode 将属性节点添加到该元素的属性集合，以下代码用于实现创建 DOM 节点的属性：

```
    void createAttribute()
    {
        XmlDocument doc = new XmlDocument();
        doc.Load(Server.MapPath ("Books.xml"));
        XmlNode rootNode = doc.DocumentElement;
        //查找元素节点
        XmlNode node = rootNode.SelectSingleNode("//Books/Book/BookName");
        //创建一个新的属性
        XmlAttribute attr = doc.CreateAttribute("ID");
        //创建属性文本
        attr.InnerText = "1001";
        //将属性添加到相应节点
        node.Attributes.Append(attr);
        doc.Save(Server.MapPath ("Books.xml"));
    }
```

13.3.5 修改 DOM 节点

创建完 DOM 节点可对其进行修改操作，可以通过以下方法修改文档中的节点和内容。

- 使用 Value 属性更改节点的值。
- 通过用新节点替换节点来修改全部节点集。此操作使用 InnerXml 属性完成。
- 使用 RemoveChild 方法将现有节点替换为新节点。
- 使用 AppendData、InsertData 或 ReplaceData 方法向从 XmlCharacterData 类继承的节点添加附加字符。
- 通过在从 XmlCharacterData 继承的节点类型上使用 DeleteData 方法移除某个范围的字符，以修改内容。

更改节点值的一个简单方法是使用 node.Value ="NewValue"。此代码（即 node.Value ="NewValue"）所作用的节点类型，以及对于该节点类型更改的确切数据如表 13.8 所示。

表 13.8　　节点类型及对节点类型更改的确切数据

节 点 类 型	更改的数据
Attribute	属性的值
CDATASection	CDATA 节的内容
Comment	注释的内容

续表

节 点 类 型	更改的数据
ProcessingInstruction	内容（不包括目标）
Text	文本的内容
XmlDeclaration	声明的内容（不包括<?xml 和?>标记）
Whitespace	空白的值。可以将该值设置为 4 个可识别的 XML 空白字符之一：空格、制表符、回车或换行
SignificantWhitespace	有效空白的值。可以将该值设置为 4 个可识别的 XML 空白字符之一：空格、制表符、回车或换行

根据以上列表对修改 DOM 节点的说明可编写如下代码来实现修改 DOM 节点的内容，代码如下：

```
public void createAttribute()
{
    XmlDocument doc = new XmlDocument();
    doc.Load(Server.MapPath("Books.xml"));
    XmlNode rootNode = doc.DocumentElement;
    //查找元素节点
    XmlNode node = rootNode.SelectSingleNode("//Books/Book/BookName");
    node.InnerText = "ASP.NET数据库系统开发案例精选";
    doc.Save(Server.MapPath("Books.xml"));
}
```

13.3.6 删除 DOM 节点

删除 DOM 节点很简单，如果用户想要移除 XML 文档对象模型（DOM）中的节点，可以使用 RemoveChild 方法移除特定节点。在移除节点时，该方法将移除属于所移除节点的子树。要移除 DOM 中的多个节点，可以使用 RemoveAll 方法移除当前节点的所有子级和属性。

如果想移除属性节点，首先要使用 Attribute 属性获取其属性的集合，然后再通过该元素的 RemoveAllAttributes、RemoveAttribute 或 RemoveAttributeAt 方法删除属性。

13.3.7 使用 DataSet 加载 XML

DataSet 是 ADO.NET 的主要组件，它是从数据源中检索到的数据缓存在内存中时的表示。DataSet 可将数据和架构作为 XML 文档进行读写。数据和架构可通过 HTTP 传输，并在支持 XML 的任何平台上被应用程序使用。

例如，使用 DataSet 的 ReadXml()方法将 XML 数据加载到 DataSet 中，本例运行结果如图 13.3 所示。

若实现如图 13.3 所示的绑定 XML 信息，首先需要在 Page_Load 事件中创建 1 个 DataSet 类型的数据集 ds，然后应用 DataSet 数据集中的 ReadXml 读取指定路径中的 XML 文件，最后将其读取的 XML 信息绑定到列表控件中，具体实现的代码如下：

BookName	Price
ASP.NET数据库系统开发案例精选	49.0元
C#数据库系统开发完全手册	68.0元
ASP.NET数据库系统开发完全手册	52.0元

图 13.3 使用 DataSet 加载 XML 数据

```
protected void Page_Load(object sender, EventArgs e)
{
    DataSet ds = new DataSet();
    ds.ReadXml(Server.MapPath ("Books.xml"));
    GridView1.DataSource = ds;
    GridView1.DataBind();
}
```

注意

这里应用到的 Books.xml 文件是自定义的 XML 文件。

13.4 实 现 过 程

13.4.1 将 XML 中的留言信息绑定到 GridView 中

将 XML 中的留言信息显示到 GridView 控件中，并且以表格的形式显示数据源中的数据，主要是在“Default.aspx”页面中实现的，当页面加载将 XML 中的留言信息绑定到 GridView 控件中，并且单击 GridView 控件上的“详细信息”按钮时，页面将跳转到留言本的详细页面，运行结果如图 13.4 所示。

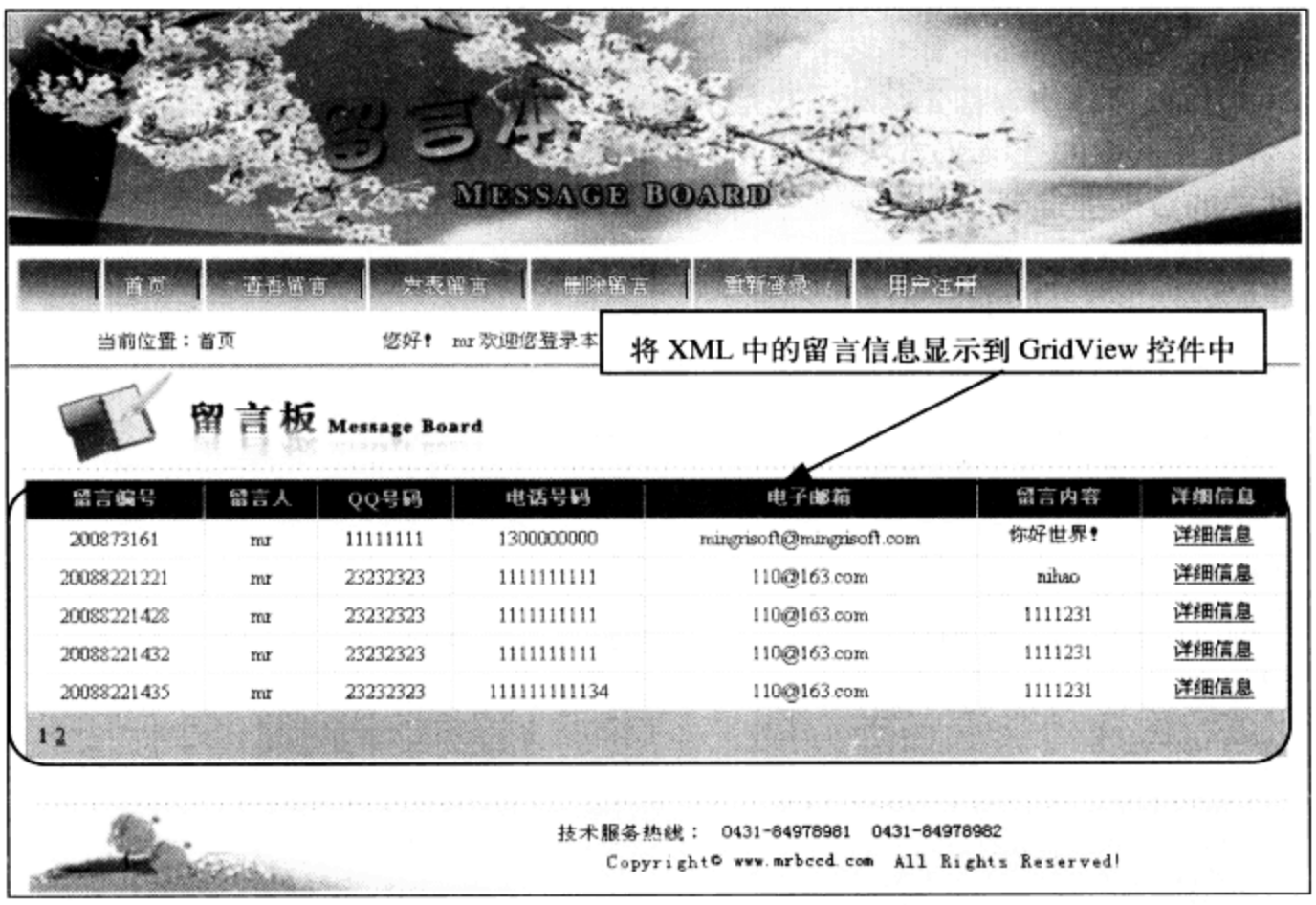


图 13.4 运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，默认名为“Default.aspx”。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Label 控件、1 个 XmlDataSource 控件和 1 个 GridView 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 13.9 所示。

表 13.9 基于 XML 技术的留言本信息页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用 途
Table	无	无	页面布局
Web 用户控件	控件的名称分别为 Header.ascx 和 footer.ascx	无	Header.ascx 实现网站的页面导航功能，footer.ascx 页面布局功能
A Label	控件的名称为 LblMessage	无	显示信息

续表

控 件 类 型	控 件 名 称	主要属性设置	用　途
XmlDataSource	控件的名称为 XmlDataSourceLW	将控件的 DataFile 属性设置为~/Leave Word.ml、将 TransformFile 属性设置为~/LeaveWord1.xsl	绑定 XML 数据的数据源
GridView	控件的名称为 GvLeaveWord	将控件的 AllowPaging 属性设置为 True、将 Auto GenerateColumns 属性设置为 False、将 DataSource ID 属性设置为 XmlDataSourceLW、将 PageSize 属性设置为 5 和 OnPageIndexChanging 属性设置为 GvLeaveWord_PageIndexChanging	实现将 XML 中的数据显示到 GridView 控件中

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 4　代码位置：光盘\mr\04\LeaveWordBook+XML\Default.aspx.cs

```
using System.Xml;//引入命名空间
```

在 Page_load 事件中，首先判断用户是否登录，如果用户已登录，则创建 XmlDocument 类的实例，并调用其中的 Load()方法加载 XML 文件；然后调用自定义方法 bindXml()将指定的 XML 文件中的数据绑定到 GridView 控件中，实现的代码如下。

例程 5　代码位置：光盘\mr\04\LeaveWordBook+XML\Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
  if (Session["UserName"] == null && Session["PassWord"] == null)
  {
    if (Session["AdminName"] != null && Session["AdminPassWord"] != null)
    {
      this.LblMessage.Text = "您好！ " + Session["AdminName"].ToString() + "  欢迎您登录本网站！ ";
    }
    else
    {
      Response.Write("<script language='javascript'>alert('您还没有登录，请您先登录！');location='Login.aspx';</script>");
    }
  }
  else
  {
      this.LblMessage.Text = "您好！ " + Session["UserName"].ToString() + "  欢迎您登录本网站！ ";
      //创建XmlDocument类的实例
      XmlDocument doc = new XmlDocument();
      //调用XmlDocument类中的Load()方法加载XML文件
      doc.Load(Server.MapPath("LeaveWord.xml"));
      //调用自定义方法bindXml()绑定GridView控件中的数据
      bindXml();
  }
}
```

在自定义 bindXml 方法中，主要实现将 xml 文件中的数据绑定到 XmldataSource 中，实现的代码如下。

例程 6　代码位置：光盘\mr\04\LeaveWordBook+XML\Default.aspx.cs

```
public void bindXml()
{
    XmlDocument mydoc = this.XmlDataSourceLW.GetXmlDocument();
}
```

在 GvLeaveWord_PageIndexChanging 事件中，实现 GridView 控件的分页功能，实现的代码如下。

例程 7　代码位置：光盘\mr\04\LeaveWordBook+XML\Default.aspx.cs

```
protected void GvLeaveWord_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    this.GvLeaveWord.PageIndex = e.NewPageIndex;
    this.GvLeaveWord.DataBind();
}
```

13.4.2 将留言信息保存到 XML 中

本模块在添加留言信息时，主要是将新添加的留言信息保存在 XML 中，实现发表留言信息，此功能是在“AddLeaveWord.aspx”页面中实现的，发表留言信息的运行效果如图 13.5 所示。

图 13.5　发表留言信息运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，默认名为“AddLeaveWord.aspx”，作为发表留言信息页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 7 个 TextBox 控件、2 个 Image 控件和 2 个 Button 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及用途如表 13.10 所示。

表 13.10　　发表留言信息页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称分别为 Header.ascx 和 footer.ascx	无	Header.ascx 实现网站的页面导航功能，footer.ascx 页面布局功能
TextBox	控件的名称分别为 TxtID、TxtName、TxtEmail、TxtQQ、TxtTel、TxthomePage 和 TxtLeaveWord	将控件名为 TxtID 的 Visible 属性设置为 False	用于向留言中输入的信息内容
Button	控件的名称分别设置为 BtnAdd LeaveWord 和 BtnCZ	将控件名称为 BtnAddLeave Word 的 OnClick 属性设置为 BtnAdd LeaveWord_Click，并将 Text 属性设置为“添加留言”，将控件名称为 BtnCZ 的 On Click 属性设置为BtnCZ_Click，并且将 Text 属性设置为重置	用于执行添加留言信息操作和清空文本框内容信息操作

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 8　代码位置：光盘\mr\04\LeaveWordBook+XML\AddLeaveWord.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 页面加载的事件当中，首先判断用户或者管理员是否登录过，如果已经登录过给出登录提示信息，否则跳转到登录页面。Page_Load 事件中的代码如下。

例程 9　代码位置：光盘\mr\04\LeaveWordBook+XML\AddLeaveWord.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        if (Session["UserName"] == null && Session["PassWord"] == null)
            {
                if (Session["AdminName"] != null && Session["AdminPassWord"] != null)
                {
                    this.LblMessage.Text = "您好！  " + Session["AdminName"].ToString() + "  欢迎您登录本网站！ ";
                }
                  else
                  {
                      Response.Write("<script language='javascript'>alert('您还没有登录，请您先登录！
');location='Login.aspx';</script>");
                  }
                }
                else
                {
                    if (!this.IsPostBack)
                    {
                    this.LblMessage.Text = "您好！  " + Session["UserName"].ToString() + "  欢迎您登录本网站！ ";
                        this.clearText();//调用用户自定义的方法
                    }
                }
            }
        }
```

双击页面中的“添加留言”按钮，触发 BtnAddLeaveWord_Click 事件，在该事件中，首先根据当前时间获取一个 11 位的数字赋值给留言编号，然后定义 1 个 XmlDocument 类对象，使用其 Load 方法加载指定的 XML 文件。并且调用 XmlDocument 对象的 CreateElement 方法生成新的元素，最后使用 AppendClild 方法将新生成的元素添加到 XML 文件中，将留言信息成功添加到指定的 Xml 文件中，弹出提示对话框“添加成功!”的字样，实现的代码如下。

例程 10　代码位置：光盘\mr\04\LeaveWordBook+XML\AddLeaveWord.aspx.cs

```
protected void BtnAddLeaveWord_Click(object sender, EventArgs e)
{ //获取当前日期的年、月、日转换成字符串类型
    string date = DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString() + DateTime.Now.Day.ToString();
    //获取当前时间的小时，分钟转换成字符串类型
    string time = DateTime.Now.Hour.ToString() + DateTime.Now.Second.ToString();
    this.TxtID.Text = date + time;        //返回一个11位的数字
    XmlDocument doc1 = new XmlDocument();
    doc1.Load(Server.MapPath("LeaveWord.xml"));
    XmlNode newNode1;
    XmlNode newNode2;
    newNode1 = doc1.CreateElement("LW");
    newNode2 = doc1.CreateElement("ID");
    newNode2.InnerText = this.TxtID.Text;
    newNode1.AppendChild(newNode2);
    newNode2 = doc1.CreateElement("name");
    newNode2.InnerText = this.TxtName.Text;
    newNode1.AppendChild(newNode2);
    newNode2 = doc1.CreateElement("QQ");
    newNode2.InnerText = this.TxtQQ.Text;
    newNode1.AppendChild(newNode2);
    newNode2 = doc1.CreateElement("tel");
    newNode2.InnerText = this.TxtTel.Text;
     newNode1.AppendChild(newNode2);
```

```
        newNode2 = doc1.CreateElement("mail");
        newNode2.InnerText = this.TxtEmail.Text;
        newNode1.AppendChild(newNode2);
        newNode2 = doc1.CreateElement("url");
        newNode2.InnerText = this.Txthomepage.Text;
        newNode1.AppendChild(newNode2);
        newNode2 = doc1.CreateElement("message");
        newNode2.InnerText = this.TxtLeaveWord.Text;
        newNode1.AppendChild(newNode2);
        doc1.DocumentElement.AppendChild(newNode1);
        doc1.Save(Server.MapPath("LeaveWord.xml"));
        Response.Write("<script>alert('添加成功！');location='javascript:history.go(-1)'</script>");
    }
```

双击页面中的“重置”按钮，触发其 BtnCZ_Click 事件，在该事件中主要调用了 ClearText 方法实现清空文本框的功能，实现的代码如下。

例程 11 代码位置：光盘\mr\04\LeaveWordBook+XML\AddLeaveWord.aspx.cs

```
protected void BtnCZ_Click(object sender, EventArgs e)
{
        this.clearText();//调用用户自定义方法
}
```

自定义 1 个 clearText 方法中，主要用于清空文本框，实现的代码如下。

例程 12 代码位置：光盘\mr\04\LeaveWordBook+XML\AddLeaveWord.aspx.cs

```
public void clearText()
{
    this.TxtID.Text = "";
    this.TxtName.Text = "";
    this.TxtQQ.Text = "";
    this.TxtTel.Text = "";
    this.TxtEmail.Text = "";
    this.Txthomepage.Text = "";
    this.TxtLeaveWord.Text = "";
}
```

13.4.3 在 XML 文件中查询留言相关内容

本程序在查询留言内容时，主要是根据留言编号查找与其对应的相关信息，此功能主要在“selectLeaveWord.aspx”页面中实现的，查询留言信息的运行效果如图 13.6 所示。

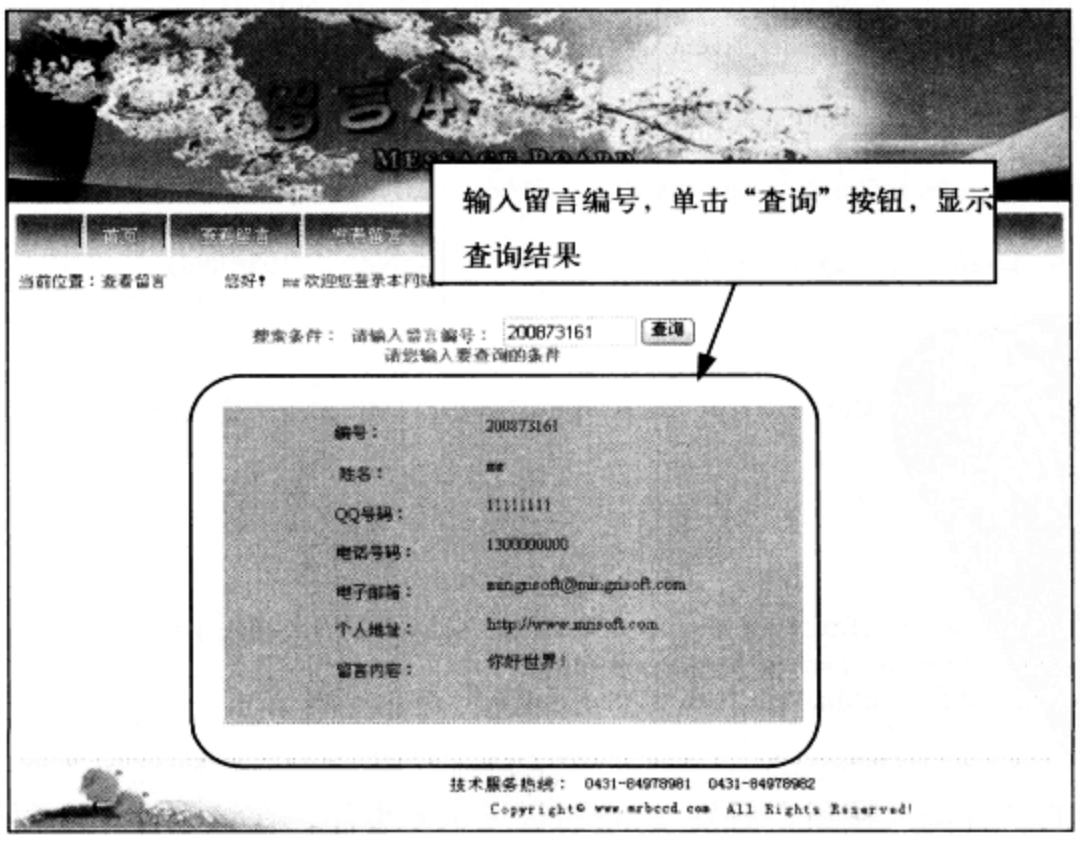


图 13.6 查询留言信息

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，默认名为“selectLeaveWord.aspx”，作为发表留言信息页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 7 个 TextBox 控件、2 个 Image 控件和 1 个 Button 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 13.11 所示。

表 13.11　　查询留言信息页控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称分别为 Header.ascx 和 footer.ascx	无	Header.ascx 实现网站的页面导航功能，footer.ascx 页面布局功能
TextBox	控件的名称为 TxtName	无	用于显示留言人姓名
Image	控件的名称分别为 ImgName 和 ImgPwd	将控件的 ImageUrl 属性分别设置为~/image/name.bmp 和~/image/pwd.bmp	用于显示信息
Button	控件的名称为 BtnSelect	将控件的 OnClick 属性设置为 BtnSelect_Click，并且将控件的 Text 属性设置为查询	用于执行查询操作

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 13　代码位置：光盘\mr\04\LeaveWordBook+XML\selectLeaveWord.aspx

```
using System.Xml;
```

在 Page_load 事件中，判断用户或者管理员是否登录过，如果已经登录过，则弹出登录提示信息，并且将 LeaveWord.Xml 中的数据绑定到 GridView 控件中，如果没有登录过则跳转到登录页面。实现的代码如下。

例程 14　代码位置：光盘\mr\04\LeaveWordBook+XML\selectLeaveWord.aspx

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["UserName"] == null && Session["PassWord"] == null)
    {
        if (Session["AdminName"] != null && Session["AdminPassWord"] != null)
        {
            this.LblMessage.Text = "您好！  " + Session["AdminName"].ToString() + "  欢迎您登录本网站！ ";
        }
        else
        {
            Response.Write("<script language='javascript'>alert('您还没有登录，请您先登录！
');location='Login.aspx';</script>");
        }
    }
    else
    {
        this.LblMessage.Text = "您好！  " + Session["UserName"].ToString() + "  欢迎您登录本网站！ ";
        XmlDocument doc = new XmlDocument();
        doc.Load(Server.MapPath("LeaveWord.xml"));
    }
}
```

双击页面中的“查询”按钮，触发 BtnSelect_Click 事件，在该事件中主要使用 foreach 语句遍历 LeaveWord.xml 文件中的节点，并将节点信息显示出来，实现的代码如下。

例程 15　代码位置：光盘\mr\04\LeaveWordBook+XML\selectLeaveWord.aspx

```
protected void BtnSelect_Click(object sender, EventArgs e)
{
    this.lblcondition.Text = "";//清空标签中的内容
    //创建XmlDocument类的实例
    XmlDocument doc = new XmlDocument();
    //调用XmlDocument类中的Load()方法加载XML文件
    doc.Load(Server.MapPath("LeaveWord.xml"));
    XmlNodeList nodes;
    XmlElement root = doc.DocumentElement;
    nodes = root.SelectNodes("descendant::LW[ID='" + TxtID.Text.Trim() + "']");
    foreach (XmlNode node in nodes)
    {
        if (lblcondition.Text == "")
        {
            for (int i = 0; i <= node.ChildNodes.Count - 1; i++)
            {
                lblcondition.Text = lblcondition.Text + node.ChildNodes[i].InnerText + "<br><br>";
            }
        }
        else
        {
            lblcondition.Text = "";
            for (int i = 0; i <= node.ChildNodes.Count - 1; i++)
            {
                lblcondition.Text = lblcondition.Text + node.ChildNodes[i].InnerText + "<br>";
            }
        }
    }
}
```

13.5　从 XML 文件中删除指定留言信息

本程序主要是根据留言编号删除基于 XML 技术的留言本中的信息，此功能主要在“DeleteLeaveWord.aspx”页面中实现，删除留言信息的运行效果如图 13.7 所示。弹出删除指定留言信息的如图 13.8 所示。

图 13.7　删除留言信息的运行的效果

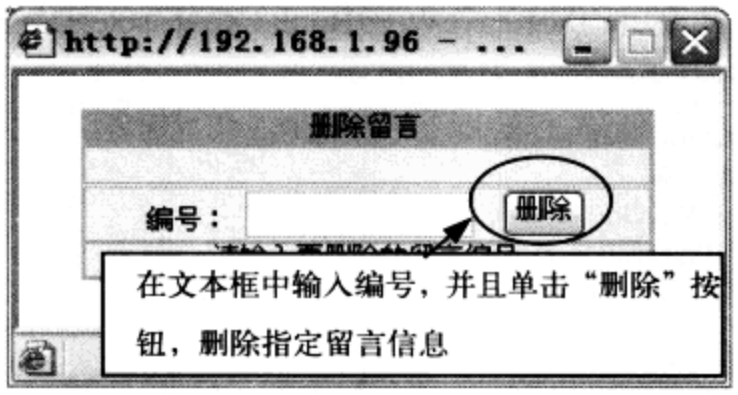


图 13.8　弹出删除指定留言信息

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，默认名为“DeleteLeaveWord.aspx”，作为发表留言信息页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1GridView 控件通过属性窗口，设置控件的属性。页面中各个控件的属性设置及用途如表 13.12 所示。

表 13.12　　删除留言信息页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
Web 用户控件	控件的名称分别为 Header.ascx 和 footer.ascx	无	Header.ascx 用于实现网站的页面导航功能，footer.ascx 用于实现页面布局功能
GridView	控件的名称为 GridView1	将控件的 DataSourceID 属性设置为 XmlDataSource1，将控件的 AllowPaging 属性设置为 True，将控件的 PageSize 属性设置为 5	绑定 XML 数据

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 16　代码位置：光盘\mr\04\LeaveWordBook+XML\DeleteLeaveWord.aspx.aspx

```
using System.Xml;
using System.Xml.Xsl;
```

在 Page_load 事件中，首先创建 XmlDocument 类的实例，并调用其中的 Load()方法加载指定的 XML 文件；然后定义 1 个 XslTransform 类的实例，并使用其 Load()方法加载指定的 XSL 文件，最后使用 XslTransform 类的对象对 XML 文件进行格式转换，实现的代码如下。

例程 17　代码位置：光盘\mr\04\LeaveWordBook+XML\DeleteLeaveWord.aspx.aspx

```
protected void Page_Load(object sender, EventArgs e)
{
    XmlDocument doc = new XmlDocument();
    doc.Load(Server.MapPath("LeaveWord.xml"));
    XslTransform trans = new XslTransform();
    trans.Load(Server.MapPath("LeaveWord.xsl"));
    Xml1.Document = doc;
    Xml1.Transform = trans;
}
```

单击 GridView 控件中的“删除”按钮，将弹出删除 XML 文件中数据的窗口，在该窗口中用户可选择要删除的留言信息编号，进行 XML 文件中数据的删除操作。“删除”按钮的 Click 事设置如下。

例程 18　代码位置：光盘\mr\04\LeaveWordBook+XML\DeleteLeaveWord.aspx.aspx

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text.Trim() != "")
    {
        XmlDocument doc = new XmlDocument();
        doc.Load(Server.MapPath("LeaveWord.xml"));
        XmlNodeList nodes;
        XmlElement root = doc.DocumentElement;
        nodes = root.SelectNodes("descendant::LW[ID='" + TextBox1.Text.Trim() + "']");
        foreach (XmlNode node in nodes)
        {
            root.RemoveChild(node);
        }
            TextBox1.Text = "";
            Response.Write("<script>alert('恭喜你！删除成功');location=javascript:history:go(-1)</script>");
            doc.Save(Server.MapPath("LeaveWord.xml"));
            XslTransform trans = new XslTransform();
            trans.Load(Server.MapPath("LeaveWord.xsl"));
            Xml1.Document = doc;
            Xml1.Transform = trans;
        }
        else
        {
            Response.Write("<script>alert('请输入要删除的编号！');;location=javascript:history:go(-1)</script>");
        }
    }
```

13.6　程序错误与调试

在对程序进行调试时，登录页面如图 13.9 所示，当输入用户名和密码时，可能出现如图 13.10 所示的错误。

图 13.9　运行效果图

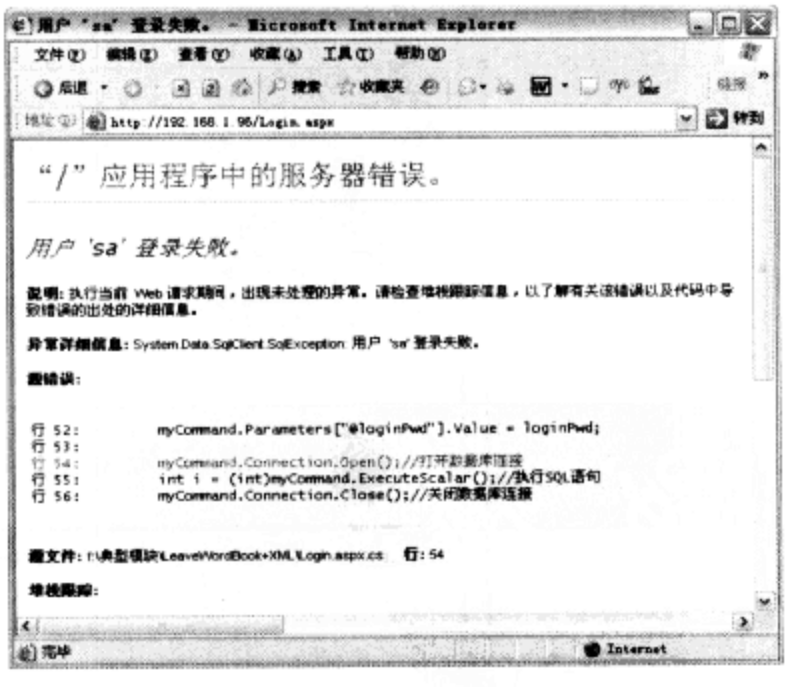


图 13.10　提示错误页

出现图 13.10 所示的错误，是因为在连接数据库时没有根据当前机器 Sql Server 2000 的用户名和密码配置连库字符串。此时可以在本程序的 LeaveWord.cs 公共类中，找到连接数据库的 getcon 方法，代码如下。

例程 19　代码位置：光盘\mr\04\LeaveWordBook+XML\App_Code\LeaveWord.cs

```
#region  连接数据库
/// <summary>
/// 创建时间:2008-8-18
/// 连接数据库
/// </summary>
public SqlConnection getcon()
{
    string strCon = "Data Source=(local);DataBase=db_LeaveWordXML;User ID=sa;PWD=sa";
    SqlConnection sqlCon = new SqlConnection(strCon);
    return sqlCon;
}
#endregion
```

设置连接字符串中的登录用户名（UserID）和密码（PWD），例如，笔者当前使用的数据库的用户名（UserID）是 sa，用户密码（PWD）为空，所以修改后的 getcon 方法如下。

例程 20　代码位置：光盘\mr\04\LeaveWordBook+XML\App_Code\LeaveWord.cs

```
  #region  连接数据库
/// <summary>
/// 连接数据库
/// </summary>
public SqlConnection getcon()
{
    string strCon = "Data Source=(local);DataBase=db_LeaveWordXML;User ID=sa;PWD=";
    SqlConnection sqlCon = new SqlConnection(strCon);
    return sqlCon;
}
#endregion
```

设置完毕后“基于 XML 技术的留言本”便能够运行，其效果如图 13.11 所示。

图 13.11　运行效果图

上传与下载模块

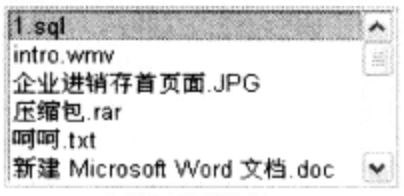
第14章

实例位置：光盘\mr\14\

在开发 Web 应用程序过程中，经常涉及对文件的上传或下载操作。在以前的 Web 应用程序中，要实现文件上传是件麻烦的事，但这一操作在 Web 应用程序中又会经常用到，因此令开发人员非常头痛。而在 ASP.NET 2.0 中，要实现文件的上传或下载不再是难事，它变得非常轻而易举。本章将介绍实现多文件上传、文件删除以及对文件搜索、断点续传等多方面的操作。通过学习本章，读者可以学习以下内容。

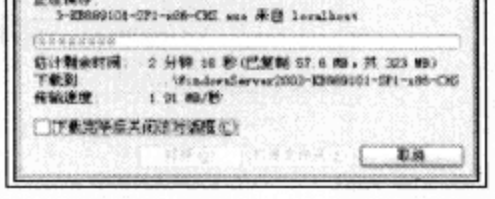
使用 FileUpload 控件将文件批量上传

将下载文件绑定到控件中



上传图片并且生成缩略图

断点续传



14.1 上传与下载概述

文件的上传是在客户端将文件上传到服务器上，文件下载是将服务器端文件下载到客户端，其中的文件的上传功能主要利用 FileUpload 控件（文件上传控件）来完成，并通过按钮来实现动态添加 FileUpload 控件。

本实例实现的具体功能如下。

- 多文件上传功能。
- 动态添加文件上传控件。
- 判断上传的文件是否为空。
- 在上传的图片中选择指定位置添加文字。
- 上传图片生成缩略图。
- 文件下载。

本章主要介绍一个简单的文件上传下载模块，具体实现过程，程序运行结果如图 14.1 所示。

图 14.1 上传下载首页运行效果图

为了方便读者阅读和有效地利用本书附赠光盘中的实例，网站页面的各部分说明以列表形式，如表 14.1 所示。

表 14.1 网站首页解析

区 域	名 称	说 明	对 应 文 件
1	网站头	主要用于网站的旗帜广告	Default.aspx
2	选择要上传的文件	主要用于选择要上传的文件	Default.aspx
3	动态添加上传文件操作	单击“上传文件”按钮，添加上传控件	Default.aspx
4	上传文件区	主要用于执行上传文件	Default.aspx
5	下载文件区	主要用于将页面跳转到文件下载页	Default.aspx
6	上传处理图片区	主要用于将页面跳转到上传处理图片页	Default.aspx
7	网站站脚标	用于显示客服的电话等信息内容	Default.aspx

14.2 上传与下载关键技术

14.2.1 上传文件存储在指定目录

文件上传是一个完整网站必须具备的功能之一，如果网站中不具备上传功能，会显得不完整。而且开发人员在开发网站时往往要求网站要具有上传的功能，因为用户不能每次上传文件时都进入 ftp 进行操作，登录 ftp 对用户名和密码要求都很严格，为了简化这一操作，ASP.NET 提出了文件上传的方法。

在 Default.aspx 页面中，即网站首页中使用 FileUpLoad 控件实现上传文件到指定目录中。具体步骤如下。

(1) 在编写上传代码之前首先判断上传文件是否存在，如果存在则将文件上传到服务器中，否则给出提示信息，给出提示信息如图 14.2 所示。实现的代码如下。

例程 1　代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
if(this.FileUpload1.PostedFile.FileName!="")
{
      //……将上传文件添加到服务器中的操作代码已省略
}
 else
{
      Response.Write("<script language='javascript'>alert('不可以为空');location=Default.aspx</script>");
}
```

图 14.2　上传文件为空，给出提示信息

(2) 对于批量上传文件，需要动态添加上传控件。在上传下载模块中，将代码封装到用户自定义的 InsertC 方法中，动态添加多个上传控件的运行效果如图 14.3 所示。动态添加上传控件的代码如下。

例程 2　代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
#region 该方法用于添加一个上传文件的控件
private void InsertC()
{
   //实例化ArrayList
   ArrayList AL = new ArrayList();
   this.Tab_UpDownFile.Rows.Clear(); //清除ID为表格里的所有行
   GetInfo();
   //表示 HtmlTable 控件中的 <tr> HTML 元素
   HtmlTableRow HTR = new HtmlTableRow();
   //表示 HtmlTableRow 对象中的 <td> 和 <th> HTML 元素
   HtmlTableCell HTC = new HtmlTableCell();
```

```
    //在单元格中添加一个FileUpload控件
    HTC.Controls.Add(new FileUpload());
    //在行中添加单元格
    HTR.Controls.Add(HTC);
    //在表中添加行
    Tab_UpDownFile.Rows.Add(HTR);
    SFUPC();
}
    #endregion
```

图 14.3　添加控件的运行效果图

（3）将当前页面的上传文件控件集保存到 Session 中，实现的代码如下。

例程 3　代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
    #region  该方法用于将当前页面上传文件控件集保存到Session中
    private void SFUPC()
    {
       ArrayList AL = new ArrayList();//动态增加数组
       foreach (Control C in Tab_UpDownFile.Controls)
       {
          //在表格中查找出FileUpload控件，并添加到ArrayList中
          if (C.GetType().ToString() == "System.Web.UI.HtmlControls.HtmlTableRow")
          {
             HtmlTableCell HTC = (HtmlTableCell)C.Controls[0];
             foreach (Control FUC in HTC.Controls)
             {
                if (FUC.GetType().ToString() == "System.Web.UI.WebControls.FileUpload")
                {
                   FileUpload FU = (FileUpload)FUC;
                   //添加FileUpload控件
                   AL.Add(FU);
                }
             }
          }
       }
       //把ArrayList添加到Session中
       Session.Add("FilesControls", AL);
    }
    #endregion
```

（4）将保存在 Session 中的上传文件控件集添加到表格中，实现的代码如下。

例程 4　代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
private void GetInfo()
{
    ArrayList AL = new ArrayList();
    if (Session["FilesControls"] != null)
    {
        AL = (ArrayList)Session["FilesControls"];
        for (int i = 0; i < AL.Count; i++)
        {
            HtmlTableRow HTR = new HtmlTableRow();
                HtmlTableCell HTC = new HtmlTableCell();
               HTC.Controls.Add((System.Web.UI.WebControls.FileUpload)AL[i]);
```

```
                HTR.Controls.Add(HTC);
                Tab_UpDownFile.Rows.Add(HTR);
            }
        }
    }
    #endregion
```

（5）最后获取文件的路径，并且将文件上传到指定的目录中，显示“上传成功”字样的提示信息。上传成功后运行效果如图 14.4 所示。实现的代码如下。

例程 5 代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
#region 该方法用于执行文件上传操作
private void UpFile()
{
    //获取文件夹路径
    string FilePath = Server.MapPath("./") + "File";
    // 获取客户端上载文件的集合
    HttpFileCollection HFC = Request.Files;
    for (int i = 0; i < HFC.Count; i++)
    {
        //访问指定的文件
        HttpPostedFile UserHPF = HFC[i];
        try
        {
            //判断文件是否为空
            if (UserHPF.ContentLength > 0)
            {
                //将上传的文件存储在指定的目录下
                UserHPF.SaveAs(FilePath + "\\" + System.IO.Path.GetFileName(UserHPF.FileName));
            }
        }
        catch
        {
            LblMessage.Text = "上传失败！";
        }
    }
    if (Session["FilesControls"] != null)
    {
        Session.Remove("FilesControls");
    }
    LblMessage.Text = "上传成功！";
}
#endregion
```

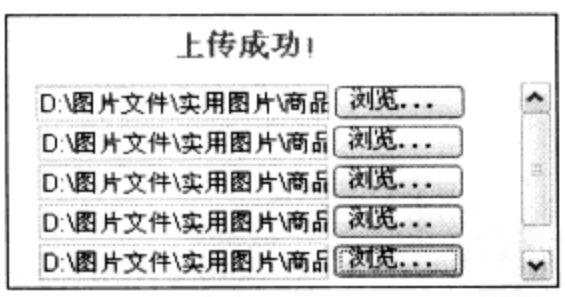


图 14.4 文件上传成功的运行效果图

14.2.2 获得指定路径中的所有文件名

下载文件前需要指定文件名，使用 Directory 对象的 GetFiles 方法来获得指定路径中的所有文件名，此方法的语法如下：

```
public static string[] GetFiles (
    string path
)
```

参数 path：用于检索文件的路径。

返回值：指定目录中文件名的 String 数组。文件名包含完整路径。

14.2.3 获取路径中的文件名

通过 GetFiles 方法获取的文件名中包含路径，这就需要使用 Path 对象的 GetFileName 方法将路径中的文件名取出。语法如下：

```
public static string GetFileName (
    string path
)
```

参数 path：用于获取文件名和扩展名。

返回值：返回 1 个 String，由 path 中最后的目录字符后的字符组成。

在上传下载模块中使用 GetFiles 方法获取文件名中包含的路径，实现的代码如下：

```
string[] name = Directory.GetFiles(Server.MapPath("File"));
```

14.2.4 设置 HTTP 标头的名称和值实现文件下载

文件的下载主要是通过 Response 对象的 AddHeader 方法来设置 HTTP 标头的名称和值。最后将当前所有缓冲的输出发送到客户端。在上传下载模块中实现的代码如下：

```
//将文件保存到本机上
Response.Clear();
Response.AddHeader("Content-Disposition", "attachment; filename=" + Server.UrlEncode(fi.Name));
Response.AddHeader("Content-Length", fi.Length.ToString());
Response.ContentType = "application/octet-stream";
Response.Filter.Close();
Response.WriteFile(fi.FullName);
Response.End();
```

14.2.5 生成图片的缩略图技术

在上传图片时，可以将图片先进行缩放，然后将其保存到服务器中，生成图片的缩略图和原图的对比图如图 14.5 所示。其主要代码如下：

```
string filePath = FileUpload1.PostedFile.FileName;//获取上传文件的路径
//生成缩略图
System.Drawing.Image image, newimage;
image = System.Drawing.Image.FromFile(filePath);
System.Drawing.Image.GetThumbnailImageAbort callb=null;
newimage = image.GetThumbnailImage(67, 90, callb, new System.IntPtr());
//把缩略图保存到指定的虚拟路径
newimage.Save(serverpath);
//释放image对象占用的资源
newimage.Dispose();
image.Dispose();
```

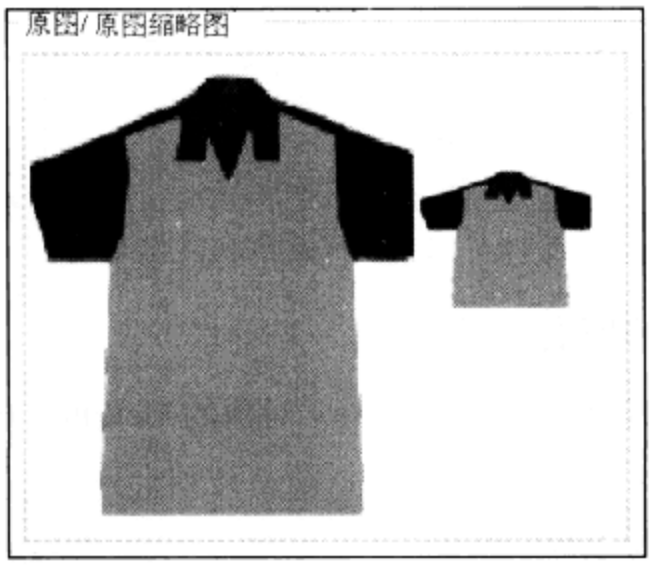


图 14.5 生成原图和缩略图的对比图

14.2.6 ASP.NET 实现断点续传

在文件下载的时候，使用断点续传可以将上次未下载完成的文件继续下载。该功能在开发下载软件时非常重要。本模块将实现比较简单的断点续传功能。运行本模块，单击文件名将实现文件的下载操作，当断开文件下载后。在重新下载文件时将会以上一次未完成下载的位置开始下载。运行效果如图 14.6 所示。

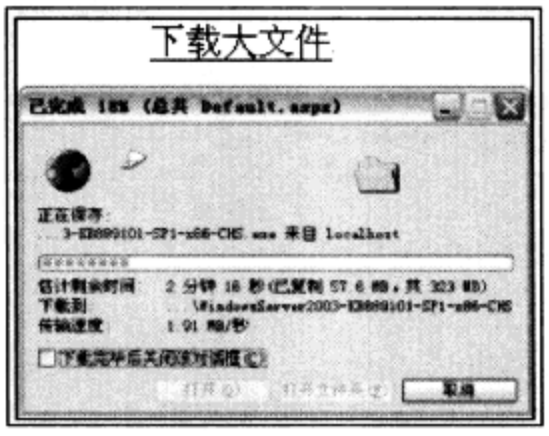


图 14.6 断点续传

断点续传实际上就是在上一次下载时断开的位置开始继续下载。本实例主要通过 Response 类中的 AddHeader 方法将一个 HTTP 头添加到输出流中。在 HTTP 头中，是由头信息和体信息组成。两者之间以一行空行为分隔。本实例利用在头中加入 Range 段，来表示客户端希望从何处继续下载，来实现续传的功能。代码如下：

```
Response.AddHeader("Content-Range", "bytes " + sum.ToString() + "-" + ((long)(fileSize)).ToString() + "/" + fileSize.ToString());
```

Asp.net 实现断点继传的步骤如下。

（1）新建 1 个主页，将其命名为 DFile.aspx。

（2）在该页中添加 1 个 LinkButton 按钮，该按钮用于执行存储过程。

（3）在 LinkButton 按钮的 Click 事件中，实现断点续传功能。代码如下。

例程 6 代码位置：光盘\mr\14\UpDownFile\DFile.aspx.aspx.cs

```
protected void LinkButton1_Click(object sender, EventArgs e)
    {
        // 创建一比特数组
        byte[] buffer = new Byte[1024];
        // 指定要下载文件的路径
        string filePath = @"E:\asp.net.rar";
        // 或取文件名包括扩展名
        string fileName = Path.GetFileName(filePath);
        Stream fileStream=null;
        try
        {
            // 打开文件
            fileStream = new FileStream(filePath, FileMode.Open, FileAccess.Read, FileShare.Read);
            Response.Clear();
            // 获取文件的大小
            long fileSize = fileStream.Length;
            long sum = 0;
            if (Request.Headers["Range"] != null)
            {
                // 表示返回客户端的 HTTP 输出状态的整数，默认值为 200
                Response.StatusCode = 206;
                sum = long.Parse(Request.Headers["Range"].Replace("bytes=", "").Replace("-", ""));
            }
            if (sum != 0)
```

```
                {
                    Response.AddHeader("Content-Range", "bytes " + sum.ToString() + "-" + ((long)(fileSize)).
ToString() + "/" + fileSize.ToString());
                }
                // 获取部分HTTP头信息
                Response.AddHeader("Content-Length", ((long)(fileSize - sum)).ToString());
                Response.ContentType = "application/octet-stream";
                //获取文件来源
                Response.AddHeader("Content-Disposition", "attachment; filename=" + HttpUtility.UrlEncode(Request.
ContentEncoding.GetBytes(fileName)));
                fileStream.Position = sum;      //设置当前流位置
                fileSize = fileSize - sum;
                // 当文件大小大于0是进入循环
                while (fileSize > 0)
                {
                    // 判断客户端是否仍连接在服务器
                    if (Response.IsClientConnected)
                    {
                        // 获取缓冲区中的总字节数
                        int length = fileStream.Read(buffer, 0, 1024);
                        // 写入数据
                        Response.OutputStream.Write(buffer, 0, length);
                        // 将缓冲区的输出发送到客户端
                        Response.Flush();
                        buffer = new Byte[1024];
                        fileSize = fileSize - length;
                    }
                    else
                    {
                        //当用户断开后退出循环
                        fileSize = -1;
                    }
                }
            }
            catch (Exception ex)
            {
                Response.Write("Error : " + ex.Message);
            }
            finally
            {
                if (fileStream != null)
                {
                    //关闭文件
                    fileStream.Close();
                }
                Response.End();
            }
        }
```

使用 ASP.NET 实现断点继传需要引用 System.IO 命名空间。

14.3 上传与下载模块实现过程

14.3.1 文件单个和批量上传

文件上传页面（FileUp.aspx）可将选定的文件上传到指定的服务器文件夹中，同时将所上

传文件的文件信息保存到数据库中。该页运行结果如图 14.7 所示。

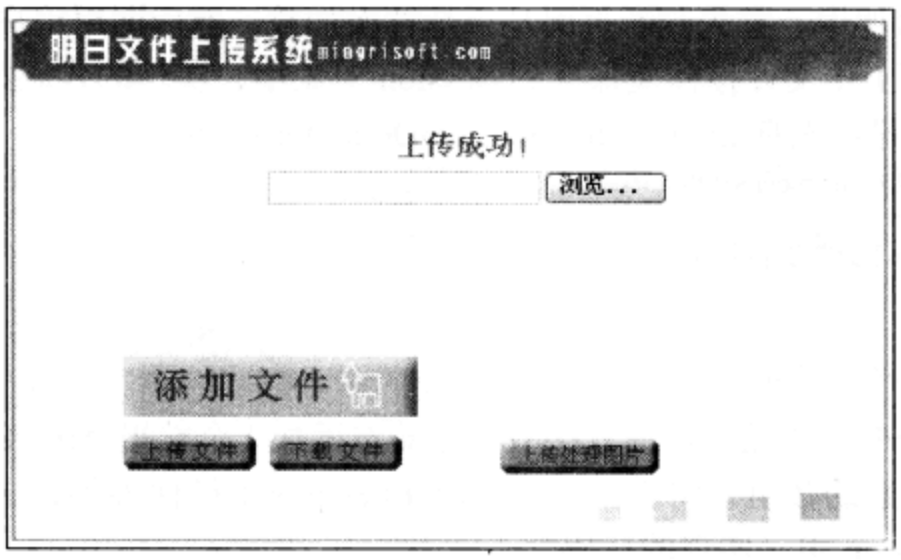


图 14.7　文件上传页面的运行效果图

1. 前台页面设计

（1）创建 1 个 Web 窗体，默认名为 Default.aspx。作为文件单个和批量上传页。页面 Default.aspx 的设计界面如图 14.8 所示。

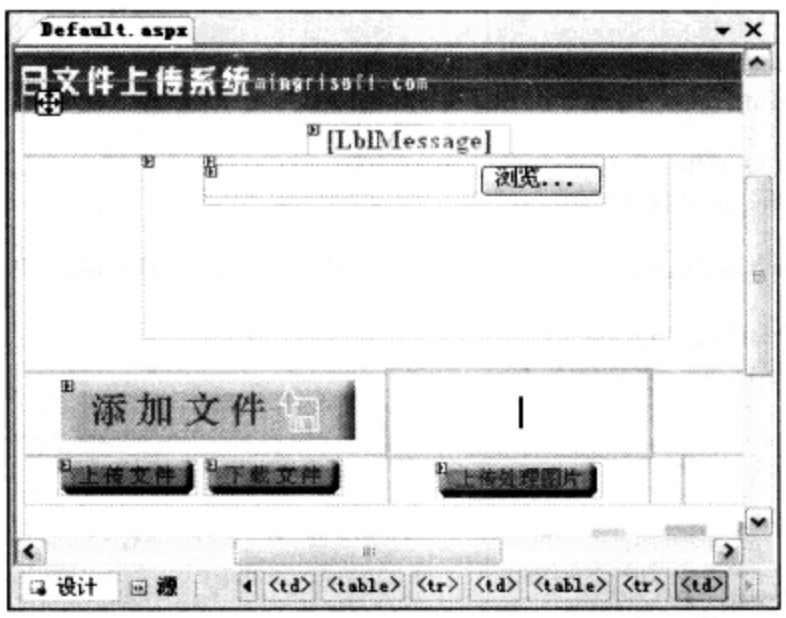


图 14.8　Default.aspx 的设计界面

（2）文件的上传页面主要使用 Table 表格、用户控件、TextBox 控件、RadioButton 控件、DropDownList 控件和 Button 控件设计完成，其主要控件如表 14.2 所示。

表 14.2　　文件单个和批量上传页控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
Panel	控件的名称为 Pan_UpFile	将控件的 ScrollBars 属性设置为 Auto	页面布局
FileUpload	控件的名称为 FileUpload1	无	用于浏览要上传的文件
A Label	控件的名称为 LblMessage	无	显示信息
ImageButton	控件的名称分别为 ImgUpFile、ImgBtnDownFile 和 ImgBtnUppic	分别将这 3 个控件的 ImageUrl 属性分别设置为~/images/sc2.gif、~/images/xz1.gif 和 ImageUrl="~/images/sc8.gif"	分别用于执行“上传文件”操作、将页面跳转到下载页面中和将页面跳转到处理图片页面中

2. 后台代码实现

在 Page_Load 事件中，首先判断页面是否是首次加载，然后调用用户自定义的 SFUPC 方法，实现将当前页面上传文件控件集保存到 Session 对象中，实现的代码如下。

例程 7 代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)//首次执行页面
    {
        SFUPC();
    }
}
```

用户自定义的 SFUPC 方法，在该方法中首先动态添加 1 个数组 ArrayList，然后应用 foreach 循环语句遍历动态创建的数组，将表格中查找出 FileUpload 控件添加到 ArrayList 中，最后把 ArrayList 添加到 Session 中。实现的代码如下。

例程 8 代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
#region  该方法用于将当前页面上传文件控件集保存到Session中
private void SFUPC()
{
   ArrayList AL = new ArrayList();//动态增加数组
   foreach (Control C in Tab_UpDownFile.Controls)
   {
     //在表格中查找出FileUpload控件添加到ArrayList中
     if (C.GetType().ToString() = = "System.Web.UI.HtmlControls.HtmlTableRow")
     {
        HtmlTableCell HTC = (HtmlTableCell)C.Controls[0];
        foreach (Control FUC in HTC.Controls)
        {
          if (FUC.GetType().ToString() == "System.Web.UI.WebControls.FileUpload")
          {
            FileUpload FU = (FileUpload)FUC;
            //添加FileUpload控件
            AL.Add(FU);
          }
        }
     }
   }
  //把ArrayList添加到Session中
 Session.Add("FilesControls", AL);
}
#endregion
```

在用户自定义的 Insert 方法中，该方法实现的是添加一个上传文件的控件的功能，实现的代码如下。

例程 9 代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
#region 该方法用于添加1个上传文件的控件
private void InsertC()
{
  //实例化ArrayList
  ArrayList AL = new ArrayList();
  this.Tab_UpDownFile.Rows.Clear(); //清除ID为F表格里的所有行
  GetInfo();
  //表示 HtmlTable 控件中的 <tr> HTML 元素
  HtmlTableRow HTR = new HtmlTableRow();
  //表示 HtmlTableRow 对象中的 <td> 和 <th> HTML 元素
  HtmlTableCell HTC = new HtmlTableCell();
  //在单元格中添加一个FileUpload控件
  HTC.Controls.Add(new FileUpload());
  //在行中添加单元格
  HTR.Controls.Add(HTC);
```

```
        //在表中添加行
        Tab_UpDownFile.Rows.Add(HTR);
        SFUPC();
    }
#endregion
```

在用户自定义的 GetInfo 方法中，该方法用于将保存在 Session 中的上传文件控件添加到表格中，实现的代码如下。

例程 10 代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
#region 该方法用于将保存在Session中的上传文件控件集添加到表格中
private void GetInfo()
{
   ArrayList AL = new ArrayList();
   if (Session["FilesControls"] != null)
   {
      AL = (ArrayList)Session["FilesControls"];
      for (int i = 0; i < AL.Count; i++)
      {
         HtmlTableRow HTR = new HtmlTableRow();
         HtmlTableCell HTC = new HtmlTableCell();
         HTC.Controls.Add((System.Web.UI.WebControls.FileUpload)AL[i]);
         HTR.Controls.Add(HTC);
         Tab_UpDownFile.Rows.Add(HTR);
       }
     }
   }
#endregion
```

在用户自定义的 UpFile 方法中，该方法用于执行文件的上传操作，实现的代码如下。

例程 11 代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
#region 该方法用于执行文件上传操作
private void UpFile()
{
   //获取文件夹路径
   string FilePath = Server.MapPath("./") + "File";
   // 获取客户端上载文件的集合
   HttpFileCollection HFC = Request.Files;
   for (int i = 0; i < HFC.Count; i++)
   {
      //访问指定的文件
      HttpPostedFile UserHPF = HFC[i];
     try
     {
        //判断文件是否为空
        if (UserHPF.ContentLength > 0)
            {
              //将上传的文件存储在指定目录下
              UserHPF.SaveAs(FilePath + "\\" + System.IO.Path.GetFileName(UserHPF.FileName));
            }
       }
       catch
       {
            LblMessage.Text = "上传失败！";
       }
     }
    if (Session["FilesControls"] != null)
    {
      Session.Remove("FilesControls");
    }
     LblMessage.Text = "上传成功！";
}
#endregion
```

双击页面中的“添加文件”按钮，触发其“ImgBtnAdd_Click”事件，在该事件中实现的是调用 InsertC 方法，实现添加 FileUpLoad 控件的功能，实现的代码如下。

例程 12　代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
#region 调用InsertC方法，实现添加FileUpLoad控件的功能
protected void ImgBtnAdd_Click(object sender, ImageClickEventArgs e)
{
    InsertC();//执行添加控件方法
    LblMessage.Text = "";
}
#endregion
```

双击页面上的“文件上传”按钮，触发其“ImgUpFile_Click”事件，在该事件中实现文件上传的功能，实现的代码如下。

例程 13　代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
#region 实现文件上传的功能
protected void ImgUpFile_Click(object sender, ImageClickEventArgs e)
{
  if(this.FileUpload1.PostedFile.FileName!="")
  //if(FileUpload1.HasFile)//判断是否选择上传文件
  {
      UpFile();//执行上传文件
      SFUPC();
  }
  else
 {
    Response.Write("<script language='javascript'>alert('不可以为空');location=Default.aspx</script>");
     }
}
    #endregion
```

双击页面上的“文件下载”按钮，触发其“ImgBtnDownFile_Click”事件，在该事件中实现的是将页面跳转到 downFile.aspx 页面中，实现的代码如下。

例程 14　代码位置：光盘\mr\14\UpDownFile\Default.aspx.cs

```
protected void ImgBtnDownFile_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("downFile.aspx");
}
```

14.3.2　对指定的文件进行下载

文件的下载先通过 ListBox 控件显示文件名，再选择要下载的文件名，单击“下载文件”按钮，将文件保存到本机，即完成文件的下载功能。该页面运行的效果如图 14.9 所示。

图 14.9　文件下载运行效果图

1. 前台页面设计

(1) 创建 1 个 Web 窗体，默认名为 DownFile.aspx。作为下载文件页面。页面 downFile.aspx 的设计界面如图 14.10 所示。

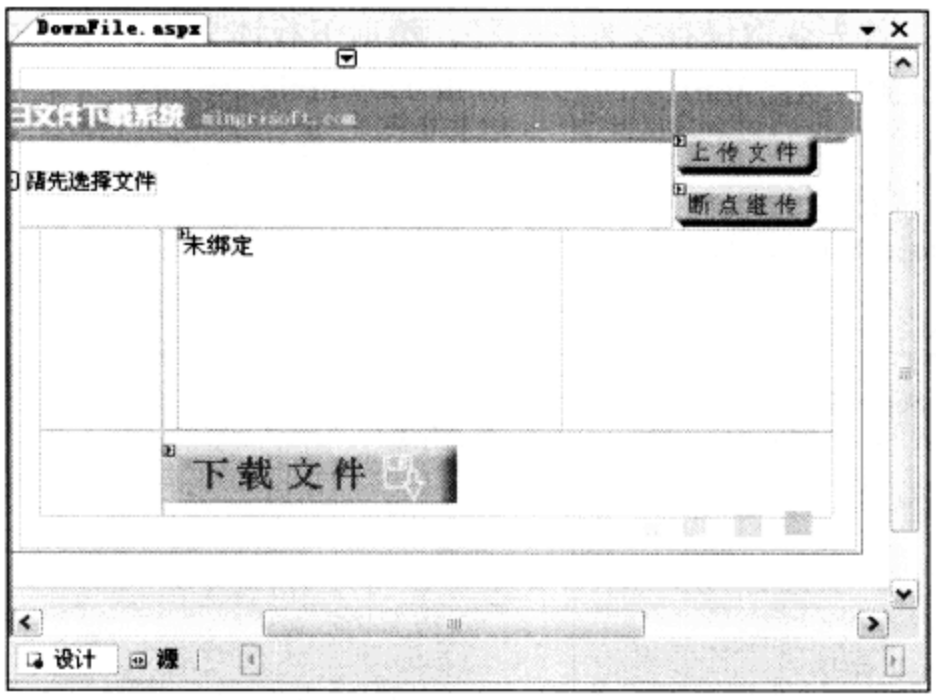


图 14.10　DownFile.aspx 的设计界面

(2) 文件的上传下载的页面主要使用 Table 表格、ImageButton 控件和 ListBox 控件设计界面。

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 15　代码位置：光盘\mr\14\UpDownFile\DownFile.aspx.cs

```
using System.IO;//引入命名空间
```

在 Page_Load 事件中，实现的是调用用户自定义的 addListBox 方法，实现的代码如下。

例程 16　代码位置：光盘\mr\14\UpDownFile\DownFile.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)//首次加载
    {
        addListBox();    //调用用户自定义的addListBox方法
    }
}
```

在用户自定义的 addListBox 方法中，实现的是将文件的名称绑定到 ListBox 控件当中，实现的代码如下。

例程 17　代码位置：光盘\mr\14\UpDownFile\DownFile.aspx.cs

```
protected void addListBox()
{
  //将指定文件夹中的文件保存到字符串数组中
  string[] name = Directory.GetFiles(Server.MapPath("File"));
  foreach (string s in name)
  {
      //将文件名添加到ListBox中
      LisBoxFile.Items.Add(Path.GetFileName(s));
  }
}
```

在 ListBox1_SelectedIndexChanged 事件中，实现在 ListBox 控件中选择的项的值赋值给 Session["txt"]变量中，实现的代码如下。

例程 18 代码位置：光盘\mr\14\UpDownFile\DownFile.aspx.cs

```
protected void ListBox1_SelectedIndexChanged(object sender, EventArgs e)
{
   Session["txt"] = LisBoxFile.SelectedValue.ToString();//从ListBox控件中选择的项赋值给Session["txt"]中
}
```

在 dFile 方法中，首先获取保存文件的路径，然后下载该文件，实现的代码如下。

例程 19 代码位置：光盘\mr\14\UpDownFile\DownFile.aspx.cs

```
protected void dFile()
{
   //判断是否选择文件名
   if (LisBoxFile.SelectedValue != "")
   {
       if (Session["txt"] != "")
     {    //获取文件路径
        string path = Server.MapPath("File/") + Session["txt"].ToString();
        //初始化 FileInfo 类的实例，它作为文件路径的包装
        FileInfo fi = new FileInfo(path);
        //判断文件是否存在
        if (fi.Exists)
        {
            //将文件保存到本机上
            Response.Clear();
            Response.AddHeader("Content-Disposition", "attachment; filename=" + Server.UrlEncode(fi.Name));
            Response.AddHeader("Content-Length", fi.Length.ToString());
            Response.ContentType = "application/octet-stream";
            Response.Filter.Close();
            Response.WriteFile(fi.FullName);
            Response.End();
        }
      }
    }
    else
   {
      Page.RegisterStartupScript("sb", "<script>alert('请您先选择文件名')</script>");
   }
}
```

双击“下载文件”按钮，触发其“ImgBtnDownFile_Click”事件，在该事件中实现的是调用用户自定义的 dFile 方法，将指定的文件下载到本地磁盘当中，实现的代码如下。

例程 20 代码位置：光盘\mr\14\UpDownFile\DownFile.aspx.cs

```
protected void ImgBtnDownFile_Click(object sender, ImageClickEventArgs e)
{
    dFile();//调用用户自定义的dFile方法
}
```

双击“断点继传”按钮，触发其 ImgBtnDd_Click 事件，在该事件中实现的是将页面跳转到 Dfile.aspx 页面中，实现的代码如下。

例程 21 代码位置：光盘\mr\14\UpDownFile\DownFile.aspx.cs

```
protected void ImgBtnDd_Click(object sender, ImageClickEventArgs e)
{
    Response.Redirect("DFile.aspx");
}
```

14.3.3 将上传图片生成缩略图并且加上文字

将上传图片生成缩略图并且加上文字是比较常用的功能，程序中提供了向图片添加文字的功能，通过此功能将上传上来的图片生成缩略图，并在原图片上绘制“明”字样的文字。

运行效果如图 14.11 所示。

图 14.11　将上传图片生成缩略图并且加上文字

1. 前台页面设计

(1) 创建 1 个 Web 窗体，默认名为 pic.aspx。作为将上传图片生成缩略图并且加上文字。页面 pic.aspx 的设计界面如图 14.12 所示。

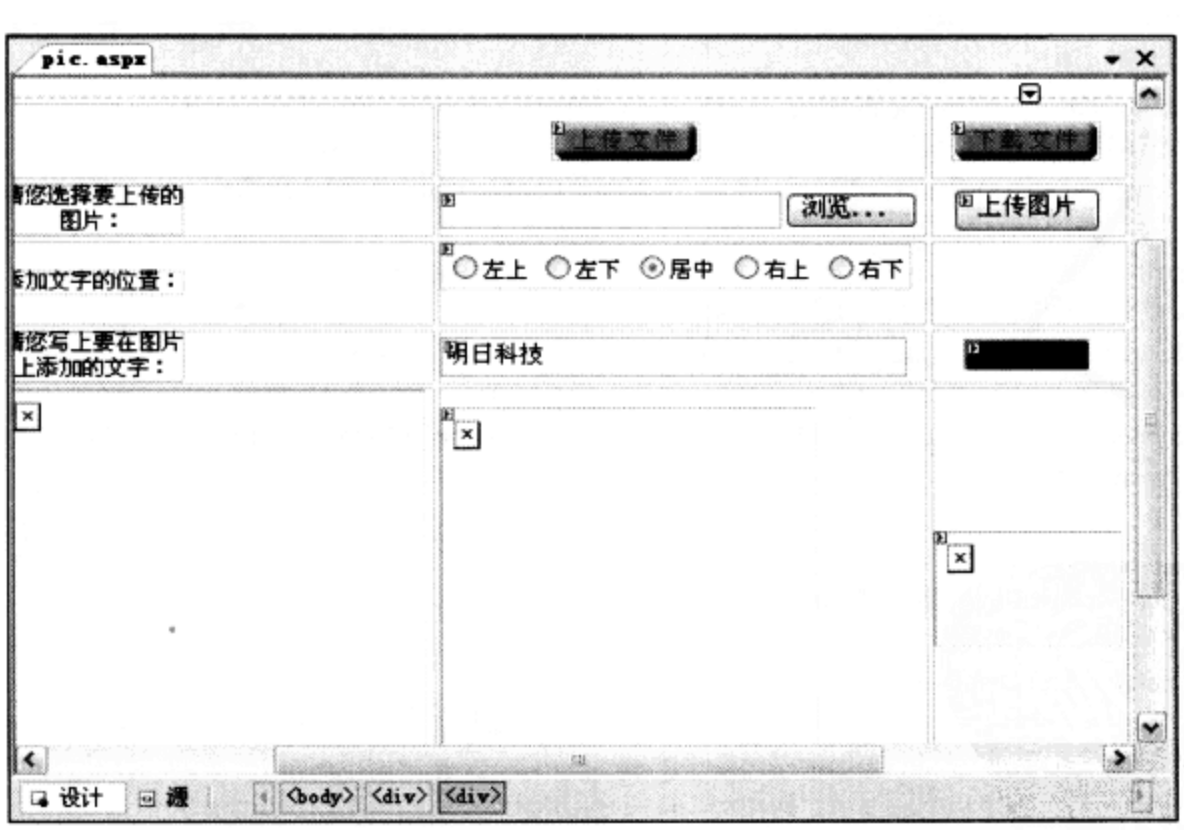


图 14.12　pic.aspx 的设计界面

(2) 该页面主要使用 Table 表格、Button 控件、TextBox 控件、Image 控件、RadioButtonList 控件和 ImageButton 控件设计完成。

2. 后台代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 22 代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
using System.IO;///声明一个命名空间
```

在 Page_Load 页面加载事件外，声明一个字符串类型的变量，实现的代码如下。

例程 23 代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
string filename = "";///定义一个字符串类型的变量
```

在 Page_Load 页面加载事件中，在该事件中实现的是将 Image 控件隐藏的功能，实现的代码如下。

例程 24 代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    this.Image1.Visible = false;
    this.Image2.Visible = false;
    this.Image3.Visible = false;
}
```

在用户自定义的 toImage 方法中，实现的是将上传的图片生成缩略图并且可以将文字添加到图片指定的位置中，实现的代码如下。

例程 25 代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
/**/
    /// <summary>
    /// 生成缩略图并将文字添加到图片指定的位置中
    /// </summary>
    /// <param name="myStream">取到的流文件对象</param>
    /// <param name="uploadPath">要保存的路径</param>
    /// <param name="picName">上传的图片原文件名</param>
    private void toImage(Stream myStream, string uploadPath, string picName)
    {
        //==========================生成缩略图==========================
        //取得后缀名
        string suffix = picName.Substring(picName.LastIndexOf("."));
        //缩略图的保存路径
        string fileXltPath = uploadPath + "\\" + picName.Replace(suffix, "x" + suffix);
        //写字图的保存路径
        string fileXztPath = uploadPath + "\\" + picName.Replace(suffix, "w" + suffix);
        //创建一个图像对象取得上传图片对象
        System.Drawing.Image myImage = System.Drawing.Image.FromStream(myStream, false);
        //对绘制前的图片产生一个缩略图(原图片一半大小)
        System.Drawing.Image thumbImage = myImage.GetThumbnailImage(myImage.Size.Width /2,myImage.Size.
Height / 2, null, System.IntPtr.Zero);
        //}
         //保存缩略图
        thumbImage.Save(fileXltPath, this.getImageFormat(suffix));
        //关闭缩略图对象
        thumbImage.Dispose();
        //==========================绘制上传图片上的文字==========================
        //创建绘制对象
        System.Drawing.Graphics g = System.Drawing.Graphics.FromImage(myImage);
        g.DrawImage(myImage, 0, 0, myImage.Size.Width, myImage.Size.Height);
        //选择字体及字体大小
        System.Drawing.Font f = new Font("隶书", 38);
        //定义字体颜色
        System.Drawing.Brush b = new SolidBrush(this.TextBox1.BackColor);//System.Drawing.Color.Cyan
        //开始绘制,根据上述两种设定,添加绘制的上左位置
        string wenzi = this.TxtName.Text;
        if (this.RadBtnlocation.SelectedValue=="左上")
```

```
        {

            g.DrawString(wenzi, f, b, 1, 1);
        }
        if (this.RadBtnlocation.SelectedValue=="居中")
        {
            g.DrawString(wenzi, f, b, myImage.Size.Width / 2, myImage.Size.Height / 2);
        }
        if (this.RadBtnlocation.SelectedValue == "左下")
        {
            g.DrawString(wenzi, f, b, 0, myImage.Size.Height*3/14.);
        }
        if (this.RadBtnlocation.SelectedValue == "右上")
        {
            g.DrawString(wenzi, f, b, myImage.Size.Width * 3 / 14., 1);
        }
        if (this.RadBtnlocation.SelectedValue == "右下")
        {
            g.DrawString(wenzi, f, b, myImage.Size.Width * 3 / 14., myImage.Size.Height * 3 / 14.);
        }
        //关闭绘制对象
        g.Dispose();
        //保存绘制后上传图片
        myImage.Save(fileXztPath, myImage.RawFormat);
        //关闭图片对象
        myImage.Dispose();
    }
```

在用户自定义的 getImageFormat 方法中，可实现根据图片的后缀名称，返回要保存图片的格式，并且返回的是 1 个 System.Drawing.Imaging.ImageForma 对象，实现的代码如下。

例程 26 代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
/// <summary>
    /// 根据图片的后缀名,返回要保存的图片格式
    /// </summary>
    /// <param name="suffix">带.号的后缀名</param>
    /// <returns>返回System.Drawing.Imaging.ImageForma对象</returns>
    private System.Drawing.Imaging.ImageFormat getImageFormat(string suffix)
    {
        System.Drawing.Imaging.ImageFormat myFormat;
        switch (suffix.ToLower())
        {
            case ".bmp":
                myFormat = System.Drawing.Imaging.ImageFormat.Bmp;
                break;
            case ".emf":
                myFormat = System.Drawing.Imaging.ImageFormat.Emf;
                break;
            case ".exif":
                myFormat = System.Drawing.Imaging.ImageFormat.Exif;
                break;
            case ".gif":
                myFormat = System.Drawing.Imaging.ImageFormat.Gif;
                break;
            case ".icon":
                myFormat = System.Drawing.Imaging.ImageFormat.Icon;
                break;
            case ".jpeg":
```

```
                case ".jpg":
                    myFormat = System.Drawing.Imaging.ImageFormat.Jpeg;
                    break;
                case ".png":
                    myFormat = System.Drawing.Imaging.ImageFormat.Png;
                    break;
                case ".tiff":
                    myFormat = System.Drawing.Imaging.ImageFormat.Tiff;
                    break;
                case ".wmf":
                    myFormat = System.Drawing.Imaging.ImageFormat.Wmf;
                    break;
                default:
                    myFormat = System.Drawing.Imaging.ImageFormat.MemoryBmp;
                    break;
            }
            return (myFormat);
        }
```

双击“上传图片”按钮，触发其“BtnUpDown_Click”事件，在该事件中，实现的是将图片的原图、图片上有文字的图和图片的缩略图上传到程序中指定的“file”文件夹下，实现的代码如下。

例程 27 代码位置：光盘\mr\14\UpDownFile\pic.aspx.cs

```
protected void BtnUpDown_Click(object sender, EventArgs e)
{
    if (FileUpload1.PostedFile.FileName != "")
    {
        //定义上传路径(在当前目录下的uploadfile文件下）
        string uploadpath = this.Server.MapPath("file");
        //取得文件名
        string tmpfilename = FileUpload1.PostedFile.FileName;
        //文件名
        filename = tmpfilename.Substring(tmpfilename.LastIndexOf("\\") + 1);
        //原文件的保存路径
        string fileSavePath = uploadpath + "\\" + filename;
        //保存原图片
        FileUpload1.SaveAs(fileSavePath);
        //调用生成缩略图程序,生成缩略图及生成写字的图片
        this.toImage(FileUpload1.PostedFile.InputStream, uploadpath, filename);
         //求取后缀名
        string suffix = filename.Substring(filename.LastIndexOf("."));
        //显示图片
        //分别为原图片/写字的图片(多一个w)/缩略图(多一个x)
        this.Image1.ImageUrl = "~/file/" + filename;
        this.Image2.ImageUrl = "~/file/" + filename.Replace(suffix, "w" + suffix);
        this.Image3.ImageUrl = "~/file/" + filename.Replace(suffix, "x" + suffix);
        //显示图像控件
        this.Image1.Visible = true;
        this.Image2.Visible = true;
        this.Image3.Visible = true;
    }
  }
}
```


14.4 程序调试与错误处理

在对程序进行调试时，可能出现如图 14.13 所示的错误。

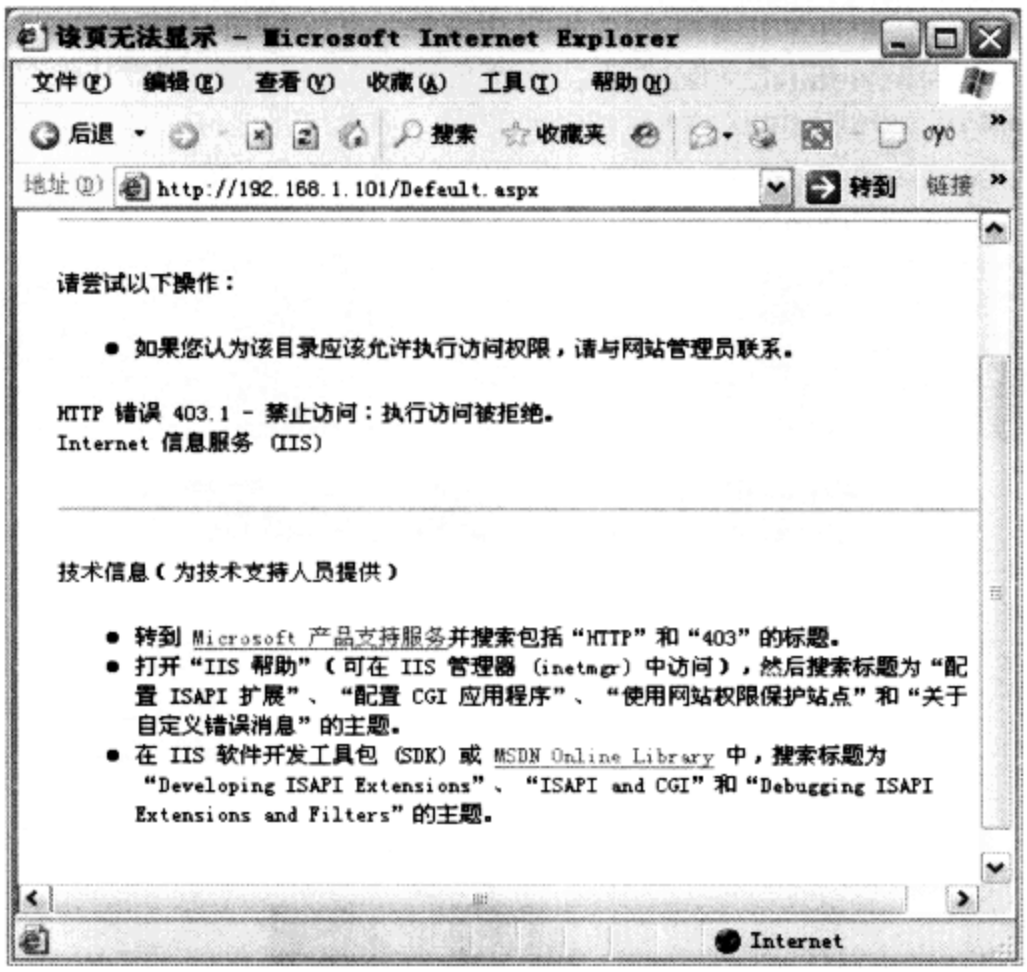


图 14.13　执行权限错误出错页面

出现图 14.13 中错误，是因为在配置 IIS 过程中，执行权限设置错误。

此时，可选择“开始”→“程序”→“管理工具”→“Internet 信息服务（IIS）管理器”，展开菜单右键单击“默认网站”选项，选择“属性”选项，单击“主目录”选项卡，如图 14.14 所示。在“执行权限”下拉框中选择“脚本和可执行文件”选项，单击“确定”按钮，设置完成。

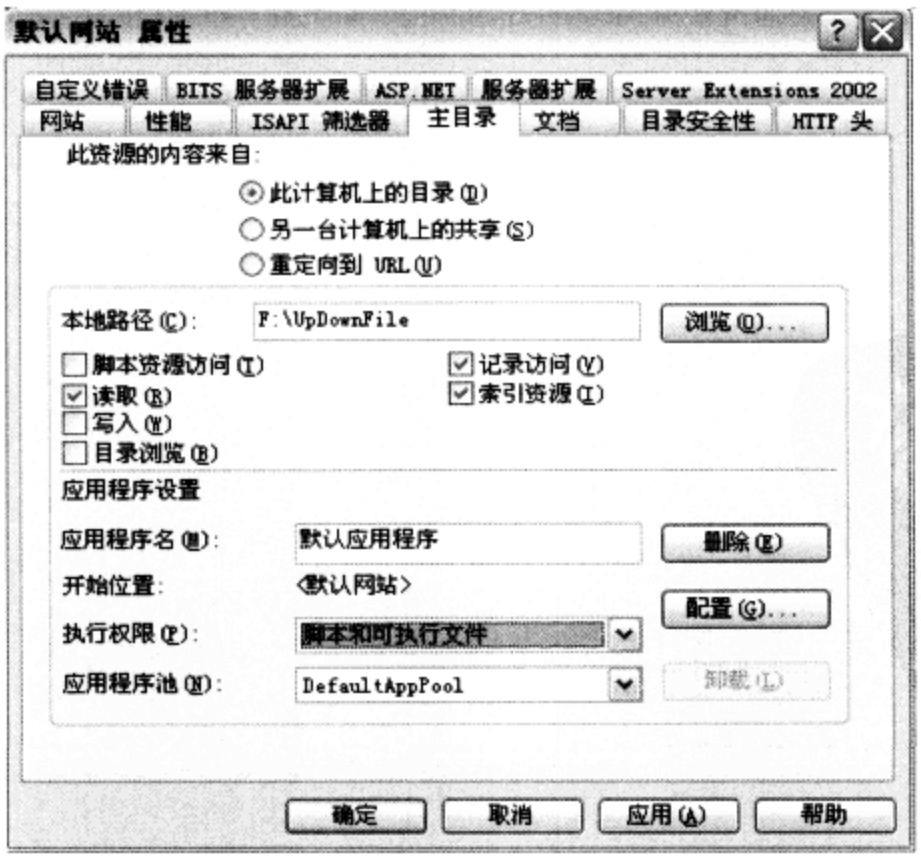


图 14.14　IIS 设置

设置完毕后上传下载模块便能运行，其效果如图 14.15 所示。

图 14.15　运行效果图

第15章 图片资源管理模块

实例位置：光盘\mr\15\

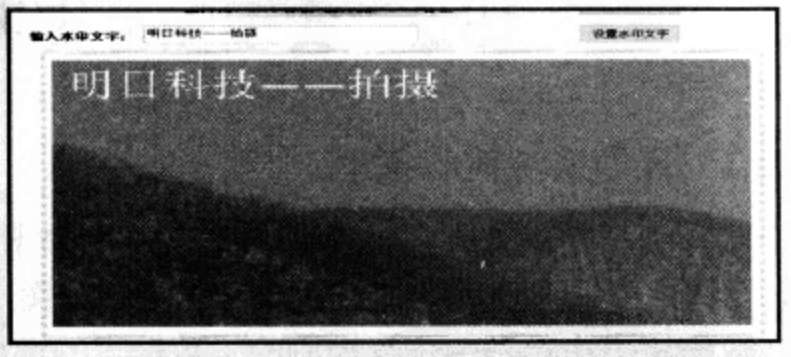
从目前的互联网信息来看，几乎所有的网站都包含图片。中小型网站中的图片数量相对而言将会少一些，如果是一个大中型网站，那么他的图片数量将是我们无法估计的。这样看来，应该根据实际情况在网站中开发一个图片资源管理模块，更好的管理整个网站的图片资源。那么通过本章图片资源管理模块的学习，读者能够学到以下内容。

递归遍历显示树状菜单

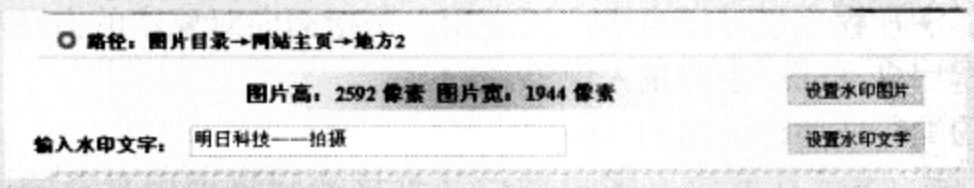
多文件上传

设置水印图片

设置水印文字



获取图片的尺寸和显示图片导航路径



15.1 概　　述

大多数网站或 Web 管理系统中，例如产品销售网、电子相册网、企业门户网等，都离不开图片资源，产品销售网需要在网站的页面中显示产品图片；电子相册网站需要显示照片；企业门户网显示企业办公环境照片等。由此看来，图片资源管理是网站中不可缺少的一个功能模块，如果图片资源少，则方便管理。如果电子商务网站中包含成百上千种产品，每一个产品都需要显示图片，那么不进行有效管理的话，网站或 Web 系统将会瘫痪。图片资源管理则是网站中不可缺少的功能模块。

图片资源管理模块数据库采用 SQL Server 2000 作为后台数据库，数据库命名为 db_Picture，该数据库中只包含一张 Images 数据表，用来存储上传的图片类别、存储路径等信息。设计 Images 数据表相对于本模块比较重要，因为表中的数据在前台是通过 TreeView 控件显示的，所以需要支持递归遍历。在此只介绍两个数据表中的列名，其中列名 isDir 用于存储文件或目录的标识（1 代表目录，0 代表图片文件），列名 ParentID 保存文件或目录属于哪个目录，其他字段列说明如图 15.1 所示。

Images *

	列名	数据类型	长度	标识种子	标识递增量	默认值	描述
🔑	ImgID	int	4	1	1		ID流水号
	isDir	bit	1				文件或目录标识
	ParentID	int	4				上级目录ID号
	ImgName	nvarchar	50				图片名称
	ImgUrl	nvarchar	100				图片保存路径
▶	CreateDate	datetime	8			(getdate())	上传时间

图 15.1　Images 数据表结构

15.2 实现图片资源管理的关键技术

15.2.1 递归实现树状菜单

图片资源管理模块主要通过 TreeView 控件导航管理图片，在 TreeView 导航控件中涉及目录与目录、目录与文件等之间的父子关系，如图 15.2 所示。这里值得注意的是，目录和文件都是在数据库表中通过标识来控制的，数据表的设计必须支持递归操作。下面介绍递归算法及其在本程序中的应用。

递归作为一种算法在程序设计语言中广泛应用，是指面向对象中的方法和子程序在运行过程序中直接或间接调用自身而产生的重入现象。

程序调用自身的编程技巧称为递归（recursion）。

一个面向对象中的方法在其定义或说明中又直接或间接调用自身的一种方法，它通常把一个大型复杂的问题层层转化为一个与原问题相似的规模较小的问题来求解，递归策略只需少量的程序就可描述出解题过程所需要的多次重复计算，大大地减少了程序的代码量。递归的能力在于用有限的语句来定义对象的无限集合。用递归思想写出的程序往往十分简洁易懂。

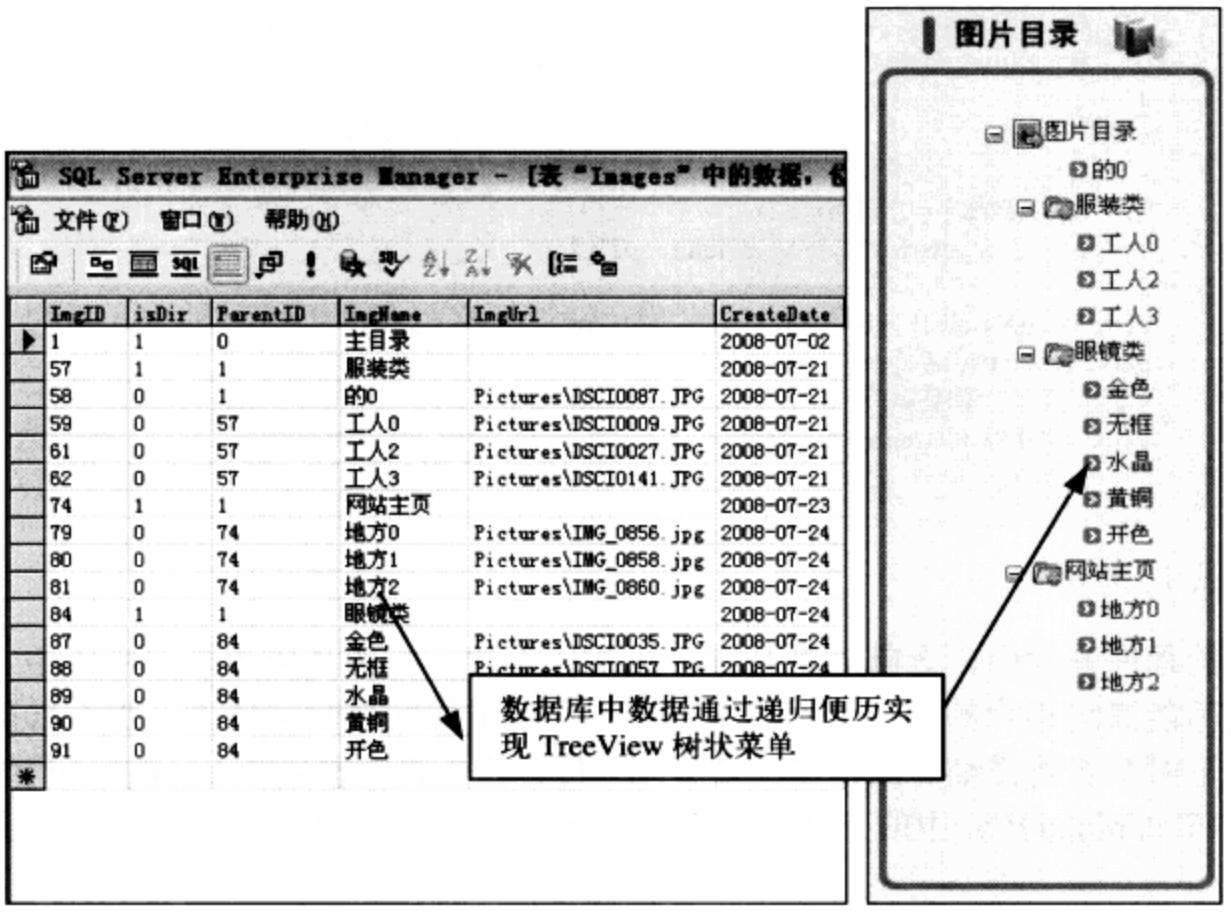


图 15.2　递归实现树状目录

一般来说，递归需要有边界条件、递归前进段和递归返回段。当边界条件不满足时，递归前进；当边界条件满足时，递归返回。

实现递归的注意事项包括以下内容。

- 递归就是在过程或函数里调用自身。
- 在使用递增归策略时，必须有一个明确的递归结束条件，称为递归出口

递归算法一般用于解决 3 类问题。

- 数据的定义是按递归定义的。
- 问题解法按递归算法实现。
- 数据的结构形式是按递归定义的。

递归算法解题的运行效率较低。在递归调用的过程中系统为每一层的返回点、局部量等开辟了栈来存储，而递归次数过多很容易造成栈溢出等问题。

下面通过递归程序代码实现递归算法。

本程序中 CreateChildNode 方法为递归方法，该方法主要用来创建 TreeView 控件的节点及子节点。

例程 1　代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
private void CreateChildNode(TreeNode parentNode, DataTable dataTable)
{
    DataRow[] rowList = dataTable.Select("ParentID='" + parentNode.Value + "'");
    foreach (DataRow row in rowList)
    {   //创建新节点
        TreeNode node = new TreeNode();
        //设置节点的属性
        node.Text = row["ImgName"].ToString();
        node.Value = row["ImgID"].ToString();
        if (row["isDir"].ToString() == "True")
        {
```

```
                node.Expanded = true;
                node.Target = "_self";
                node.ImageUrl = "~/Image/Dir.gif";
            }
            else
            {
                node.Target = "_blank";
                node.ImageUrl = "~/Image/Picture.gif";
            }
            node.ImageToolTip = row["ImgUrl"].ToString();
            parentNode.ChildNodes.Add(node);
            //递归调用，创建其他节点
            CreateChildNode(node, dataTable);
        }
    }
```

15.2.2 多文件上传

在过去的程序开发中，大部分都是进行单文件上传。在图片资源管理模块中，单张图片文件上传已不能满足用户的需求，而且单张图片文件上传既费时又费力，那么如何实现多文件上传呢？下面笔者将详细介绍，实现多文件上传，首先要动态生成 FileUpload 控件；其次，通过循环将FileUpload控件中的上传文件上传到服务器中。多文件上传功能运行结果如图 15.3 所示。

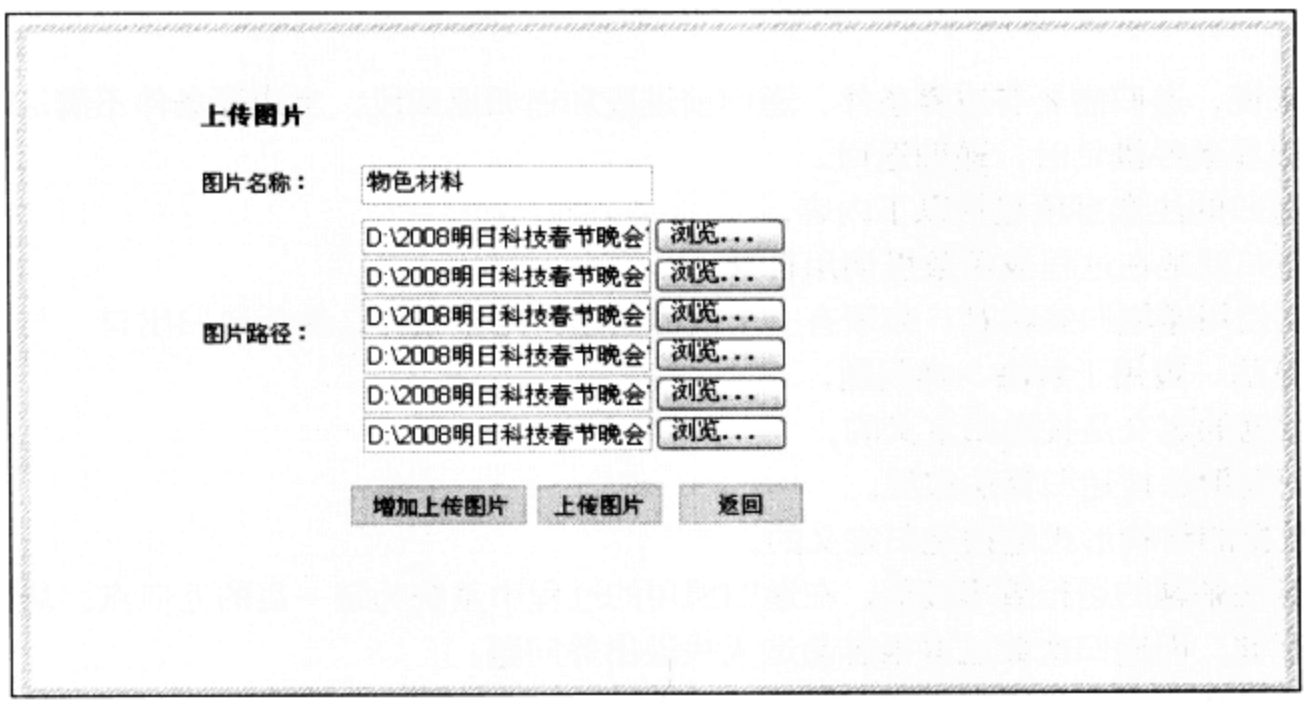


图 15.3 多文件上传

- 动态生成 FileUpload 控件。

首先需要 1 个 Table 表格，并设置为服务器控件，然后将 FileUpload 控件添加到表格中，最后利用类 HtmlTableRow 和 HtmlTableCell 的方法将 FileUpload 控件添加到 Table 表格中。

- HtmlTableRow 类表示 HtmlTable 控件中的<tr> HTML 元素。
- HtmlTableRow. Controls.Add 方法用来将 HtmlTableCell 单元格添加到表格行中。
- HtmlTableCell 类表示 HtmlTableRow 对象中的<td>和<th>HTML 元素。
- HtmlTableCell. Controls.Add 方法将初始化生成的 FileUlpload 控件添加到表格单元格中。
- FileUpload 控件实现文件上传。

下面主要介绍用于实现上传功能的 FileUload 控件的主要属性和方法。

● FileUpload.FileName 属性

该属性用于获取客户端上使用 FileUpload 控件上载的文件的名称。

属性值：该字符串指定客户端上使用 FileUpload 上载的文件的名称。FileName 属性返回的文件名不包含此文件在客户端上的路径。

● FileUpload.SaveAs 方法

使用 FileUpload 控件将上载文件的内容保存到 Web 服务器上的指定路径。

其语法如下：

```
public void SaveAs (
    string filename
)
```

filename 是一个字符串，用于指定服务器上保存上载文件的位置的完整路径。

下面介绍实现多文件上传的关键代码。

生成多个 FileUpLoad 上传控件，完成代码如下：

例程 2 代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
    /// <summary>
    /// 添加1个上传文件的控件（FileUpload）
    /// </summary>
    private void AddFileUpload()
    {
        //定义数组
        ArrayList arrayList = new ArrayList();
        //清除Table中的行
        tab_FileUpload_Area.Rows.Clear();
        //调用自定义方法，获取上传文件控件集
        GetFileUpload();
        HtmlTableRow tabRow = new HtmlTableRow();
        HtmlTableCell tabCell = new HtmlTableCell();
        //添加上传控件
        tabCell.Controls.Add(new FileUpload());
        tabRow.Controls.Add(tabCell);
        tab_FileUpload_Area.Rows.Add(tabRow);
        //调用自定义方法，保存控件集信息到缓存中
        SetFileUpload();
    }
    //<summary>
    //获取缓存中上传文件控件集
    //</summary>
    private void GetFileUpload()
    {
        //声明数组对象
        ArrayList arrayList = new ArrayList();
        if (Session["FilesControls"] != null)
        {
            //将生成的上传控件，显示在Table表格中
            arrayList = (System.Collections.ArrayList)Session["FilesControls"];
            for (int i = 0; i < arrayList.Count; i++)
            {
                HtmlTableRow tabRow = new HtmlTableRow();
                HtmlTableCell tabCell = new HtmlTableCell();
                tabCell.Controls.Add((System.Web.UI.WebControls.FileUpload)arrayList[i]);
                tabRow.Controls.Add(tabCell);
                tab_FileUpload_Area.Rows.Add(tabRow);
            }
        }
```

```
}
//<summary>
//保存当前页面上传文件控件集到缓存中
//</summary>
private void SetFileUpload()
{
    //创建动态数组
    ArrayList arrayList = new ArrayList();
    foreach (Control C in tab_FileUpload_Area.Controls)
    {
        //判断控件类型
        if (C.GetType().ToString() == "System.Web.UI.HtmlControls.HtmlTableRow")
        {
            HtmlTableCell tabCell = (HtmlTableCell)C.Controls[0];
            //在Table单元格中检索FileUpload控件
            foreach (Control control in tabCell.Controls)
            {
                //判断控件类型
                if (control.GetType().ToString() == "System.Web.UI.WebControls.FileUpload")
                {
                    //将FileUpload控件信息保存到ArrayList控件中
                    FileUpload f = (FileUpload)control;
                    arrayList.Add(f);
                }
            }
        }
    }
    Session.Add("FilesControls", arrayList);
}
```

如果上传图片为一张，则不按编号进行命名；如果为多张图片上传，则按指定编号命名。执行代码如下。

例程 3 代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
protected void btnUploadPic_Click(object sender, EventArgs e)
{
    //设置上传图片保存的路径
    string FilePath = Server.MapPath("./") + "Pictures";
    //获取上串图片的集合
    HttpFileCollection HFC = Request.Files;
    for (int i = 0; i < HFC.Count; i++)
    {
        HttpPostedFile UserHPF = HFC[i];
        if (UserHPF.ContentLength > 0)
        {
            // 获取上传图片的文件名
            String fileName = UserHPF.FileName.Substring(UserHPF.FileName.LastIndexOf("\\"));
            //保存图片到指定的位置
            UserHPF.SaveAs(FilePath + "\\" + System.IO.Path.GetFileName(UserHPF.FileName));
        //如果上传1张图片，则不按编号进行命名，如果为多张图片上传，则按编号命名
            if (UserHPF.ContentLength == 1)
                imgData.AddNode(txtPicName.Text.Trim(), "Pictures" + fileName,"0",ViewState ["ImgID"].
ToString());
            else
                imgData.AddNode(txtPicName.Text.Trim() + i.ToString(), "Pictures" + fileName, "0",View
State["ImgID"].ToString());
        }
    }
    //删除所有的FileUpload控件
    if (Session["FilesControls"] != null)
    {
        Session.Remove("FilesControls");
    }
}
```

15.2.3 设置水印图片

在图片资源管理模块中实现水印图片，如图 15.4 所示。这里首先介绍设置水印图片的设计思路和程序代码中使用类的关键属性、方法，然后再给出实现水印图片功能代码，代码中配有详细注释。

图 15.4 设置水印图片

首先获取需要水印图片，然后设置水印图片的透明度。最后将水印图片合成到要加水印图片的指定位置。

下面程序代码中使用类的关键属性和方法。

- Bitmap 类

Bitmap 类是封装 GDI+位图，此位图由图形图像及其属性的像素数据组成。Bitmap 是用于处理由像素数据定义的图像的对象。

- Bitmap 构造函数（String）

从指定的文件初始化 Bitmap 类的新实例。

该函数的语法如下：

```
public Bitmap (
    string filename
)
```

其中 filename 表示位图文件的名称。

- Bitmap.GetPixel 方法

该方法用于获取此 Bitmap 中指定像素的颜色。

其语法如下：

```
public Color GetPixel (
    int x,
    int y
)
```

参数说明如下。

- x：表示要检索的像素的 x 坐标。
- y：表示要检索的像素的 y 坐标。

返回值：Color 结构，它表示指定像素的颜色。

- ImageAttributes 类

ImageAttributes 对象包含有关在呈现时如何操作位图和图元文件颜色的信息。ImageAttributes 对象维护多个颜色调整设置，包括颜色调整矩阵、灰度调整矩阵、灰度校正值、颜色映射表和颜色阈值。在此过程中，可以对颜色进行校正、调暗、调亮和移除。要应用这些操作，应初始化 1 个 ImageAttributes 对象，并将该 ImageAttributes 对象的路径（连同 Image 的路径）传递给 DrawImage 方法。

ImageAttributes.SetColorKey（Color, Color）方法

该方法为默认类别设置颜色键。

其语法如下：

```
public void SetColorKey (
    Color colorLow,
    Color colorHigh
)
```

参数说明如下

- colorLow：表示低颜色键值。
- colorHigh：表示高颜色键值。

此方法无返回值。

此方法设置高低颜色键值，以便颜色范围可以成为透明的。对于任何颜色，只要它的 3 种颜色成分（红、绿、蓝）都处于高低颜色键的对应成分之间，它的颜色就会成为透明的。

- Graphics 类

该类用于封装一个 GDI+绘图图面。

- Graphics.FromImage 方法

该方法从指定的 Image 创建新的 Graphics。

其语法如下：

```
public static Graphics FromImage (
    Image image
)
```

其中 image 表示从中创建新 Graphics 的 Image。

返回值：此方法为指定的 Image 返回 1 个新的 Graphics。

- Graphics.DrawImage 方法(Image、Rectangle、Single、Single、Single、Single、 GraphicsUnit、ImageAttributes)

该方法用于在指定位置并且按指定大小绘制指定的 Image 的指定部分。

其语法如下：

```
public void DrawImage (
    Image image,
    Rectangle destRect,
    float srcX,
    float srcY,
    float srcWidth,
    float srcHeight,
    GraphicsUnit srcUnit,
    ImageAttributes imageAttrs
)
```

参数说明如下。

- image：表示要绘制的 Image。
- destRect：表示 Rectangle 结构，它指定所绘制图像的位置和大小。将图像进行缩放以适合该矩形。

- srcX：表示要绘制的源图像部分的左上角的 x 坐标。
- srcY：表示要绘制的源图像部分的左上角的 y 坐标。
- srcWidth：表示要绘制的源图像部分的宽度。
- srcHeight：表示要绘制的源图像部分的高度。
- srcUnit：表示 GraphicsUnit 枚举的成员，它指定用于确定源矩形的度量单位。
- imageAttrs：表示 ImageAttributes，它指定 image 对象的重新着色和伽玛信息。

实现设置水印图片的代码如下：

例程 4 代码位置：光盘\mr\15\PictureManager\App_Code\PictureSet.cs

```
        /// <summary>
        /// 设置水印图片
        /// </summary>
        /// <param name="path">要设置水印图片的路径</param>
        public void WaterMark(string path)
        {
            //获取要水印的图片
            Bitmap bmp = new Bitmap(HttpContext.Current.Server.MapPath(".") + "/Alex.gif");
            //设置水印图片的透明度
            ImageAttributes imageAttr = new ImageAttributes();
            imageAttr.SetColorKey(bmp.GetPixel(20, 20), bmp.GetPixel(20, 20));
            //获取要设置水印图片的扩展名
            string extension = Path.GetExtension(path).ToUpper();
            //设置临时图片名称
            string fileName = DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString() + DateTime.Now.Day.
ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.ToString() + DateTime.Now.Second.ToString();
            //初始化要加水印的图片
            System.Drawing.Image image = System.Drawing.Image.FromFile(path);
            //初始化水印图片
            System.Drawing.Image copyImage = System.Drawing.Image.FromFile(HttpContext.Current.Server.
MapPath(".") + "/Alex.gif");
            //创建绘图区域
            Graphics g = Graphics.FromImage(image);
            //将水印图片合成到指定图片上
            g.DrawImage(copyImage, new Rectangle(image.Width - copyImage.Width, image.Height - copyImage.Height,
copyImage.Width, copyImage.Height), 0, 0, copyImage.Width, copyImage.Height, GraphicsUnit.Pixel, imageAttr);
            g.Dispose();
            //保存加水印过后的图片，删除原始图片
            string newPath = HttpContext.Current.Server.MapPath(".") + "/" + fileName + "_new" + extension;
            //保存设置完成后的水印图片到临时位置
            image.Save(newPath);
            image.Dispose();
            File.Copy(newPath, path, true);
            //删除水印
            if (File.Exists(newPath))
            {
                //删除临时存储的图片
                File.Delete(newPath);
            }
        }
```

15.2.4 设置水印文字

图片资源管理模块中实现水印文字，如图 15.5 所示。设置水印文字与设置水印图片相比，设计思路和实现方法基本一致，在实现过程中比设置水印图片简单，实现的关键

技术这里不再详细介绍，直接给出关键代码，具体内容读者可参看代码中标注的详细注释。

图 15.5　设置水印文字

实现设置水印文字的代码如下：

例程 5　代码位置：光盘\mr\15\PictureManager\App_Code\PictureSet.cs

```
    /// <summary>
    /// 设置水印文字
    /// </summary>
    /// <param name="path">要设置水印图片的路径</param>
    /// <param name="str">水印文字</param>
    public void WaterLetter(string path, string str)
    {
        //获取文件扩展名
        string extension = Path.GetExtension(path).ToUpper();
        //设置临时文件名称
        string fileName = DateTime.Now.Year.ToString() + DateTime.Now.Month.ToString() + DateTime.Now.Day.
ToString() + DateTime.Now.Hour.ToString() + DateTime.Now.Minute.ToString() + DateTime.Now.Second.ToString();
        //加文字水印
        System.Drawing.Image image = System.Drawing.Image.FromFile(path);
        Graphics g = Graphics.FromImage(image);
        g.DrawImage(image, 0, 0, image.Width, image.Height);
        Font f = new Font("Verdana", 64);
        Brush b = new SolidBrush(Color.White);
        g.DrawString(str, f, b, 10, 10);
        g.Dispose();
        //保存加水印过后的图片，删除原始图片
        string newPath = HttpContext.Current.Server.MapPath(".") + "/" + fileName + "_new" + extension;
        image.Save(newPath);
        image.Dispose();
        //将水印图片替换为原来图片
        File.Copy(newPath, path, true);
        //删除水印
        if (File.Exists(newPath))
        {
            File.Delete(newPath);
        }
    }
```

15.3 图片资源管理模块主页设计

图片资源管理模块主页中包括管理图片资源的菜单（上传图片、新建目录、修改项目和删除项目）、显示指定的图片、设置水印文字、设置水印图片、获取图片的像素等功能，如图 15.6 所示。

图 15.6 图片资源管理模块主页

15.3.1 页面设计

图片资源管理模块主页添加主要控件及用途如表 15.1 所示。

表 15.1　　图片资源管理模块主页控件及用途

控　件	ID 属性	主 要 属 性	用　途
ImageButton	imgBtnAddPicture	ImageUrl 属性值为 images/btn1.jpg	选中目录后，进入到上传页面
ImageButton	imgBtnNewDir	ImageUrl 属性值为 images/btn2.jpg	新建图片保存的目录
ImageButton	imgBtnEditNode	ImageUrl 属性值为 images/btn3.jpg	编写选中的图片或目录
ImageButton	imgBtnDeleteNode	ImageUrl 属性值为 images/btn4.jpg	删除选中的图片
ImageButton	btnPicSet	ImageUrl 属性值为 images/ sztp.jpg	为选中的图片设置水印图片
ImageButton	btnWaterWZ	ImageUrl 属性值为 images/szwz.jpg	为选中的图片设置水印文字
Label	lblPath	默认	显示当前图片的路径
Label	lblWidth	默认	显示图片宽度
Label	lblHeight	默认	显示图片高度
TextBox	txtWZ	默认	输入要设置的水印文字
TreeView	treeList	默认	通过 TreeView 控件显示所有的图片
Image	imagePic	默认	显示选中的图片

15.3.2 实现代码

声明全局数据库操作类对象、图片设置类对象、保存图片字段，便于程序开发调用。在页面装载的 Page_Load 事件中，调用 BindListView 自定义方法在 TreeView 控件上显示所有的图片数据。实现代码如下。

例程 6 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
//声明数据库操作类对象
ImageDataBase imgData = new ImageDataBase();
//声明图片设置类对象
PictureSet pSet = new PictureSet();
//声明图片存储字段
static string savePicPath = "";
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        //调用自定义方法
        BindListView(treeList);
    }
}
```

自定义 BindListView 方法用于将数据库中的图片信息绑定到 TreeView 控件，并且通过自定义 CreateChildNode 方法来完成递归操作。实现代码如下。

例程 7 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
/// <summary>
/// 自定义方法，绑定TreeView控件
/// </summary>
/// <param name="treeView"></param>
public void BindListView(TreeView treeView)
{
    //清空树的所有节点
    treeView.Nodes.Clear();
    //创建根节点
    TreeNode rootNode = new TreeNode();
    rootNode.Text = "图片目录";
    rootNode.Value = "1";
    rootNode.ImageUrl = "~/Image/Root.gif";
    //展开所有节点
    rootNode.Expanded = true;
    rootNode.Selected = true;
    //添加根节点
    treeView.Nodes.Add(rootNode);
    //获取所有节点信息
    DataTable dataTable = imgData.GetPicInfo();
    //创建其他节点
    CreateChildNode(rootNode, dataTable);
}
```

自定义 CreateChildNode 方法，主要用于实现递归操作，将目录中含有子目录中的数据显示到 TreeVew 控件中。实现代码如下。

例程 8 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
/// <summary>
/// 为TreeView控件绑定子节点（递归）
/// </summary>
/// <param name="parentNode"></param>
/// <param name="dataTable"></param>
private void CreateChildNode(TreeNode parentNode, DataTable dataTable)
{
    DataRow[] rowList = dataTable.Select("ParentID='" + parentNode.Value + "'");
```

```
        foreach (DataRow row in rowList)
        {    //创建新节点
             TreeNode node = new TreeNode();
             //设置节点的属性
             node.Text = row["ImgName"].ToString();
             node.Value = row["ImgID"].ToString();
             if (row["isDir"].ToString() == "True")
             {
                  node.Expanded = true;
                  node.Target = "_self";
                  node.ImageUrl = "~/Image/Dir.gif";
             }
             else
             {
                  node.Target = "_blank";
                  node.ImageUrl = "~/Image/Picture.gif";
             }
             node.ImageToolTip = row["ImgUrl"].ToString();
             parentNode.ChildNodes.Add(node);
             //递归调用，创建其他节点
             CreateChildNode(node, dataTable);
        }
    }
```

单击 TreeView 控件中节点图片的名称，在 Image 控件中显示选中的图片、显示图片的存储路径以及获取图片的大小尺寸。实现代码如下。

例程 9 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
protected void treeList_SelectedNodeChanged(object sender, EventArgs e)
{
    if (treeList.SelectedNode.ImageToolTip != "")
    {
        //存放到临时文件中
        string pPath = treeList.SelectedNode.ImageToolTip;
        //临时文件夹是否存放临时文件
        if (!System.IO.File.Exists(Server.MapPath(@"Pictures\temp\" + pPath.Substring(9, pPath.Length - 9))))
            System.IO.File.Copy(Server.MapPath(pPath), Server.MapPath(@"Pictures\temp\" + pPath.Substring
(9, pPath.Length - 9)), true);
        imagePic.ImageUrl = @"Pictures\temp\" + pPath.Substring(9, pPath.Length - 9);
        savePicPath = treeList.SelectedNode.ImageToolTip;
        imagePic.Visible = true;
        //获取图片尺寸
        FileWebRequest q = (FileWebRequest)FileWebRequest.Create(Server.MapPath(imagePic.ImageUrl));
        FileWebResponse p = (FileWebResponse)q.GetResponse();
        System.Drawing.Image image = System.Drawing.Image.FromStream(p.GetResponseStream());
        lblHeight.Text = "图片宽：" + image.Height.ToString() + "    像素";
        lblWidth.Text = "图片高：" + image.Width.ToString() + "    像素";
    }
    //在根节点
    if (treeList.SelectedNode.Depth == 0)
        lblPath.Text = treeList.SelectedNode.Text;
    //在二级节点
    if (treeList.SelectedNode.Depth == 1)
        lblPath.Text = treeList.SelectedNode.Parent.Text + "→" + treeList.SelectedNode.Text;
    //在三级节点
    if (treeList.SelectedNode.Depth == 2)
        lblPath.Text = treeList.SelectedNode.Parent.Parent.Text + "→" + treeList.SelectedNode.Parent.Text + "→
" + treeList.SelectedNode.Text;
    //在四级节点
    if (treeList.SelectedNode.Depth == 3)
        lblPath.Text = treeList.SelectedNode.Parent.Parent.Parent.Text + "→" + treeList.SelectedNode.Parent.
Parent.Text + "→" + treeList.SelectedNode.Parent.Text + "→" + treeList.SelectedNode.Text;
}
```

单击“上传图片”按钮进入图片上传页面，如果指定目录，则进入上传页面；如果未指定目录，则提示相关信息。实现代码如下。

例程 10 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
//上传图片
protected void btnAddPicture_Click(object sender, EventArgs e)
{
    if (treeList.SelectedNode == null)
    {
        //显示提示信息
        Response.Write("<script>window.alert('请选择合适的目录上传')</script>");
        return;
    }
    else if (!imgData.CheckDir(treeList.SelectedValue))
    {
        //显示提示信息
        Response.Write("<script>window.alert('请选择合适的目录上传')</script>");
        return;
    }
    Response.Redirect("~/Upload.aspx?ImgID=" + treeList.SelectedValue);
}
```

单击“新建目录”按钮进入添加目录页面，如果指定在某个目录中建立目录，则进入目录添加页面。如果未指定目录，则提示相关信息。实现代码如下。

例程 11 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
// 新建目录
protected void btnNewDir_Click(object sender, EventArgs e)
{
    if (treeList.SelectedNode == null)
    {
        //显示提示信息
        Response.Write("<script>window.alert('请选择合适的目录添加')</script>");
        return;
    }
    else if (!imgData.CheckDir(treeList.SelectedValue))
    {
        //显示提示信息
        Response.Write("<script>window.alert('请选择合适的目录添加')</script>");
        return;
    }
    Response.Redirect("~/Building.aspx?ImgID=" + treeList.SelectedValue);
}
```

单击“修改项目”按钮进入项目（图片或目录）修改页面，如果指定目录或图片，则进入修改页面；如果未指定目录或图片，则提示相关信息。实现代码如下。

例程 12 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
//修改项目名称
protected void btnEditNode_Click(object sender, EventArgs e)
{
    if (treeList.SelectedNode == null)
    {
        //显示提示信息
        Response.Write("<script>window.alert('请选择合适的修改项目')</script>");
        return;
    }
    else if (treeList.SelectedValue == "1")
    {
        //显示提示信息
        Response.Write("<script>window.alert('根目录不能修改名称')</script>");
        return;
```

```
        }
        Response.Redirect("~/Edit.aspx?" +
                          "ImgID=" + treeList.SelectedValue + "&" +
                          "ImgName=" + Server.HtmlEncode(treeList.SelectedNode.Text));
    }
```

单击“删除项目”按钮进入项目（图片或目录）删除页面，如果指定目录或图片，则进入修改页面；如果未指定目录或图片，则提示相关信息。实现代码如下。

例程 13 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
    //删除项目
    protected void btnDeleteNode_Click(object sender, EventArgs e)
    {
        if (treeList.SelectedNode == null)
        {
            //显示提示信息
            Response.Write("<script>window.alert('请选择合适的删除对象')</script>");
            return;
        }
        else if (treeList.SelectedValue == "1")
        {
            //显示提示信息
            Response.Write("<script>window.alert('根目录不能删除')</script>");
            return;
        }
        else if (imgData.CheckChildNode(treeList.SelectedValue))
        {
            //显示提示信息
            Response.Write("<script>window.alert('请先删除目录下的文件和子目录')</script>");
            return;
        }
        //删除节点
        imgData.DeleteNode(treeList.SelectedValue);
        System.IO.File.Delete(Server.MapPath(treeList.SelectedNode.ImageToolTip));
        //显示提示信息
        Response.Write("<script>window.alert('删除成功！')</script>");
        BindListView(treeList);
        imagePic.Visible = false;
    }
```

单击“设置水印图片”按钮为选中的图片设置水印图片。实现代码如下。

例程 14 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
    protected void btnPicSet_Click(object sender, EventArgs e)
    {
        //设置整个目录图片文件
        if (treeList.SelectedNode == null)
        {
            //显示提示信息
            Response.Write("<script>window.alert('请选择合适的目录添加')</script>");
            return;
        }
        else if (!imgData.CheckDir(treeList.SelectedValue))       //对选择的图片进行设置水印
        {
            //设置水印图片
            imagePic.ImageUrl = "";
            imagePic.Dispose();
            pSet.WaterMark(Server.MapPath(savePicPath));
            imagePic.ImageUrl = savePicPath;
        }
        else          //对选中的目录进行设置水印
        {
            foreach (DataRow dr in imgData.GetPicture(treeList.SelectedValue).Rows)
            {
```

```
                pSet.WaterMark(Server.MapPath(dr[0].ToString()));
                imagePic.ImageUrl = savePicPath;
            }
        }
    }
```

在文字文本框中输入要设置水印的文字，然后单击“设置水印文字”按钮为选中的图片设置水印文字。实现代码如下。

例程 15 代码位置：光盘\mr\15\PictureManager\Default.aspx.cs

```
protected void btnWaterWZ_Click(object sender, EventArgs e)
{
    //设置水印文字
    imagePic.ImageUrl = "";
    imagePic.Dispose();
    pSet.WaterLetter(Server.MapPath(savePicPath), txtWZ.Text);
    imagePic.ImageUrl = savePicPath;
}
```

15.4 新 建 目 录

在图片资源管理模块主页中，当指定在某个目录下建立子目录时，单击“新建目录”按钮进入新建目录页，如图 15.7 所示。这里的新建目录并不是真正在磁盘驱动器中建立目录（文件夹），而是在本程序中达到更好的形象及程序的人性化。

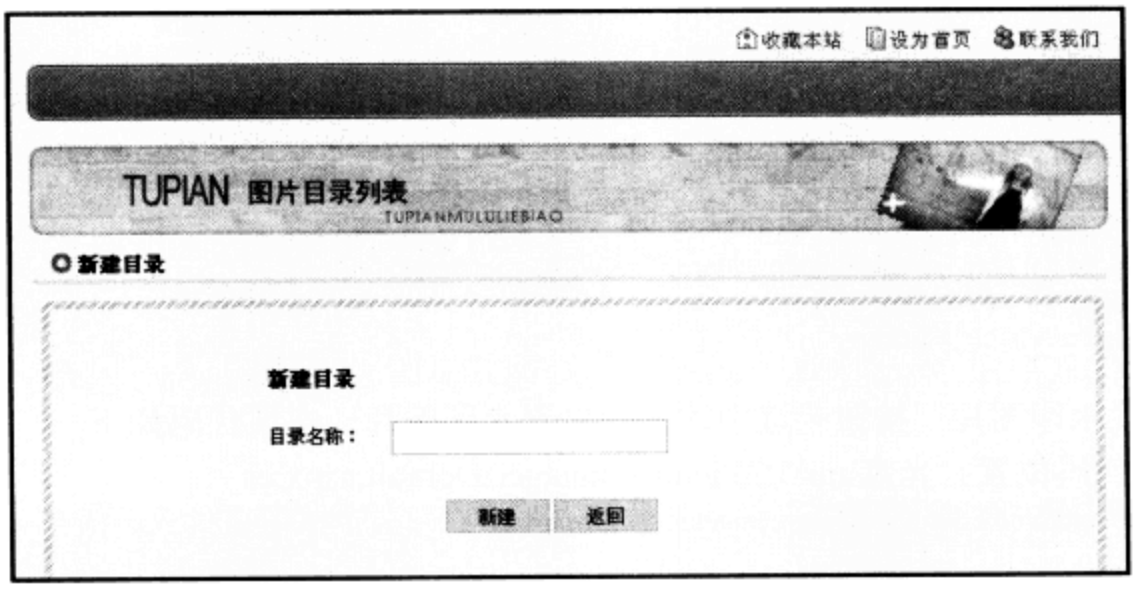


图 15.7 新建目录

15.4.1 页面设计

新建目录页添加主要控件及用途如表 15.2 所示。

表 15.2 新建目录页控件及用途

控 件	ID 属性	主 要 属 性	用 途
Button	BtnAddDir	Text 属性值为“新建”	执行新建目录命令
Button	btnReturn	Text 属性值为“返回”、CausesValidation 属性为 False	返回主页
TextBox	txtDirName	默认	编写选中的图片或目录
RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate 属性为 txtDirName、ErrorMessage 属性值为“*名字不能为空”	删除选中的图片

15.4.2 实现代码

声明全局数据库操作类对象 imgData，便于程序开发过时调用类中的相关方法。在 Page_Load 事件中主要获取通过地址栏中传入的 ID 值，保存到 ViewState 中。传入的 ID 值是将要创建目录的父目录的 ID 值。实现代码如下：

例程 16 代码位置：光盘\mr\15\PictureManager\Building.aspx.cs

```
ImageDataBase imgData = new ImageDataBase();
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //判断参数值是否为空，如果不为空获取参数值
        if (Request.Params["ImgID"] != null)
        {
            btnAddDir.Enabled = true;
            ViewState["ImgID"] = Request.Params["ImgID"].ToString();
        }
        else
        {
            btnAddDir.Enabled = false;
            ViewState["ImgID"] = "";
        }
    }
}
```

单击“新建”按钮，将创建的目录名称保存到到数据库中。保存到数据库中主要调用数据库操作类中的 AddNode 方法实现，其中参数值 1 标识目录的意思。实现代码如下：

例程 17 代码位置：光盘\mr\15\PictureManager\Building.aspx.cs

```
protected void btnAddDir_Click(object sender, EventArgs e)
{
    //在指定的参数值（目录）中创建新的目录，保存到数据库中
    imgData.AddNode(tbxDirName.Text,"","1",ViewState["ImgID"].ToString());
    Response.Write("<script>window.alert('目录添加成功！')</script>");
}
```

单击“返回”按钮，将当前页定向到主页。实现代码如下：

例程 18 代码位置：光盘\mr\15\PictureManager\Building.aspx.cs

```
protected void btnReturn_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Default.aspx");
}
```

15.5 上 传 图 片

当在图片资源管理模块主页中，指定在某个目录下上传图片时，单击“上传图片”按钮进入上传图片页，如图 15.8 所示，这里除了可以进行单张图片上传，还支持多文件上传，多文件上传主要通过动态增加多个 FileUpload 服务器控件实现。另外，如果是多文件进行上传，其图片名称尾部自动增加一个流水号。

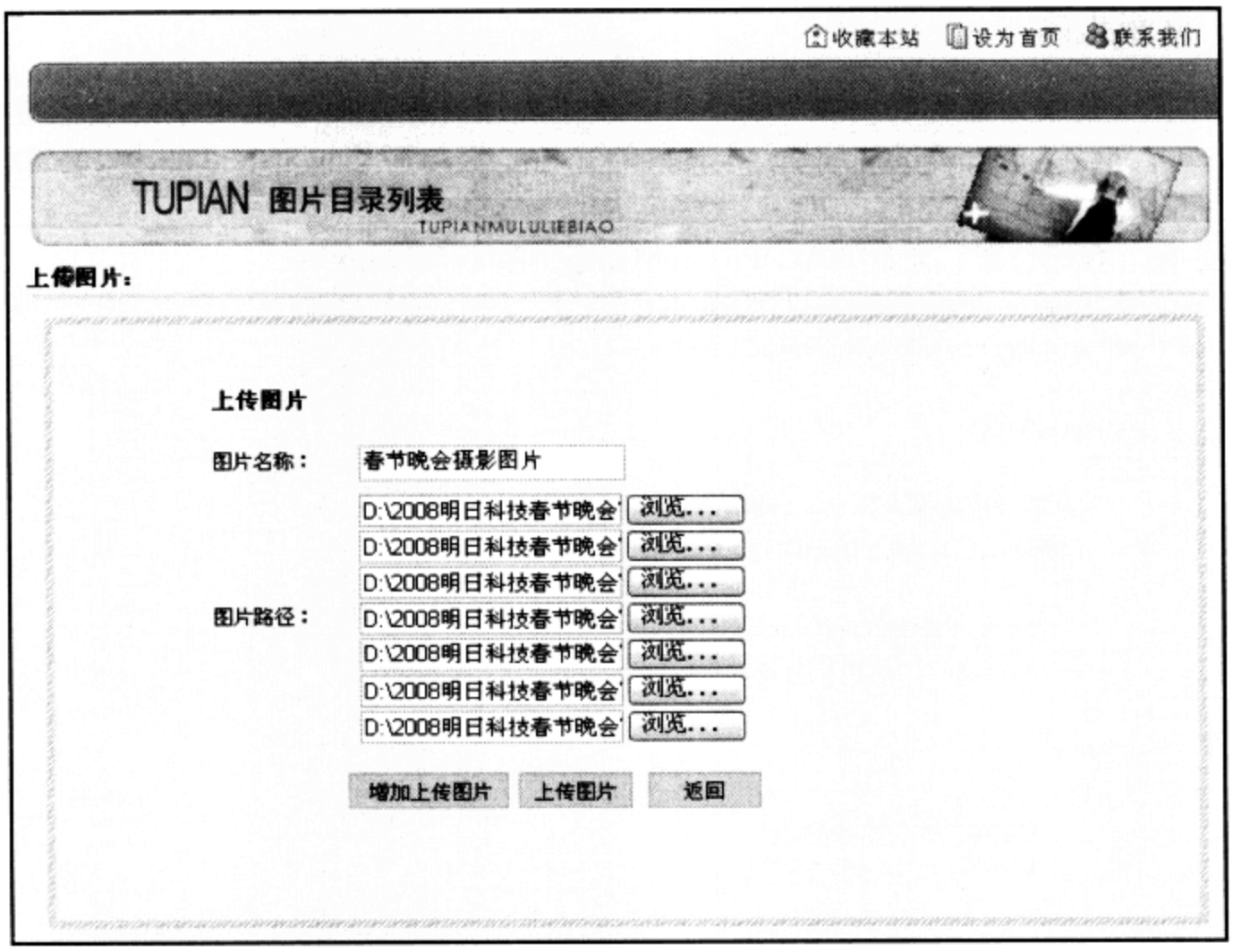


图 15.8　上传图片

15.5.1　页面设计

上传图片页添加主要控件及用途如表 15.3 所示。

表 15.3　　上传图片页控件及用途

控　件	ID 属性	主 要 属 性	用　途
Button	btnUploadPic	Text 属性值为“上传图片”	执行上传命令
Button	btnReturn	Text 属性值为“返回”、CausesValidation 属性为 True	返回主页
Button	btnAdd	Text 属性值为“返回”、CausesValidation 属性为 False	动态增加 FileUpload 控件
TextBox	txtDirName	默认	输入图片名称

15.5.2　实现代码

声明全局数据库操作类对象 imgData，便于程序开发过时调用类中的相关方法。在 Page_Load 事件中主要获取通过地址栏中传入的 ID 值，保存到 ViewState 中。传入的 ID 值是上传图片父目录的 ID 值。实现代码如下。

例程 19　代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
ImageDataBase imgData = new ImageDataBase();
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Request.Params["ImgID"] != null)
        {
```

```
                btnUploadPic.Enabled = true;
                ViewState["ImgID"] = Request.Params["ImgID"].ToString();
            }
            else
            {
                btnUploadPic.Enabled = false;
                ViewState["ImgID"] = "";
            }
        }
    }
```

自定义 AddFileUpload 方法，用于在指定的 Table 表格中动态添加 FileUpload 控件。在下面的代码中需要调用自定义 GetFileUpload 方法来判断是否已经添加过 FileUpload 控件，如果已添加 FileUpload 控件，则在原来的基础上继续增加。最后调用自定义 SetFileUpload 方法保存添加的 FileUpload 控件。实现代码如下。

例程 20 代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
//<summary>
//添加1个上传文件的控件（FileUpload）
//</summary>
private void AddFileUpload()
{
    //定义数组
    ArrayList arrayList = new ArrayList();
    //清除Table中的行
    tab_FileUpload_Area.Rows.Clear();
    //调用自定义方法，获取上传文件控件集
    GetFileUpload();
    HtmlTableRow tabRow = new HtmlTableRow();
    HtmlTableCell tabCell = new HtmlTableCell();
    //添加上传控件
    tabCell.Controls.Add(new FileUpload());
    tabRow.Controls.Add(tabCell);
    tab_FileUpload_Area.Rows.Add(tabRow);
    //调用自定义方法，保存控件集信息到缓存中
    SetFileUpload();
}
```

自定义 GetFileUpload 方法用于获取动态生成添加 FileUpload 信息。实现代码如下。

例程 21 代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
//<summary>
//获取缓存中上传文件控件集
//</summary>
private void GetFileUpload()
{
    //声明数组对象
    ArrayList arrayList = new ArrayList();
    if (Session["FilesControls"] != null)
    {
        //将生成的上传控件，显示在Table表格中
        arrayList = (System.Collections.ArrayList)Session["FilesControls"];
        for (int i = 0; i < arrayList.Count; i++)
        {
            HtmlTableRow tabRow = new HtmlTableRow();
            HtmlTableCell tabCell = new HtmlTableCell();
            tabCell.Controls.Add((System.Web.UI.WebControls.FileUpload)arrayList[i]);
            tabRow.Controls.Add(tabCell);
            tab_FileUpload_Area.Rows.Add(tabRow);
        }
    }
}
```

自定义 SetFileUpload 方法，用于将添加的 FileUpload 控件信息保存到缓存中。实现代码如下。

例程 22 代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
//<summary>
//将当前页面中的上传文件控件集保存到缓存中
//</summary>
private void SetFileUpload()
{
    //创建动态数组
    ArrayList arrayList = new ArrayList();
    foreach (Control C in tab_FileUpload_Area.Controls)
    {
        //判断控件类型
        if (C.GetType().ToString() == "System.Web.UI.HtmlControls.HtmlTableRow")
        {
            HtmlTableCell tabCell = (HtmlTableCell)C.Controls[0];
            //在Table单元格中检索FileUpload控件
            foreach (Control control in tabCell.Controls)
            {
                //判断控件类型
                if (control.GetType().ToString() == "System.Web.UI.WebControls.FileUpload")
                {
                    //将FileUpload控件信息保存到ArrayList控件中
                    FileUpload f = (FileUpload)control;
                    arrayList.Add(f);
                }
            }
        }
    }
    Session.Add("FilesControls", arrayList);
}
```

单击“增加上传图片”按钮，则在页面中添加多个 FileUpload 控件。实现代码如下。

例程 23 代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
protected void btnAdd_Click(object sender, EventArgs e)
{
    this.AddFileUpload();              //调用自定义方法
}
```

单击“上传图片”按钮，将上传的图片上传的服务器中，如果多张图片上传，其名称是末尾加流水号（例如，图片 0、图片 1 等）。实现代码如下。

例程 24 代码位置：光盘\mr\15\PictureManager\Upload.aspx.cs

```
protected void btnUploadPic_Click(object sender, EventArgs e)
{
    //设置上传图片保存的路径
    string FilePath = Server.MapPath("./") + "Pictures";
    //获取上传图片的集合
    HttpFileCollection HFC = Request.Files;
    for (int i = 0; i < HFC.Count; i++)
    {
        HttpPostedFile UserHPF = HFC[i];
        if (UserHPF.ContentLength > 0)
        {
            //获取上传图片的文件名
            String fileName = UserHPF.FileName.Substring(UserHPF.FileName.LastIndexOf("\\"));
            //保存图片到指定的位置
            UserHPF.SaveAs(FilePath + "\\" + System.IO.Path.GetFileName(UserHPF.FileName));
        //如果上传图片为1张，则不按编号进行命名，如果为多张图片上传，则进行按编号命名
```

```
                    if (UserHPF.ContentLength == 1)
                        imgData.AddNode(txtPicName.Text.Trim(), "Pictures" + fileName,"0",ViewState["ImgID"].
ToString());
                    else
                        imgData.AddNode(txtPicName.Text.Trim() + i.ToString(), "Pictures" + fileName,"0",View
State["ImgID"]. ToString());
                }
            }
            //删除所有的FileUpload控件
            if (Session["FilesControls"] != null)
            {
                Session.Remove("FilesControls");
            }
        }
```

15.6 常见开发技术问题总结

在开发图片资源管理模块程序中，笔者遇到很多开发常见问题，开发常见问题主要包括以下内容。

- 打开新的 Web 窗口并传送参数。
- 传送参数。

```
Response.write ("<script>window.open (Default.aspx?id="+ str1+"&id1="+str2+"")</script>")
```

- 接收参数。

```
String a    = Request.QueryString ("id");
String b    = Request.QueryString ("id1");
```

- 为“Button”按钮添加确认对话框。

```
Button1.Attributes.Add("onclick","return confirm('确认?')");
```

- 删除 GirdView 控件中选定的记录。

```
int intD = (int) GridView1.DataKeys[e.Item.ItemIndex];
string delCmd = "DELETE from Employee where emp_id = " + intID.ToString()
```

- 删除 GirdView 控件表格中记录，弹出确认对话框。

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        //删除指定行数据时，弹出询问对话框
        ((LinkButton)(e.Row.Cells[3].Controls[0])).Attributes.Add("onclick", "return confirm('是否删除当前行
数据！')");
    }
}
```

- 在 GirdView 控件中格式日期。

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
    e.items.cell["列"].text=DateTime.Parse(e.items.cell["列"].text.ToString("yyyy-MM-dd"));
    }
}
```

- 捕获错误信息后程序跳转到指定页面。

该过程在 Global.asax 全局文件中完成，使用 Server.Transfer 跳转到指定页，代码如下：

```
protected void Application_Error(Object sender, EventArgs e)
{
    if (Server.GetLastError() is HttpUnhandledException)
        Server.Transfer("ErrorPage.aspx");
}
```

- 清空 Cookie，代码如下：

```
Cookie.Expires=[DateTime];
Response.Cookies("UserName").Expires = 0
```

- 大小写转换，代码如下：

```
HttpUtility.HtmlEncode(string);
HttpUtility.HtmlDecode(string)
```

- 设置全局变量，代码如下：

设置全局变量主要通过 Global.asax 文件中 Application_Start 事件完成，代码如下：

```
    Application[属性名] = xxx;
```

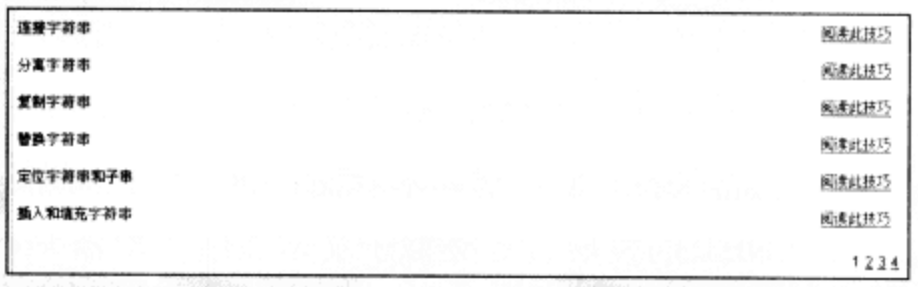
第16章 搜索引擎模块

实例位置：光盘\mr\16\

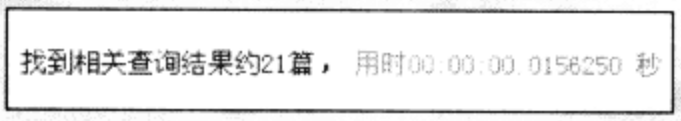
从目前情况看，讲解搜索引擎技术的资料并不多，如果有，一般都是停留在理论阶段，并非讲解实现搜索引擎的开发过程。另外，SQL 语句查询已经无法满足用户的需求，为何不在自己所开发的网站中，实现像百度和谷歌那样的技术搜索引擎呢？根据广大开发者和客户的需求，本章将详细介绍通过 Lucene.NET 技术实现搜索引擎的方法。通过本章的学习，读者能够学到以下内容。

查询分词

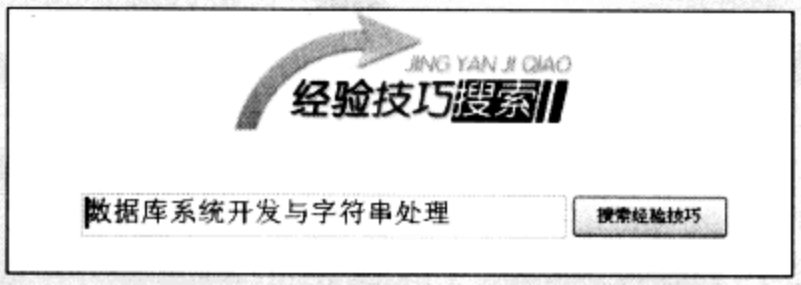
查询关键字描红



统计查询结果与时间



主页搜索引擎



16.1 搜索引擎概述

Google（谷歌）搜索引擎取得巨大的成功，撼动了整个互联网世界，之后各种各样搜索引擎服务席卷而来，从最初的 google、yahoo 到如今的 baidu、msn 等，搜索引擎的品牌越来越多，服务也越来越丰富。随着 Web 2.0 的发展与普及，网络信息的膨胀速度呈指数级增长，各种各样的网站都希望加入搜索功能，来满足用户的需要。

另外，在企业级应用的市场上，全文信息检索的需求也一直在增加，各种文档处理、内容管理软件都需要加入全文检索搜索的功能。

在上述市场需求的背景下，搜索引擎的技术迅速发展。

其实，搜索引擎并不是一门大众技术，从其出现开始，就是一门高门槛技术，搜索引擎实质上包括学术领域的众多先进思想和设计理念，其涉及的学科包括自然语言处理、人工智能、离散数学、排列组合、编译原理等。因此，设计一个性能良好并且实用性强的搜索引擎并非是一件很容易的事。

16.2 实现搜索引擎关键技术

16.2.1 了解 Lucene.NET 技术及其基本语法

1. Lucene.NET 技术概述

Lucene.NET 起初是一个开源项目，从 Lucene.NET 2.0 开始转为商业化并且开始收费，Lucene.NET 的发展有点类似于文本在线编辑器 FreeTextBox。

Lucene.NET 提供的方法可以为文本资料创建索引，并提供检索。也就是说 Lucene.NET 就是一个信息检索的函数库（Library），通过使用 Lucene.NET 可以为所开发的应用程序加上索引和搜索的功能。

值得注意的是，Lucene.NET 可以实现像 google 和百度那样的搜索引擎，从开发者的使用角度看，Lucene.NET 仅仅是一个工具，一个 Library。可以把 Lucene.NET 理解为一个将索引和搜索功能封装好的一套简单易用的 API，它可以满足对一个应用做简单的全文搜索，作为应用的开发者来说，它的功能足以满足用户需求。

2. 创建索引

Lucene.NET 创建全文搜索最核心的内容是创建索引和数据检索，而创建索引是后面数据检索的基础，因为后面的数据检索是使用索引来搜索的。对于创建索引，Lucene.NET 专门提供了相关的类来实现，其中 Lucene.Net.Index.IndexWrite 创建索引并将索引写入文件，对应的 Lucene.Net.Index.IndexReader 实现从文件夹中将索引读出来，以便对索引进行修改等操作。

索引文件创建成功后的效果图，如图 16.1 所示。

图 16.1 创建索引文件效果图

创建索引的构造函数代码如下：

```
public IndexWriter(string path, Analyzer a, bool create);
public IndexWriter(FileInfo path, Analyzer a, bool create);
public IndexWriter(Directory d, Analyzer a, bool create);
```

程序开发人员经常用的是第一种形式。下面对构造函数中的参数进行介绍。

- 参数 string path：为要存储索引所在目录（如 D:\search\index）。
- 参数 Analyzer a：是选用词语分析器（如 new StandardAnalyzer（））, 也是一个分析对象，主要用于从文本中抽取那些需要建立索引的内容，把不需要的文本内容去掉，如去掉一些 a 或 the 之类的常用词，还有决定是否大小写敏感。不同的选项通过指定不同的分析对象控制。
- 参数 bool create：用于确定是否覆盖原有索引，True 表示新创建的索引将覆盖掉原来的索引，false 将重新创建并保留原有索引。

在本模块中实现创建搜索引擎的索引代码如下：

```
Lucene.Net.Analysis.Standard.StandardAnalyzer a = new Lucene.Net.Analysis.Standard.StandardAnalyzer();
//Server.MapPath("Index")为索引文件存放地址
IndexWriter iw = new IndexWriter(Server.MapPath("Index"), a, true);
```

3. 创建文档

创建完成搜索引擎的索引后就可以开始创建索引中的搜索文档了。创建文档主要通过 Document 类和字段（域）Field 实现。其中 Document 类中的相关方法如表 16.1 所示。

表 16.1　　Document 类中的方法及说明

方 法 名 称	说　　明
Add()	增加索引
Get()	获取索引字段存储的值
GetBoost()	获取权重
SetBoost()	设置权重
GetField()	返回赋予字段的名
RemoveField()	从文档中删除特定名的字段

下面举一个例子创建一个文档对象，并通过例子讲解 Field 的使用；

```
Document doc = new Document();
doc.Add(Field.Keyword("resTitle", title));
doc.Add(Field.Text("resContent", info));
doc.Add(Field.Text("resTitle", title));
doc.Add(Field.Text("id", id));
doc.Add(Field.UnStored("text", context);
doc.Add(Field.Keyword("path", path));
doc.Add(Field .Text ("filename",filename));
writer.AddDocument(doc);                                    //将doc添加到索引中
```

上面代码中主要创建文档对象及给文档添加属性，Add 方法用于将一个属性添加到 Document 类对象中，Field 类中的具体方法说明如下。

- Keyword 方法，该方法中的内容不经过分析但会被索引并直接保存到索引中，例如，good，file 等可以是字符串常量，也可以是一个字符串数组，如 string [] contex={“doc”，“pdf”，“html”，“txt”｝。

```
doc.Add(Filed.Keyword("text", context);
```

- UnIndexed 不被分析，不被索引，但却保存在索引中。

- Unstrored 和 UnIndexed 刚好相反。
- Text 和 UnStrored 类似。如果值的类型为 string 还会被保存，如果值的类型为 Reader 就不会被保存和 UnStored 一样。

本章创建索引的代码如下。

例程 1 代码位置：光盘\mr\16\SearchEngine\resAdd.aspx.cs

```
//创建查询索引
protected void imgBtnIndex_Click(object sender, ImageClickEventArgs e)
{
    Lucene.Net.Analysis.Standard.StandardAnalyzer a = new Lucene.Net.Analysis.Standard.StandardAnalyzer();
//Server.MapPath("Index")为索引文件存放地址
    IndexWriter iw = new IndexWriter(Server.MapPath("Index"), a, true);
    string conn = "Server=.;DataBase=db_res;Uid=sa;pwd=;";
    using (SqlConnection con = new SqlConnection(conn))
    {
        SqlDataAdapter dap = new SqlDataAdapter("select * from tb_res", con);
        DataSet ds = new DataSet();
        dap.Fill(ds);
        foreach (DataRow dr in ds.Tables[0].Rows)
        {
        // 这是关键
          IndexBook(dr["id"].ToString(), dr["resTitle"].ToString(), dr["resContent"].ToString(), iw);
        }
    }
    iw.Optimize();   //优化索引
    iw.Close();      //关闭索引
}
private void IndexBook(string id, string title, string info, IndexWriter writer)
{
    try
    {
        Document doc = new Document();
        doc.Add(Field.Keyword("resTitle", title));
        doc.Add(Field.Text("resContent", info));
        doc.Add(Field.Text("resTitle", title));
        doc.Add(Field.Text("id", id));
        writer.AddDocument(doc);
    }
    catch (Exception ex)
    { }
}
```

16.2.2 Lucene.NET 技术多字段搜索

使用 MultiFieldQueryParser 类的 Parse 方法传入相关参数可以指定多个搜索字段，其中第二个参数必须为字符串数组，数组中存储多个字段。实现关键代码如下：

```
Query query = MultiFieldQueryParser.Parse("name*", new string[] { FieldName, FieldValue }, analyzer);
IndexReader reader = IndexReader.Open(directory);        //读取查询索引文件
IndexSearcher searcher = new IndexSearcher(reader);
Hits hits = searcher.Search(query);
```

16.2.3 Lucene.NET 技术多条件查询

除了使用 QueryParser.Parse 方法分解复杂的搜索语法外，还可以通过组合多个 Query 来达到多条件查询的目的：

```
Query query1 = new TermQuery(new Term(FieldValue, "name1"));                    // 词语搜索
Query query2 = new WildcardQuery(new Term(FieldName, "name*"));                 // 通配符
//Query query3 = new PrefixQuery(new Term(FieldName, "name1"));                 //字段搜索Field:Keyword，自动在结尾添加*
//Query query4 = new RangeQuery(new Term(FieldNumber, NumberTools.LongToString(11L)), new Term(FieldNumber,
```

```
NumberTools.LongToString(13L)), true);                    // 范围搜索
        //Query query5 = new FilteredQuery(query, filter);  // 带过滤条件的搜索
        BooleanQuery query = new BooleanQuery();
        query.Add(query1, BooleanClause.Occur.MUST);
        query.Add(query2, BooleanClause.Occur.MUST);
        IndexSearcher searcher = new IndexSearcher(reader);
        Hits hits = searcher.Search(query);
```

16.2.4 关键字分词技术

开发搜索引擎模块中分词功能是非常重要的，本章搜索引擎模块实现分词功能如图 16.2 所示，主要通过第 3 方动态链接库（DictSeg.dll）来实现，并且使用 DictSeg.dll 自带的中文词库。另外，在分词过程中，程序开发人员需要过滤掉标点符号以及轻声字（如吧、吗、么等），否则，这些标点符号和轻声字都将分词为一个词，并不是用户查询搜索的关键内容。

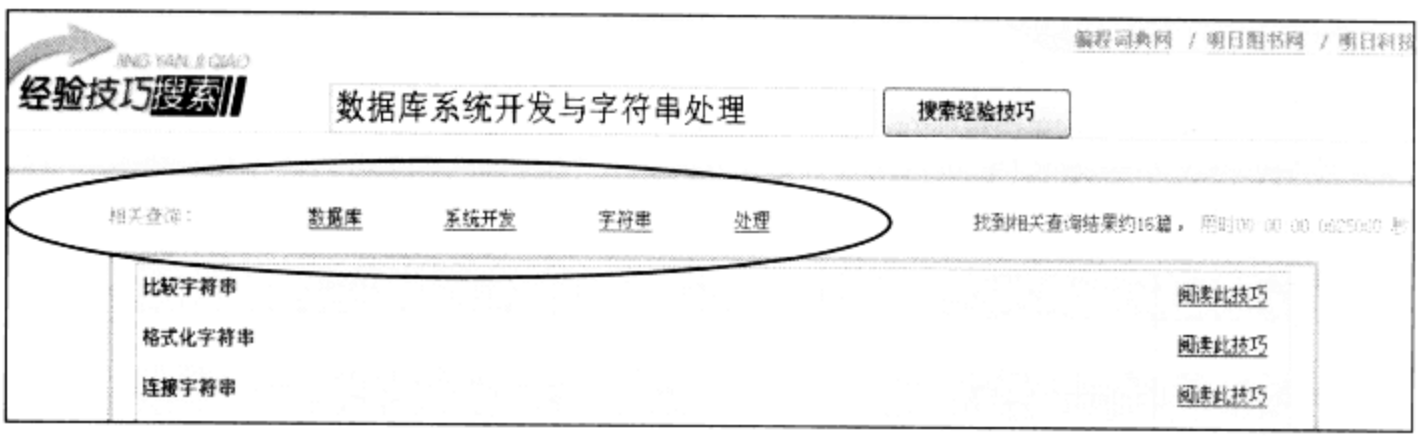


图 16.2 关键字分词

实现分词的核心代码如下。

例程 2 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
//分词
public static CSimpleDictSeg m_SimpleDictSeg;      //第三方分词对象
private static string[] strKey;                    //将关键分词，存储在数组中
private void fc(string keyWord)                    //自定义分词方法
{
    if (m_SimpleDictSeg == null)
    {
        try
        {
            m_SimpleDictSeg = new CSimpleDictSeg();
            m_SimpleDictSeg.DictPath = Server.MapPath("Data/");
            m_SimpleDictSeg.LoadDict();
        }
        catch (Exception e1)
        {
            m_SimpleDictSeg = null;
            Response.Write(String.Format("Load Dict Fail! ErrMsg:{0}", e1.Message));
            return;
        }
    }
    m_SimpleDictSeg.FilterStopWords = false;          //过滤停用词
    m_SimpleDictSeg.MatchName = true;                 //识别中文人名
    Stopwatch watch = new Stopwatch();
    watch.Start();
    //过滤字符和标点符号
    string key = keyWord;                             //赋值查询关键字
```

```
            key = key.Replace("的", " ");
            key = key.Replace("么", " ");
            key = key.Replace("吗", " ");
            key = key.Replace("与", " ");
            key = key.Replace('，', ' ');                    //将标点符号替换为空格
            key = key.Replace('。', ' ');
            key = key.Replace('！', ' ');
            key = key.Replace('-', ' ');
            key = key.Replace('—', ' ');
            key = key.Replace('：', ' ');
            key = key.Replace('；', ' ');
            key = key.Replace(':', ' ');
            key = key.Replace(';', ' ');
            key = key.Replace('*', ' ');
            key = key.Replace('.', ' ');
            key = key.Replace(',', ' ');
            key = key.Replace('“', ' ');
            key = key.Replace('’', ' ');
            key = key.Replace('‘', ' ');
            key = key.Replace('”', ' ');
            key = key.Replace('？', ' ');
            key = key.Replace('?', ' ');
            key = key.Replace("'", " ");
            key = key.Replace("'", ' ');
            key = key.Replace('(', ' ');
            key = key.Replace(')', ' ');
            key = key.Replace('）', ' ');
            key = key.Replace('（', ' ');
            key = key.Replace('_', ' ');
            key = key.Replace('、', ' ');
            key = key.Replace(" ", "");                      //最后再替换掉空格
            ArrayList words = m_SimpleDictSeg.Segment(key);  //输入分词
            watch.Stop();
            StringBuilder wordsString = new StringBuilder();
            foreach (String str in words)
            {
                wordsString.AppendFormat("{0}/", str);
            }
            key = wordsString.ToString();                    //去掉空格
            key = key.Substring(0, key.Length - 1);          //去掉最后一个 / 字符
            strKey = key.Split(new Char[] { '/', ' ' });
        }
```

16.2.5 高亮显示查询关键字

高亮显示查询关键字效果如图 16.3 所示，高亮显示查询关键字可以调用 Highlighter.Net.Dll 类库中提供的相关方法实现，也可以自定义编写方法实现。本程序实现高亮显示查询关键字通过自定义 HightLight 方法实现。

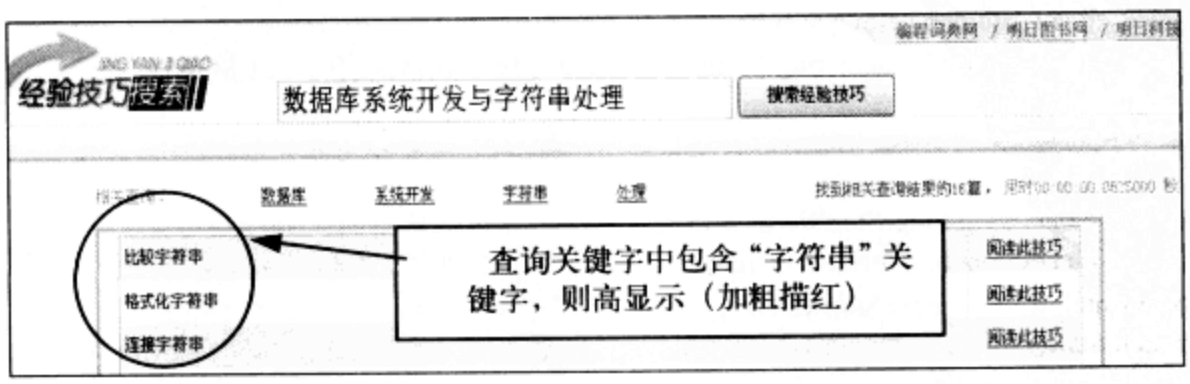


图 16.3 高亮显示查询关键字

实现高亮显示查询关键字代码如下。

例程 3 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
/// <summary>
/// 高量显示查询关键字
/// </summary>
/// <param name="title">关键字</param>
/// <param name="result">数据库中查询出的结果</param>
/// <returns></returns>
public string HightLight(string title, string result)
{
    if (strKey.Length > 0)              //strKey是分词结果 字符串数组
    {
        for (int k = 0; k < strKey.Length; k++)
        {
            //高亮显示设置（设置红色字体并且加粗）
            result = result.Replace(strKey[k], "<font color='red'><strong>" + strKey[k] + "</strong></font>");
        }
    }
    else
    {
        result = result.Replace(title, "<font color='red'><strong>" + title + "</strong></font>");
    }
    return result;
}
```

16.3 搜索引擎主页设计

当用户在搜索引擎主页输入关键字到文本框中，然后单击“搜索经验技巧”按钮，搜索关键字将提交到执行搜索引擎功能页面（so.aspx）。搜索引擎网站主页如图 16.4 所示。

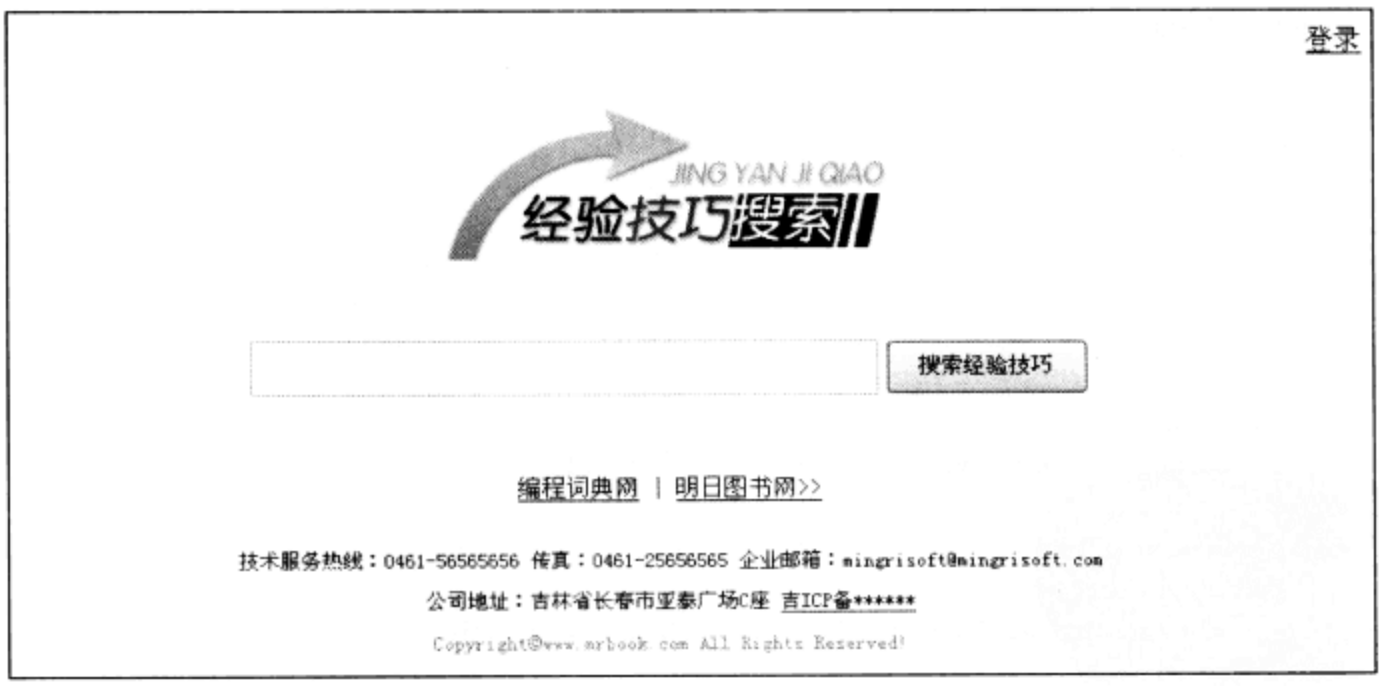


图 16.4 搜索引擎网站主页

搜索引擎主页主要通过 DIV+CSS 布局实现，起关键核心作用的代码如下。

例程 4 代码位置：光盘\mr\16\SearchEngine\Default.htm

```
<form action="so.aspx" name="f" target="_blank">
    <input id="q" name="key" style="width: 356px; font-size: 16pt; height: 27px;" type="text" />
    <input id="so" type="submit" value="搜索经验技巧" style="height: 33px" />
</form>
```

搜索引擎主页完整代码如下。

例程 5 代码位置：光盘\mr\16\SearchEngine\Default.htm

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>搜索引擎网站</title>
    <meta http-equiv="Content-Type" content="text/html; charset=gb2312">
    <style type="text/css">
<!--
#Layer1 {
}
-->
</style>
    <link href="css/css.css" rel="stylesheet" type="text/css" />
    <style type="text/css">
<!--
#Layer2 {
}
#kw {      FONT: 16px Verdana; PADDING-TOP: 2px; HEIGHT: 1.78em
}
#sb { WIDTH: 5.6em; HEIGHT: 2em
}
#sb { FONT-SIZE: 14px
}
a:link {
     color: #0000CC;
}
a:visited {
     color: #0000CC;
}
a:hover {
     color: #FF6600;
}
a:active {
     color: #FF6600;
}
-->
</style>
</head>
<body>
    <div id="denglu">
        <div align="right">
            <a target="_blank" href="logon.aspx">登录</a></div>
    </div>
    <div class="logo00" id="logo">
        <div align="center">
            <img src="images/logo.gif" width="179" height="98" /></div>
    </div>
    <div class="huizi" id="biaodan">
        <div align="center">
            <span class="biaodan">最新最全的经验技巧等待您搜索....</span>    
   
                               
   
            </div>
        </div>
        <div class="biaodan" id="wenben">
```

```
            <div align="center">
                <form action="so.aspx" name="f" target="_blank">
                    <input id="q" name="key" style="width: 356px; font-size: 16pt; height: 27px;" type="text" />
                    <input id="so" type="submit" value="搜索经验技巧" style="height: 33px" />
                </form>
            </div>
        </div>
        <p align="center">
            <span class="shenlan14">
                <br />
                编程词典网| 明日图书网&gt;&gt;</span></p>
        <p align="center" class="shenlan">
            技术服务热线：-84978981 传真：-84978982 企业邮箱：mingrisoft@mingrisoft.com<br />
            公司地址：吉林省长春市亚泰广场C座 吉ICP备122345678<br />
            <span class="qianlan">Copyright&copy;www.mrbook.com All Rights Reserved!</span></p>
    </body>
    </html>
```

16.4 搜索引擎结果页设计

本节主要讲解创建搜索引擎索引文档和实现搜索引擎。其中创建索引文档主要是将数据库中要实现查询的数据通过 Lucene.net 技术创建搜索引擎索引文档。搜索引擎结果页中主要实现查询关键字分词功能、关键字高亮显示及使用 Lucene.NET 技术查询索引文档，共界面如图 16.5 所示。

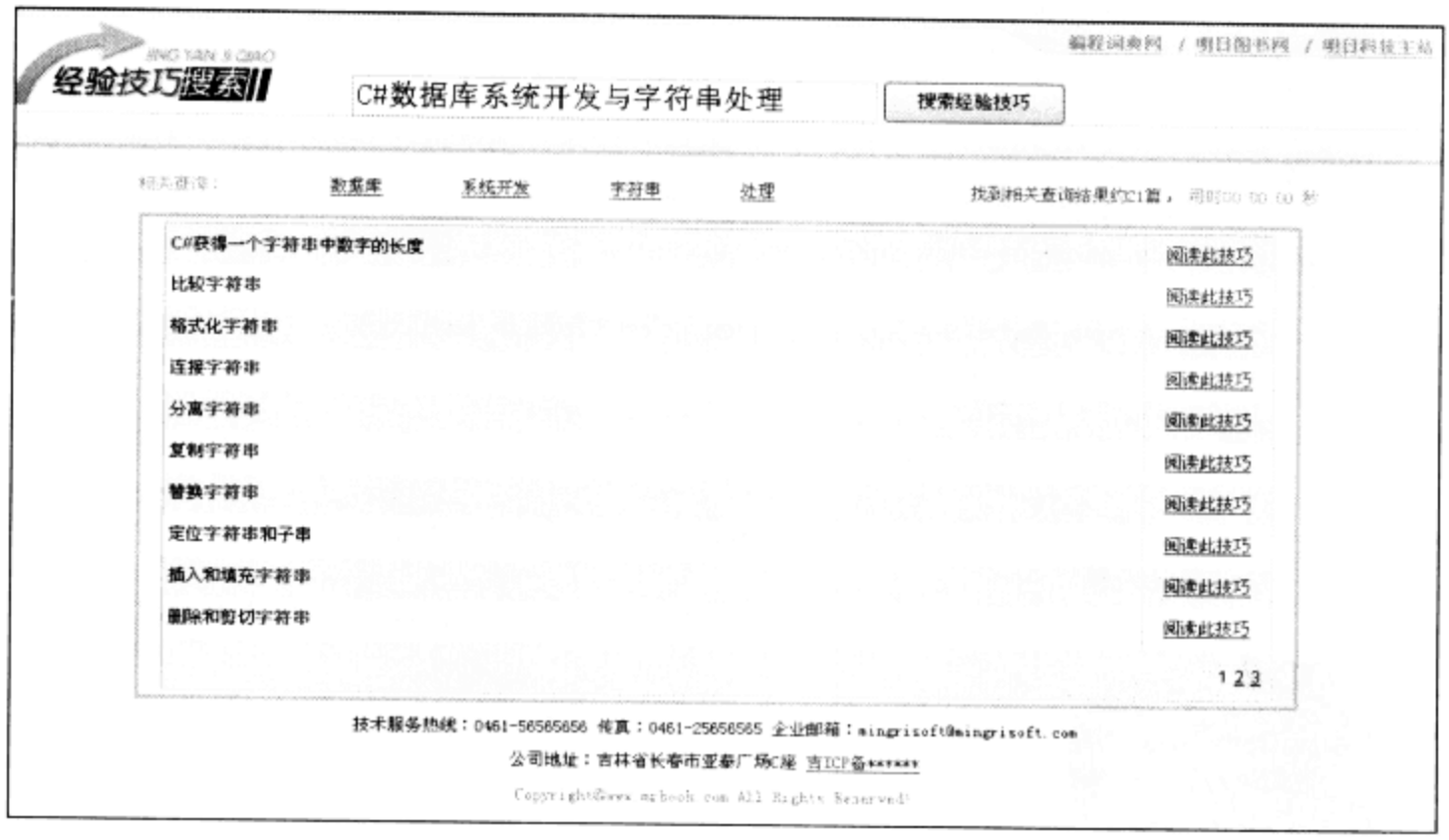


图 16.5 搜索引擎结果

16.4.1 创建索引文档

创建的索引文档可以把他看作是一个数据库，因为它本身存储要查询的数据，并且还可以不断地向索引文档中添加数据。实现 lucene.net 搜索引擎技术，创建索引是其中核心技术之一。索引文件一共包含 3 个，索引文件扩展名是自定义文件格式，如图 16.6 所示。

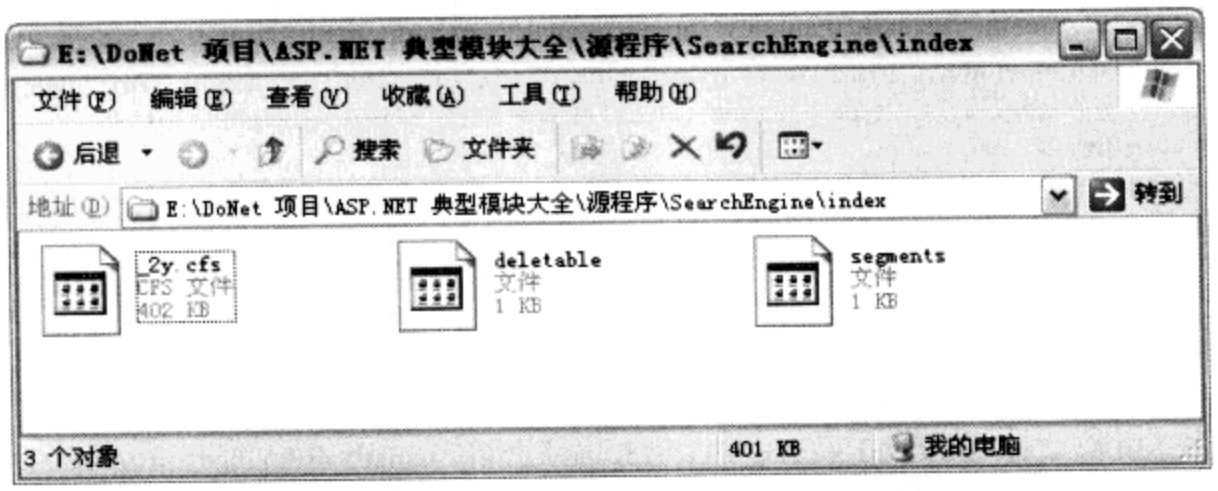


图 16.6　索引文件

下面代码主要实现创建索引文件功能，首先，将数据库中要查询的数据读取出来，然后调用自定义 IndexBook 方法，将从数据库中读取的数据添加到索引文件中。实现创建搜索引擎的索引文件完整代码如下。

例程 6　代码位置：光盘\mr\16\SearchEngine\resAdd.aspx.cs

```
//创建查询索引
protected void imgBtnIndex_Click(object sender, ImageClickEventArgs e)
{
    //建立搜索分析对象
    Lucene.Net.Analysis.Standard.StandardAnalyzer a = new Lucene.Net.Analysis.Standard.StandardAnalyzer();
    //指定索引文档创建位置
  //Server.MapPath("Index")为索引文件存放地址
    IndexWriter iw = new IndexWriter(Server.MapPath("Index"), a, true);
    //数据库连接字符串
    string conn = "Server=.;DataBase=db_res;Uid=sa;pwd=;";
    //获取要查询的数据表中数据，然后添加索引文档中
    using (SqlConnection con = new SqlConnection(conn))
    {
        SqlDataAdapter dap = new SqlDataAdapter("select * from tb_res", con);
        DataSet ds = new DataSet();
        dap.Fill(ds);
        foreach (DataRow dr in ds.Tables[0].Rows)
        {
            //调用自定义方法IndexBook，将数据库中查询的值添加到索引文档中
          //这是关键
            IndexBook(dr["id"].ToString(), dr["resTitle"].ToString(), dr["resContent"].ToString(), iw);
        }
    }
    iw.Optimize();   //优化索引
    iw.Close();      //关闭索引
}
```

自定义 IndexBook 方法主要用于创建索引字段，将查询的数据添加到字段中。实现代码如下。

例程 7　代码位置：光盘\mr\16\SearchEngine\resAdd.aspx.cs

```
/// <summary>
/// 将查询数据添加到索引文档中
/// </summary>
/// <param name="id">字段ID值</param>
/// <param name="title">字段标题值</param>
/// <param name="info">字段内容值</param>
```

```
/// <param name="writer">写入索引对象</param>
private void IndexBook(string id, string title, string info, IndexWriter writer)
{
    try
    {
        Document doc = new Document();                    //创建文档对象
        doc.Add(Field.Keyword("resTitle", title));          //添加字段值
        doc.Add(Field.Text("resContent", info));
        doc.Add(Field.Text("resTitle", title));
        doc.Add(Field.Text("id", id));
        writer.AddDocument(doc);                          //写入到索引文档中
    }
    catch
    {
        throw;                                            //抛出异常
    }
}
```

16.4.2 实现搜索引擎

使用 Lucene.net 技术实现搜索引擎查询的结果执行流程如下。

- 查询结果将与查询关键字完全匹配的内容显示在最前方。
- 如果查询内容没有与关键字完全匹配，那么查询结果按与查询关键字最匹配的内容显示在最前方。
- 如果关键字不与查询内容不匹配，则对关键字进行分词，然后再查询。

实现搜索引擎是在 so.aspx 页面中完成的，在 so.aspx 页面中主要添加控件及控件说明，如表 16.2 所示。

表 16.2　　主要添加控件及控件说明

控　件	数　量	主要属性设置	用　途
TextBox	1	ID 属性为 txtKey	输入查询关键字
Button	1	ID 属性为 btnSo	执行搜索功能
		Text 属性为“搜索经验技巧”	
Label	1	ID 属性为 lblFC	显示关键字分词，并且可以进行查询
GridView	1	ShowHeader 属性为 false，不显示 GridView 表格表头	显示查询结果
		添加可用字段 BoundField，用来显示查询结果标题，DataField 属性为 resTitle	
		添加可用字段 HyperLinkField，DataNavigateUrlFields 属性值为 id，DataNavigateUrlFormatString 属性值为~/resDetail.aspx?id={0}，Text 属性值为“阅读此技巧”，Target 属性为_blank	

注　意　在设置 GridView 的可用字段时，需要在 GridView 属性对话框中选择 Columns 属性，单击“...”按钮，弹出“字段”对话框，如图 16.7 所示，在该“字段”对话框，中设置可用字段。

在搜索引擎结果页 Page_Load 事件中调用自定义 BindGridView 方法，执行搜索引擎并将查询结果显示到 GridView 中。实现代码如下。

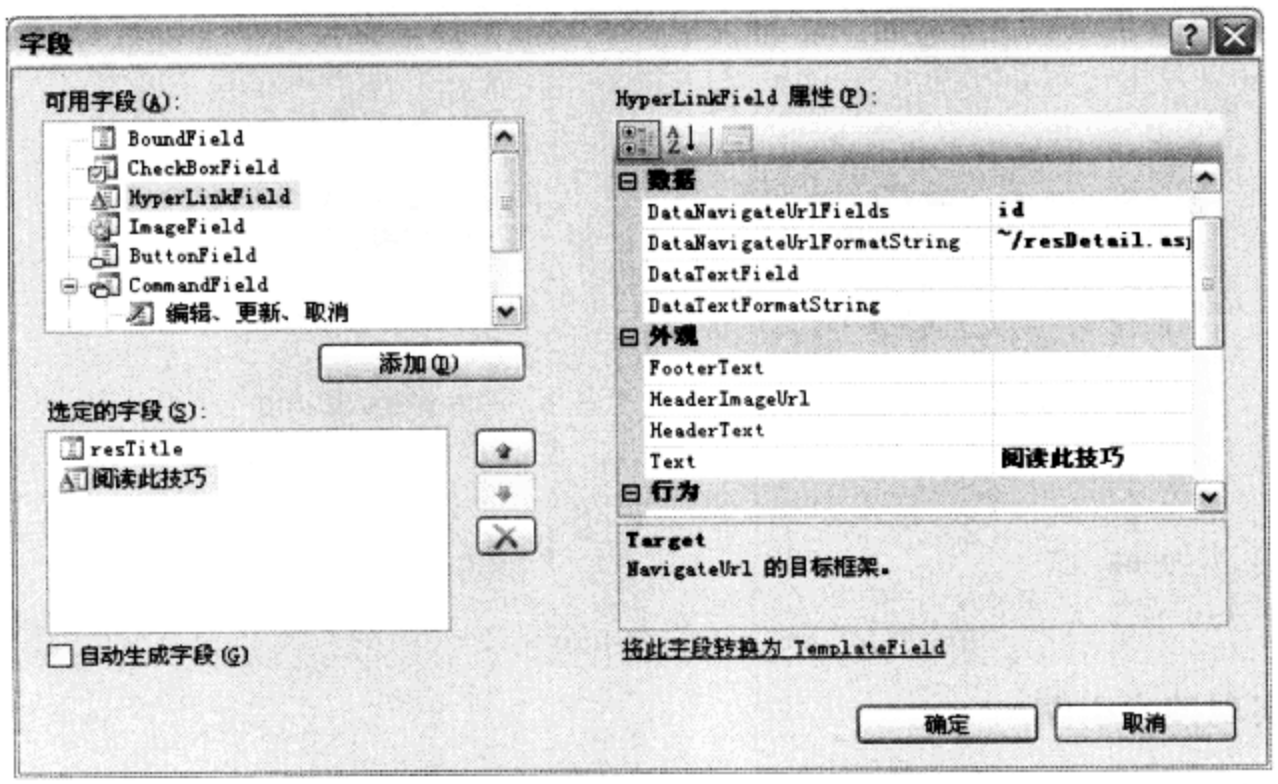


图 16.7　添加可用字段

例程 8　代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        this.BindGridView();
    }
}
```

单击“搜索经验技巧”按钮，将查询的关键字进行编码，然后通过地址栏传值到搜索引擎页（这里指的是本页）。由于搜索引擎主页使用的该种方法传递查询关键字，因此在此也使用同样方法，否则页面刷新时，将查询显示在地址栏中的关键字。实现代码如下。

例程 9　代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
protected void btnSo_Click(object sender, EventArgs e)
{
    //将查询关键字 以地址参数传值的形式 传入查询页（这里是本页）
    Response.Redirect("so.aspx?key=" + Server.UrlEncode(txtKey.Text));
}
```

自定义 BindGridView 方法主要实现将获取地址栏传值（关键字）显示在关键字输入文本框中；设置查询结果页标题；调用自定义 FindResult 方法获取查询结果并显示在 GridView 中。自定义 BindGridView 方法实现代码如下。

例程 10　代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
private void BindGridView()
{
    //获取地址栏传值，并为其解码
    string keyWord = Server.UrlDecode(Request.QueryString["key"]);
    //将查询关键字，显示在查询文本框中
    txtKey.Text = keyWord;
    //设置查询结果页标题
    this.Title = txtKey.Text + "——Asp.net经验技巧搜索";
    //调用自定义FindResult方法，将查询结果显示在GridView控件中
    GridView1.DataSource = this.FindResult(keyWord);
    GridView1.DataKeyNames = new string[] { "id" };
    GridView1.DataBind();
}
```

自定义 setTable 方法主要初始化数据源结构（这里是 DataTable 类对象 Results）。自定义 setTable 方法实现代码如下。

例程 11 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
//将存储查询结果的数据源  DataTable对象
DataTable Results = new DataTable();
//初始化数据源  DataTable对象，将查询结果保存到数据源对象中
private void setTable()
{
    //创建结果记录集
    //定义字段
    this.Results.Columns.Add("resTitle", typeof(string));
    this.Results.Columns.Add("resContent", typeof(string));
    this.Results.Columns.Add("id", typeof(string));
}
```

自定义 FindResult 方法主要实现核心查询技术，并且返回查询结果 DataTable 数据源。自定义 FindResult 方法不做详细介绍，下面代码给出了详细的代码注释。实现代码如下。

例程 12 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
//执行查询
private DataTable FindResult(string keyWord)
{
    //调用自定义方法，初始化数据源对象
    this.setTable();                                          //创建查询结果存储表
    this.fc(keyWord);                                         //关键字分词
    DateTime dt = DateTime.Now;                               //开始记时
    IndexSearcher search = new IndexSearcher(Server.MapPath("Index")); //把刚才建立的索引取出来
    string[] sField = new string[strKey.Length];              //初始化长度
    sField.SetValue("resTitle", 0);
    for (int s = 1; s < strKey.Length; s++)
    {
        sField.SetValue("resTitle", s);                       //创建将要查询resTitle字段
    }
    lblFC.Text = "";                                          //清空分词结果
    //显示分词结果
    foreach (string k in strKey)
    {
        //显示分词结果在2个以上的词，并设置超级链接
        if (k.Length > 1)
            lblFC.Text = lblFC.Text + "        " + "<A
target='black' href='http://donet/so.aspx?key=" + k + "'>" + k + "</A>";
    }
    //执行查询
    Query qq = MultiFieldQueryParser.Parse(strKey,sField,newLucene.Net.Analysis.Standard.StandardAnal
yzer ());
    BooleanQuery q = new BooleanQuery();
    //q.Add(qq, true, false);
    q.Add(qq, BooleanClause.Occur.MUST);
    Hits hit = search.Search(q);                              //获取查询结果
    TimeSpan ts = System.DateTime.Now.Subtract(dt);   //计算用时
    for (int i = 0; i <= hit.Length() - 1; i++)
    {
        Document doc = hit.Doc(i);
        DataRow row = Results.NewRow();
        //添加到DataRow中
        row["resTitle"] = this.HightLight(Server.UrlDecode(Request.QueryString["key"]), doc.Get("resTitle"));
//显示高亮描红查询关键字
        row["resContent"] = doc.Get("resContent");
        row["id"] = doc.Get("id");
        //将结果行添加到数据源DataTable对象中
        this.Results.Rows.Add(row);
```

```
        }
        search.Close();
        lblSum.Text = "<font color=red>共找到" + hit.Length() + "条记录</font> -----" + ts.ToString() + " 毫秒  ";
        return Results;
    }
```

自定义 fc 方法主要实现关键字分词功能，这里实现分词功能主要是通过第 3 方组件（DictSeg.dll）实现的。在分词过程中，过滤标点符号和轻声字（吧、吗、么等）非常重要，因为标点符号和轻声字都被看作是一个词。自定义 fc 方法实现代码如下。

例程 13 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
//分词
public static CSimpleDictSeg m_SimpleDictSeg;                    //第三方分词对象
private static string[] strKey;                                  //将关键分词，存储在数组中
private void fc(string keyWord)                                  //自定义分词方法
{
    if (m_SimpleDictSeg == null)
    {
        try
        {
            m_SimpleDictSeg = new CSimpleDictSeg();
            m_SimpleDictSeg.DictPath = Server.MapPath("Data/");          //指定分词词库位置
            m_SimpleDictSeg.LoadDict();                                  //装载字库到内存
        }
        catch (Exception e1)
        {
            m_SimpleDictSeg = null;
            Response.Write(String.Format("Load Dict Fail! ErrMsg:{0}", e1.Message));
            return;
        }
    }
}
m_SimpleDictSeg.FilterStopWords = false;      //过滤停用词
m_SimpleDictSeg.MatchName = true;             //识别中文人名
Stopwatch watch = new Stopwatch();
watch.Start();
//过滤字符和标点符号
string key = keyWord;                         //赋值查询关键字
key = key.Replace("的", " ");
key = key.Replace("么", " ");
key = key.Replace("吗", " ");
key = key.Replace("与", " ");
key = key.Replace('，', ' ');                 //将标点符号替换为空格
key = key.Replace('。', ' ');
key = key.Replace('！', ' ');
key = key.Replace('-', ' ');
key = key.Replace('—', ' ');
key = key.Replace('：', ' ');
key = key.Replace('；', ' ');
key = key.Replace(':', ' ');
key = key.Replace(';', ' ');
key = key.Replace('*', ' ');
key = key.Replace('.', ' ');
key = key.Replace(',', ' ');
key = key.Replace('“', ' ');
```

```
        key = key.Replace("'", ' ');
        key = key.Replace("""", ' ');
        key = key.Replace("“", ' ');
        key = key.Replace('？ ', ' ');
        key = key.Replace('?', ' ');
        key = key.Replace("""", " ");
        key = key.Replace("‘", ' ');
        key = key.Replace('(', ' ');
        key = key.Replace(')', ' ');
        key = key.Replace('） ', ' ');
        key = key.Replace(' （', ' ');
        key = key.Replace('_', ' ');
        key = key.Replace('、 ', ' ');
        key = key.Replace(" ", "");                          //最后再替换掉空格
        ArrayList words = m_SimpleDictSeg.Segment(key);      //输入分词
        watch.Stop();
        StringBuilder wordsString = new StringBuilder();
        foreach (String str in words)
        {
            wordsString.AppendFormat("{0}/", str);
        }
        key = wordsString.ToString();                        //去掉空格
        key = key.Substring(0, key.Length - 1);              //去掉最后一个 / 字符
        strKey = key.Split(new Char[] { '/', ' ' });
    }
```

GridView 控件的 GridView1_PageIndex 事件主要实现查询结果分页功能，实现代码如下。

例程 14 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    //查询结果分页
    GridView1.PageIndex = e.NewPageIndex;
    this.BindGridView();
}
```

自定义 HightLight 方法主要实现关键高亮显示（这里设置字天变红和加粗）。自定义 HightLight 方法实现代码如下。

例程 15 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
/// <summary>
/// 高量显示查询关键字
/// </summary>
/// <param name="title">关键字</param>
/// <param name="result">数据库中查询出的结果</param>
/// <returns></returns>
public string HightLight(string title, string result)
{
    if (strKey.Length > 0)              //strKey是分词结果 字符串数组
    {
        for (int k = 0; k < strKey.Length; k++)
        {
            //高亮显示设置（设置红色字体并且加粗）
            result = result.Replace(strKey[k], "<font color='red'><strong>" + strKey[k] + "</strong></font>");
        }
```

```
        }
        else
        {
            result = result.Replace(title, "<font color='red'><strong>" + title + "</strong></font>");
        }
        return result;
    }
```

16.5 程序调试与错误处理

在实现关键分词的初期，通常程序开发人员不会考虑到标点符号和轻声字也算一个词（例如，“，”的、吗等），所以搜索出来的结果与关键字不符合（搜索结果中包含逗号或语气词）。

综合上述，程序开发人员需要将查询关键字中的标点符号和轻声字过滤掉。主要实现代码如下。

例程 16 代码位置：光盘\mr\16\SearchEngine\so.aspx.cs

```
//过滤字符和标点符号
string key = keyWord;                                    //赋值查询关键字
key = key.Replace("的", " ");
key = key.Replace("么", " ");
key = key.Replace("吗", " ");
key = key.Replace("与", " ");
key = key.Replace('，', ' ');                            //将标点符号替换为空格
key = key.Replace('。', ' ');
key = key.Replace('！', ' ');
key = key.Replace('-', ' ');
key = key.Replace('—', ' ');
key = key.Replace('：', ' ');
key = key.Replace('；', ' ');
key = key.Replace(':', ' ');
key = key.Replace(';', ' ');
key = key.Replace('*', ' ');
key = key.Replace('.', ' ');
key = key.Replace(',', ' ');
key = key.Replace('“', ' ');
key = key.Replace('’', ' ');
key = key.Replace('‘', ' ');
key = key.Replace('”', ' ');
key = key.Replace('？', ' ');
key = key.Replace('?', ' ');
key = key.Replace("'", " ");
key = key.Replace("", ' ');
key = key.Replace('(', ' ');
key = key.Replace(')', ' ');
key = key.Replace('）', ' ');
key = key.Replace('（', ' ');
key = key.Replace('_', ' ');
key = key.Replace('、', ' ');
key = key.Replace(" ", "");                              //最后再替换掉空格
```

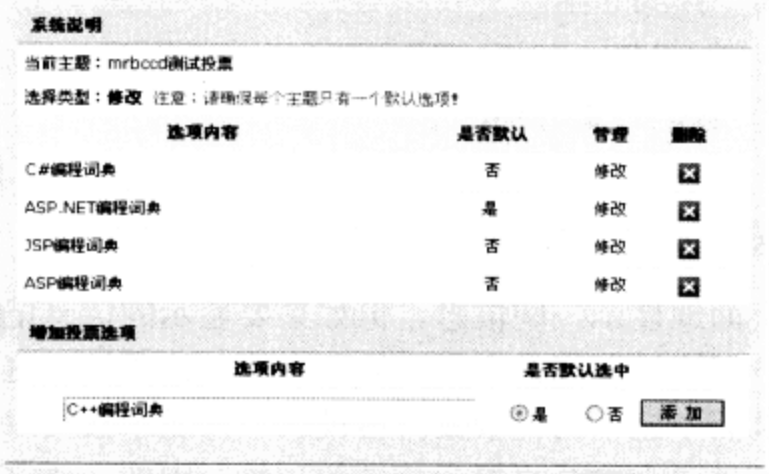
网上问卷调查模块

第 17 章

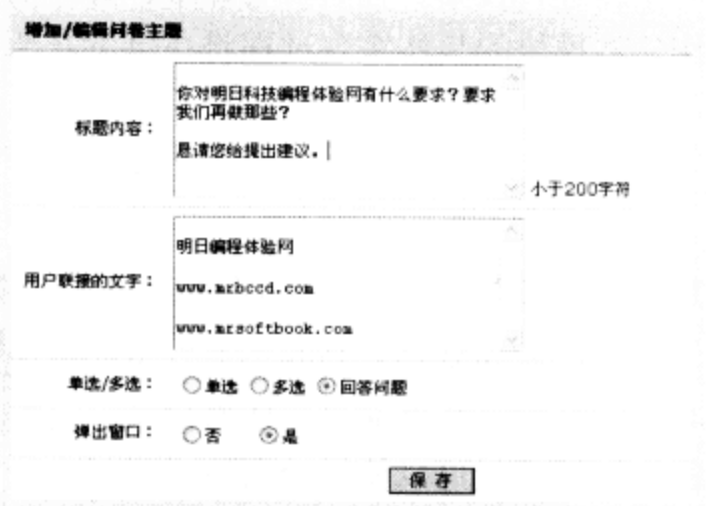
实例位置：光盘\mr\17\

问卷调查，就是根据调查目的，制定调查问卷，由被调查者按调查问卷所提的问题和给定的选择答案进行回答的一种专项调查形式。问卷调查是一种常用的专项调查手段，是国际通行的一种专项调查形式，也是我国近年来进行专项调查的一种主要形式。通过本章的学习，读者能够学到以下内容。

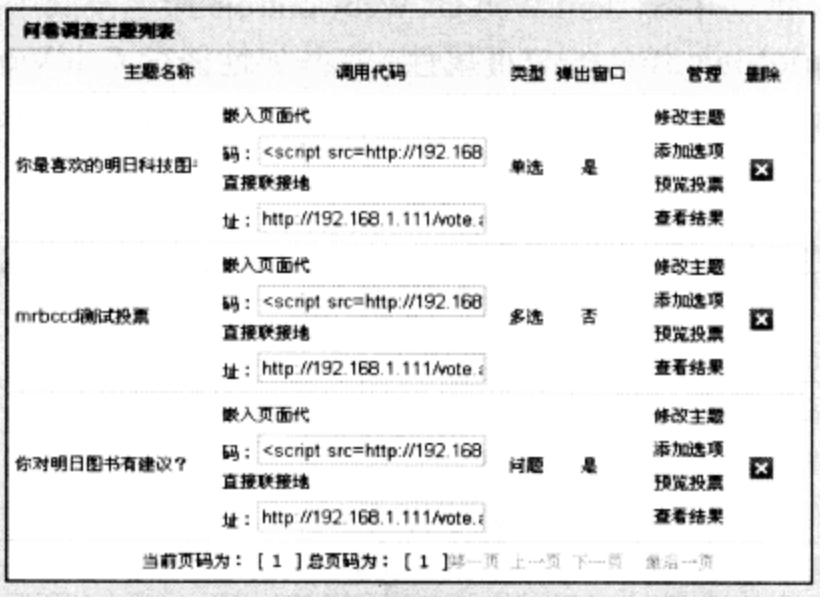
问卷主题选项的添加



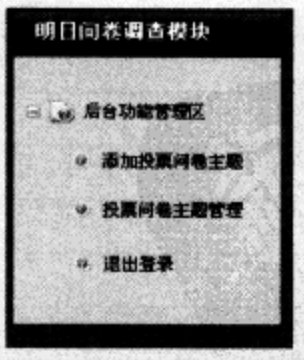
添加/编辑问卷主题



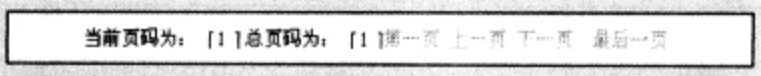
问卷调查主题管理



TreeView 控件绑定 XML 实现导航



Repeater 控件分页



17.1 网上问卷调查模块概述

首先来了解一下网上问卷调查的特点。

(1) 通俗易懂，实施方便。

采用问卷形式进行专项调查，由于将调查的问题和可供选择的答案均提供给被调查者，由其从中选择，因此，容易被调查者所接受。

(2) 适用范围广。

问卷调查既适用于对社会政治经济现象进行专项调查，也适用于对社会广大群众关心的问题进行专项调查，还适用于对其他关心的问题进行专项调查。

(3) 节省调查时间，提高调查效率。

由于在调查问卷中已将调查目的、内容和问题及可供选择的答案均列出，因此，除特殊情况外，无须再详细说明，只需由调查者进行选择回答即可。从而，节省了时间，加快了调查进度。

本网上问卷调查模块问卷题型分为 3 种，即单选、多选和问答，用户可根据自己所感兴趣的主题进行投票。在后台管理中，管理员可对问卷投票主题进行添加、编辑、删除和预览等操作。本模块的具体实现过程将在以下几节中给予详细介绍。

17.2 关 键 技 术

17.2.1 使用 Repeater 控件分页显示数据

在开发网站时，常常需要在页面中比较全面地显示一些信息，但如果要显示的信息记录较多，此时用一个页面显示所有记录，可能会给用户的浏览带来很大的不方便。为了解决这个问题，开发人员可以使用分页技术来限定一个页面中显示的记录数。本模块中在显示数据库中信息时主要应用 Repeater 控件。由于 Repeater 控件没有与分页显示相关的属性，因此，要实现分页必须使用其他方式。

本模块在实现 Repeater 控件的分页时，借助了 System.Web.UI.WebControls 命名空间中提供的 PagedDataSource 来实现。该类封装了 GridView 控件的分页属性，这些属性使得 GridView 控件可以实现分页，因此在这里也可以使用该类实现 Repeater 控件的分页操作。PageDataSource 类中的常用属性如表 17.1 所示。

表 17.1　PageDataSource 类中的常用属性

名　称	说　明
AllowPaging	获取是否启用分页
AllowCustomPaging	获取或设置是否启用自定义分页
CurrentPageIndex	获取或设置当前显示页的索引
DataSource	获取或设置用于填充控件中项的数据源
PageSize	获取或设置要在 GridView 控件的每页显示的项数
PageCount	获取显示 GridView 控件中各项所需的总页数
FirstIndexPage	获取页中的第一个索引
IsFirstPage	获取一个值，该值指示当前显示的页是否为首页
IsLastPage	获取一个值，该值指示当前显示的页是否为最后一页

以显示用户信息列表页 MemberList.aspx 为例具体讲解如何使用 Repeater 控件分页显示数

据。实际运行效果如图 17.1 所示。

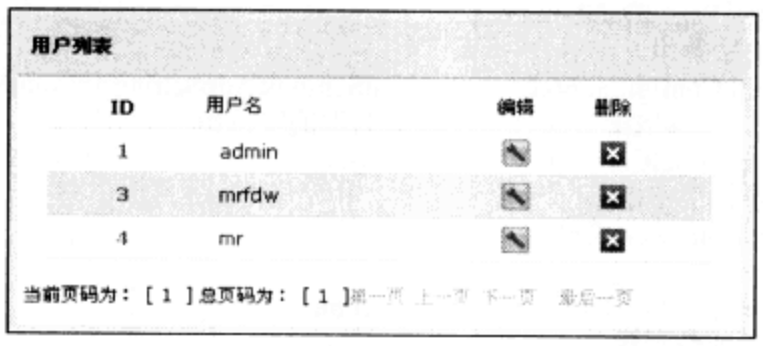


图 17.1　使用 Repeater 控件分页显示数据

实现过程如下。

首先，编辑 Repeater 控件的模板，需要在 HTML 源视图中通过代码进行编辑。主要在该模板中添加了 LinkButton 控件来绑定数据库中用户的信息。关键代码如下。

例程 1　代码位置：光盘\mr\17\OnlineSurvey\admin\MemberList.aspx

```
<asp:repeater id="lstMember" Runat="server">
<ItemTemplate>
<tr bgColor="#f0f0f0" onmouseover="this.style.backgroundColor='#D0E8FF';" onmouseout="this.style. backgroundColor='#ffffff';">
<td height="28" align="center"><strong><font color="#FF0000">
<%# DataBinder.Eval(Container, "DataItem.MemberID") %>
</font></strong>
</td><td height="28"> <a href='MemberEdit.aspx?pid=<%# DataBinder.Eval(Container, "DataItem.MemberID") %>' ><%# DataBinder.Eval(Container, "DataItem.MemberName") %>
</a></td>
<td height="28" align="center"><a  href='MemberEdit.aspx?id=<%# DataBinder.Eval(Container, "DataItem.MemberID") %>' ><img src='images/edit_x.gif' border="0"></a></td>
<td height="28" align="center">
<asp:linkbutton runat="server" Text="<img src='images/cancel_x.gif' border=0>" CommandName="Delete" CausesValidation="false" OnLoad="Delete_Load" CommandArgument='<%# DataBinder.Eval(Container, "DataItem.MemberID") %>' ID="Linkbutton2" NAME="Linkbutton2">
</asp:linkbutton></td>
</tr><tr><td height="1" colspan="4" bgcolor="#D4D0C8"></td>
</tr>
</ItemTemplate>
</asp:repeater>
```

接着编写后台主要代码。在 MemberList.aspx 页的 Page_Load 事件中，调用用户自定义 BindData()对 Repeater 控件进行数据绑定并分页。代码如下。

例程 2　代码位置：光盘\mr\17\OnlineSurvey\admin\MemberList.aspx.cs

```
protected void Page_Load(object sender, System.EventArgs e)
{
    if (!IsPostBack)
    {
        // 调用自定义方法BindData()，用于分页显示Repeater控件中的数据
        BindData();
    }
}
```

上述 Page_Load 事件代码中调用了自定义方法 BindData。BindData 方法为自定义的无返回类型方法，该方法主要用来从数据库中查询出符合指定条件的记录，并绑定在 Repeater 控件中，然后通过设置 PageDataSource 类对象的 AllowPaging 属性为 True，来实现 Repeater 控件的分页功能。BindData()方法关键代码如下。

例程 3　代码位置：光盘\mr\17\OnlineSurvey\admin\MemberList.aspx.cs

```
private void BindData()
{
  //定义一个整型变量用来保存当前页的索引
```

```
int curpage = Convert.ToInt32(this.labPage.Text);
//创建分页类PagedDataSource的一个实现ps
PagedDataSource ps = new PagedDataSource();
//定义一个连接数据库的字符串
string connection = System.Configuration.ConfigurationSettings.AppSettings["ConnString"].ToString();
 //创建连接数据库对象，并读取配置文件中连接数据库的字符串
   SqlConnection conn = new SqlConnection(connection);
   //定义查询数据库记录的字符串
   string sqlstr = "select * from Member";
   //创建数据库适配器
   SqlDataAdapter myda = new SqlDataAdapter(sqlstr, conn);
   //创建一个DataSet对象的数据集
   DataSet ds = new DataSet();
   //打开数据库连接
   conn.Open();
   //应用数据适配器的Fill方法填充数据集
   myda.Fill(ds, "Member");
   //设定数据源
   ps.DataSource = ds.Tables["Member"].DefaultView;
   //设定分页
   ps.AllowPaging = true;
   //显示的数量
   ps.PageSize = 6;
   //取得当前页的页码
   ps.CurrentPageIndex = curpage - 1;
   this.lnkbtnUp.Enabled = true;
   this.lnkbtnNext.Enabled = true;
   this.lnkbtnBack.Enabled = true;
   this.lnkbtnOne.Enabled = true;
   if (curpage == 1)
   {
     //不显示第一页按钮
     this.lnkbtnOne.Enabled = false;
              //不显示上一页按钮
              this.lnkbtnUp.Enabled = false;
          }
          if (curpage == ps.PageCount)
          {
              //不显示下一页
              this.lnkbtnNext.Enabled = false;
              //不显示最后一页
              this.lnkbtnBack.Enabled = false;
          }
          //计算总页数据，并显示出来
          this.labBackPage.Text = Convert.ToString(ps.PageCount);
          //设定lstMember的数据源为Pages类
          lstMember.DataSource = ps;
          //将数据绑定到lstMember控件中
          lstMember.DataBind();
    }
```

当用户单击用于操作分页的 LindButton 控件时，程序会根据当前页码执行指定操作。用于分页的 LinkButton 控件的 Click 事件代码如下。

例程 4 代码位置：光盘\mr\17\OnlineSurvey\admin\MemberList.aspx.cs

```
protected void lnkbtnOne_Click(object sender, EventArgs e)
    {
        this.labPage.Text = "1";
        this.BindData();
    }
    protected void lnkbtnUp_Click(object sender, EventArgs e)
    {
        this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) - 1);
        this.BindData();
```

```
    }
    protected void lnkbtnNext_Click(object sender, EventArgs e)
    {
        this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) + 1);
        this.BindData();
    }
    protected void lnkbtnBack_Click(object sender, EventArgs e)
    {
        this.labPage.Text = this.labBackPage.Text;
        this.BindData();
    }
protected void Delete_Load(object sender, System.EventArgs e)
{
    ((LinkButton)sender).Attributes["onclick"]="return confirm('您确定要删除这个会员吗？');";
}
```

17.2.2 应用 ViewState 保存状态信息

在 ASP 开发中，要实现多页面回传时保持原来的页状态必须编写大量代码，但是在 ASP.NET 中通过启用页面或控件的 ViewState 属性就可以轻松实现保持其状态的功能。在编写该问卷调查模块后台代码中如在保存问卷调查主题编号时就应用了 ViewState 来保存主题编号信息。

1. 启用 ViewState

ViewState 是 ASP.NET 中用来保存 Web 控件回传时状态值的一种机制。在 Web 窗体（FORM）的设置为 runat="server"，这个窗体（FORM）会被附加一个隐藏的属性_VIEWSTATE。_VIEWSTATE 中存放了所有控件在 ViewState 中的状态值。例如，用户在 TextBox 控件中输入的内容默认情况下就保存在 ViewState 中，在后续的回传中，可以自动将 ViewState 中保存信息取出并填充到控件。因此，除了可以减少繁琐的工作和代码外，ViewState 通常还可以减少数据库的往返次数。

ViewState 是类 Control 中的一个域，其他所有控件通过继承 Control 来获得 ViewState 功能。它的类型是 system.Web.UI.StateBag，一个名称/值的对象集合。

使用 ViewState 的条件：如果要使用 ViewState，则在 ASPX 页面中必须有一个服务器端窗体标记（<form runat=server>）。窗体字段是必需的，这样包含 ViewState 信息的隐藏字段才能回传给服务器。而且，该窗体还必须是服务器端的窗体，这样在服务器上执行该页面时，ASP.NET 页面框架才能添加隐藏的字段。除此之外，还需分别设置 Page 的 EnableViewState 属性值为 True 和控件的 EnableViewState 属性值为 True。

2. 禁用 ViewState

默认情况下页面和控件的 ViewState 是被启用的。使用 ViewState 会明显增加发送到浏览器的页面的大小，因此如果不需要使用 ViewState，最好将其关闭。

ViewState 有 4 种级别：控件级、页面级、应用程序级和全局配置。每种级别关闭的方式如下。

- 控件级：将控件的 EnableViewState 属性值设为 False。
- 页面级：在<%@ Page%>指令中添加“EnableViewState= false”。
- 应用程序级：在 Web 配置文件中添加如下代码：

```
<configuration>
 <system.web>
   <pages enableViewState="false"/>
 <system.web>
<configuration>
```

- 全局配置：修改 machine.config 文件，将<pages/>标签的 enableViewState 属性设置为 False。

上述 4 种设置的优先级由高到低为控件级、页面级、应用程序级及全局配置。

17.2.3 TreeView 控件绑定 XML 数据

TreeView 控件有着对数据源控件良好的绑定支持。TreeView 控件可分别与 SiteMap DataSource 控件和 XMLDataSource 控件进行数据绑定来实现站点导航功能。本模块中主要是与 XMLDataSource 控件结合应用来实现后台导航功能。实际运行效果如图 17.2 所示。

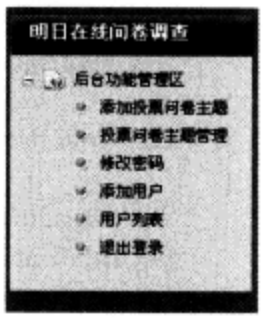


图 17.2 TreeView 控件绑定 XML 数据实现导航

在绑定过程中通过设置 TreeView 控件的 DataSourceID 属性值为数据源控件的 ID 属性值，完成数据绑定任务。TreeView 控件绑定数据源控件实质就是绑定到 XML 文件。

说明

“可扩展标记语言”（XML）提供一种描述结构化数据的方法。其自描述性使其非常适用于不同应用间的数据交换，而且这种交换不以预先定义的一组数据结构为前提，因此具备很强的开放性。XML 是可扩展标记语言，其本身有一套定义语义标记的规则，而不像 HTML 那样定义了许多标记。在定义 XML 时必须做到格式良好且有效，不要只有定义了标记而在应用程序中没有使用。

下面详细介绍如何使用 TreeView 控件绑定 XML 中的数据。

1. 创建执行导航功能的 XML 数据

创建一个 menu.xml 文件，具体步骤如下。

右键单击解决方案资源管理器中的 Web 站点，并选择“添加新项”命令，在弹出的对话框中选择“XML 文件”图标，并将其命名为 TreeView.xml，然后单击“添加”按钮即可。如图 17.3 所示。

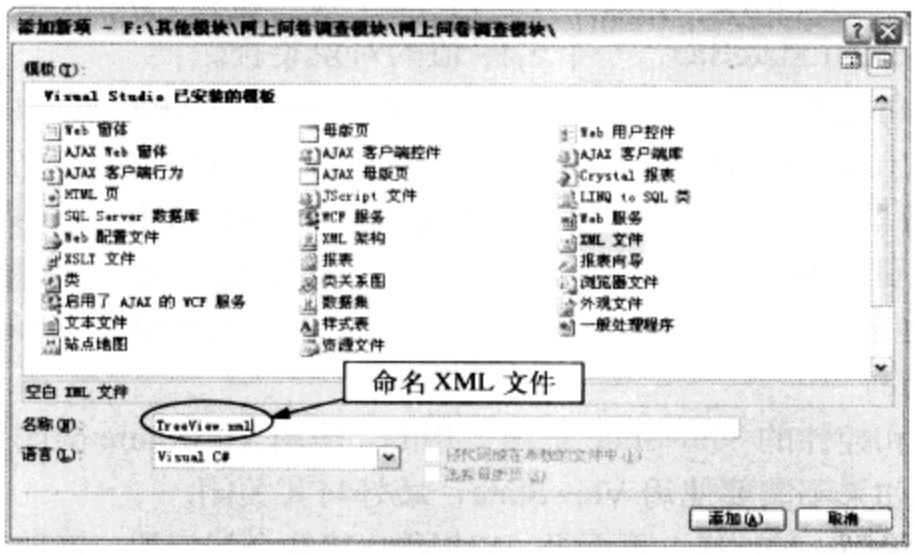


图 17.3 创建 XML 文件

本模块创建的 XML 文件名称为 TreeView.xml，其文件源代码如下。

例程 5 代码位置：光盘\mr\17\OnlineSurvey\admin\modules\menu.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<items>
        <item Text="添加投票问卷主题" NavigateUrl="../addtitle.aspx" />
        <item Text="投票问卷主题管理" NavigateUrl="../titlelist.aspx" />
        <item Text="修改密码" NavigateUrl="../PasswordEdit.aspx" />
        <item Text="添加用户" NavigateUrl="../MemberEdit.aspx" />
        <item Text="用户列表" NavigateUrl="../MemberList.aspx" />
    <item Text="退出登录" NavigateUrl="../../Logout.aspx" />
</items>
```

2. 配置 XMLDataSource 数据源控件

建立完名为 TreeView.xml 的 XML 文件后，在应用程序 adimin 文件夹下的 modules 文件中添加一个名为 TreeView.ascx 的页，在此页的窗体中分别放置 1 个 TreeView 控件和 1 个 XMLDataSource 控件。

首先配置 XMLDataSource 数据源控件。如图 17.4 所示 XMLDataSource 控件的任务栏中包括两个常用任务：配置数据源和刷新架构。单击“配置数据源”选项，将会弹出如图 17.5 所示对话框。此实例只需配置 XML 文件路径。结束以上配置后，单击“确定”按钮。

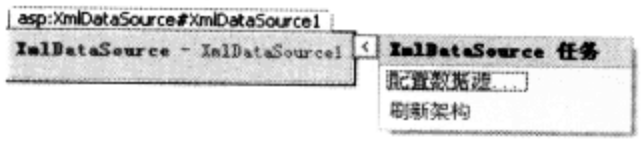


图 17.4 XmlDataSource 控件及其常用任务列表

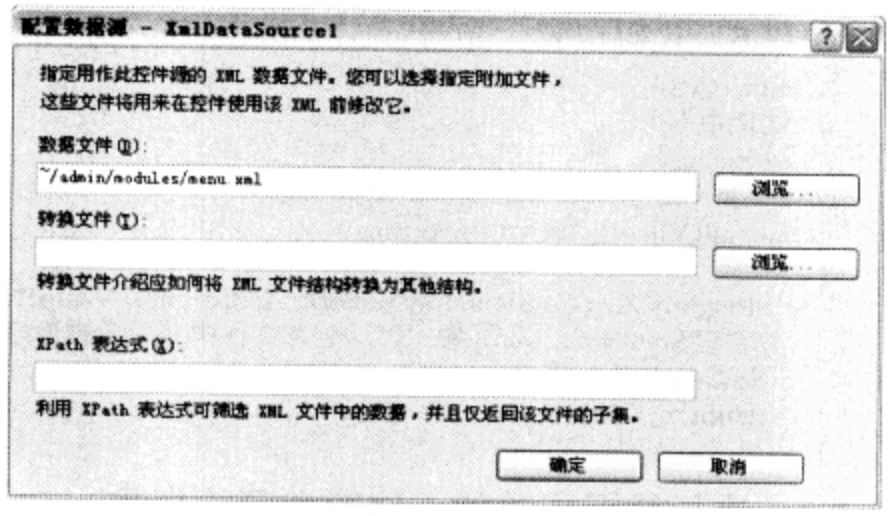


图 17.5 配置数据源窗口

其次，配置 TreeView 控件的相关属性。主要包括两个方面内容：一是选择数据源，二是设置节点数据绑定。在如图 17.6 所示 TreeView 控件常用任务列表中选择“数据源数据”选项即可完成与 XMLDataSource 控件的数据绑定。

另外，设置节点数据绑定可单击 TreeView 控件常用任务列表中的“编辑 TreeNode 数据绑定”选项，此时将会弹出如图 17.7 所示的对话框。

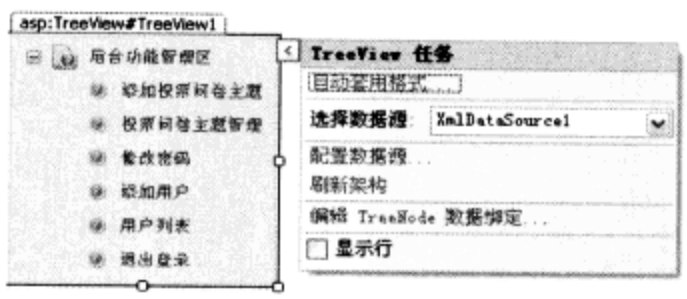


图 17.6 选择数据源

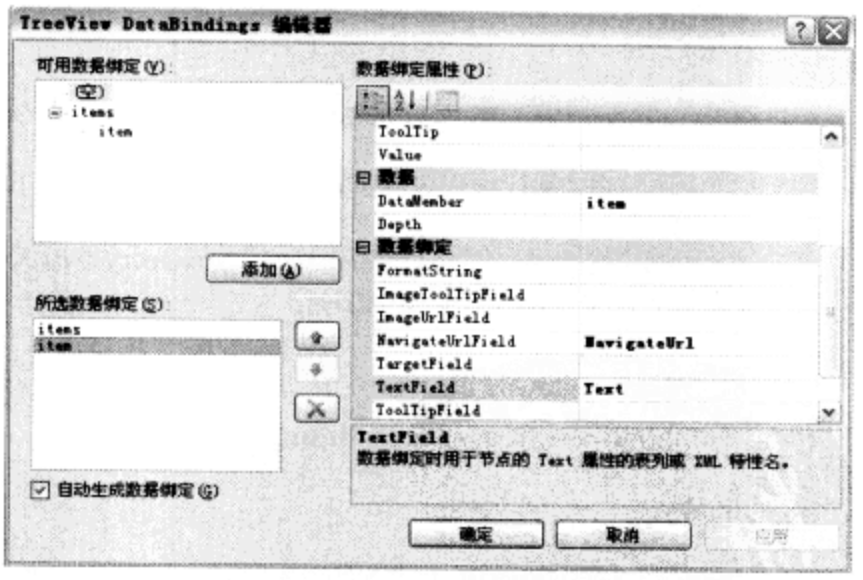


图 17.7 配置 TreeView 控件的数据绑定

图中左列上部是可用数据绑定选项，显示的内容为 TreeView.xml 文件中可供数据绑定的选项。可用数据绑定的选项为树形结构，该结构表现出与 XML 文件一样的嵌套结构；左列下部为选中数据绑定项，此时图的右列将会显示出详细的属性列表信息，XML 文件中的属性值将配置为 TreeView 控件中的节点属性值。如图中为 item 节点配置的属性。

完成以上配置工作后，单击“确定”按钮即可完成 TreeView 控件与 XML 文件的集成。

17.3 公共类的封装与设计

以类的形式来组织、封装一些常用的方法和事件，不仅可以提高代码的重用率，也大大方便了代码的管理。这里主要介绍操作数据库的公共类 DBClass 类的编写过程。

17.3.1 数据库连接操作

数据库连接方法为 GetConnection，在该方法中主要从配置文件中获取连接数据库的字符串，然后创建一个数据库连接对象 SqlConnection，最后返回 SqlConnection 类型的变量。代码如下。

例程 6 代码位置：光盘\mr\17\OnlineSurvey\App_Code\DBClass.cs

```
/// <summary>
/// 连接数据库
/// </summary>
/// <returns>返回SqlConnection对象</returns>
public SqlConnection GetConnection()
{
    string myStr = ConfigurationManager.AppSettings["ConnString"].ToString();
    //创建SqlConnection对象，并从配置文件中获取数据库连接字符串
    SqlConnection myConn = new SqlConnection(myStr);
    return myConn;
}
```

17.3.2 执行数据库添加、修改和删除操作

执行 SQL 命令，如添加、修改和删除操作的方法为 ExecNonQuery 方法，在该方法中通过创建 SqlCommand 对象来执行 SQL 语句命令，并最终返回受影响的行数。代码如下。

例程 7 代码位置：光盘\mr\17\OnlineSurvey\App_Code\DBClass.cs

```
/// <summary>
/// 执行SQL语句，并返回受影响的行数
/// </summary>
/// <param name="myCmd">执行SQL语句命令的SqlCommand对象</param>
public void   ExecNonQuery(SqlCommand myCmd)
{
    try
    {
        if (myCmd.Connection.State != ConnectionState.Open)
            {
            myCmd.Connection.Open(); //打开与数据库的连接
        }
        //使用SqlCommand对象的ExecuteNonQuery方法执行SQL语句，并返回受影响的行数
        myCmd.ExecuteNonQuery();
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        if (myCmd.Connection.State == ConnectionState.Open)
        {
            myCmd.Connection.Close(); //关闭与数据库的连接
        }
    }
```

17.3.3 返回结果集中第一行的第一列

返回结果集中第一行的第一列方法为 ExecScalar，在该方法中主要使用 SqlCommand 对象的 ExecuteScalar 方法执行查询，并返回结果集中第一行的第一列，而所有其他的列和行将被忽略。代码

如下。

例程 8 代码位置：光盘\mr\17\OnlineSurvey\App_Code\DBClass.cs

```
/// <summary>
/// 执行查询，并返回查询所返回的结果集中第一行的第一列。所有其他的列和行将被忽略。
/// </summary>
/// <param name="myCmd"></param>
/// <returns>执行SQL语句命令的SqlCommand对象</returns>
public string ExecScalar(SqlCommand myCmd)
{
    string strSql;
    try
    {
        if (myCmd.Connection.State == ConnectionState.Closed)
        {
            myCmd.Connection.Open(); //打开与数据库的连接
        }
        //使用SqlCommand对象的ExecuteScalar方法执行查询，并返回查询所返回的结果集中第一行的第一列。所有其他的列和行将被忽略。
            strSql=Convert.ToString(myCmd.ExecuteScalar());
            return strSql ;
        }
        catch (Exception ex)
        {
            throw new Exception(ex.Message, ex);
        }
        finally
        {
            if (myCmd.Connection.State == ConnectionState.Open)
            {
                myCmd.Connection.Close();//关闭与数据库的连接
            }
        }
    }
```

17.3.4 执行数据库的查询操作

GetDataSet 方法主要用来执行创建的命令对象，返回一个 DataSet 类型的数据集，在该方法中首先打开数据库连接，然后应用数据适配器从命令对象中读取数据添充到创建的 DataSet 数据集中。代码如下。

例程 9 代码位置：光盘\mr\17\OnlineSurvey\App_Code\DBClass.cs

```
/// <summary>
/// 说　明：　返回数据集的表的集合
///　返回值：　数据源的数据表
///　参　数：　myCmd 执行SQL语句命令的SqlCommand对象，TableName 数据表名称
/// </summary>
public DataSet GetDataSet(SqlCommand myCmd, string TableName)
{
    SqlDataAdapter adapt;
    DataSet ds = new DataSet();
    try
    {
        if (myCmd.Connection.State != ConnectionState.Open)
        {
            myCmd.Connection.Open();
        }
        adapt = new SqlDataAdapter(myCmd);
        adapt.Fill(ds,"TableName");
        return ds;
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
```

```
        finally
        {
            if (myCmd.Connection.State == ConnectionState.Open)
            {
                myCmd.Connection.Close();
            }
        }
    }
```

17.3.5 创建命令对象

创建命令对象的方法为 GetCommandStr，该方法中首先创建一个数据库连接对象，接着创建一个命令对象执行程序中定义的 SQL 语句。代码如下。

例程 10 代码位置：光盘\mr\17\OnlineSurvey\App_Code\DBClass.cs

```
public SqlCommand GetCommandStr(string strSql)
{
    //创建数据库连接对象
    SqlConnection myConn = GetConnection();
    //创建命令对象
    SqlCommand myCmd = new SqlCommand();
    //连接数据库
    myCmd.Connection = myConn;
    //执行定义的SQL语句
    myCmd.CommandText = strSql;
    //获取命令对象执行的类型
    myCmd.CommandType = CommandType.Text;
    return myCmd;
}
```

17.4 问卷调查主页

17.4.1 问卷调查主页概述

该问卷调查的主题类型可分为单选、多选和问答题 3 种类型，在主界面中可对列出的问卷调查主题进行投票，同时可查看投票结果。实际运行效果如图 17.8 所示。

图 17.8 问卷调查主页

17.4.2 问卷调查主页实现过程

1. 设计步骤

（1）在网站根目录下创建一个 Web 窗体，命名为 VoteIndex.aspx，作为网上问卷调查模块的主页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Repeater 控件实现绑定网上问卷调查主题信息。展示其他内容如友情链接内容的控件这里不做介绍。

2. 实现过程

在编写 Page_Load 事件前，需要声明创建的 TitleRule 公共类，以便调用其中编写的方法，声明如下。

例程 11 代码位置：光盘\mr\17\OnlineSurvey\VoteIndex.aspx.cs

```
TitleRule title = new TitleRule();
```

在页面 Page_Load 事件中，首先判断页面是否首次加载，然后调用自定义方法 BindData 绑定 Repeater 列表中的数据，代码如下。

例程 12 代码位置：光盘\mr\17\OnlineSurvey\VoteIndex.aspx.cs

```
protected void Page_Load(object sender, System.EventArgs e)
{
    //首先判断页面是否首次加载
    if(!IsPostBack)
    {
        //调用自定义方法绑定列表控件中的数据
        BindData();
    }
}
```

自定义方法 BindData 主要是将数据库问卷调查信息表中数据绑定到 Repeater 列表中，在该自定义方法中还实现了 Repeater 控件分页功能，实现的代码如下。

例程 13 代码位置：光盘\mr\17\OnlineSurvey\VoteIndex.aspx.cs

```
private void BindData()
{
    int curpage = Convert.ToInt32(this.labPage.Text);
    PagedDataSource ps = new PagedDataSource();
    string connection = System.Configuration.ConfigurationSettings.AppSettings["ConnString"].ToString();
     SqlConnection conn = new SqlConnection(connection);
     string sqlstr = "select * from Title";
     SqlDataAdapter myda = new SqlDataAdapter(sqlstr, conn);
     DataSet ds = new DataSet();
     conn.Open();
     myda.Fill(ds, "Title");
     ps.DataSource = ds.Tables["Title"].DefaultView;
     ps.AllowPaging = true; //是否可以分页
     ps.PageSize = 6; //显示的数量
     ps.CurrentPageIndex = curpage - 1; //取得当前页的页码
     this.lnkbtnUp.Enabled = true;
     this.lnkbtnNext.Enabled = true;
     this.lnkbtnBack.Enabled = true;
     this.lnkbtnOne.Enabled = true;
     if (curpage == 1)
     {
         this.lnkbtnOne.Enabled = false;//不显示第一页按钮
         this.lnkbtnUp.Enabled = false;//不显示上一页按钮
     }
     if (curpage == ps.PageCount)
     {
```

```
            this.lnkbtnNext.Enabled = false;//不显示下一页
            this.lnkbtnBack.Enabled = false;//不显示最后一页
        }
        this.labBackPage.Text = Convert.ToString(ps.PageCount);
        lstTitle.DataSource = ps;
        lstTitle.DataBind();;
    }
```

17.5 问卷调查主题管理

17.5.1 问卷调查主题管理概述

问卷调查主题管理由网站根目录下的 admin 文件夹下的 titlelist.aspx 页面实现，在该页面中可对问卷主题进行修改、删除、添加问卷主题选项、预览问卷投票和查看投票的结果，另外，还可查看问卷主题直接的链接地址等。问卷调查主题管理页运行效果如图 17.9 所示。

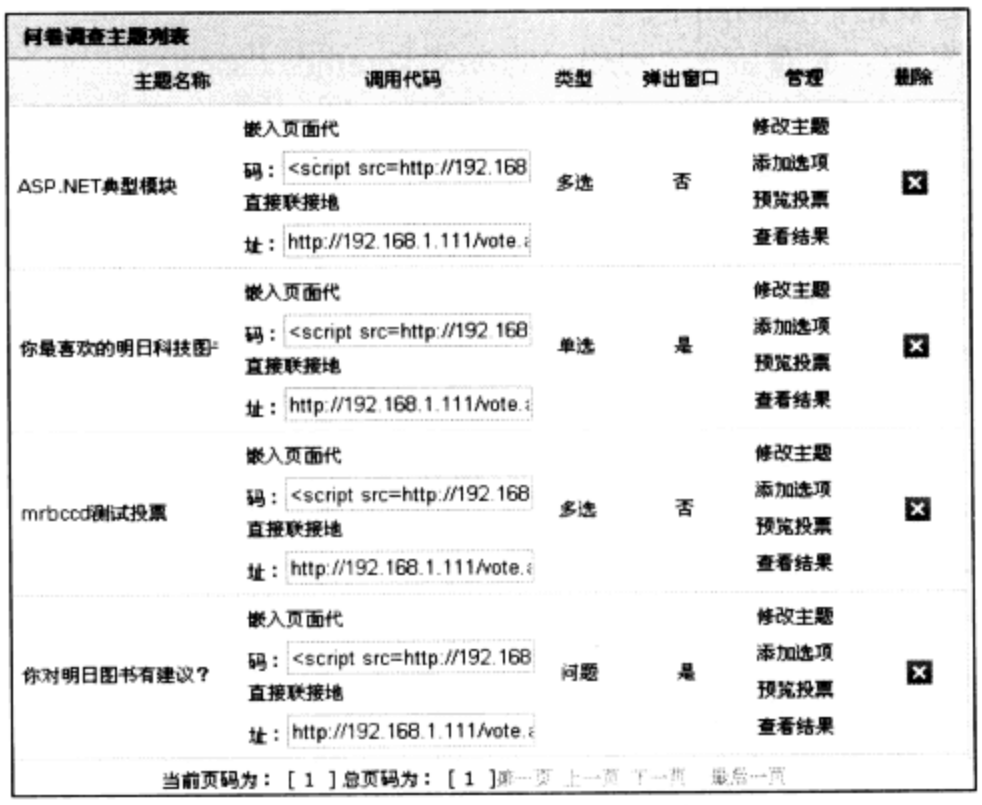


图 17.9 问卷调查主题管理页

17.5.2 问卷调查主题管理实现过程

1. 设计步骤

(1) 在应用程序中创建的名为 admin 文件夹下，创建 1 个 Web 窗体，命名为 titlelist.aspx，作为实现问卷调查主题管理页。

(2) 在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Repeater 控件。在 Repeater 控件模板列中添加 1 个 LinkButton 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 17.2 所示。

表 17.2 addchoice.aspx 页的主要控件属性说明

控件类型	控件名称	主要属性设置	用途
LinkButton	Linkbutton2	无	删除问卷主题操作
Repeater	lstTitle	无	显示问卷调查主题内容

2. 实现过程

前台代码主要用来绑定数据库中问卷主题信息，如问卷主题的名称、调用的代码、问卷主题类型等信息，具体绑定代码如下。

例程 14 代码位置：光盘\mr\17\OnlineSurvey\admin\titlelist.aspx

```
<asp:repeater id="lstTitle" Runat="server" onitemcommand="lstTitle_ItemCommand">
<ItemTemplate>
<TR>
    <TD height="28"><%# DataBinder.Eval(Container, "DataItem.title") %></TD>
    <TD height="28" style="line-height:24px">嵌入页面代码：<input  type=text value="&lt;script src=<%=base.WebUrl%>
voteWindow.aspx?id=<%# DataBinder.Eval(Container, "DataItem.id") %>&w=<%# ToUrl(DataBinder.Eval(Container,
"DataItem. windows")) %>&width=400&height=400&gt;&lt;/script&gt;">
    <br>直接连接地址：<input  type=text value="<%=base.WebUrl%>vote.aspx?id=<%# DataBinder.Eval(Container,
"DataItem.id") %>"></TD>
    TD align="center" height="28"><%# ToChoice(DataBinder.Eval(Container, "DataItem.choice")) %>
    </TD>
    <TD align="center" height="28"><%# ToWindows(DataBinder.Eval(Container, "DataItem.windows")) %></TD>
    <TD height="28" style="line-height:24px">
<a href='addtitle.aspx?id=<%# DataBinder.Eval(Container, "DataItem.id") %>'>修改主题</a>
<br>
<a href='addchoice.aspx?id=<%# DataBinder.Eval(Container, "DataItem.id") %>'>添加选项</a><br>
<a href='../vote.aspx?id=<%# DataBinder.Eval(Container, "DataItem.id") %>' target="_blank">预览投票</a><br>
<a href='<%# ToResult(DataBinder.Eval(Container, "DataItem.choice")) %>?id=<%# DataBinder.Eval(Container,
"DataItem. id") % >' target="_blank">查看结果</a></TD>
<TD height="28" align="center">
<asp:linkbutton runat="server" Text="<img src='images/cancel_x.gif' border=0>" CommandName="Delete" Causes
Validation="false" OnLoad="Delete_Load" CommandArgument='<%# DataBinder.Eval(Container, "DataItem.id") %>' ID=
"Linkbutton2">
</asp:linkbutton></TD>
</TR>
<TR>
  <TD height="1" bgcolor="#CCCCCC" colspan="6"></TD></TR>
</ItemTemplate>
...  //省略程序中自动生成的代码
</asp:repeater>
```

在后台代码中，首先声明创建的公共类，以便调用其中的方法，声明代码如下。

例程 15 代码位置：光盘\mr\17\OnlineSurvey\admin\titlelist.aspx.cs

```
TitleRule title = new TitleRule();
```

在 Page_Load 事件中，主要调用了 1 个自定义方法 BindData()来绑定 Reapter 控件中的问卷主题信息，代码如下。

例程 16 代码位置：光盘\mr\17\OnlineSurvey\admin\titlelist.aspx.cs

```
protected void Page_Load(object sender, System.EventArgs e)
{
    if(!IsPostBack)
    {
        BindData();
    }// 在此处放置用户代码以初始化页面
}
```

自定义 BindData()方法主要是用来绑定 Reapter 控件中数据，并实现分页显示数据功能，具体代码如下。

例程 17 代码位置：光盘\mr\17\OnlineSurvey\admin\titlelist.aspx.cs

```
private void BindData()
{
    int curpage = Convert.ToInt32(this.labPage.Text);
    PagedDataSource ps = new PagedDataSource();
        string connection = System.Configuration.ConfigurationSettings.AppSettings["ConnString"].ToString();
        SqlConnection conn = new SqlConnection(connection);
```

```
        string sqlstr = "select * from Title";
        SqlDataAdapter myda = new SqlDataAdapter(sqlstr, conn);
        DataSet ds = new DataSet();
        conn.Open();
        myda.Fill(ds, "Title");
        ps.DataSource = ds.Tables["Title"].DefaultView;
        ps.AllowPaging = true; //是否可以分页
        ps.PageSize = 6; //显示的数量
        ps.CurrentPageIndex = curpage - 1; //取得当前页的页码
        this.lnkbtnUp.Enabled = true;
        this.lnkbtnNext.Enabled = true;
        this.lnkbtnBack.Enabled = true;
        this.lnkbtnOne.Enabled = true;
        if (curpage == 1)
        {
            this.lnkbtnOne.Enabled = false;//不显示第一页按钮
            this.lnkbtnUp.Enabled = false;//不显示上一页按钮
        }
        if (curpage == ps.PageCount)
            {
                this.lnkbtnNext.Enabled = false;//不显示下一页
                this.lnkbtnBack.Enabled = false;//不显示最后一页
            }
            //分页显示问卷主题信息
            this.labBackPage.Text = Convert.ToString(ps.PageCount);
            //设置问卷主题数据源为创建的数据集
            lstTitle.DataSource = ps;
            //从数据库中绑定问卷主题信息
            lstTitle.DataBind();;
    }
```

17.6 添加或编辑问卷主题

17.6.1 添加或编辑问卷主题概述

为了使代码更加优化，将问卷调查主题的添加和修改都设计在了一个页面中，即addtitle.aspx 页。用户在后台导航列表中单击“添加投票问卷主题”和单击投票问卷主题列表主页 titlelist.aspx 页中的“修改主题”链接时都会链接到该页。添加或编辑问卷主题运行效果如图 17.10 所示。

图 17.10 添加或编辑问卷主题

17.6.2 添加或编辑问卷主题实现过程

1. 设计步骤

（1）在应用程序中创建的名为 admin 文件夹下，创建 1 个 Web 窗体，命名为 addtitle.aspx，作为实现添加和修改投票问卷主题页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 TextBox 控件、2 个 RButton 控件和 1 个 Button 控件，通过属性窗口设置控件的属性。页面中各个控件的属性设置及其用途如表 17.3 所示。

表 17.3　addtitle.aspx 页的主要控件属性说明

控件类型	控件名称	主要属性设置	用　途
TextBox	txtTitle	MaxLength 属性设置为 200、TextMode 属性设置为 MultiLine	添加或显示问卷调查标题内容
	txtContant	MaxLength 属性设置为 200、TextMode 属性设置为 MultiLine	添加或显示问卷调查中用户连接的文字
RadioButtonList	rblChoice	RepeatDirection 属性设置为 Horizontal	用于选择问卷调查的类型（单选、多选或问答题）
	rbtWindows	RepeatDirection 属性设置为 Horizontal	用于选择问卷调查是否弹出窗口
Button	btnSave	Text 属性设置为“保存”	保存用户对问卷调查的添加或修改操作

2. 实现过程

创建 TitleRule 类的一个实例，以便调用其中定义的方法，声明如下。

例程 18　代码位置：光盘\mr\17\OnlineSurvey\admim\addtitle.aspx.cs

```
TitleRule title= new TitleRule();
```

在页面的 Paga_Load 事件中，首先判断页面是否首次加载，然后判断页面传过来的值是否为空，如果不为空则调用自定义方法绑定添加的投票的信息，代码如下。

例程 19　代码位置：光盘\mr\17\OnlineSurvey\admim\addtitle.aspx.cs

```
protected void Page_Load(object sender, System.EventArgs e)
{
    btnSave.Attributes["onclick"]="return Check_Input();";
   //判断页面是否首次加载
    if(!IsPostBack)
    {
        //获取传过来的id值
       ViewState["Tid"] = Request.QueryString["id"];
       //如果传来的值不为空
         if(ViewState["Tid"]!=null)
           //调用自定义方法绑定添加投票信息
              BindData();
    }
}
```

自定义方法 BindData，分别从数据库中绑定投票主题的名称、投票主题类型、是否添加连接文字等信息，代码如下。

例程 20　代码位置：光盘\mr\17\OnlineSurvey\admim\addtitle.aspx.cs

```
public void BindData()
{
    //创建一个数据表行视图，并调用公共类中的Info方法绑定数据库中数据
   DataRow dr = title.Info(ViewState["Tid"]);
```

```
        //将从数据库中读取的投票主题读取来
        txtTitle.Text = dr["title"].ToString();
        //绑定投票类型
        rblChoice.Items.FindByValue(dr["choice"].ToString()).Selected=true;
        //判断是否弹出窗口
        if(bool.Parse(dr["windows"].ToString()))
        rbtWindows.Items.FindByValue("1").Selected=true;
    }
```

上述自定义 BindData 方法中调用了公共类中定义的 1 个 DataRow 类型的 Info 方法，调用该方法主要用来从数据库中绑定投票主题信息，具体实现代码如下。

例程 21 代码位置：光盘\mr\17\OnlineSurvey\admim\addtitle.aspx.cs

```
public DataRow Info(object id)
 {
     //定义一个查询语句的字符串
     string sqlstr = "select * from title where id=@id order by desc";
     //定义一个SqlCommad命令对象，并调用公共类中的GetCommandStr执行定义的SQL语句
     SqlCommand myCmd = dbobj.GetCommandStr(sqlstr);
     //定义一个SqlParameter参数
     SqlParameter myid = new SqlParameter("@id", SqlDbType.Int, 4);
     //获取参数的值
     myid.Value = id;
     //添加参数
    myCmd.Parameters.Add(myid);
    //执行命令对象
    dbobj.ExecNonQuery(myCmd);
    //定义数据表中行数据
    DataRow ds = dbobj.GetDataSet(myCmd, "title").Tables[0].Rows[0];
    return ds;
}
```

双击页面中的“保存“按钮，触发其 btnSave_Click 事件。在该事件中调用公共类中的 Update 方法和 Add 方法，分别执行投票主题更新操作和执行问卷调查投票主题操作，代码如下。

例程 22 代码位置：光盘\mr\17\OnlineSurvey\admim\addtitle.aspx.cs

```
protected void btnSave_Click(object sender, System.EventArgs e)
{
    if(ViewState["Tid"]!=null)
    {
      //调用公共类中的Update方法执行投票主题更新操作
title.Update(ViewState["Tid"], txtTitle.Text, Convert.ToByte(rblChoice.SelectedValue), rblChoice.SelectedValue, Convert.ToByte
(rbtWindows.SelectedValue));
    }
        else
    {
            //调用公共类中的Add方法执行问卷调查投票主题操作
title.Add(txtTitle.Text.ToString(), Convert.ToByte(rblChoice.SelectedValue), rblChoice.SelectedValue, Convert.ToByte
(rbtWindows.SelectedValue));
          }
    this.AlertAndRedir("保存成功，确认后返回投票主题列表","titlelist.aspx");
    }
```

上述 btnSave_Click 事件代码中，调用了公共类中的 Update 方法，主要用来更新数据库问卷调查信息表中的数据，代码如下。

例程 23 代码位置：光盘\mr\17\OnlineSurvey\App_code\rules\ReplyRule.cs

```
public void Update(object id, object title, object current, object choice, object windows)
  {
    //定义一个更新操作的SQL语句
    string sqlstr = "update title set [title]=@title,[current]=@current,choice=@choice,windows=@windows where id=@id";
```

```
        //创建一个SqlComma命令对象，并调用公共类中的GetCommandStr方法执行SQL语句
        SqlCommand myCmd = dbobj.GetCommandStr(sqlstr);
        SqlParameter[] oParms = {
                                    new SqlParameter("@id",SqlDbType.Int,4),
                                    new SqlParameter("@title",SqlDbType.NVarChar,200),
                                    new SqlParameter("@current",SqlDbType.Bit,1),
                                    new SqlParameter("@choice",SqlDbType.SmallInt,2),
                                    new SqlParameter("@windows",SqlDbType.Bit,1)
                                };
        oParms[0].Value = id;
        oParms[1].Value = title;
        oParms[2].Value = current;
        oParms[3].Value = choice;
        oParms[4].Value = windows;
        foreach (SqlParameter parmater in oParms)
        {
            myCmd.Parameters.Add(parmater);
        }
        string s = myCmd.CommandText;
        //执行命令对象SqlCommd中存储的查询SQL语句
        dbobj.ExecNonQuery(myCmd);
    }
```

上述 btnSave_Click 事件代码中，除了调用 Update 方法外，还调用了公共类中的 Add 方法，主要用来添加数据库问卷调查信息表中的数据，代码如下。

例程 24 代码位置：光盘\mr\17\OnlineSurvey\App_code\rules\ReplyRule.cs

```
public void Add(object title, object current, object choice, object windows)
{
        //定义添加的SQL字符串
        string sqlstr = "insert into title ([title],[current],[choice],[windows]) values(@title,@current, @choice, @windows)";
        ////创建一个SqlCommdw命令对象，并调用公共类中的GetCommandStr执行定义的SQL语句
        SqlCommand myCmd = dbobj.GetCommandStr(sqlstr);
        SqlParameter[] oParms = {
                                    new SqlParameter("@title",SqlDbType.NVarChar,200),
                                    new SqlParameter("@current",SqlDbType.Bit,1),
                                    new SqlParameter("@choice",SqlDbType.SmallInt,2),
                                    new SqlParameter("@windows",SqlDbType.Bit,1)
                                };
        oParms[0].Value = title;
        oParms[1].Value = current;
        oParms[2].Value = choice;
        oParms[3].Value = windows;
        foreach (SqlParameter parmater in oParms)
        {
            myCmd.Parameters.Add(parmater);
        }
        dbobj.ExecNonQuery(myCmd);
}
```

17.7 问卷调查主题选项管理

17.7.1 问卷调查主题选项管理概述

在问卷调查主题列表页 titlelist.aspx 中，添加完问卷调查主题后便可添加相应的主题选项内容。添加问卷调查主题选项由 addchoice.aspx 页实现。在该页中用户除了添加主题选项外，还可以对添加的主题选项进行修改和删除操作。该页运行效果如图 17.11 所示。

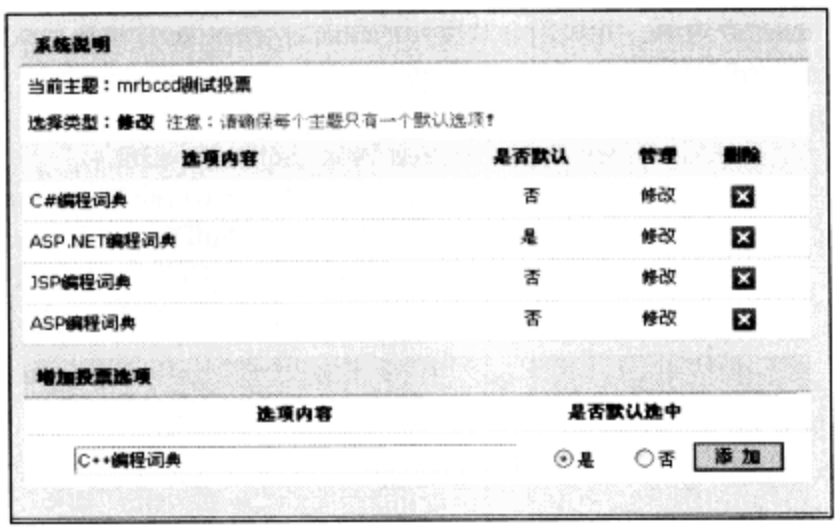


图 17.11　问卷调查主题选项管理页

17.7.2　问卷调查主题选项管理实现过程

1. 设计步骤

(1)在应用程序中创建的名为 admin 文件夹下，创建 1 个 Web 窗体，命名为 addchoice.aspx，作为实现问卷调查主题选项管理页。

(2) 在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 Repeater 控件、1 个 TextBox 控件、1 个 RadioButtonList 和 1 个 Button 控件，通过属性窗口，设置控件的属性。页面中各个控件的属性设置及其用途如表 17.4 所示。

表 17.4　addchoice.aspx 页的主要控件属性说明

控 件 类 型	控 件 名 称	主要属性设置	用　途
TextBox	txtTitle	MaxLength 属性设置为“200”	添加问卷调查主题选项内容
RadioButtonList	rblChoice	RepeatDirection 属性设置“Horizontal”	用于选择问卷调查主题选项内容的选项
Repeater	lstChoice	无	显示添加的问卷调查选项内容
Button	btnSave	Text 属性设置为“添加”	保存用户对问卷调查的添加或修改操作

2. 实现过程

在编写 Page_Load 事件前，需要先声明创建的两个公共类 TitleRule 和 ChoiceRule，以便调用它们中的方法实现相应的操作，声明代码如下。

例程 25　代码位置：光盘\mr\17\OnlineSurvey\admin\addchoice.aspx.cs

```
TitleRule title = new TitleRule();
ChoiceRule choice = new ChoiceRule();
```

在 Page_Load 事件中，首先判断页面是否首次加载，然后应用 ViewState 保存页面间的值，最后调用自定义方法 BindData 绑定列表中的数据。实现代码如下。

例程 26　代码位置：光盘\mr\17\OnlineSurvey\admin\addchoice.aspx.cs

```
protected void Page_Load(object sender, System.EventArgs e)
{
    btnSave.Attributes["onclick"]="return Check_Input();";
    if(!IsPostBack)//判断页面是否首次加载
    {
        //应用ViewState保存页面间的值
        ViewState["Tid"] = Request.QueryString["id"];
        //调用自定义方法绑定列表控件中的数据
        BindData();
    }
}
```

自定义 BindData 方法中主要调用创建的公共类中的 GetTitle 和 List 方法，分别绑定问卷主题信息和绑定问卷主题的选项信息，具体实现的代码如下。

例程 27 代码位置：光盘\mr\17\OnlineSurvey\admin\addchoice.aspx.cs

```
private void BindData()
{
    //将保存的主题编号存储在定义的字符串型变量id中
    id=ViewState["Tid"].ToString();
    //调用公共类中的GetTitle方法绑定问卷主题信息
    strtitle=title.GetTitle(ViewState["Tid"]);
    //调用公共类中的List方法绑定问卷主题t选项信息
    lstChoice.DataSource = choice.List(ViewState["Tid"]);
    //调用DataBind方法从数据库中绑定数据
    lstChoice.DataBind();
    //绑定页面中数据
    Page.DataBind();
}
```

上述自定义 BindData 方法调用了公共类中的 GetTitle 方法。该方法中首先定义一个查询数据库问卷主题列表中的 SQL 语句，接着创建一个命令对象执行定义的 SQL 语句，然后，创建一个 SqlParameter 参数进行参数传递，最后创建一个 DataSet 数据集来存储问卷主题信息。具体实现代码如下。

例程 28 代码位置：光盘\mr\17\OnlineSurvey\App_code\rules\TitleRule.cs

```
public string GetTitle(object id)
{
    //定义查询数据库的字符串
    string sqlstr = "select [title] from title where id=@id";
    //创建SqlCommand命令对象
    SqlCommand myCmd = dbobj.GetCommandStr(sqlstr);
    //定义一个参数
    SqlParameter myid = new SqlParameter("@id", SqlDbType.Int, 4);
    //参数赋值
    myid.Value = id;
    //添加参数
    myCmd.Parameters.Add(myid);
    //执行命令对象中存储的定义的SQL语句
    dbobj.ExecNonQuery(myCmd);
    //创建一个数据集对象
    DataSet ds = dbobj.GetDataSet(myCmd, "title");
        //判断存储在数据集中的内存表有数据
        if (ds.Tables[0].Rows.Count > 0)
            //有数据将主题列表信息显示出来
            return ds.Tables[0].Rows[0]["title"].ToString();
        else
            //没有数据，弹出提示框
            return "该选票主题不存在!!! ";
}
```

自定义 BindData 方法除了调用了公共类中的 GetTitle 方法外，还调用了公共类中的 List 方法绑定问卷主题选项内容，具体实现代码如下。

例程 29 代码位置：光盘\mr\17\OnlineSurvey\App_code\rules\TitleRule.cs

```
public DataSet List(object tid)
{
    //定义查询数据表中的字符串
    string sqlstr = "select    * from choice    where extends=@tid order by id";
    //创建一个命令对象执行定义的SQL语句
    SqlCommand myCmd = dbobj.GetCommandStr(sqlstr);
    //定义一个参数
    SqlParameter mytid = new SqlParameter("@tid", SqlDbType.Int, 4);
    //参数赋值
```

```
        mytid.Value = tid;
        //添加参数
        myCmd.Parameters.Add(mytid);
        //执行查询操作
        dbobj.ExecNonQuery(myCmd);
        //调用公共类中的GetDataSet返回一个DataSet类型的数据集
        DataSet myds= dbobj.GetDataSet(myCmd, "choice");
        return myds;
    }
```

双击页面中的“添加”按钮触发其 btnSave_Click 事件，在该事件中主要调用了公共类中的 Add 方法向数据库中添加信息，添加完信息后调用自定义方法 BindData 重新绑定下数据库中信息，代码如下。

例程 30 代码位置：光盘\mr\17\OnlineSurvey\admin\addchoice.aspx.cs

```
protected void btnSave_Click(object sender, System.EventArgs e)
{
    //调用公共类中的Add方法向数据库中添加信息
            choice.Add(txtChoice.Text.Trim(),Convert.ToInt32(ViewState["Tid"]),transfer(rblChoice.SelectedValue));
     //弹出保存成功对话框
    this.DisplayAlert("保存成功");
    txtChoice.Text="";
    //调用自定义方法重新绑定数据库中信息
    BindData();
}
```

上述 btnSave_Click 事件代码中调用了公共类中的 Add 方法，该方法中首先定义了一个执行添加操作的 SQL 语句，接着创建一个命令对象并应用 SqlParameter 进行参数传值，最后调用公共类中的 ExecScalar 方法返回数据表中第一行第一列的数据信息。代码如下。

例程 31 代码位置：光盘\mr\17\OnlineSurvey\App_code\rules\TitleRule.cs

```
public object Add(string choice, int extends, bool IsDefault)
{
    //定义添加的SQL语句
    string sqlstr = "insert into choice([choice],[extends],[IsDefault]) values(@choice,@extends,@IsDefault)";
     //创建一个命令对象，执行定义的QL语句
     SqlCommand myCmd = dbobj.GetCommandStr(sqlstr);
     SqlParameter mychoice = new SqlParameter("@choice", SqlDbType.NVarChar, 50);
     mychoice.Value = choice;
     myCmd.Parameters.Add(mychoice);
     SqlParameter myextends = new SqlParameter("@extends", SqlDbType.Int, 4);
     myextends.Value = extends;
     myCmd.Parameters.Add(myextends);
     SqlParameter myIsDefault = new SqlParameter("@IsDefault", SqlDbType.Bit, 1);
     myIsDefault.Value = IsDefault;
     myCmd.Parameters.Add(myIsDefault);
     //调用公共类中的ExecScalar方法返回数据表中第一行第一列的数据信息
     return dbobj.ExecScalar(myCmd).ToString();
}
```

在问卷主题选项列表中，可对选项内容进行删除操作，该删除操作主要编写在 Repeater 控件的 ItemCommand 事件中，代码如下。

例程 32 代码位置：光盘\mr\17\OnlineSurvey\admin\addchoice.aspx.cs

```
protected void lstChoice_ItemCommand(object source, RepeaterCommandEventArgs e)
{
    //接收传过来的CommandName的值
    if (e.CommandName == "Delete")
    {
        //调用公共类中的Delete方法执行删除操作
        choice.Delete(e.CommandArgument);
    }
    //调用自定义方法重新绑定列表信息
    BindData();
}
```

上述 lstChoice_ItemCommand 事件代码中，调用了公共类中的 Delete 方法，该方法代码如下。

例程 33 代码位置：光盘\mr\17\OnlineSurvey\App_code\rules\TitleRule.cs

```
public void Delete(object id)
{
    //定义一个删除操作的SQL语句
    string sqlstr = "delete from choice where id=@id";
    //创建一个命令对象执行定义的SQL语句
    SqlCommand myCmd = dbobj.GetCommandStr(sqlstr);
    //定义一个参数
    SqlParameter myid = new SqlParameter("@id", SqlDbType.Int, 4);
    //参数赋值
    myid.Value = id;
    //添加参数
    myCmd.Parameters.Add(myid);
    //执行命令对象
    dbobj.ExecNonQuery(myCmd);
}
```

17.8 程序错误与调试

在如图 17.12 所示的问卷主题选项管理页 addchoice.aspx 中，在实现“是否默认选中”功能时应用了 RadioButtonList 控件，并在数据库中设置了其值为布尔类型。如果在对问卷主题选项进行保存时没有对此布尔类型的值进行转换就会出现如图 17.13 所示的错误。

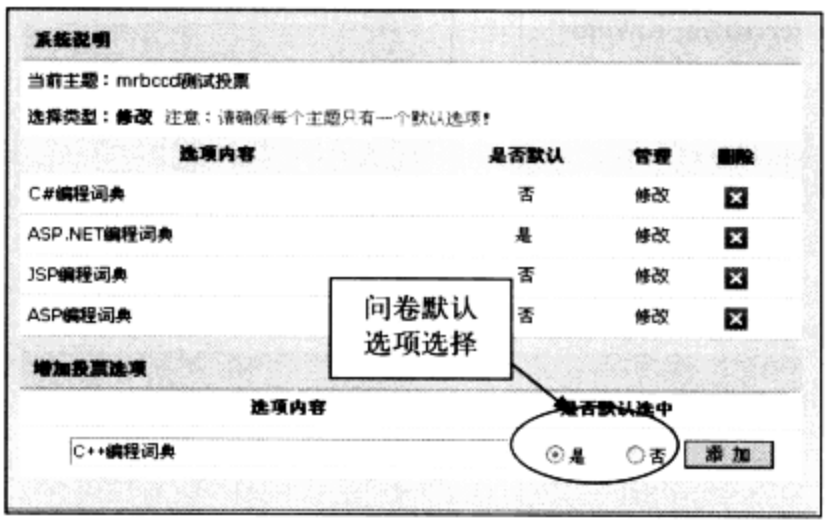


图 17.12 问卷主题选项管理

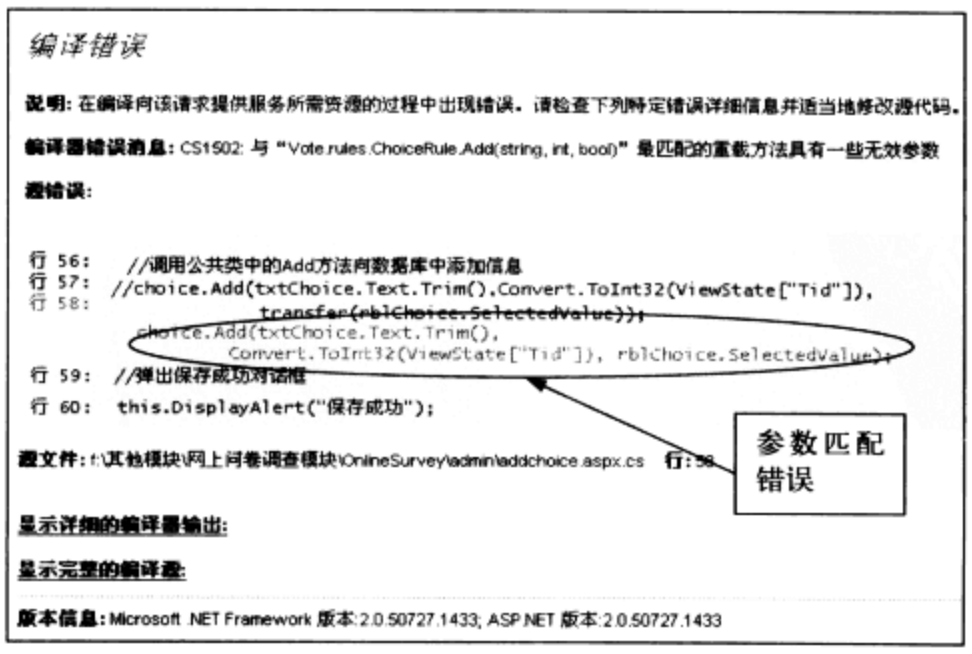


图 17.13 参数类型匹配错误

编写的错误代码如下：

```
protected void btnSave_Click(object sender, System.EventArgs e)
{
    //调用公共类中的Add方法向数据库中添加信息
    choice.Add(txtChoice.Text.Trim(), Convert.ToInt32(ViewState["Tid"]), rblChoice.SelectedValue);
    //弹出保存成功对话框
    this.DisplayAlert("保存成功");
    txtChoice.Text="";
    BindData();
}
```

解决此错误只需要编写一个方法对布尔类型的值进行转换，正确代码编写如下。

例程 34 代码位置：光盘\mr\17\OnlineSurvey\admin\addchoice.aspx.cs

```
protected void btnSave_Click(object sender, System.EventArgs e)
{
    //调用公共类中的Add方法向数据库中添加信息
     choice.Add(txtChoice.Text.Trim(), Convert.ToInt32(ViewState["Tid"]), transfer(rblChoice.SelectedValue));
     //弹出保存成功对话框
          this.DisplayAlert("保存成功");
          txtChoice.Text="";
          BindData();
}
    /// <summary>
    ///   转化为Bool值
    /// </summary>
    /// <param name="strValue">需要转化的值</param>
    /// <returns>返回转化后的值</returns>
    protected bool transfer(string strValue)
    {
        if (strValue == "是")
        {
            return true;
        }
        else
        {
            return false;
        }
    }
```

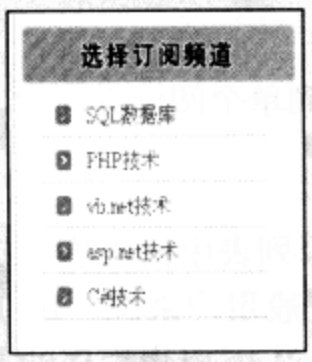
RSS 在线订阅与阅读模块

第 18 章

实例位置：光盘\mr\18\

RSS 频道订阅技术，出现在几年前新闻网站的新闻频道功能中，如今一个 RSS 包含很多新闻频道，一个新闻频道的介绍包含了该频道中的所有内容，这些频道的链接通常链接该频道中的所有新闻链接，当网站打不开的情况下，借助支持 RSS 的新闻聚合工具软件进行新闻阅读。另外，目前 RSS 频道订阅不仅仅应用到新闻类网站中，而且在博客、论坛等网站中应用也比较广泛。为了使读者更好的学习和深入的了解 RSS，本模块既实现了 RSS 在线订阅，也开发了一个简单的 RSS 订阅器。通过本章，读者能够学到以下内容。

订阅频道列表



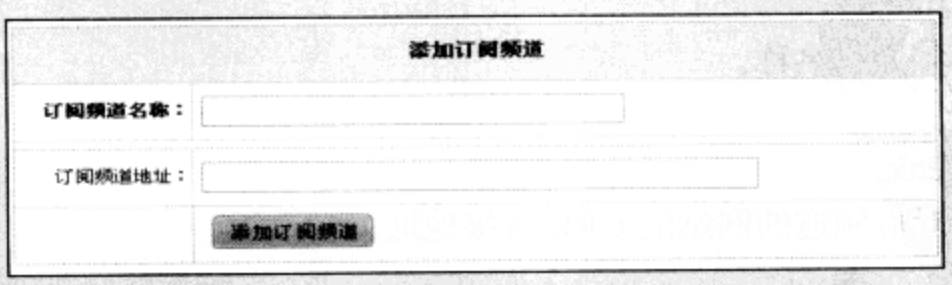
管理订阅频道

SQL数据库	http://localhost:2205/rss.ashx	编辑	删除
PHP技术	http://localhost:2205/rss.ashx	编辑	删除
vb.net技术	http://localhost:2205/rss.ashx	编辑	删除
asp.net技术	http://localhost:2205/rss.ashx	编辑	删除
C#技术	http://localhost:2205/rss.ashx	编辑	删除

生成订阅频道

```
<?xml version="1.0" encoding="utf-8" ?>
- <rss version="2.0">
- <channel>
    <title>Sample Channel</title>
    <link>http://localhost:2205/Default.aspx</link>
    <description>asp.net技术频道</description>
    <ttl>10</ttl>
    <name />
    <user />
  - <item>
      <title>图书大本营</title>
      <description>.net图书 c#图书 vb.net图书 JAVA图书</description>
      <link>http://localhost/docLink/java.aspx</link>
    </item>
  - <item>
```

添加订阅频道



18.1 RSS 在线订阅与阅读模块概述

18.1.1 RSS 简介

RSS 是一种站点内容聚合，在英文中被不同的技术团体做了不同的解释，分别是“Rich Site Summary”（丰富站点摘要）、“RDF Site Summary”（RDF 站点摘要）、“Really Simple Syndication”（真正简易聚合）。

RSS 是基于文本的格式。它是 XML（可扩展标识语言）的一种形式。通常 RSS 文件都是标为 XML，RSS files（通常也被称为 RSS feeds 或者 channels）通常只包含简单的项目列表。通常情况下，每一个项目都含标题、简单的描述和 URL 链接，以及其他的信息，包括日期、作者等。

18.1.2 RSS 订阅特点

随着越来越多的站点对 RSS 的支持，RSS 已经成为目前最成功的 XML 应用。RSS 搭建了信息迅速传播的一个技术平台，使得每个人都成为潜在的信息提供者。相信很快就会看到大量基于 RSS 的专业门户、聚合站点和更精确的搜索引擎。RSS 订阅具有如下特点：

- 内容简洁，是自己需要的内容，而没有多余的其他内容；
- RSS 阅读器会自动更新订阅的内容，保持内容的及时性；
- 用户可以加入多个订阅的 RSS 栏目，从多个来源搜集内容整合到单个阅读界面中。

18.1.3 如何使用 RSS

首先，需要有一个 RSS 阅读器。其次，从网站提供的聚合订阅目录列表中订阅感兴趣的栏目内容。订阅后，将会及时获得所订阅新闻频道的最新内容。对于一般用户来说，用 RSS 订阅新闻可以像使用 Outlook Express 收取订阅的邮件一样简单；而对于 Web 应用程序的开发者而言，RSS 的工作过程也不是那么复杂，至少比大多数其他常见的 Web 技术都更容易被理解和被实现。

18.1.4 RSS 技术规范

无论为网站添加一个订阅功能，还是开发一个 RSS 订阅器，熟悉 RSS 技术规范是必须掌握的知识。

1. 必需的频道组成元素

下面是必须包含的频道（channel）元素的列表。

- title
  - 名称：title。
  - 描述：频道的名称。
- link
  - 名称：link。
  - 描述：对应频道的网站的 URL 链接地址。
- description
  - 名称：description。
  - 描述：关于频道的描述。

2. 可选的频道元素

下面是一个可选的频道（channel）元素的列表。

- language
  - 名称：language。
  - 描述：频道使用的语言。比如，在一个网站上，允许聚合所有的简体中文站点到相应的分组。
  - 例如，en-us（英文）。
- copyright
  - 名称：copyright。
  - 描述：频道内容的版权声明。
  - 例如，Copyright 2008， mingrisoft。
- managingEditor
  - 名称：managingEditor。
  - 描述：频道内容责任编辑的电子邮件地址。
  - 例如，****abc@mrbccd.com。
- webMaster
  - 名称：webMaster。
  - 描述：频道技术支持人员的电子邮件地址。
- pubDate
  - 名称：pubDate。
  - 描述：频道内容发布的日期。所有的日期和时间都必须遵循 RFC 822 规范，但年份可以用 2 个或 4 个字母表示（首选 4 个字母）。
  - 例如，Sat, 07 Sep 2002 00:00:01 GMT。
- lastBuildDate
  - 名称：lastBuildDate
  - 描述：频道内容的最后修改时间。
  - 例如，Fri, 08 Aug 2008 08:08:08 GMT。
- category
  - 名称：category。
  - 描述：指定频道所属的一个或多个分类。遵循与 item 级 category 元素相同的规则。
  - 例如，<category>News</category>。
- generator
  - 名称：generator。
  - 描述：表明生成频道的程序名称的字符串。
- docs
  - 名称：docs。
  - 描述：指向该 RSS 文件所用格式说明文档的 URL 链接地址。
  - 例如，http://www.mrbccd.com/rss。
- cloud
  - 名称：cloud。
  - 描述：允许通过注册一个 cloud 来处理获得频道的更新通知，并为 rss 种子实现一个轻量级的发布订阅协议。
  - 例如，<cloud domain="www.mrbccd.com" port="80" path="/RPC2" registerProcedure="pingMe" protocol="soap"/>。

- ttl
  - 名称：ttl。
  - 描述：ttl 是 Time to live 的缩写，表示生存时间。它表示频道从源更新之前可以缓存的时间。
  - 例如，<ttl>60</ttl>。
- image
  - 名称：image。
  - 描述：指定一个可以在频道中显示的 GIF、JPEG 或者 PNG 图像。
- rating
  - 名称：rating。
  - 描述：频道的 PICS 内容分级信息。
- textInput
  - 名称：textInpu。
  - 描述：指定一个可以在频道中显示的文本输入框。
- skipHours
  - 名称：skipHours。
  - 描述：提示聚合器，可以跳过那些小时的时间段。
- skipDays
  - 名称：skipDays。
  - 描述：提示聚合器，可以跳过那些天的时间段。

3. <item>中的元素

一个频道可以包含多个<item>元素。一个项目可以代表一个“故事”，例如一份报纸或杂志上的故事。如果是这样的话，那么项目的描述则是故事的摘要，项目的链接则指向整个故事的链接位置。一个项目也可以本身是完整的，如果是这样的话，项目的描述就包含了文本，而链接和标题可以省略。项目的所有元素都是可选的，但是至少要包含一个标题（title）或描述（description）。

- title
  - 名称：title。
  - 描述：item 的标题。
- link
  - 名称：link。
  - 描述：item 的 URL 链接地址。
  - 例如，http://nytimes.com/2004/12/07FEST.html。
- description
  - 名称：description。
  - 描述：item 的摘要。
- author
  - 名称：author。
  - 描述：item 作者的电子邮件地址。
- category
  - 名称：category。
  - 描述：包含 item 在一个或多个分类中。

- comments
  - 名称：comments。
  - 描述：与 item 相关的评论的 URL 链接地址。
- enclosure
  - 名称：enclosure。
  - 描述：item 附加的媒体对象。
- guid
  - 名称：guid。
  - 描述：可以唯一确定 item 身份的字符串。
- pubDate
  - 名称：pubDate。
  - 描述：item 发布的时间。
- source
  - 名称：source。
  - 描述：rss 频道来源。

18.2 实现 RSS 在线订阅与阅读的关键技术

18.2.1 微软提供 RSS 工具包

在 ASP.NET 中可以很方便地对 RSS 技术进行开发，因为微软公司提供了一个 RSS 开发工具包 RssToolkit.dll，程序开发人员可以通过 RssToolkit.dll 工具包实现 RSS 订阅和阅读功能。

RssToolkit.dll 工具包提供 RssDataSource 和 RssHyperLink 两个服务器控件，其中 RssDataSource 控件用来实现一个只有一列的网格频道列表，并通过导航的方式显示频道的内容，RssHyperLink 控件用来实现为站点提供订阅功能。本模块并没有用到这两个控件，主要通过 RssToolkit.dll 工具包提供的处理接口来生成订阅频道和读取解析订阅频道。

注意

RssToolkit.dll 工具包，读者可以到 Microsoft 官方网站上下载。

下面介绍如何将 RssToolkit.dll 工具包添加到 Visual Studio 2008 开发环境工具箱中，具体步骤如下。

（1）进入 Visual Studio 2008 开发环境中的 Web 窗体的设计视图，展开工具箱，在工具箱上单击鼠标右键弹出快捷菜单，如图 18.1 所示。

（2）在弹出的快捷菜单上，选择“选择项”命令，弹出“选择工具箱项”对话框，如图 18.2 所示。

（3）在“选择工具箱”对话框中单击“浏览”按钮，弹出“打开”对话框，如图 18.3 所示，找到 RssToolkit.dll 工具包，并将其打开。

（4）成功打开 RssToolkit.dll 工具包后，分别在“解决方案资源管理器”中显示引用的 DLL 和“选择工具项”，.NET 组件列表框中相关控件如图 18.4 所示。

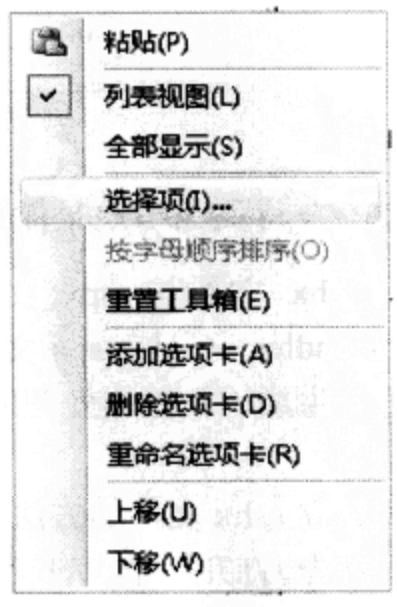


图 18.1 工具箱快捷菜单

图 18.2　选择工具箱项

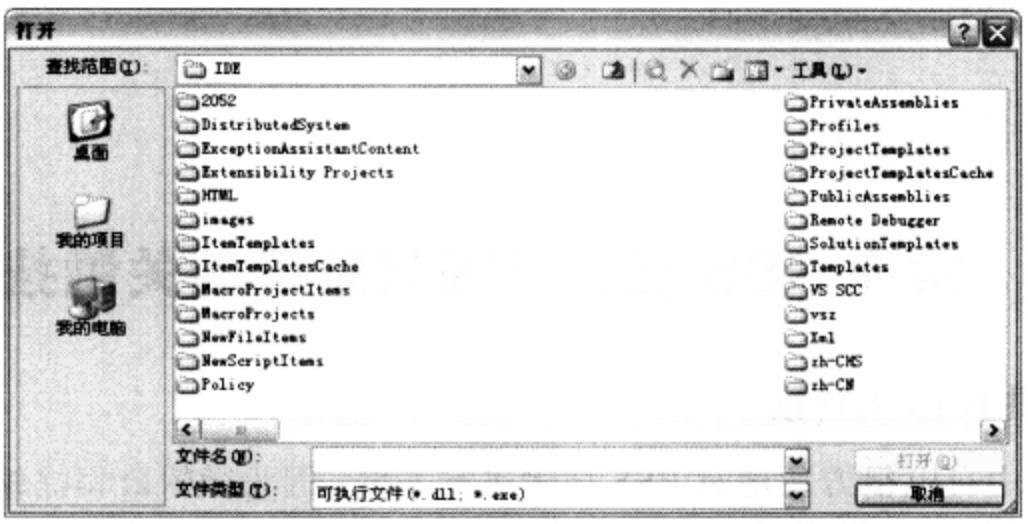


图 18.3　“打开”对话框

图 18.4　成功添加 RssToolkit.dll 工具包

18.2.2　订阅.ashx 文件的介绍与创建

.ashx 文件与.aspx 文件类似，.ashx 文件用于写 WebHandler，并且可以通过它来调用 HttpHandler 类。与 aspx 文件相比，省去了.aspx 页面的控件解析以及页面处理的过程。.ashx 文件适合生成供浏览器处理的、不需要回发处理的数据格式，例如用于生成动态文本、动态图片等内容。

建立 ashx 文件的方法如下。

首先，打开一个 Web 项目，然后在任意目录下使用 Visual Studio 2008 解决方案资源管理器，单击鼠标右键，在弹出的菜单中选择“添加新项”命令，在弹出的对话框中选择“文本文件”，

然后在输入文件名位置输入 Rss.ashx，如图 18.5 所示。

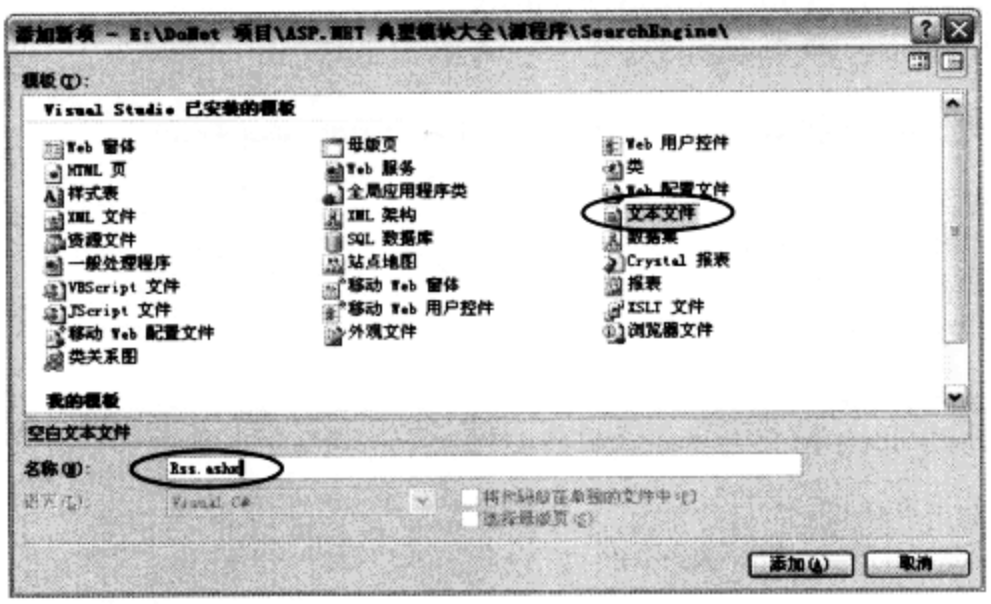


图 18.5　创建 ashx 文件

其次，在创建的 Rss.ashx 文件中编写初始化文件代码，在此以本模块订阅频道为例。实现代码如下。

```
<%@ WebHandler Language="C#" Class="RSS" %>
using System;
using System.Web;
using System.Data;
using System.Data.OleDb;
using RssToolkit;

public class RSS : GenericRssHttpHandlerBase
{
        …省略代码
}
```

18.2.3　创建 RSS 订阅频道

创建 RSS 订阅频道，主要在.ashx 文件中开发，通过 RSS 开发工具包提供的 Channel 集合来定义频道元素、GenericRssElement 类来定义频道内的项，最后通过 Channel.Items.Add 方法将所有频道中的内容添加到频道中。另外，在创建频道中项时，通过 For 循环将数据库中的所有数据读出来添加到频道中的项里。按以上操作开发完成后，通过 IE 浏览器浏览.ashx 文件，XML 文件结构如图 18.6 所示。

图 18.6　RSS 订阅文件

实现代码如下。

例程 1　代码位置：光盘\mr\18\RSS\ rss.ashx

```
public class RSS : GenericRssHttpHandlerBase
{
    protected override void PopulateChannel(string channelName, string userName)
    {
        //添加频道
        Channel["title"] = "Sample Channel";
        //如果频道名称不为空
        if (!string.IsNullOrEmpty(channelName))
        {
            //设置频道名称
            Channel["title"] += " '" + channelName + "'";
        }
        //如果访问用户名不为空
        if (!string.IsNullOrEmpty(userName))
        {
            //设置用户名名称
            Channel["title"] += " (generated for " + userName + ")";
        }
        //设置频道的属性
        Channel["link"] = "~/Default.aspx";
        Channel["description"] = "asp.net技术频道";
        Channel["ttl"] = "10";
        Channel["name"] = channelName;
        Channel["user"] = userName;
        //获取保存到数据库中的订阅频道
        DataSet ds = new DataSet();
        using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
HttpContext.Current.Server.MapPath("rss.mdb")))
        {
            OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_link order by id desc", conn);
            da.Fill(ds);
        }
        GenericRssElement item;
        for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
        {
            //创建一个频道内的项
            item = new GenericRssElement();
            //为项的基本属性赋值
            item["title"] = ds.Tables[0].Rows[i][1].ToString();
            item["description"] = ds.Tables[0].Rows[i][3].ToString();
            item["link"] = ds.Tables[0].Rows[i][2].ToString();
            //将项添加到频道内
            Channel.Items.Add(item);
        }
    }
}
```

18.2.4　读取 RSS 订阅频道

读取 RSS 订阅频道中的所有项如图 18.7 所示，主要通过 RSS 开发工具包中提供的相关类和方法，实现具体步骤如下。

（1）通过 GenericRssChannel 类的 LoadChannel 方法加载订阅频道地址。

（2）步骤 1 的返回值赋值给 GenericRssChannel 类对象 channel。

（3）类对象 channel 调用 SelectItems 方法检索订阅频道的所有项目。

（4）检索订阅频道的所有项目通过 GridView 控件显示出来。

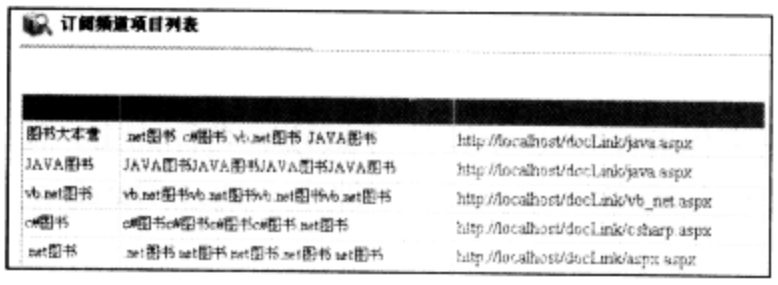


图 18.7　读取 RSS 订阅频道

实现关键代码如下。

例程 2　代码位置：光盘\mr\18\RSS\Default.aspx.cs

```
//创建一个频道对象,然后装载订阅地址
GenericRssChannel channel = GenericRssChannel.LoadChannel(ds.Tables[0].Rows[0][2].ToString());
//为GridView绑定数据源，数据源来自订阅频道中的所有项目
GridView1.DataSource = channel.SelectItems();
GridView1.DataBind();
```

18.3　RSS 在线订阅与阅读模块主页设计

RSS 在线订阅与阅读模块主页主要包括一些导航链接（如在线订阅、添加订阅、管理订阅），以及可以选择订阅频道来阅读订阅的文章内容，如图 18.8 所示。

图 18.8　RSS 在线订阅与阅读主页

18.3.1　页面设计

RSS 在线订阅与阅读模块主页中涉及主要控件及控件用途如表 18.1 所示。

表 18.1　　**RSS 在线订阅与阅读模块主页中主要控件及用途**

控　件	ID 属性	主 要 属 性	用　途
DataList	DataList1	编辑 DataList 模板，在 ItemTemplate 控件中添加 LinkButton 控件，设置 LinkButton 控件的 CommandName 属性值为 Edit，当 DataList 控件绑定数据时，通过 LinkButton 控件实现导航功能（选择订阅频道）	选择订阅频道
GridView	GridView1	编辑 GridView 列，添加可用字段 BoundField，用来显示订阅文章标题，DataField 属性为 Title	显示订阅频道的内容
		编辑 GridView 列，添加可用字段 BoundField，用来显示订阅文章描述，DataField 属性为 description	
		编辑 GridView 列，添加可用字段 HyperLinkField，用来链接订阅文章地址，DataNavigateUrlFields 属性值为 Link，DataTextField 属性值为 link，Target 属性为_blank	

18.3.2 实现代码

声明全局变量 id，用来保存 DataList 控件订阅频道名称的 ID 值。在 Page_Load 事件中获取数据库中所有的订阅频道并绑定到 DataList 控件中，通过 DataList 控件显示订阅频道。实现代码如下。

例程 3　代码位置：光盘\mr\18\RSS\Default.aspx.cs

```
static string id = "";
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //获取数据库中数据订阅频道，绑定到DataList控件上
        DataSet ds = new DataSet();
         using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" + HttpContext.Current.Server.MapPath("rss.mdb")))
        {
            OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_rss order by id desc", conn);
            da.Fill(ds);
        }
        DataList1.DataSource = ds;
        DataList1.DataKeyField = "ID";
        DataList1.DataBind();
    }
}
```

当用户在 DataList 控件选择订阅频道后，那么将在 GridView 控件上显示频道的中所有订阅内容。实现代码如下。

例程 4　代码位置：光盘\mr\18\RSS\Default.aspx.cs

```
protected void DataList1_EditCommand(object source, DataListCommandEventArgs e)
{
    //获取编辑数据库行的ID
    id = DataList1.DataKeys[e.Item.ItemIndex].ToString();
    BindGridView(id);
}
```

当订阅频道中的内容非常多时，不可能在一个页面内显示所有的订阅内容，根据笔者多年的经验，这里必须分页显示订阅内容。GridView 控件的 PageIndexChanging 事件主要用来在切换到其他页内容时触发，在 GridView1_PageIndexChanging 中实现分页代码如下。

例程 5　代码位置：光盘\mr\18\RSS\Default.aspx.cs

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    GridView1.PageIndex = e.NewPageIndex;
    BindGridView(id);
}
```

自定义 BindGridView 方法，用来绑定指定订阅频道中的订阅内容，其中通过 GenericRssChannel 类的 LoadChannel 方法加载订阅频道地址，然后利用 channel.SelectItems() 方法检索订阅中所有的项，将检索结果绑定到 GridView 控件中显示。实现代码如下。

例程 6　代码位置：光盘\mr\18\RSS\Default.aspx.cs

```
private void BindGridView(string id)
{
    DataSet ds = new DataSet();
    //根据ID查询订阅地址
    using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
HttpContext.Current.Server.MapPath("rss.mdb")))
    {
        OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_rss where id=" + id, conn);
```

```
            da.Fill(ds);
        }
        //创建一个频道对象,然后装载订阅地址
        GenericRssChannel channel = GenericRssChannel.LoadChannel(ds.Tables[0].Rows[0][2].ToString());
        //为GridView绑定数据源，数据源来自订阅频道中的所有项目
        GridView1.DataSource = channel.SelectItems();
        GridView1.DataBind();
    }
```

18.4 添加 RSS 订阅频道

当今的互联网上，像新闻网站、Blog 网站等这类网站都提供 RSS 订阅频道，为用户提供在线订阅。本模块也为用户提供了添加多个订阅频道的功能，如图 18.9 所示。

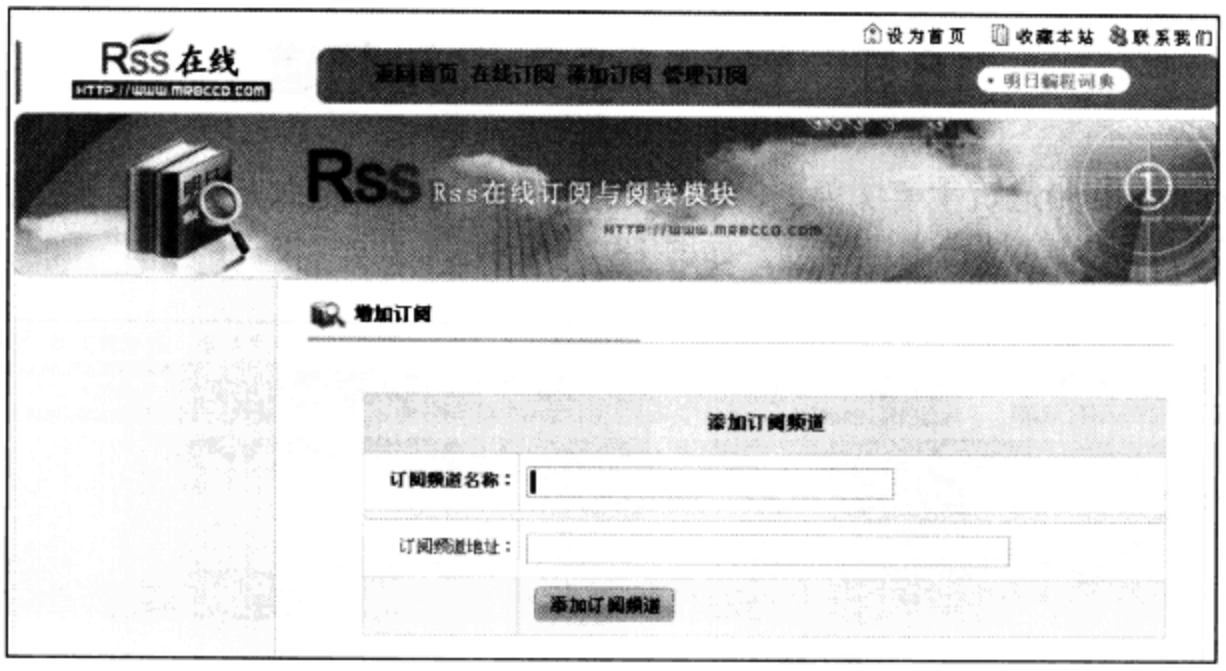


图 18.9 添加 RSS 订阅频道

18.4.1 页面设计

添加 RSS 订阅频道页中涉及的主要控件及控件用途如表 18.2 所示。

表 18.2 **RSS 在线订阅与阅读模块主页中主要控件及用途**

控　件	ID 属性	主 要 属 性	用　途
TextBox	txtTitle	默认	输入订阅频道名称
TextBox	txtAddress	默认	输入订阅频道地址
RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate 属性值为 txtTitle，ErrorMessage 属性值为“必填项”	验证订阅频道名称输入文本框不能为空
RegularExpression Validator	RegularExpression Validator1	ControlToValidate 属性值为 txtAddress，ErrorMessage 属性值为“格式不正确”	验证订阅频道地址文本框文本框输入的订阅地址
ImageButton	imgBtnRssAdd	ImageUrl 属性值为 images/an_13.jpg	执行订阅频道添加

18.4.2 实现代码

在添加 RSS 订阅频道页面中，主要完成的就是 RSS 订阅频道的添加，并没有其他辅助功能。在页面中的相关文本框输入订阅频道的名称和地址，单击“添加订阅频道”按钮，将订阅信息添加到数据库中。实现代码如下。

例程 7　代码位置：光盘\mr\18\RSS\addRss.aspx.cs

```
protected void imgBtnRssAdd_Click(object sender, ImageClickEventArgs e)
{
    using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
HttpContext.Current.Server.MapPath("rss.mdb")))
    {
        OleDbCommand cmd = new OleDbCommand("insert into tb_rss (channelName,channelLink) values ('" +
txtTitle.Text + "','" + txtAddress.Text + "')", conn);
        conn.Open();
        if (cmd.ExecuteNonQuery() > 0)
            Response.Write("添加成功！");
        conn.Close();
    }
}
```

18.5　管理 RSS 订阅频道

管理 RSS 订阅频道页主要实现编辑订阅频道、删除订阅频道以及分页显示数据，另外，当鼠标移动某数据行时，该行高量显示，如图 18.10 所示。

图 18.10　管理 RSS 订阅频道

18.5.1　页面设计

管理 RSS 订阅频道的页面设计，主要通过 GridView 控件实现。GridView 控件可以通过列表的形式显示所有订阅频道，而且 GridView 附带编辑、删除等功能，不过使用这些功能还需要编写代码实现，这里值得注意的是，如果使用 DataSource 数据源绑定 GridView 控件，GridView 控件编辑和删除功能可以直接使用。

下面开始编辑 GridView 控件，用于显示数据和编辑管理订阅频道。在 GridView 属性对话框中选择 Columns 属性，在 Columns 属性值文本框中单击 [...] 按钮弹出如图 18.11 所示对话框，在该对话框中添加 BoundField 字段，用于显示指定列数据，前提必须设置 DataFile 属性（绑定数据库字段），设置 BoundField 字段后，还需要添加 CommandField 字段中的编辑和删除功能，用户执行订阅频道的管理工作。

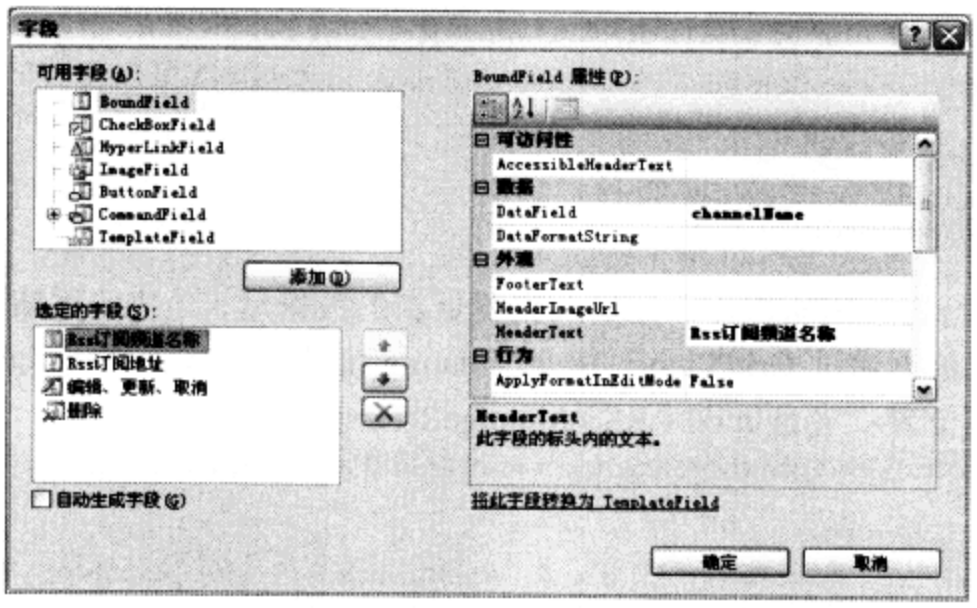


图 18.11　编辑 GridView 控件

18.5.2　实现代码

在 Page_Load 事件下调用自定义方法 BindGridView，用于将订阅频道的名称和地址绑定到 GridView 控件中，并显示出来。实现代码如下。

例程 8　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //调用自定义方法
        this.BindGridView();
    }
}
```

自定义 BindGridView 方法，用于将订阅频道绑定到 GridView 控件显示出来。实现代码如下。

例程 9　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
private void BindGridView()
{
    DataSet ds = new DataSet();
     using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=" + HttpContext.Current.Server.MapPath("rss.mdb")))
    {
        OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_rss order by id desc", conn);
        da.Fill(ds);
    }
    //将查询数据，绑定到GridView控件上显示出来
    GridView1.DataSource = ds;
    GridView1.DataKeyNames = new string[] { "ID" };          //设置主键
    GridView1.DataBind();
}
```

在选定行单击“编辑”按钮，GridView 控件切换到行编辑状态，等待用户进行数据的更新，如图 18.12 所示。

SQL数据库	http://localhost:2205/rss.ashx	编辑	删除
PHP技术	://localhost:2205/rss.ashx	更新 取消	
vb.net技术	http://localhost:2205/rss.ashx	编辑	删除
asp.net技术	http://localhost:2205/rss.ashx	编辑	删除
C#技术	http://localhost:2205/rss.ashx	编辑	删除

图 18.12　GridView 控件编辑状态

GridView 控件实现编辑状态通过 GridView1_RowEditing 事件来完成，实现代码如下。

例程 10　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
    //指定要编辑行的索引值
    GridView1.EditIndex = e.NewEditIndex;
    this.BindGridView();
}
```

当用户在选定行进行编辑操作，且输入完要更新的数据后，单击“更新”按钮后更新数据，GridView 控件更新数据通过 GridView1_RowUpdating 事件来完成。实现代码如下。

例程 11　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    //获取修改的字段值
    string RssName = ((TextBox)(GridView1.Rows[e.RowIndex].Cells[0].Controls[0])).Text;
    string RssLink = ((TextBox)(GridView1.Rows[e.RowIndex].Cells[1].Controls[0])).Text;
    //将修改的数据保存到数据库中
    using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
HttpContext.Current.Server.MapPath("rss.mdb")))
    {
      OleDbCommand cmd = new OleDbCommand("Update tb_Rss set ChannelName='" + RssName + "',ChannelLink='" +
RssLink + "'where id=" + GridView1.DataKeys[e.RowIndex].Value.ToString(), conn);
        conn.Open();
        if (cmd.ExecuteNonQuery() > 0)
            Response.Write("更新成功！");
        conn.Close();
    }
    GridView1.EditIndex = -1;          //恢复编辑状态
    this.BindGridView();
}
```

当用户在选定行进行编辑操作，输入完要更新的数据后，不打算更新数据了，此时单击“取消”按钮，取消编辑状态，实现代码如下。

例程 12　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
protected void GridView1_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
{
    //取消编辑状态
    GridView1.EditIndex = -1;
    this.BindGridView();
}
```

在 GridView 控件上指定的行上单击“删除” 按钮，可以删除指定的数据，在 GridView 控件下的 GridView1_RowDeleting 事件中完成。实现代码如下。

例程 13　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    string id = GridView1.DataKeys[e.RowIndex].Value.ToString();
    //将修改的数据保存到数据库中
    using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
HttpContext.Current.Server.MapPath("rss.mdb")))
    {
        OleDbCommand cmd = new OleDbCommand("delete from tb_Rss where id=" + id, conn);
        conn.Open();
        if (cmd.ExecuteNonQuery() > 0)
            Response.Write("更新成功！");
        conn.Close();
    }
    GridView1.EditIndex = -1;
    this.BindGridView();
}
```

当有上百条订阅频道数据时，需要进行分页显示，在 GridView 控件中实现分页是通过 GridView1_PageIndexChanging 事件完成的。实现代码如下。

例程 14　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    //实现分页，获取新页的索引值
    GridView1.PageIndex = e.NewPageIndex;
    this.BindGridView();
}
```

18.6　程序调试与错误处理

本例中管理 RSS 订阅频道数据主要是在 GridView 控件中实现编辑和删除管理工作的，为了使程序达到人性化标准，在删除订阅频道时弹出友情提示信息，主要在 GridView 控件的 GridView1_RowDataBound 事件下完成，实现代码如下。

例程 15　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        //高亮显示指定行
        e.Row.Attributes.Add("onMouseOver", "Color=this.style.backgroundColor;this. style.backgroundColor='
#336699'");
        e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=Color;");
        //删除指定行数据时，弹出询问对话框
        ((LinkButton)(e.Row.Cells[3].Controls[0])).Attributes.Add("onclick", "return confirm('是否删除当前行
数据！')");
    }
}
```

那么当编辑 GridView 控件中数据时，GridView 将自动编程 3 列，也就是将“删除”暂时取消如图 18.13 所示。这样，运行程序代码 e.Row.Cells[3]将无法找到第 4 列，错误提示如图 18.14 所示。

SQL数据库	http://localhost:2205/rss.ashx	编辑	删除
PHP技术	//localhost:2205/rss.ashx	更新 取消	
vb.net技术	http://localhost:2205/rss.ashx	编辑	删除
asp.net技术	http://localhost:2205/rss.ashx	编辑	删除
C#技术	http://localhost:2205/rss.ashx	编辑	删除

图 18.13　编辑 GridView 控件中数据

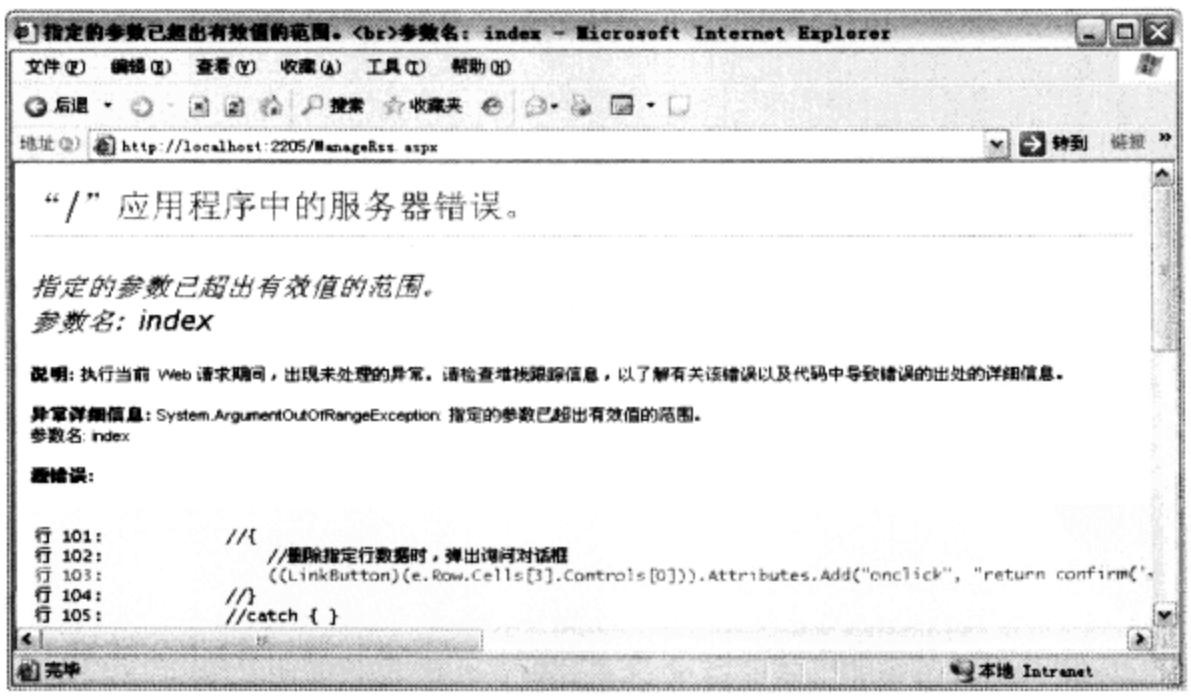


图 18.14　错误提示信息

根据笔者多次对问题的分析与调试，发现当编辑 GridView 控件中数据时，图 18.14 所示第 103 行错误代码在编辑 GridView 中数据时根本不起任何作用，那么需要使用 try...catch 语句忽略此错误。正确代码如下。

例程 16　代码位置：光盘\mr\18\RSS\ManageRss.aspx.cs

```
        protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
        {
            if (e.Row.RowType == DataControlRowType.DataRow)
            {
                //高亮显示指定行
                e.Row.Attributes.Add("onMouseOver",
"Color=this.style.backgroundColor;this.style.backgroundColor='#336699'");
                e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=Color;");
                try
                {
                    //删除指定行数据时，弹出询问对话框
                    ((LinkButton)(e.Row.Cells[3].Controls[0])).Attributes.Add("onclick", "return confirm('是否删除当
前行数据！')");
                }
                catch { }
            }
        }
```

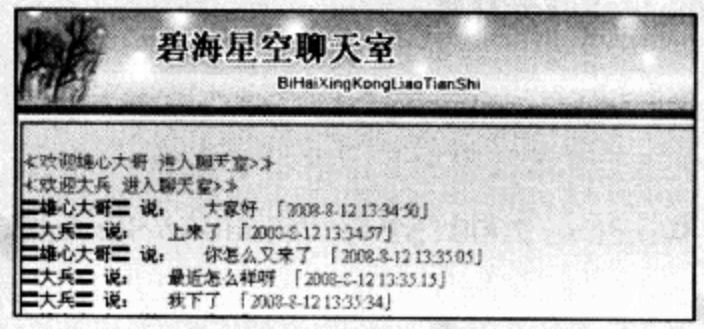
聊天室模块

第19章

实例位置：光盘\mr\19\

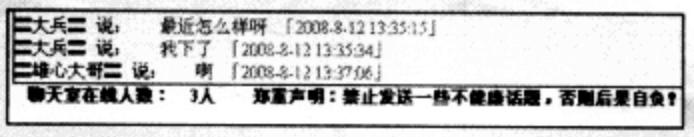
聊天室拉近了人与人之间的距离，是人与人之间交流的互动平台。自推出以来，一直倍受网民的青睐。在本章中将对聊天室的整体框架结构的设计、聊天功能的实现、显示在线用户、输出聊天信息和发送聊天信息等内容进行详细地分析和讲解。通过本章，读者能够学到以下内容。

欢迎用户进入聊天室



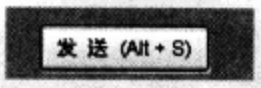
显示当前在线用户

统计在线人数



自定义聊天信息字体颜色

快捷键发送聊天信息



19.1 聊天室概述

19.1.1 概述

网络带给人类的不仅仅是信息，它同样给人类带来了快乐，拉近了人与人之间的距离，聊天室是人与人之间交流的另一个平台，它自推出以来，一直倍受网民的青睐。聊天室是 Web 服务中最常见的服务，除了 BBS 之外，聊天室是使用频率最高的网络服务之一。目前许多网络服务商推出了多媒体聊天室，能够支持语音聊天、视频聊等。本章将介绍一个文本聊天室，功能包括用户注册、登录、聊天、显示在线用户、自动滚屏等。

19.1.2 开发环境

1. 网站开发环境

- 网站开发环境：Microsoft Visual Studio 2008 集成开发环境。
- 网站开发语言：ASP.NET 3.5+C#。
- 网站后台数据库：Access 2000。
- 开发架构：ASP.NET 3.5 + Ajax。
- 开发环境运行平台：Windows XP（SP2）/ Windows 2000（SP4）/ Windows Server 2003（SP1）/Windows Vista。

注意

SP（Service Pack）为 Windows 操作系统补丁。

2. 服务器端

- 操作系统：Windows Vista。
- Web 服务器：IIS 7.0。
- 数据库服务器：Access 2000。
- 浏览器：IE 7.0。
- 网站服务器运行环境：Microsoft .NET Framework SDK v3.5。

3. 客户端

- 浏览器：Internet Explorer 7.0。
- 分辨率：最佳效果 1024×768 像素。

19.2 实现聊天室关键技术

19.2.1 Iframe 框架介绍与应用

聊天室信息页面的结构设计主要通过 Iframe 框架实现。浮动框架 Iframe 是一种特殊的框架结构，它是在浏览的窗口中嵌套另外的网页文件。

语法：

```
<IFRAME></IFRAME>
```

浮动框架标记<IFRAME>的属性如表 19.1 所示。

表 19.1　　浮动框架的属性

属　性	描　述
Src	浮动框架的文件路径
Name	浮动框架的名称
Align	浮动框架水平方向的对齐方式
Width	浮动框架的宽度
Height	浮动框架的高度
Scrolling	是否显示浮动框架滚动条
Frameborder	是否显示浮动框架的边框
Marginwidth	浮动框架左右边缘宽度
Marginheight	浮动框架上下边缘高度

下面是浮动框架属性的详细讲解。

- 浮动框架的文件路径属性 SRC

语法：

```
<IFRAME SRC="file_name">
```

- file_name：指明浮动框架文件的文件名或者其他超链接的网址。
- 浮动框架的名称属性 NAME

语法：

```
<IFRAME SRC="file_name"NAME="frame_name">
```

- frame_name：定义的浮动框架名称。
- 浮动框架的对齐属性 ALIGN

语法：

```
<IFRAME SRC="file_name"ALIGN="left/center/right">
```

- left：居左对齐。
- center：居中对齐。
- right：居右对齐。
- 浮动框架的宽度和高度属性 WIDTH、HEIGHT

语法：

```
<IFRAME SRC="file_name" WIDTH="value" HEIGHT="value">
```

- WIDTH：浮动框架的宽度。
- HEIGHT：浮动框架的高度。
- 浮动框架滚动条显示属性 SCROLLING

语法：

```
<IFRAME SRC="file_name" SCROLLING="value">
```

value 有 3 个取值。

- YES：显示滚动条。
- NO：不显示滚动条。
- AUTO：根据窗口内容决定是否有滚动条。
- 浮动框架边框属性 FRAMEBORDER

语法：

```
<IFRAME SRC="file_name" FRAMEBORDER="value">
```

- value：值为 yes 代表显示框架边框，值为 no 代表隐藏框架边框。
- 浮动框架边缘的宽度和高度属性 MARGINWIDTH、MARGINHEIGHT

语法：

```
<IFRAME SRC="file_name"MARGINWIDTH="value"MARGINHEIGHT="value">
```

- MARGINWIDTH：设定浮动框架左右边缘与边框的宽度。
- MARGINHEIGHT：设定浮动框架上下边缘与边框的高度。

例如，在聊天室模块中定义一个浮动框架页面，并在该页面中声明、定义浮动框架，浏览器显示的内容写在文件 MsgContent.aspx 中。程序代码如下：

```
<iframe id="msgFrame" width="100%" style="height: 440px; visibility: inherit; z-index: 1;
    border-style: groove" src="MsgContent.aspx" scrolling="no" bordercolor="green"   frameborder="0">
</iframe>
```

19.2.2 Ajax 技术应用讲解

Ajax 主要用于局部区域更新。更新范围小，往返数据量小，没有屏幕抖动，用户体验好。

微软在 ASP. NET 框架基础上，创建了 ASP. NET Ajax 技术，能够实现 Ajax 功能。在.NET Framework 3.5 中整合了 Ajax 框架，用户可以直接使用 ASP. NET Ajax 服务器端控件（如 ScriptManager、UpdatePanel、Timer 等控件）来实现 Ajax 功能。

下面介绍开发聊天室模块主要用到的 ScriptManager 控件、UpdatePanel 控件和 Timer 控件。

1. ScriptManager 控件

ScriptManager 控件用来处理页面上的所有 AJAX 组件以及页面局部更新，生成相关的客户端代理脚本以便能够在 JavaScript 中访问 Web Service，所有需要支持 ASP.NET Ajax 的 ASP.NET 页面上有且只能有一个 ScriptManager 控件。在 ScriptManager 控件中可以指定需要的脚本库，或者指定通过 JS 来调用的 Web Service，还可以指定页面错误处理等。ScriptManager 控件的主要属性和方法如表 19.2 所示。

表 19.2　　ScriptManager 控件的主要属性和方法

属性/方法	描　述
AllowCustomError	和 Web.config 中的自定义错误配置区<customErrors>相联系，是否使用它，默认值为 True
AsyncPostBackErrorMessage	异步回传发生错误时的自定义提示错误信息
AsyncPostBackTimeout	异步回传时超时限制，默认值为 90，单位为秒
EnablePartialRendering	是否支持页面的局部更新，默认值为 True，一般不需要修改
ScriptMode	指定 ScriptManager 发送到客户端的脚本的模式，有 4 种模式：Auto，Inherit，Debug，Release，默认值为 Auto
ScriptPath	设置所有的脚本块的根目录，作为全局属性，包括自定义的脚本块或者引用第三方的脚本块。如果在 Scripts 中的<asp:ScriptReference/>标签中设置了 Path 属性，它将覆盖该属性
OnAsyncPostBackError	异步回传发生异常时的服务端处理函数，在这里可以捕获一场信息并作相应的处理
OnResolveScriptReference	指定 ResolveScriptReference 事件的服务器端处理函数，在该函数中可以修改某一条脚本的相关信息如路径、版本等

注意　当设置 EnablePartialRendering 属性为 True 时，表示仅对 UpdatePanel 区域范围内的部分刷新。如果设置为 False，对整个页面刷新。

2. UpdatePanel 控件

UpdatePanel 控件主要用于规划和配置更新区域。UpdatePanel 控件主要属性如表 19.3 所示。

表 19.3　　UpdatePanel 控件主要属性

属　性	说　明
ChildrenAsTriggers	当 UpdateMode 属性为 Conditional 时，UpdatePanel 中的子控件的异步回送是否会引发 UpdatePanle 的更新
RenderMode	表示 UpdatePanel 最终呈现的 HTML 元素。Block（默认）表示 ，Inline 表示
UpdateMode	表示 UpdatePanel 的更新模式，有两个选项：Always 和 Conditional。Always 是不管有没有 Trigger，其他控件都将更新该 UpdatePanel，Conditional 表示只有当前 UpdatePanel 的 Trigger，或 ChildrenAsTriggers 属性为 true 时当前 UpdatePanel 中控件引发的异步回送或者整页回送，或是服务器端调用 Update()方法才会引发更新该 UpdatePanel

注意　当设置 UpdateMode 属性为 Conditional 时，UpdatePanel 自成一区域，不会被自动刷新。只有当区域内触发事件、异步回送或者是整页回送时才会刷新。UpdateMode 的默认属性为 Always，允许被自动刷新。

下面举一个异步回送例子，代码如下：

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server" UpdateMode="Conditional">
  <ContentTemplate>
    <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox> 
  </ContentTemplate>
  <Triggers >
    <asp:AsyncPostBackTrigger ControlID="Button1" />
    <asp:AsyncPostBackTrigger ControlID="Button2"/>
  </Triggers>
</asp:UpdatePanel>
```

3. Timer 控件

Timer 控件是一个定时器控件。可以设定以特定时间规律触发。

- 主要的属性
- Enabled 属性：是否启用。属性值 True 为启用，False 为停用。
- EnableViewState 属性：是否自动保存其状态以用于往返过程。
- Interval 属性：用于定义触发间隔时间。单位是 ms，1000ms=1s。
- 主要事件
- Tick：当到达间隔触发时间后，自动触发的事件。
- 使用技巧
- Timer 放在不同的区域，其应用的范围是不一样的。
- 如果放在所有的 UpdatePanel 外，那么是整个页面的刷新。
- 如果将 Timer 放在 UpdatePanel 内，当 ScriptManager 控件没有设置 EnablePartialRendering 属性或者将其设置 True 时，那么仅在特定时间对所有的 UpdatePanel 区域刷新。当 EnablePartialRendering 设置为 False 时，对整个页面刷新。
- 将一个 Timer 放入任意一个可以接受自动刷新的 UpdatePanel 中，如果 Updatepanel 不想被其他影响，可以设置 UpdateMode 属性为 Conditional。如果涉及 Master 母版页，且内容区域（contentplaceholder）被放在 UpdatePanel 内，那么内容区域都受其最外层环境影响。

注意　在调试时，发现程序在更新时容易报回调时错误，可以做以下设置避免这些情况的发生。在页面<page>中设置 ValidateRequest="false"，在发布环境下，可以设置 web.config 中的<compilation debug="false">。

19.2.3 快捷键发送聊天信息

巧妙地利用快捷键，可以大大加快我们操作计算机的速度。因此，为操作用户设置有针对性地记忆快捷键对操作用户及一个合格的软件是很有好处的。

每天使用计算机一定会和键盘打交道，可以非常熟练地用它敲出各种字母、汉字、数字、符号。除了这些字母、数字、符号键外，键盘上还有一些不能输入字符的按键（Ctrl、Alt等键），这些按键有的可以单独使用，有些则需要和其他键配合使用，通过在键盘上的单键或多键组合实现一些相对鼠标单击复杂的操作，我们通常把这些按键称为快捷键，俗名又叫热键。如在开发聊天室程序时，将发送聊天信息时，使用 Alt+S 快捷键完成发送聊天信息功能，也就是当用户输入聊天信息到文本框后，单击“发送”按钮，才能将聊天信息发送出去，此时使用 Alt+S 快捷键即可完成发送聊天信息功能。

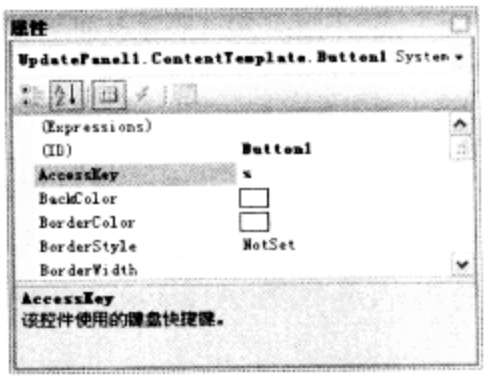


图 19.1 设置 Button 的快捷键

发送聊天信息使用 Button 控件完成，只需要为 Button 控件设置快捷键。Button 控件设置快捷键很简单，设置 AccessKey 即可。如图 19.1 所示为 Button 按钮设置 Alt + S 快捷键，AccessKey 属性值等于 S 即可。

19.2.4 统计在线人数

统计在线人数主要在 Global.asax 文件中编写统计代码完成，Global.asax 文件是 ASP.NET 应用程序文件，提供了一种在一个中心位置响应应用程序级或模块级事件的方法。可以使用这个文件实现应用程序安全性以及其他一些任务。下面给出 3 个事件，用来实现聊天室统计在线人数的功能。

- Application_End：在 HttpApplication 类的最后一个实例被销毁时，该事件被触发。在一个应用程序的生命周期内它只被触发一次。
- Session_Start：在一个新用户访问应用程序 Web 站点时，该事件被触发。
- Session_End：在一个用户的会话超时、结束或他们离开应用程序 Web 站点时，该事件被触发。

下面给出实现统计在线人数的关键代码。

当应用程序启动时，初始化记数器为零。实现代码如下。

例程 1 代码位置：光盘\mr\19\AjaxChatRoom\ Global.asax

```
void Application_Start(object sender, EventArgs e)
{
    // 在应用程序启动时运行的代码
    Application["count"] = 0;
}
```

当新用户启动并运行程序时，计数器加 1。实现代码如下。

例程 2 代码位置：光盘\mr\19\ AjaxChatRoom\ Global.asax

```
void Session_Start(object sender, EventArgs e)
{
    // 在新会话启动时运行的代码
    Application.Lock();
    Application["count"] = int.Parse(Application["count"].ToString()) + 1;
    Application.UnLock();
}
```

当访问用户关闭程序时，计数器减 1。实现代码如下。

例程 3 代码位置：光盘\mr\19\ AjaxChatRoom\ Global.asax

```
    void Session_End(object sender, EventArgs e)
    {
        // 在会话结束时运行的代码
        // 注意: 只有在 Web.config 文件中的 sessionstate 模式设置为
        // InProc 时，才会引发 Session_End 事件。如果会话模式设置为 StateServer
        // 或 SQLServer，则不会引发该事件
...其他功能代码
        Application.Lock();
        Application["count"] = int.Parse(Application["count"].ToString()) - 1;
        Application.UnLock();
...其他功能代码
    }
```

19.2.5 聊天信息自动滚屏

实现聊天信息自动滚屏主要通过 JavaScript 语言实现，在 JavaScript 语言中主要使用 scroll 函数来定位网页滚动条的位置。

scroll 函数用来控制窗口滚动条坐标。

语法：

```
scroll（x , y）
```

参数：

x 代表控制窗口左右方向滚动的尺寸。

Y 代表控制窗口上下方向滚动的尺寸。

实现聊天信息自动滚屏的关键代码如下：

```
<script language="javascript">
    var position=0;
    function scroller()
    {
        if(position!=700)                    //当滚动条位置在纵向滚动条700像素的位置
        {
            position=position+3;             //每次增加3像素
            scroll(0,position);
            clearTimeout(timer);                              //取消setTimeout设置
            var timer=setTimeout("scroller()",1);             //每1秒调用scroller函数
            timer;
        }
    }
</script>
```

进入聊天信息在聊天信息页的源视图，在 Body 元素的 onload 事件下调用滚屏代码如下：

```
<body onload="scroller()" class="bgg" >
```

19.3 聊天室实现过程

19.3.1 登录聊天室

登录聊天室主要用于对进入聊天室的用户进行安全性检查，以防止非法用户进入聊天室。只有合法的用户，才可以进入聊天室进行聊天活动，登录聊天室页面如图 19.2 所示。

图 19.2 聊天室登录页面

1. 页面设计

聊天室登录页中涉及的主要控件及控件用途如表 19.4 所示。

表 19.4 聊天室登录页中主要控件及用途

控 件	ID 属性	主 要 属 性	用 途
TextBox	txtUser	默认	输入登录用户名
TextBox	txtPwd	TextMode 属性值为 Password	输入登录用户密码
ImageButton	imgBtnLogon	ImageUrl 属性值为 images/dl_06.jpg	登录聊天室
ImageButton	imgBtnLogin	ImageUrl 属性值为 images/dl_08.jpg，PostBackUrl 属性值~/BackGround/Login.aspx	进入用户注册页面

2. 代码设计

首先，声明全局数据库操作类对象 bc，在本页中使用 bc 对象调用 SelectSql 方法查询用户名称和密码是否正确。其次，当用户单击“进入聊天室”时，则验证用户名称和密码是否正确，如果正确，进入聊天室，否则重新登录。另外，限制一个用户名称不能重复进行登录。实现代码如下。

例程 4 代码位置：光盘\mr\19\ AjaxChatRoom\ Logon.aspx.cs

```
        //声明数据库操作类对象
        BaseClass bc = new BaseClass();
        protected void imgBtnLogon_Click(object sender, ImageClickEventArgs e)
        {
            DataSet ds = bc.SelectSql("Select * from tb_User where UserName='" + txtUser.Text + "' and UserPwd='" +
txtPwd.Text + "'");
            //查询结果行数大于0 则存在该用户
            if (ds.Tables[0].Rows.Count > 0)
            {
                //判断是否唯一登录
                if (ds.Tables[0].Rows[0][4].ToString() == "1")
                    WebMessageBox.Show("您现在的状态为已经已经登录，如果不能进入聊天室，请在 1 分钟后
重新登录！", "Default.aspx");
                //修改登录状态
                if (!bc.ExecuteSQL("update tb_User set state='1' where UserName='" + txtUser.Text + "' "))
                    WebMessageBox.Show("Error");
                //存储登录信息，并且进入聊天室,
                Session["UserName"] = txtUser.Text;
                  Application.Set("Msg", "" + Application["Msg"] + "<br><font color='#666666' size=2>★欢迎
" + Session["UserName"].ToString() + " 进入聊天室>★</</font>");
                Response.Redirect("Default.aspx");
            }
            else
            {
                WebMessageBox.Show("登录用户名称或密码不正确！");
            }
        }
```

19.3.2 聊天室

在聊天室页面中可以实时获取在线用户，如图 19.3 所示中右侧部分；可以设置用户名称的字体颜色及发送聊天信息的字体颜色；可以使用快捷键<Alt> + <S> 发送聊天信息；可以提示用户进入或离开聊天室的状态。在显示聊天信息区域中，当信息满屏时，信息将自动滚屏显示聊天信息。聊天室页面如图 19.3 所示。

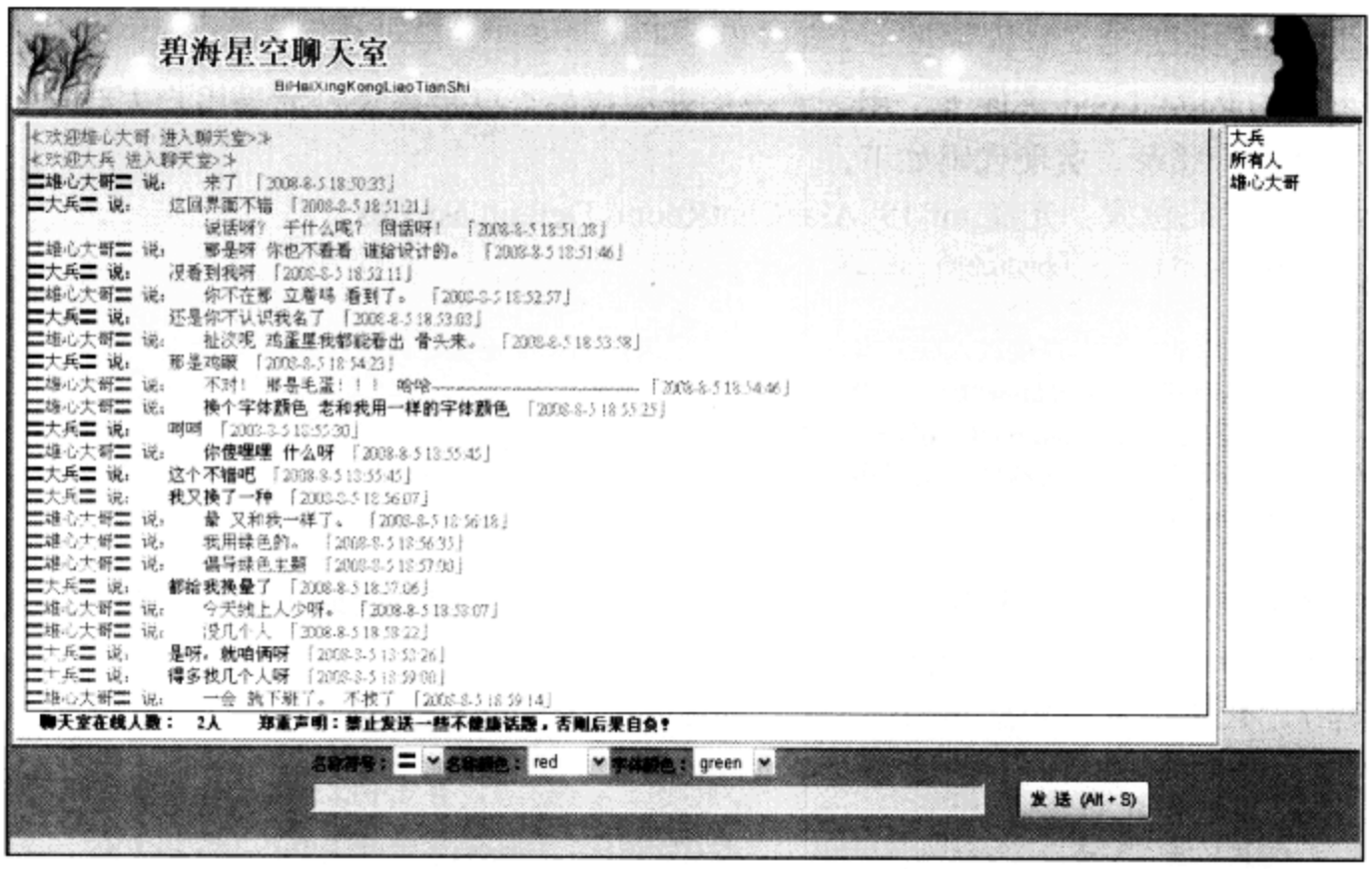


图 19.3 聊天室页面

1. 页面设计

聊天室页中涉及的主要控件及控件用途如表 19.5 所示。

表 19.5 聊天室页中主要控件及用途

控　件	ID 属性	主 要 属 性	用　途
AJAX/ ScriptManager	ScriptManager1	默认	管理 AJAX 控件
AJAX/UpdatePanel	UpdatePanel1	UpdateMode 属性值为 Conditional	局部更新发送聊天信息
DropDownList	ddlSign	设置 Items 属性为控件添加项列表	用于选择用户名称显示符号
DropDownList	ddlSign	设置 Items 属性为控件添加项列表	用于设置用户名称字体颜色
DropDownList	ddlContent	设置 Items 属性为控件添加项列表	用于设置聊天信息字体颜色
TextBox	txtMsg	默认	输入发送的聊天信息
Button	btnSendMsg	Text 属性值为“发送 (Alt + S)”	执行发送聊天信息命令
AJAX/UpdatePanel	UpdatePanel2	UpdateMode 属性值为 Conditional	局部更新在线用户
AJAX/Timer1	Timer1	Interval 属性值为 5000	每隔 5s 获取在线用户
ListBox	lsbOnlineUser	默认	显示在线用户

注意 聊天室页面使用了局部更新的 Ajax 技术，那么在页面上添加控件时需要将控件 ID 为 ddlSign、ddlSign、ddlContent、txtMsg、btnSendMsg 的控件放置在 UpdatePanel1 控件中。将控件 ID 为 Timer1、lsbOnlineUser 的控件放置在 UpdatePanel2 控件中。

在聊天室页面的源视图中，通过 Iframe 框架显示聊天信息内容页，实现代码如下。

例程 5 代码位置：光盘\mr\19\ AjaxChatRoom\ Default.aspx

```
<iframe id="msgFrame" width="100%" style="height: 440px; visibility: inherit; z-index: 1;
                    border-style: groove" src="MsgContent.aspx" scrolling="no" bordercolor="green"
frameborder="0"></iframe>
```

说 明

聊天信息内容页 MsgContent.aspx 在本节中不做介绍，在下一节中做详细介绍。

2. 代码设计

在页面的 Page_Load 事件下，用于实现检查用户是否已经登录，如果用户未登录，则进入到登录页面进行登录。实现代码如下。

例程 6 代码位置：光盘\mr\19\ AjaxChatRoom\ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //防止非法用户直接进入聊天室，如果非法登录，则页面连接定向到登录页
    if (Session["UserName"] == null)
        Response.Redirect("Logon.aspx");
    //聊天信息输入文本框，获得焦点
    txtMsg.Focus();
}
```

聊天用户单击“发送”按钮，或按<Alt> + <S>快捷键，则发送聊天信息，聊天信息主要存储在 Application 对象中。实现代码如下。

例程 7 代码位置：光盘\mr\19\ AjaxChatRoom\ Default.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
    //发送聊天信息
    Application.Set("Msg", Application["Msg"] + "<br> <font color=" + ddlName.Text + " size='2px'> " + ddlSign.Text +
Session["UserName"].ToString() + ddlSign.Text + " 说： </font> <font color=" + ddlContent.Text + " size='2px'>  " + txtMsg.Text +
" </font><font size='2px'>「" + DateTime.Now.ToString() + "」 </font>");
    //清空聊天信息输入文本框
    txtMsg.Text = "";
}
```

在 Ajax 中的 Timer 控件的 Tick 事件下，主要完成每过 5s 获取聊天室当前在线用户，并显示在 ListBox 中的功能。实现代码如下。

例程 8 代码位置：光盘\mr\19\ AjaxChatRoom\ Default.aspx.cs

```
//声明全局数据库操作类对象
BaseClass bc = new BaseClass();
protected void Timer1_Tick(object sender, EventArgs e)
{
    //查询数据，检索出在线用于并在ListBox控件中显示
    lsbOnlineUser.DataSource = bc.SelectSql("Select * from tb_User where state='1' order by id desc");
    lsbOnlineUser.DataTextField = "UserName";
    lsbOnlineUser.DataValueField = "UserName";
    lsbOnlineUser.DataBind();
}
```

19.3.3 显示聊天信息内容页

显示聊天信息内容页嵌入在聊天室中用于显示聊天信息内容和聊天用户的当前状态。该页显示的所有信息都是从 Application 对象中获取的。聊天信息内容页运行结果如图 19.4 所示。

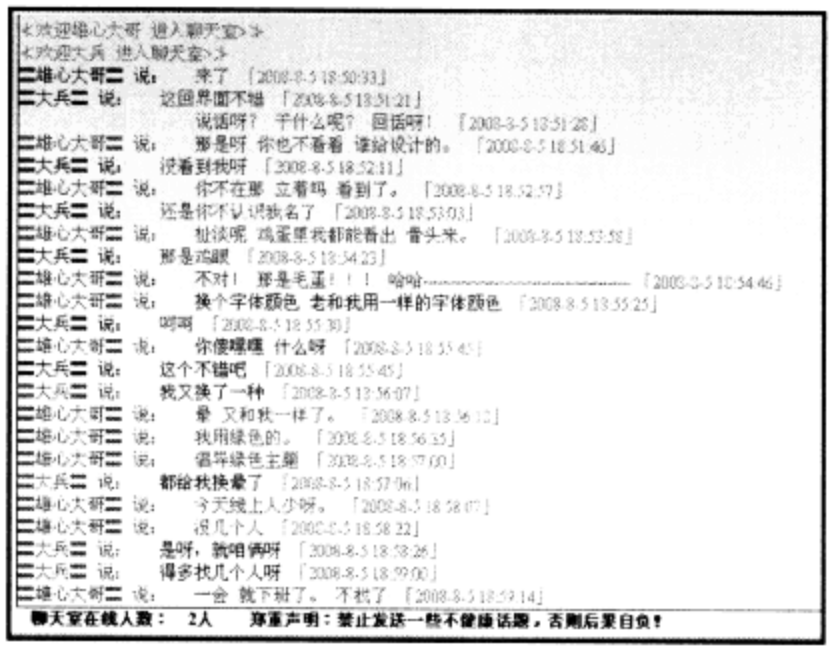


图 19.4 聊天信息内容页

1. 页面设计

聊天信息内容页中涉及的主要控件及控件用途如表 19.6 所示。

表 19.6 聊天信息内容页中主要控件及用途

控 件	ID 属性	主 要 属 性	用 途
Ajax/ ScriptManager	ScriptManager1	默认	管理 Ajax 控件
Ajax/UpdatePanel	UpdatePanel1	默认	局部更新聊天信息
Ajax/Timer1	Timer1	Interval 属性值为 2000	每隔 2s 更新聊天信息
Label	lblMsg	默认	显示所有聊天信息
Label	lblcount	默认	显示当前用户在线个数

2. 代码设计

聊天信息内容页实时显示聊天室的所有聊天信息，使用 Ajax 中 Timer 控件的 Tick 事件，每隔 2s 就获取最新的聊天信息，并且显示统计的在线人数。实现代码如下。

例程 9 代码位置：光盘\mr\19\ AjaxChatRoom\ MsgContent.aspx.cs

```
protected void Timer1_Tick(object sender, EventArgs e)
{
    try
    {
        //获取聊天信息
        lblMsg.Text = Application["Msg"].ToString();
        //统计在线人数
        lblcount.Text = "聊天室在线人数：  " + Application["count"].ToString() + "人  郑重声明：禁止发送一些不健康话题，否则后果自负！";
    }
    catch (Exception ex)
    {
        throw new Exception(ex.Message, ex);
    }
}
```

19.4 程序调式与错误处理

19.4.1 ASP.NET 版本错误

在对程序进行调试时，如果操作系统为 Windows Server 2003 可能出现如图 19.5 所示的提

示错误。

图 19.5　ASP.NET 版本错误的出错页面

原因：IIS 服务器在安装时，Windows Server 2003 系统将默认安装 Microsoft .NET Framework SDK v1.1，那么在默认状态下，IIS 服务器默认支持并运行 Microsoft .NET Framework SDK v1.1 的 Web 程序，无法运行 Microsoft .NET Framework SDK v2.0 的 Web 程序。所以在安装完 Microsoft .NET Framework SDK v2.0 之后一定要将默认站点的 ASP.NET 版本改为 3.5 版本。

解决方法如下。

选择“开始”→“程序”→“管理工具”→“Internet 信息服务（IIS）管理器”，展开菜单右键单击“默认网站”选择“属性”，单击“ASP.NET”选项卡，如图 19.6 所示。在“ASP.NET 版本”的下拉框中选择 ASP.NET 的版本，单击“确定”按钮，设置完成。

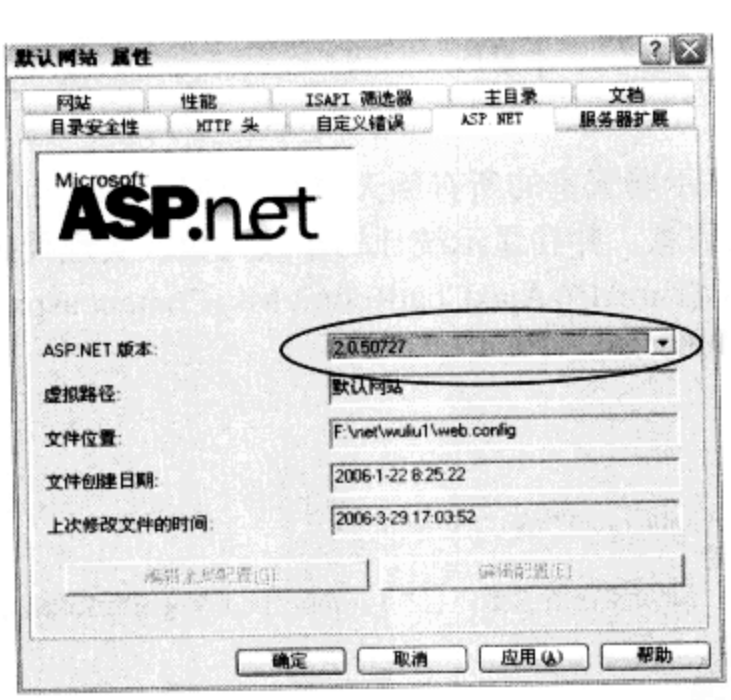


图 19.6　设置 Microsoft .NET Framework SDK v2.0

19.4.2　执行权限错误

在对程序进行调试时，出现如图 19.7 所示的错误。

原因：因为在配置 IIS 过程中，执行权限设置错误。

解决方法：选择“开始”→“程序”→“管理工具”→“Internet 信息服务（IIS）管理器”，右键单击“默认网站”，在弹出菜单中选择“属性”，弹出图 19.8 所示对话框。单击“主目录”选项卡，在“执行权限”下拉框中选择“脚本和可执行文件”，单击“确定”，设置完成。

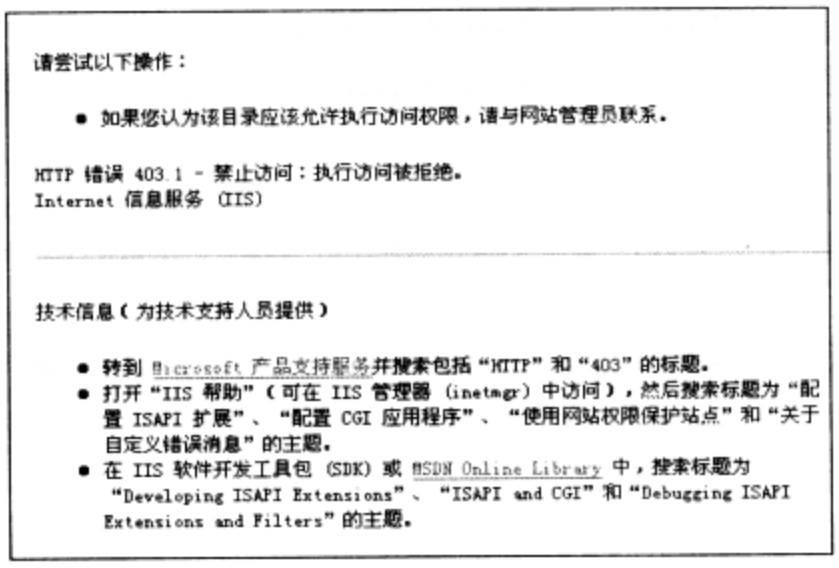


图 19.7　执行权限错误出错页面

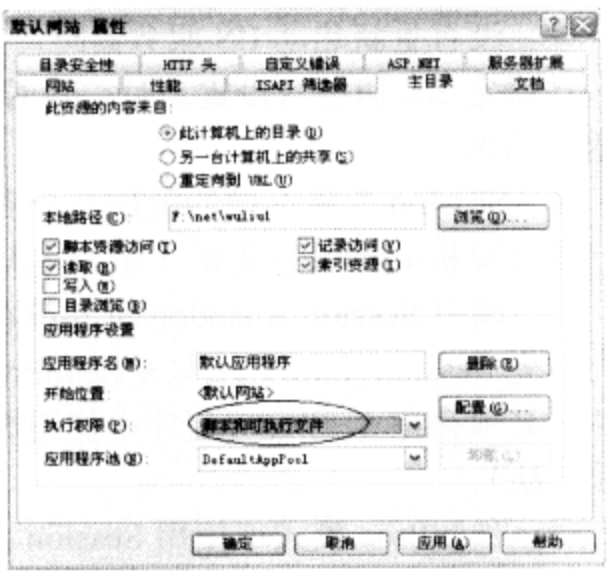


图 19.8　修改 IIS 执行权限

19.5　常见开发技术问题总结

开发聊天室模块程序中，笔者遇到很多开发常见问题，笔者都一一记录下来，希望对读者在程序开发过程当中会有一定的参考价值，下面给出的开发常见技术问题即包括开发本模块遇到的问题，也包括笔者开发其他项目遇到的问题，常见开发技术问题如下。

● 网站显示乱码。

网站出现乱码与默认的编码格式有关，此时可以在网站的 web.Config 配置文件中，统一网站编码格式，代码如下：

```
<?xml version="1.0" encoding="gb2312" ?>
<configuration>
    <system.web>
        <globalization requestEncoding="gb2312" responseEncoding="gb2312" />
    </system.web>
</configuration>
```

● 限制网站上传文件大小。

在开发 ASP.NET 网站中默认上传文件大小不能超过 4MB，如果需要上传大文件，可以在网站的 web.Config 配置文件中进行相关设置，代码如下。

```
<configuration>
    <system.web>
        <httpRuntime   maxRequestLength="10240" />
    </system.web>
</configuration>
```

代码说明如下。

（1）httpRuntime 属性表示 ASP.NET 支持的最大文件上载大小。限制因用户将大量文件传递到该服务器，从而导致的服务器带宽被攻击。httpRuntime 属性指定的大小以 KB 为单位，默认值为 4096KB（4 MB）。

（2）如本例子设置 maxRequestLength="10240"，限制文件上传大小为 10MB。

（3）字段名称不能使用 Password。

在开发 ASP.NET 网站过程中，为数据库表和字段命名时不能用关键字，关键字指的是 ASP.NET 中语法中用到的单词，如 Session、Request、Response、User、Password 等。

● 在 Application_Start 事件中不能使用 Response 对象。

此时在 Application_Star 事件中并不能响应页面，还没有执行到 Request_Begin 事件，可以通过使用 Context.Response 来解决此问题。

- 没有重新启动 IIS 服务器时，是否继续执行 Application_Start 事件？

在确定 IIS 服务器没重启时，看看是否动态更改了网站中 Web.config 配置文件，而导致程序周期结束。

- Session 变量存储的数据在个别计算机上会丢失。

与计算机的环境有关系（如，防火墙或者杀毒软件等），尝试关闭防火墙，方能解决问题。

- 调用 Session.Abandon 时并没有激发 Session_End 方法。

首先 Session_End 方法只支持 InProc（进程内的）类型的 Session。其次要激发 Session_End 方法，必须存在 Session（程序中使用 Session），并且至少要完成一次请求（在这次请求中会调用该方法）。

- 在 InProc 模式下使用 Session 会经常丢失。

该问题通常是由于应用程序被回收导致的，因为当使用进程内 Session 时，Session 是保存在 aspnet_wp 进程中，当该进程被回收 Session 自然也就消失了，确定该进程是否被回收可以通过查看系统的事件查看器获得信息。

- 每次请求的 SessionID 都不相同。

原因是在 Session 里面保存任何信息引起的，即程序中任何地方都没有使用 Session。当 Session 中保存信息之后 SessionID 将一直和浏览器相关，此时的 SessionID 将不会在变化。

- 在 Session_End 中不能使用 Response.Redirect 和 Server.Transfer 方法跳转页面。

Session_End 是一个在服务器内部激发的事件处理函数。它是基于一个服务器内部的计时器的，在激发该事件时服务器上并没有相关的 HttpRequest 对象，因此并不能使用 Response.Redirect 和 Server.Transfer 方法。

- 自定义 HttpHandler 时，不能使用 Session 对象。

在自定义 HttpHandler 的时候，如果希望使用 Session 必须实现下面的两个标记接口中的一个：IRequiresSessionState 和 IReadOnlySessionState，这两个接口没有任何方法需要实现，只是一个标记接口和使用 INamingContainer 接口的方法一样。

- Session 对象在 Application_OnAcquireRequestState 时间中无效。

Session 对象只有在 HttpApplication.AcquireRequestState 事件调用以后才会有效。

- Session 对象在 global.asax 全局文件中生效事件。

Session 对象只有在触发 AcquireRequestState 事件之后有效，在触发该事件后就可以使用 Session 对象了。

- 获得当前 Session 中保存的所有对象。

通过遍历所有的 Session.Keys 来获得。代码如下：

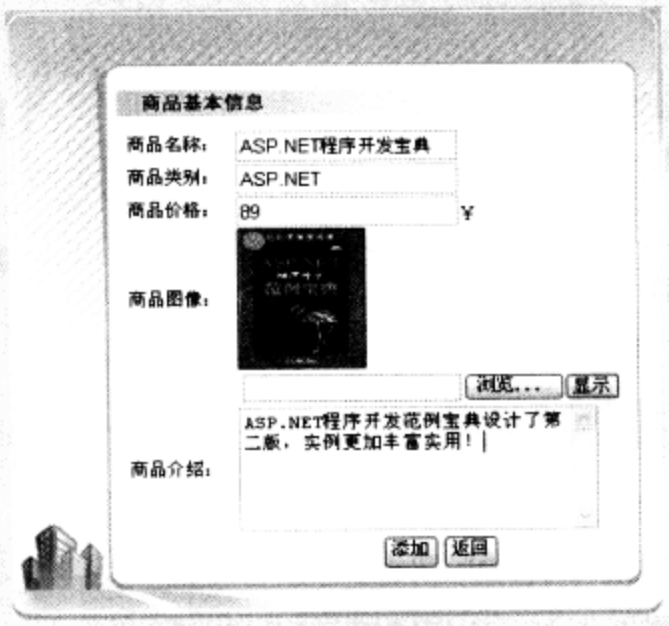
```
ArrayList sessionCollection = new ArrayList();
foreach (string strKey in Session.Keys){
   sessionCollection.Add(Session[strKey]);
}
```

第 20 章 购物车模块

实例位置：光盘\mr\20\

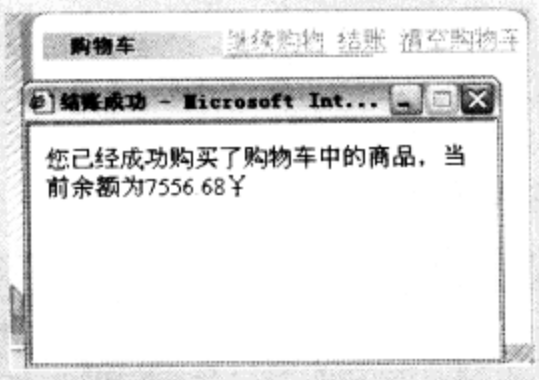
对一个商业企业来说，电子商务网站是其生存的理由和基础，同时也是企业对外展示信息、从事商务活动的窗口和界面。而实现电子商务网站的关键就是购物车功能，购物车主要用于显示及管理用户的购物信息。用户在浏览商品的过程中，如果遇到想要购买的商品，单击商品下方的“购买”按钮，即可将该商品的信息添加到购物车中。通过本章购物车模块的学习，读者能够学到以下内容。

商品信息添加

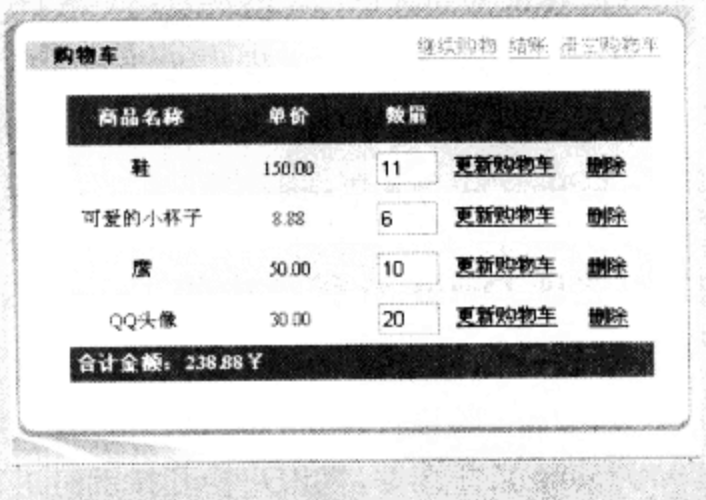


商品详细信息

商品结账



商品购物车



20.1 购物车功能概述

电子商务网站中购物车功能是否合理及安全，将直接影响网站的发展。本模块中允许游客浏览商品，并查看商品信息，但不允许购物。只有登录的用户才可以进行购物。本例可实现的主要功能如下。

- 商品浏览。
- 商品信息查看。
- 继续购物、结账、清空购物车等。
- 后台管理。

购物车页运行结果如图 20.1 所示。

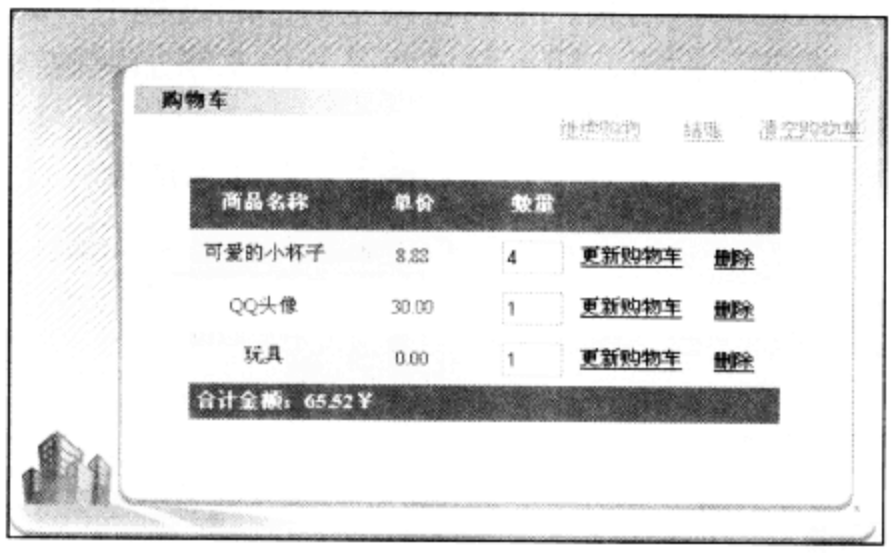


图 20.1 购物车运行结果图

20.2 购物车关键技术

20.2.1 ASP.NET 中使用 Attributes 属性运行 Javascript 脚本

Attributes 属性用于获得不与控件上的属性对应的属性集合，这样设置的属性作为 HTML 属性生成。该集合包含在 Web 服务器控件的开始标记中声明的所有属性的集合。这样一来就可以通过编程方式控制与 Web 服务器控件关联的属性。它可以将属性添加到此集合或从此集合中移除属性。

例如，TextBox 控件文本发生改变时，使用 Attributes 属性运行 Javascript 命令。代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    TextBox1.Attributes["onchange"] = "javascript:alert('Text is My Message!');";
}
```

20.2.2 验证 DataList 控件中的 TextBox 控件允许输入数字

控制 TextBox 控件中允许输入数字可以通过 CompareValidator 验证控件进行验证，也可以通过该控件的 Attributes 属性设置，当用户松开按键时进行验证，如果不为数字则取消操作，运行效果如图 20.2 所示。

在页面加载事件 Page_Load 事件中首先需进行如下设置，代码如下。

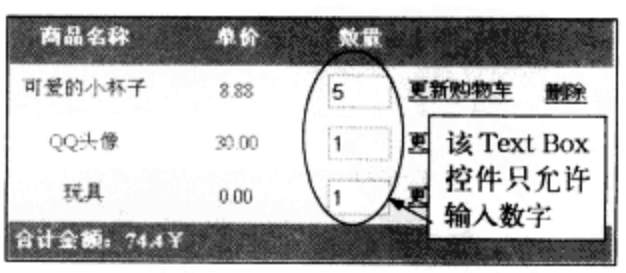


图 20.2 DataList 控件中的 TextBox 控件只允许输入数字

例程 1 代码位置：光盘\mr\20\ShoppingCart\ShoppingCart.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //应用正则表达式屏蔽用户输入非数字字符
TextBox1.Attributes["onkeyup "] = "value=value.replace(/[^\\d]/g,'')";
}
```

如果该 TextBox 控件在 DataList 控件的 ItemTemplate 模板中，那么就需要在当前项被数据绑定到 DataList 控件时引发的 ItemDataBound 事件中进行设置。首先，要在 DataList 控件中找到该控件，然后，设置它的 Attributes 属性。代码如下。

例程 2 代码位置：光盘\mr\20\ShoppingCart\ShoppingCart.aspx.cs

```
protected void dlShoppingCart_ItemDataBound(object sender, DataListItemEventArgs e)
{
    //用来实现数量文本框中只能输入数字
    TextBox txtGoodsNum = (TextBox)e.Item.FindControl("txtGoodsNum");
    if (txtGoodsNum != null)
    {
        //应用正则表达式屏蔽用户输入非数字字符
txtGoodsNum.Attributes["onkeyup"] = "value=value.replace(/[^\\d]/g,'')";
    }
}
```

20.2.3 计算购物车中账户余额

在购物车页中如果用户要购买购物车中的商品，那么可以通过单击“结账”按钮来实现。实际运行效果如图 20.3 所示。

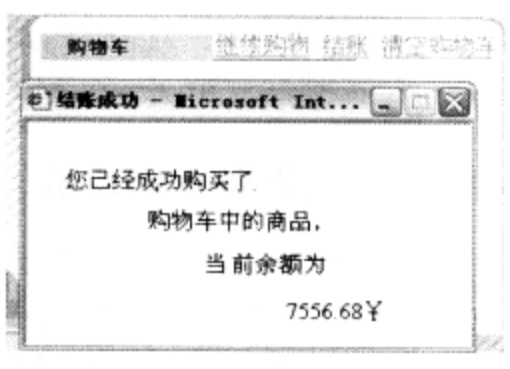


图 20.3 计算购物车中用户账户余额

单击该“结账”按钮时，将触发按钮的 Click 事件。在事件处理代码中，首先判断购物车中是否有商品，如果没有商品，则弹出提示信息；如果有商品，则判断用户的钱袋余额是否可以支付所购买商品的总价，如果余额不足，则不能进行购买，如果余额充足将跳转到 SuccessShop.aspx 页面中。具体实现的代码如下。

例程 3 代码位置：光盘\mr\20\ShoppingCart\ShoppingCart.aspx.cs

```
protected void lnkbtnSettleAccounts_Click(object sender, EventArgs e)
{
    //判断购物车中商品是否为空，如果为空给予相应提示
if (M_str_Count == "")
    {
        Response.Write("<script>alert('您的购物车中没有任何物品!');</script>");
    }
        else
        {
            //调用公共类中的reDs方法执行查询的SQL语句，并返回一个DataSet类型的数据集
DataSet ds = DB.reDs("select Money from tb_User where UserID=" + Session["UserID"].ToString());
            //结账功能
            decimal P_str_Money = Convert.ToDecimal (ds.Tables [0].Rows [0][0].ToString ());
            //判断钱袋中钱数金额
if (P_str_Money < Convert.ToDecimal (M_str_Count))
            {
                Response.Write("<script>alert('您的余额不足，请重新充值后再购买！');</script>");
            }
            else
            {
                bool P_bool_reVal1 = DB.ExSql("Delete from tb_Cart where CartID=" + Session["UserID"]);
              //调用公共类中的ExSql方法执行更新的SQL语句
bool P_bool_reval2 = DB.ExSql("update tb_User set Money=Money-"+M_str_Count+" where
```

```
UserID="+Session["UserID"]);
                        if (!P_bool_reVal1 & !P_bool_reval2)
                        {
                            Response.Write("<script>结账失败，请重试！ </script>");
                        }
                        else
                        {
                            //调用自定义方法重新绑定下表格中数据
        Bind();
                        //弹出结账余额窗口
          Response.Write("<script>window.showModalDialog ('SuccessShop.aspx','','dialog Width=300px;dialogHeight = 250px;
status=no;help=no;scrollbars=no');</script>");
                        }
                    }
                }
            }
```

用户结账后将会跳转到 SuccessShop.aspx 页，此页面主要用来显示该用户的账户余额，具体实现的代码如下。

例程 4 代码位置：光盘\mr\20\ShoppingCart\SuccessShop.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
        //调用公共类中的reDs方法返回一个DataSet类型的数据集
DataSet ds = DB.reDs("select Money from tb_User where UserID=" + Session["UserID"].ToString());
        //进行结账计算
        string P_str_Money = ds.Tables[0].Rows[0][0].ToString() ;
        labMessage.Text = "您已经成功购买了购物车中的商品，当前余额为" + P_str_Money + "￥";
    }
```

20.2.4 无刷新验证码技术

验证码技术是保护网站安全的最基本环节，它可以防止非法人员利用注册工具或登录工具来攻击网站（也就是常说的灌水），从而保护网站的安全。本模块主要在登录窗口中应用了验证码技术，该登录窗口中如果用户看不清验证码可单击图片来刷新验证码，而页面实现的是无刷新效果。实际运行效果如图 20.4 所示。

图 20.4 无刷新验证码

本模块中的验证码是通过引用 using System.Drawing 命名空间，利用 Graphice 的 FromImage 方法创建一个画布，并设置画布的宽和高，设置完成后，通过 Graphice 类的 DrawString 方法随机生成的字符串验证码绘制到画布中，绘制验证码的同时，在画布中利用 SetPixel 方法绘制一些色点，以防止非法人员通过机器人软件强行登录。下面介绍在验证码页 CheckCode.aspx 中应用的 FromImage 方法、DrawString 方法和 SetPixel 方法，语法格式及参数说明如下。

- FromImage 方法

语法：

```
public static Graphics FromImage(Image image)
```

- image：从中创建新 Graphics 的 Image。
- DrawString 方法

语法：

```
public void DrawString(string s,Font font,Brush brush,float x,float y)
```

参数说明如下。

- s：要绘制的字符串。
- font:：它定义字符串的文本格式。
- brush：它确定所绘制文本的颜色和纹理。
- x：所制文本的左上角的 *x* 坐标。
- y：所制文本的左上角的 *y* 坐标。
- SetPixel 方法

语法：

```
public void SetPixel (int x,int y,Color color)
```

参数说明如下。

- x：要设置的像素的 *x* 坐标。
- y：要设置的像素的 *y* 坐标。
- clolor：它表示要分配给指定像素的颜色。

下面介绍绘制验证码页 CheckCode.aspx 的主要代码，关键代码如下。

例程 5 代码位置：光盘\mr\20\ShoppingCart\CheckCode.aspx.cs

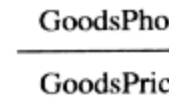
```
private void CreateCheckCodeImage(string checkCode)
{
    if (checkCode == null || checkCode.Trim() == String.Empty)
        return;
    System.Drawing.Bitmap image = new System.Drawing.Bitmap((int)Math.Ceiling((checkCode.Length * 12.5)), 22);
        Graphics g = Graphics.FromImage(image);
        try
        {
            //生成随机生成器
            Random random = new Random();
            //清空图片背景色
            g.Clear(Color.White);
            //画图片的背景噪音线
            for (int i = 0; i < 2; i++)
            {
                int x1 = random.Next(image.Width);
                int x2 = random.Next(image.Width);
                int y1 = random.Next(image.Height);
                int y2 = random.Next(image.Height);
                g.DrawLine(new Pen(Color.Black), x1, y1, x2, y2);
            }
            Font font = new System.Drawing.Font("Arial", 12, (System.Drawing.FontStyle.Bold));
             System.Drawing.Drawing2D.LinearGradientBrush brush = new System.Drawing. Drawing2D. Linear
GradientBrush(new Rectangle(0, 0, image.Width, image.Height), Color.Blue, Color.DarkRed, 1.2f, true);
            g.DrawString(checkCode, font, brush, 2, 2);
            //画图片的前景噪音点
            for (int i = 0; i < 100; i++)
            {
                int x = random.Next(image.Width);
                int y = random.Next(image.Height);
                image.SetPixel(x, y, Color.FromArgb(random.Next()));
            }
            //画图片的边框线
            g.DrawRectangle(new Pen(Color.Silver), 0, 0, image.Width - 1, image.Height - 1);
            System.IO.MemoryStream ms = new System.IO.MemoryStream();
```

```
                image.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
                Response.ClearContent();
                Response.ContentType = "image/Gif";
                Response.BinaryWrite(ms.ToArray());
            }
            finally
            {
                g.Dispose();
                image.Dispose();
            }
        }
```

绘制验证码的页面实现后，在需要验证代码的登录页面中利用 Image 控件将其显示出来，实现验证码的无刷新主要就是这个 Image 控件的 HTML 代码，其源代码如下。

例程 6 代码位置：光盘\mr\20\ShoppingCart\ Login.aspx

```
<img id="Img1" src="CheckCode.aspx" alt="看不清，请单击我！" onclick="this.src=this.src+'?'" style="width: 65px; height: 22px" />
```



注 意 在加入 src 属性时，程序会提示“属性‘src’不是元素‘Image’的有效属性”，这不影响程序的执行。

20.3 数据库设计

本实例采用 SQL Server 2005 数据库系统，数据库系统中创建一个名为 db_NetShop 的数据库。在该数据库中新建以下 3 个表，分别为 tb_User（用户表）、tb_GoodsInfo（商品信息表）、tb_Cart（购物车表）。

创建用户信息表（tb_User），用于保存用户名及密码信息，表结构如表 20.1 所示。

表 20.1 用户信息表 tb_ User 的结构

字段名称	类　型	是否为空	描　述
UserID	int	否	用户编号，标志为自动增量，增量为 1
UserName	int	否	用户名
UserPassword	varchar	否	用户密码
Money	decimal	否	钱袋余额

商品信息表（tb_GoodsInfo），用于保存商品的基本信息，表结构如表 20.2 所示。

表 20.2 商品信息表 tb_ GoodsInfo 的结构

字段名称	类　型	是否为空	描　述
GoodsID	int	否	用来存储商品信息编号，标志为自动增量，增量为 1
GoodsName	varchar	否	用来存储商品名称
GoodsKind	varchar	否	用为存储商品类别
GoodsPhoto	varchar	否	用为存储商品的照片
GoodsPrice	decimal	否	用为存储商品价格
GoodsIntroduce	varchar	否	用为存储商品的描述信息

购物车信息表（tb_Cart），用于保存用户购买商品的信息，表结构如表 20.3 所示。

表 20.3　　购物车信息表 tb_ Cart 的结构

字段名称	类　型	是否为空	描　述
ID	int	否	自动编号，标志为自动增量，增量为 1
CartID	int	否	用来存储购物车编号，这里购物车编号即为用户编号
GoodsID	int	否	用来存储商品信息编号
GoodsName	varchar	否	用来存储商品名称
GoodsPrice	decimal	否	用为存储商品价格
Num,	int	否	用为存储购买商品的数量

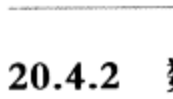
20.4　公共类的封装与设计

20.4.1　Web.Config 配置文件

本实例在 Web.Config 文件中主要是配置连接数据库的字符串。在配置文件中配置连接数据库字符串的好处就是可以省略不同据库的字符串。设置方法如下。

例程 7　代码位置：光盘\mr\20\ShoppingCart\ Web.config

```
<configuration>
    <appSettings>
    <add key="Con" value="server=FDW\MR;Uid=sa;pwd=;database=db_Counter;" />
  </appSettings>
    ...//省略其他代码
</configuration>
```



说　明

这里的用户名 Uid 及密码 pwd 要根据本地机器的用户名及密码对应上。

20.4.2　数据库操作类

在本购物车模块中建立了一个公共类 DB.cs，用来执行数据库操作。该类包括 3 个方法，分别为 GetCon 方法、sqlEx 方法和 reDs 方法，它们的功能说明及设计如下。

1. 数据库连接操作

该方法主要用来连接数据库的，使用 ConfigurationManager 对象的 AppSettings 属性值获取配置节中连接数据库的字符串，实例化 SqlConnection 数据库连接对象，并返回该对象，其代码如下。

例程 8　代码位置：光盘\mr\20\ShoppingCart\App_Code\ DB.cs

```
/// <summary>
/// 配置连接字符串
/// </summary>
/// <returns>返回SqlConnection对象</returns>
public static SqlConnection GetCon()
{
    //创建数据库连接对象，并读取Web.config文件中连接数据库的字符串
return new SqlConnection(ConfigurationManager.AppSettings["GetCon"]);
}
```

2. 执行数据库的添加、删除和修改操作

sqlEx 方法主要使用 SqlCommand 对象执行数据库操作，如添加、修改、删除等。该方法包括一个 string 类型的参数，目的是用来接收具体执行的 SQL 语句。执行该方法后，返回一个布尔类型的变量，如果成功返回 True，否则失败返回 False。代码如下。

例程 9　代码位置：光盘\mr\20\ShoppingCart\App_Code\ DB.cs

```
/// <summary>
/// 执行SQL语句
/// </summary>
/// <param name="P_str_cmdtxt">用来执行的SQL语句</param>
/// <returns>返回是否成功,成功返回True,否则返回False</returns>
public static bool ExSql(string P_str_cmdtxt)
{
    //连接数据库
SqlConnection con = DB.GetCon();
//打开连接
    con.Open();
    //创建SqlCommand命令对象
    SqlCommand cmd = new SqlCommand(P_str_cmdtxt, con);
    try
    {
        //执行SQL 语句并返回受影响的行数
cmd.ExecuteNonQuery();
        return true;
    }
    catch (Exception e)
    {
        return false;
    }
    finally
    {
        //释放连接对象资源
con.Dispose();
    }
}
```

3. 执行数据库的查询操作

reDs 方法主要使用 SqlDataAdapter 对象的 Fill 方法填充 DataSet 数据集。包括一个 string 字符类型的参数，是用来接收具体查询的 SQL 语句。执行该方法后，将返回一个 DataSet 类型的对象 ds。实现的代码如下。

例程 10　代码位置：光盘\mr\20\ShoppingCart\App_Code\ DB.cs

```
/// <summary>
/// 返回DataSet结果集
/// </summary>
/// <param name="P_Str_Condition">用来查询的SQL语句</param>
/// <returns>结果集</returns>
public static DataSet reDs(string P_str_cmdtxt)
{
    //连接上数据库
SqlConnection con = DB.GetCon();
//定义并初使化数据适配器
    SqlDataAdapter da = new SqlDataAdapter(P_str_cmdtxt, con);
    //创建数据集ds
    DataSet ds = new DataSet();
    da.Fill(ds);
    //返回DataSet对象
    return ds;
}
```

20.5　模块设计说明

20.5.1　商品信息浏览页

商品浏览页面（Default.aspx）也是该购物车模块的起始页，主要用来实现用户登录及商品

显示的功能。在该页中用户可以浏览商品信息，如果您的身份为已登录的用户，您还可以将物品放入购物车内。该页运行结果如图 20.5 所示。

图 20.5　商品浏览页面

实现商品浏览页面的步骤如下。

1. 界面设计

在该页面添加 2 个 Panel 控件、2 个 TextBox 控件、1 个 Button 控件、1 个 Label 控件、1 个 DataList 控件和 1 个 HyperLink 控件。具体属性设置如表 20.4 所示。

表 20.4　　Default.aspx 页面中控件属性设置及其用途

控 件 类 型	名　　称	主要属性设置	用　　途
标准/TextBox 控件	txtUserName	无	用来输入用户名
	txtPwd	TextMode 属性设置为“Password”	用于用户输入密码
标准/Panel 控件	pl1	无	用于布局
	pl2	无	用于布局
标准/Button 控件	btnLogin	Text 属性设置为“登录”	用于用户登录验证
标准/Label 控件	labMessage		用于显示登录用户名
数据/DataList 控件	dlGoodsInfo	RepeatColumns 属性设置为“2”	以2列表格形式显示商品信息
标准/HyperLink 控件	hylinkGoback	ImageUrl 属性设置为“～/Image/购物车/进入后台按钮.jpg”	进入后台页面
		NavigateUrl 属性设置为“～/Good sInfo. aspx”	

DataList 控件用于显示商品信息，在它的 ItemTemplate 模板中添加 1 个 Image 控件、3 个 Lable 控件和 2 个控件 LinkButton 控件。具体属性设置如表 20.5 所示。

表 20.5　　ItemTemplate 模板中控件属性设置及其用途

控 件 类 型	控 件 名 称	主要属性设置	用　　途
标准/Image 控件	imgGoodsPhoto	ImageUrl 属性设置为 Eval("GoodsPhoto")	用来显示绑定数据源中的 GoodsPhoto 字段
标准/Label 控件	labGoodsName	Text 属性设置为 Eval("GoodsName ")	用来显示绑定数据源中的 GoodsName 字段
	labGoodsKind	Text 属性设置为 Eval("GoodsKind ")	用来显示绑定数据源中的 GoodsKind 字段
	labGoodzPrice	Text 属性设置为 Eval("GoodsPrice ")	用来显示绑定数据源中的 GoodsPrice 字段

续表

控件类型	控件名称	主要属性设置	用途
标准/LinkButton控件	lnkbtnGoods Describe	CommandArgument 属性设置为 Eval ("GoodsID")	与命令按钮相关联的命令参数为数据源中的 GoodID 字段
		CommandName 属性设置为 describe	命令按钮名称为 describe
		Text 属性设置为详细信息	按钮的显示文本
	lnkbtnBuy	CommandArgument 属性设置为 Eval("GoodsID")	与命令按钮相关联的命令参数为数据源中的 GoodID 字段
		CommandName 属性设置为 buy	命令按钮名称为 buy
		Text 属性设置为“购买”	按钮的显示文本

2. 实现过程

页面初始化时将触发 Page_Load 事件。在该事件中，首先调用 DB 类的 reDs 方法，返回查询结果集，并且作为 DataList 控件的数据源。具体实现的代码如下。

例程 11 代码位置：光盘\mr\20\ShoppingCart\ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //调用公共类中的reDs方法执行查询的SQL语句，并返回DataSet类型的数据集
DataSet ds = DB.reDs("select * from tb_GoodsInfo");
    //指定数据源
dlGoodsInfo.DataSource = ds;
//从数据库中绑定表格数据
    dlGoodsInfo.DataBind();
}
```

单击“详细信息”按钮或“购物”按钮，将触发 DataList 控件的 ItemCommand 的事件。因为，在设置控件属性时，将“详细信息”按钮的 CommandName 属性设置为“describe”，将“购物”按钮的 CommandName 属性设置为“buy”。所以在该事件中，首先判断 CommandName 的值，如果 CommandName 属性值为 describe，则执行打开商品详细信息页面；如果 CommandName 属性值为“buy”，则打开购物车页面。具体实现的代码如下。

例程 12 代码位置：光盘\mr\20\ShoppingCart\ Default.aspx.cs

```
protected void dlGoodsInfo_ItemCommand(object source, DataListCommandEventArgs e)
{
    //获取命令对象名称
if (e.CommandName == "describe")
    {
        //获取用户编号
string P_str_GoodsID = e.CommandArgument.ToString();
        Response.Write("<script>window.open('Describe.aspx?GoodsID=" + P_str_GoodsID + "','','width=637px,
height= 601px')</script>");
    }
//用户触发“购买”命令
    if (e.CommandName == "buy")
    {
        if (Session["UserID"] != null)
        {
            //获取商品编号
string P_str_GoodsID = e.CommandArgument.ToString();
//转向商品购物车页面
            Response.Redirect("~/ShoppingCart.aspx?GoodsID=" + P_str_GoodsID);
        }
        else
        {
```

```
                //用户没有登录则给予提示
Response.Write("<script>alert('您还没有登录，请先登录再购买！');</script>");
            }
        }
    }
```

20.5.2 查看商品详细信息

查看商品详细信息页面（Describe.aspx）主要显示用户所选商品的详细信息。当用户在商品浏览页面时，单击某件商品的详细信息按钮，将打开查看商品详细信息页面。页面的运行结果如图 20.6 所示。

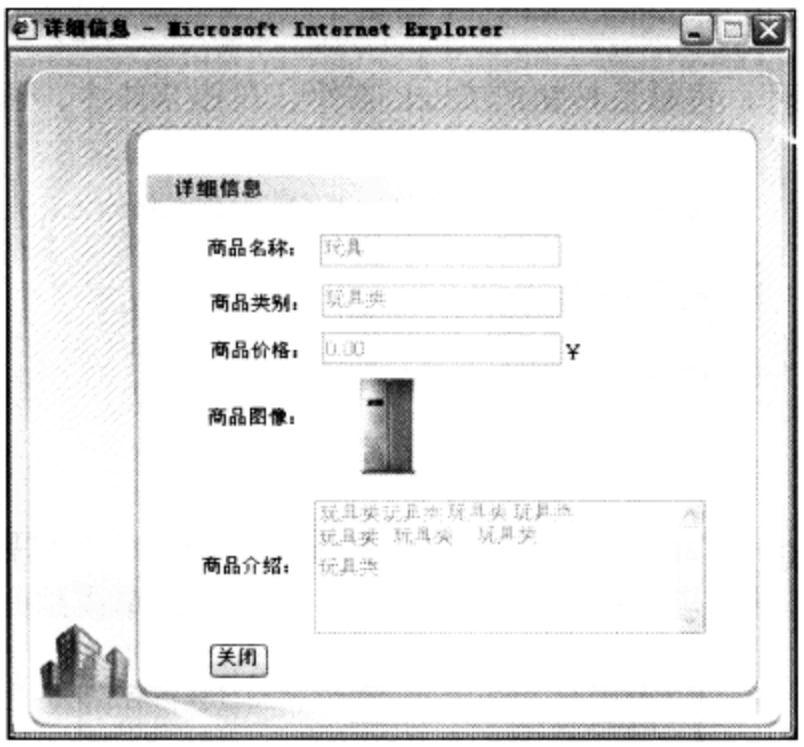


图 20.6 查看商品详细信息页面

实现查看商品详细信息页面的步骤如下。

1. 界面设计

在该页面添加 4 个 TextBox 控件、1 个 Image 控件和 1 个 Button 控件。具体属性设置如表 20.6 所示。

表 20.6 Describe.aspx 页面中控件属性设置及其用途

控 件 类 型	名 称	主要属性设置	用 途
标准/TextBox 控件	txtGoodsName	Enabled 属性设置为“False”	用来显示商品名称
	txtKind	Enabled 属性设置为“False”	用来显示商品种类
	txtGoodsPrice	Enabled 属性设置为“False”	用来显示商品单价
	txtGoodsDesc	Enabled 属性设置为“False”	用来显示商品的描述信息
		TextMode 属性设置为“MultiLine”	
标准/Image 控件	imgGoodsPhoto	无	用于显示该商品的照片
标准/Button 控件	btnClose	Text 属性设置为“关闭”	用于用户登录验证

2. 实现过程

页面初始化时将触发 Page_Load 事件。在该事件中，首先使用 Request 对象获得页面传递的参数“GoodsID”。然后，调用 DB 类的 reDs 方法查询商品编号的商品，并且将查询到的商品信息显示在 TextBox 控件和 Image 控件中。具体实现的代码如下。

例程 13　代码位置：光盘\mr\20\ShoppingCart\ Describe.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //获取传过来的商品编号
string P_str_GoodsID=Request["GoodsID"];
//调用公共类中的reDs方法执行查询SQL语句，并返回DataSet类型的数据集ds
    DataSet ds = DB.reDs("select * from tb_GoodsInfo where GoodsID=" + P_str_GoodsID);
    //分别绑定商品名称、类别、价格、商品图像和商品介绍
    txtGoodsName.Text = ds.Tables[0].Rows[0][1].ToString();
    txtKind.Text = ds.Tables[0].Rows[0][2].ToString();
    imgGoodsPhoto.ImageUrl = ds.Tables[0].Rows[0][3].ToString();
    txtGoodsPrice.Text = ds.Tables[0].Rows[0][4].ToString();
    txtGoodsDesc.Text = ds.Tables[0].Rows[0][5].ToString();
}
```

单击“关闭”按钮时，将触发按钮的 Click 事件。在该事件中，通过 javascript 脚本关闭当前窗体。代码如下。

例程 14　代码位置：光盘\mr\20\ShoppingCart\ Describe.aspx.cs

```
//关闭窗口
protected void btnClose_Click(object sender, EventArgs e)
{
    Response.Write("<script>window.close();</script>");
}
```

20.5.3　购物车页面

购物车页面（ShoppingCart.aspx）主要将用户选择的商品添加到购物车内，用户可以增加某件购买商品的数量、删除某件商品、继续购物、清空购物车或结账等，页面的运行结果如图 20.7 所示。

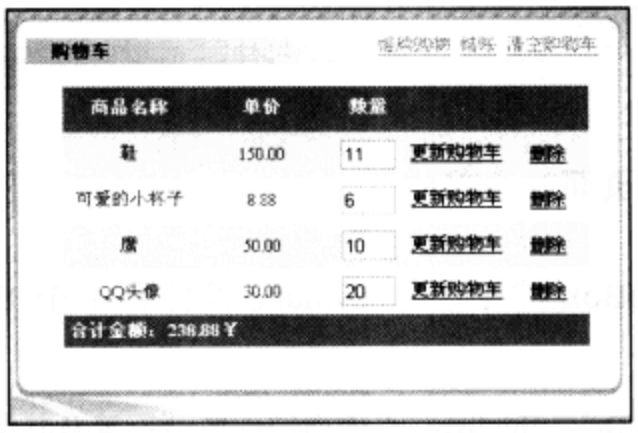


图 20.7　购物车页面

实现购物车页面的步骤如下。

1. 界面设计

在该页面添加 3 个 LinkButton 控件和 1 个 DataList 控件。具体属性设置如表 20.7 所示。

表 20.7　**Describe.aspx 页面中控件属性设置及其用途**

控 件 类 型	名　　称	主要属性设置	用　　途
标准/LinkButton 控件	lnkbtnContinue	Text 属性设置为“继续购物”	跳转到 Default.aspx 页
	lnkbtnClear	Text 属性设置为“清空购物车”	用来显示商品种类
	lnkbtnSettleAccounts	Text 属性设置为“结账”	对购物车中的商品进行支付
数据/DataList 控件	dlShoppingCart	无	显示购物车中的商品信息

DataList 控件用于显示用户添加到购物车的商品信息，在 DataList 控件中的 ItemTemplate

模板中添加了 1 个 TextBox 控件、2 个 Lable 控件和 2 个控件 LinkButton 控件。具体属性设置如表 20.8 所示。

表 20.8　　ItemTemplate 模板中控件属性设置及其用途

控 件 类 型	名　　称	主要属性设置	用　　途
标准/ TextBox 控件	txtGoodsNum	Text 属性设置为 Eval("Num ")	用来显示绑定数据源中的 Num 字段
标准/Label 控件	labGoodName	Text 属性设置为 “Eval("GoodsName ")”	用来显示绑定数据源中的 Good sName 字段
	labGoodsPrice	Text 属性设置为 Eval("GoodsPrice ")	用来显示绑定数据源中的 Good sKind 字段
标准/LinkButton 控件	lnkbtnUpdateCart	CommandArgument 属性设置为 Eval ("GoodsID")	与命令按钮相关联的命令参数为数据源中的 GoodID 字段
		CommandName 属性设置为 updateNum	命令按钮名称为 updateNum
		Text 属性设置为 “更新购物车”	按钮的显示文本
	lnkbtnDel	CommandArgument 属性设置为 Eval ("GoodsID")	与命令按钮相关联的命令参数为数据源中的 GoodID 字段
		CommandName 属性设置为 delete	命令按钮名称为 delete
		Text 属性设置为 “删除”	按钮的显示文本

2. 实现过程

已登录用户在 Default.aspx 页面中单击某件商品的” 购买” 按钮时，将打开 ShoppingCart.aspx 购物车页。在该页的 Page_Load 事件中，首先，设置购物车的编号，这里以用户编号作为购物车编号，并获得页面间传递的参数商品编号；然后，判断该用户的购物车内是否已经存在该商品，如果存在，则该商品的数量加 1，如果不存在，则在购物车内添加一条该商品的信息；最后，调用 Bind 方法，将购物车中的信息显示在 DataList 控件中。具体实现的代码如下。

例程 15　代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //向购物车中添加商品，如果购物车中已经存在该商品，则商品数量加 1，如果是第一次购买，则向购物车中添加一条商品信息
        string P_str_CartID = Session["UserID"].ToString();
        string P_str_GoodsID = Request["GoodsID"];
        //调用公共类中的reDs执行SQL查询语句，并返回一个DataSet类型的数据集
        DataSet ds = DB.reDs("select count(*) from tb_Cart where CartID=" + P_str_CartID + "and GoodsID=" + P_str_GoodsID);
        if (ds.Tables[0].Rows[0][0].ToString() == "0")
        {
            DataSet ds1 = DB.reDs("select GoodsName,GoodsPrice from tb_GoodsInfo where GoodsID=" + P_str_GoodsID);
            //读出数据集中的商品名称、价格和数量
            string P_str_GoodsName = ds1.Tables[0].Rows[0][0].ToString();
            string P_str_GoodsPrice = ds1.Tables[0].Rows[0][1].ToString();
            string P_str_Num = "1";
            //调用公共类中的ExSql方法执行SQL插入操作
            DB.ExSql("insert into tb_Cart values(" + P_str_CartID + "," + P_str_GoodsID + ",'" + P_str_GoodsName + "'," + P_str_GoodsPrice + "," + P_str_Num + ")");
        }
        else
        {
```

```
                //调用公共类中的ExSql方法执行更新购物车操作
        DB.ExSql("update tb_Cart set Num=Num+1 where CartID=" + P_str_CartID + "and GoodsID=" + P_str_GoodsID);
            }
            //显示购物车中的商品信息
            Bind();
        }
    }
```

以下操作是用户在购物车页可操作的功能，如更新购物车中商品数量、清空购物车功能等。

- 更改购买商品数量

在购物车页面中，用户还可以在文本框中更改购买商品的数量，该文本框中只允许输入数字，具体使用方法请参见本章的关键技术详解，这里就不在复赘。更改数量后，单击“更新购物车”按钮，以当前数量更新数据库。代码如下。

例程 16 代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
//更新购物车
protected void dlShoppingCart_ItemCommand(object source, DataListCommandEventArgs e)
{
    if (e.CommandName == "updateNum")
    {
        string P_str_Num = ((TextBox)e.Item.FindControl("txtGoodsNum")).Text;
        //调用公共类中的ExSql方法执行更改购买商品的数量操作
        bool P_bool_reVal = DB.ExSql("update tb_Cart set Num=" + P_str_Num + "where CartID=" + Session["UserID"]
+ "and GoodsID=" + e.CommandArgument.ToString());
        if (P_bool_reVal)
            //调用自定义方法重新绑定表格中数据
            Bind();
    }
}
```

- 删除购物车内某件商品的实现

当用户已经将商品添加到购物车中，如果不想购买某件商品，可以通过单击“删除”按钮来实现。当用户单击该按钮将触发 DataList 控件的 DeleteCommand 事件。具体实现的代码如下。

例程 17 代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
//删除购物车中的商品
protected void dlShoppingCart_DeleteCommand(object source, DataListCommandEventArgs e)
{
    //调用公共类中的ExSql方法执行删除购物车中商品操作
    bool P_bool_reVal = DB.ExSql("Delete from tb_Cart where CartID=" + Session["UserID"]+" and GoodsID="+e.
CommandArgument.ToString ());
    if (!P_bool_reVal)
        Response.Write("<script>删除失败，请重试！</script>");
    Else
        //调用自定义方法Bind()重新绑定表格中数据
        Bind();
}
```

注意 单击删除按钮之前将该按钮的 CommandName 属性设置为“delete”。

在完成删除操作前，会弹出删除确认信息。如果选择是，则完成删除操作，否则，不执行删除操作。添加提示消息的代码如下：

```
//删除购物车中的商品时的提示信息
protected void lnkbtnDel_Load(object sender, EventArgs e)
{
    ((LinkButton)sender).Attributes["onclick"] = "javascript:return confirm('你确定要删除该物品吗？')";
}
```

- 继续购物功能的实现

用户单击“继续购物”按钮，将页面跳转到 Default.aspx 页面继续浏览商品。具体实现的代码如下。

例程 18 代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
//继续购物
protected void lnkbtnContinue_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Default.aspx");
}
```

- 清空购物车功能的实现

如果用户不想购买购物车中的任何一件商品，用户可以单击“清空购物车”按钮删除购物车中的所有商品。具体实现的代码如下。

例程 19 代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
//清空购物车
protected void lnkbtnClear_Click(object sender, EventArgs e)
{
    //调用公共类中的ExSql方法执行清空购物车操作
bool P_bool_reVal=DB.ExSql("Delete from tb_Cart where CartID="+Session["UserID"]);
    if (!P_bool_reVal)
        Response.Write("<script>清空失败，请重试！</script>");
    else
        //调用自定义方法Bind()重新绑定表格中数据
Bind();
}
```

同样，在清空购物车之前，也会弹出删除确认信息。添加提示消息的代码如下。

例程 20 代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
//清空购物车时的提示信息
protected void lnkbtnClear_Load(object sender, EventArgs e)
{
    lnkbtnClear.Attributes["onclick"] = "javascript:return confirm('你确定要清空购物车吗？')";
}
```

20.5.4 后台商品管理页

后台管理页面（GoodsInfo.aspx）主要添加商品信息。该页中的每项内容都必须填写。管理员添加商品照片时，会将本地图片上传到服务器中，并显示在 Image 控件上。页面的运行结果如图 20.8 所示。

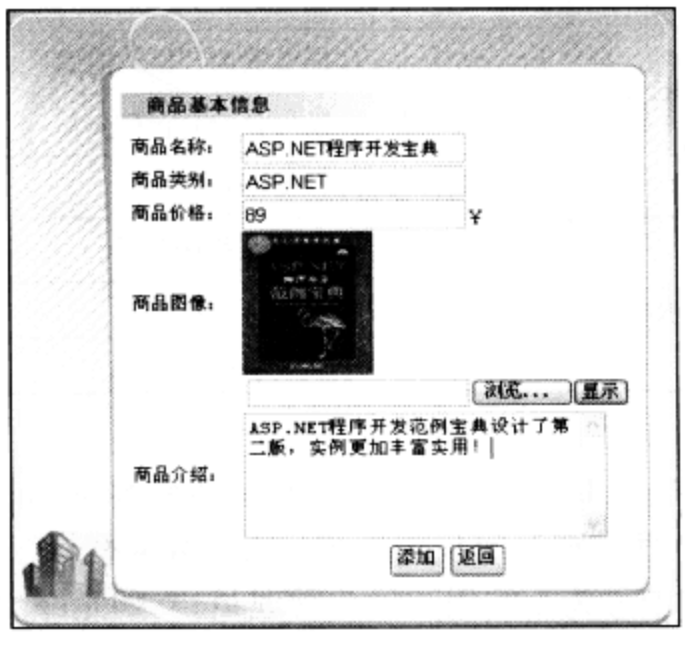


图 20.8 后台管理页面

实现后台管理页面的步骤如下。

1. 界面设计

在该页面添加 4 个 TextBox 控件、1 个 Image 控件、1 个 FileUpLoad 控件、3 个 Button 控件、3 个 RequiredFieldValidator 控件、1 个 CompareValidator 控件和 1 个 Label 控件。具体属性设置如表 20.9 所示。

表 20.9　　GoodsInfo.aspx 页面中控件属性设置及其用途

控件类型	名　称	主要属性设置	用　途
标准/TextBox 控件	txtGoodsName	无	用来显示商品名称
	txtKind	无	用来显示商品种类
	txtGoodsPrice	Text 属性设置为“0”	用来显示商品单价
	txtGoodsDesc	无	用来显示商品的描述信息
标准/Image 控件	imgGoodsPhoto	无	用于显示该商品的照片
标准/FileUpLoad 控件	fulPhoto	无	将商品照片保存到服务器上
标准/Button 控件	btnInsert	Text 属性设置为“添加”	将商品信息添加到数据库
	btnBack	Text 属性设置为“返回”	返回商品浏览页
		CausesValidation 属性设置为 False	
	btnShow	Text 属性设置为“显示”	显示商品照片并保存在服务器上
		CausesValidation 属性设置为 False	
验证/RequiredField Validator 控件	RequiredField Validator1	ControlToValidate 属性设置为 txtGoodsName	验证商品名称不能为空
		ErrorMessage 属性设置为“请输入商品名称！”	
	RequiredField Validator2	ControlToValidate 属性设置为 txtKind	验证商品类别不能为空
		ErrorMessage 属性设置为“请输入商品类别！”	
	RequiredField Validator3	ControlToValidate 属性设置为 txtGoodsDesc	验证商品介绍不能为空
		ErrorMessage 属性设置为“请输入商品介绍！”	
验证/ Compare Validator 控件	Compare Validator1	ControlToValidate 属性设置为 txtGoodsPrice	验证文本框中的内容应为货币类型
		ErrorMessage 属性设置为“格式错误!”	
		Operator 属性设置为 DataTypeCheck	
		Type 属性设置为“Currency”	
标准/ Label 控件	labMessage	Text 属性设置为“请选择图片!”	图片不能为空
		ForeColor 属性设置为 Red	
		Visible 属性设置为 False	

2. 实现过程

初始化页面时，将在 Page_Load 事件中首先通过判断 Session["UserID"]是否为 null，以确定用户是否登录，如果没有登录则返回到购物车主页；如果登录了则判断用户身份。如果用户的自动编号为 1（本例中将编号为 1 的用户设为管理员），则允许添加商品信息；如果不为 1，那么返回到“Default.aspx”。具体实现的代码如下。

例程 21　代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Session["UserID"] == null)
        {
            Response.Write("<script>alert('请先登录！');</script>");
            Response.Redirect("~/Default.aspx");
        }
        else
        {
            if (Session["UserID"].ToString() != "1")
            {
                Response.Write("<script>alert('您还没有此权限！');</script>");
                Response.Redirect("~/Default.aspx");
            }
        }
    }
}
```

若要实现加载商品照片，用户单击 FileUpLoad 控件的“浏览”按钮，将弹出对话框，用户可以通过对话框选择商品照片，或在 FileUpLoad 控件的文本框中直接输入文件的名称。单击“显示”按钮将图片上传到服务上，并显示在 Image 控件中。在保存到服务器之前，还要判断是否选择文件，文件类型是否为图片。具体实现的代码如下。

例程 22　代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
//显示商品图片
protected void btnShow_Click(object sender, EventArgs e)
{
    //获取上载文件的名称
string P_str_name = this.fulPhoto.FileName;
    bool P_bool_fileOK = false;
    if (fulPhoto.HasFile)
    {
        String fileExtension =System.IO.Path.GetExtension(fulPhoto.FileName).ToLower();
        //获取上传图片的格式
        String[] allowedExtensions = { ".gif", ".png", ".jpeg", ".jpg", ".bmp" };
        for (int i = 0; i < allowedExtensions.Length; i++)
        {
            if (fileExtension == allowedExtensions[i])
            {
                P_bool_fileOK = true;
            }
        }
    }
    if (P_bool_fileOK)
    {//将文件保存在相应的路径下
        this.fulPhoto.PostedFile.SaveAs(Server.MapPath("~/Image/") + P_str_name);
         this.imgGoodsPhoto.ImageUrl = "~/Image/" + P_str_name;//将图片显示在Image控件上
    }
    else
    {
         Response.Write("<script>alert('请选择.gif,.png,.jpeg,.jpg,.bmp格式的图片文件!');</script>");
    }
}
```

单击“添加”按钮时，将触发按钮的 Click 事件。在该事件中，首先判断是否已加载图片，如果没加载，则显示错误提示，如果已加载，则调用 DB 类的 ExSql 方法将填写的商品信息添加到数据库中。执行 ExSql 方法后，将返回一个布尔类型的值，如果该值为 True，则清空文本框，否则，弹出操作失败的错误提示。具体实现的代码如下。

例程 23 代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
protected void btnInsert_Click(object sender, EventArgs e)
{
    if (imgGoodsPhoto.ImageUrl != "")
    {
        labMessage.Visible = false;
        bool P_Bool_reVal = DB.ExSql("insert into tb_GoodsInfo values('" + txtGoodsName.Text + "','" + txtKind.Text + "','" +
imgGoodsPhoto.ImageUrl + "','" + txtGoodsPrice.Text + "','" + txtGoodsDesc.Text + "')");
        if (!P_Bool_reVal)
        {
            Response.Write("<script>alert('操作失败，请重试！');</script>");
        }
        else
        {
            txtGoodsName.Text = "";
            txtKind.Text = "";
            txtGoodsPrice.Text = "0";
            txtGoodsDesc.Text = "";
            imgGoodsPhoto.ImageUrl = "";
        }
    }
    else
    {
        labMessage.Visible = true;
    }
}
```

用户单击“返回”按钮，将触发按钮的 Click 事件，在该事件中，调用 Response 对象的 Redirect 方法实现将页面跳转到 Default.aspx 页面中。具体实现的代码如下。

例程 24 代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
protected void btnBack_Click(object sender, EventArgs e)
{
    Response.Redirect("~/Default.aspx");
}
```

20.6 程序错误与调试

在开发完购物车模块后，为了保证程序正常运行，一定要对模块进行单元测试。单元测试在程序开发中非常重要，只有通过单元测试才能发现模块中的不足之处，才能及时的弥补程序中出现的错误，在开发购物车模块时需注意以下问题。

在商品信息浏览页 Default.aspx 页面中登录用户欲购买所选商品单击“购买”按钮跳转到购物车页 ShoppingCart.aspx，如果在购物车页 ShoppingCart.aspx 编写以下代码将会出现如图 20.9 所示的提示错误。

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!IsPostBack)
        {
```

```
                //向购物车中添加商品，如果购物车中已经存在该商品，则商品数量加 1，如果是第一次购买，则向购物车中添加一条商品信息
                string P_str_CartID = Session["UserID"].ToString();
                string P_str_GoodsID = Request["GoodsID"];
                DataSet ds = DB.reDs("select count(*) from tb_Cart where CartID=" + P_str_CartID + "and GoodsID="
+ P_str_GoodsID);
                if (ds.Tables[0].Rows[0][0].ToString() == "0")
                {
                    DataSet ds1 = DB.reDs("select GoodsName,GoodsPrice from tb_GoodsInfo
whereP_str_GoodsID);
                    string P_str_GoodsName = ds1.Tables[0].Rows[1][0].ToString();
                    string P_str_GoodsPrice = ds1.Tables[0].Rows[0][1].ToString();
                    string P_str_Num = "1";
                 DB.ExSql("insert into tb_Cart values(" + P_str_CartID + "," + P_str_GoodsID + ",'" + P_str_GoodsName + "'," +
P_str_GoodsPrice + "," + P_str_Num + ")");
                }
                else
                {
                    DB.ExSql("update tb_Cart set Num=Num+1 where CartID=" + P_str_CartID + "and GoodsID=" +
P_str_GoodsID);
                }
                //显示购物车中的商品信息
                Bind();
            }
```

程序错误出处

图 20.9　编写购物页时出现的错误信息

原因分析如下，出现该错误主要是数组的索引值出现问题。从数组 Rows[i][j]中取值时，应该从第一下标元素开始取值，即 Rows[0][0]，而出现上面的错误就是数组 Rows[i][j]的初始值是 Rows[1][j]，说明数组是从第二个元素开始取值的，所以会在应用程序中提示“确保列表中的最大索引值小于列表的大小”错误信息。

解决方法：

应用 foreach 循环语句将数组中的元素值赋于新的商品数量，从数组$array 中取值时，应该从第一个下标元素（即数组的第 0 个元素）开始取值到数组的最大下标-1 结束，即可正确获取自己的购物车功能，更改后的代码如下。

例程 25　代码位置：光盘\mr\20\ShoppingCart\ ShoppingCart.aspx.cs

```
...//省略部分源代码
if (ds.Tables[0].Rows[0][0].ToString() == "0")
```

```
        {
            DataSet ds1 = DB.reDs("select GoodsName,GoodsPrice from tb_GoodsInfo where GoodsID=" + P_str_GoodsID);
            string P_str_GoodsName = ds1.Tables[0].Rows[0][0].ToString();
            string P_str_GoodsPrice = ds1.Tables[0].Rows[0][1].ToString();
            string P_str_Num = "1";
            DB.ExSql("insert into tb_Cart values(" + P_str_CartID + "," + P_str_GoodsID + ",'" + P_str_GoodsName + "'," +
P_str_GoodsPrice + "," + P_str_Num + ")");
        }
    }
```

第21章 在线银行支付模块

实例位置：光盘\mr\21\

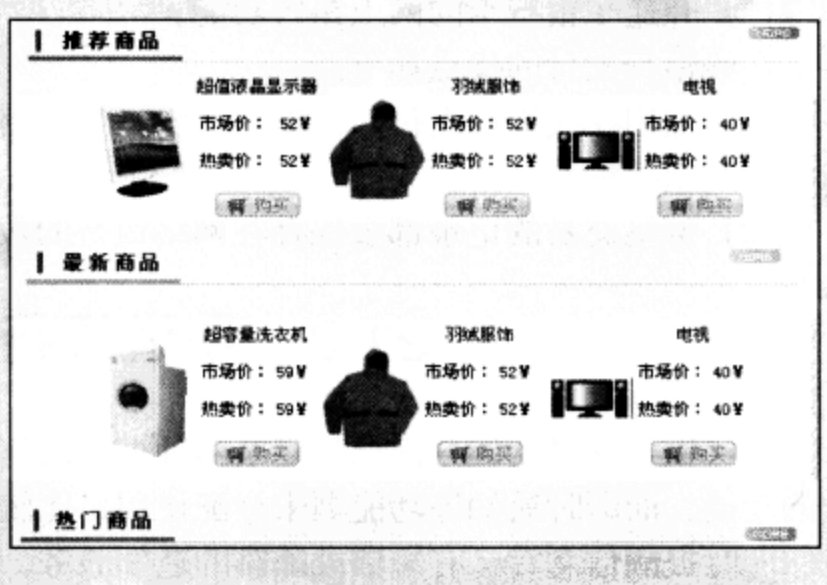
随着网络购物的兴起，在线支付的安全性成为商家和用户关注的问题，如何有效地保障商家和用户的账号信息，并实现账户之间的转账功能，成为在线支付模块发展的核心技术问题。目前大部分的网上商城都采用第三方提供的在线支付模块。通过本章的学习，读者能够学到以下内容。

在线支付方式选择

生成购物车商品列表单

生成收货人详细地址列表单

DataList 控件显示商品信息



21.1 在线银行支付模块概述

建立电子商务网站中最重要的一个环节就是在线支付功能，所谓在线，就是购买商品的整个过程完全在网络上实现，现在在线银行支付已经成为网络购物不可缺少的付款方式。

本模块主要通过一个简单的电子商务平台来介绍一个典型的在线支付——B2C 在线银行支付业务。B2C 在线银行支付业务是指企业（卖方）与个人（买方）通过因特网上的电子商务网站进行交易时，银行为其提供网上资金结算服务的一种业务。

为了拓展银行业务，许多大型银行都在网上开设了网上银行，并提供相应的网上银行支付的接口。下面以工商银行在线支付为例具体讲解。

21.1.1 在线银行支付的安全保障

既然是支付，就牵扯到买家的财产和卖家的利益，保障使用者的账户信息，并实现安全转账，是选择第三方支付的主要因素。

在线银行支付的安全性则银行方面保障。当用户选择了在线银行支付后，在填写银行卡资料时，实际上已经离开支付软件所在的网站，到达了银行的支付网关。国内各大银行的支付网关都采用了国际流行的 SSL 或 SET 方式加密，可以保障用户的信息不被窃取。在线银行支付是在银行的支付网关中完成的，用户不必担心银行卡资料在支付过程中泄露。

21.1.2 在线银行支付的优点

使用在线银行支付功能，需要具备以下两个条件。

（1）拥有支持在线银行支付的银行卡。

（2）开通了银行卡的网上银行功能。

在线银行支付的优点如下。

（1）使用在线银行支付，只需填写账户号和密码，简单快速不需要去银行排队、填表，且不需要汇款手续费。

（2）每笔交易的记录都会保存在网络银行的数据库中，查询方便。

21.2 在线银行支付的流程

在线银行支付功能分简单和复杂两种。比较简单的在线银行支付软件，只实现了付款和收款的功能，而即时通知等功能则未分配使用；比较复杂的在线银行支付软件，既有安全警告信息、即时提醒信息，又有短信或邮箱的通知服务，既方便了买家，又方便了卖家。在线银行支付的流程如图 21.1 所示。

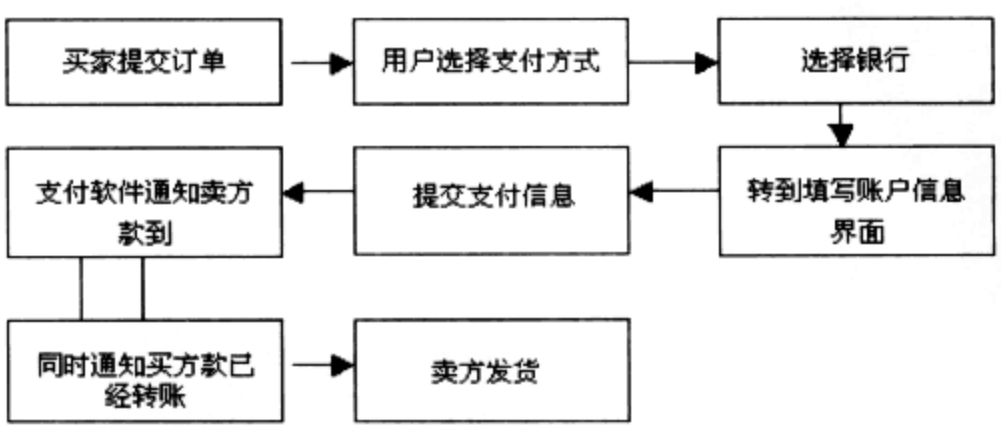


图 21.1 在线银行支付的流程图

21.3 关键技术

21.3.1 商户提交表单接口定义

本在线银行支付模块主要应用了工商银行网上银行 B2C 在线银行支付接口，根据接口数据格式定义通过接口名称和接口版本号来标识，以便将来的扩展。以下数据格式为“1.0.0.0”版的“ICBC_PERBANK_B2C”接口定义。具体介绍如表 21.1 所示。

表 21.1　　B2C 商户提交表单接口定义表

变量名称	变量命名	长度定义	说　明
接口名称	interfaceName	MAX(30)	必填，签名。 取值：“ICBC_PERBANK_B2C”
接口版本号	interfaceVersion	MAX(15)	必填，签名。 取值：“1.0.0.0”
订单号	orderid	MAX(30)	必填，签名。 客户支付后商户网站产生的一个惟一的订单号，该订单号应该在相当长的时间内不重复。工行通过订单号加订单日期来唯一确认一笔订单的重复性
订单金额	amount	MAX(10)	必填，签名。 客户支付订单的总金额，一笔订单一个，以分为单位。不可以为零，必需符合金额标准
支付币种	curType	=3	必填，签名。 用来区分一笔支付的币种，目前工行只支持使用人民币（001）支付。 取值：“001”
商户代码	merID	MAX(20)	必填，签名。 唯一确定一个商户的代码，由商户在工行开户时，由工行告知商户
商城账号	merAcct	MAX(19)	必填，签名。 商城收费入账账号（只能交易时指定）
检验联名标志	verfyJoinFlag	=1	必填，签名。 取值“1”：客户支付时，网银判断该客户是否与商户联名，是则按上送金额扣账，否则展现未联名错误； 取值“0”：不检验客户是否与商户联名，按上送金额扣账
通知类型	notifyType	=2	必填，签名。 在交易转账处理完成后把交易结果通知商户的处理模式。 取值“HS”：在交易完成后实时将通知信息以 HTTP 协议 POST 方式，主动发送给商户，发送地址为商户端随订单数据提交的接收工行支付结果的 URL 即表单中的 merURL 字段；取值“AG”：在交易完成后不通知商户。商户需使用浏览器登录工行的 B2C 商户服务网站，或者使用工行提供的客户端程序 API 主动获取通知信息
接收支付结果信息通知程序地址	merURL	MAX(200)	选填，签名。 使用 HS 通知类型的商户用来接收工行订单支付结果的 URL；银行使用 HTTP 协议 POST 方式向此地址发送通知信息；目前只支持 80 端口。使用“AG”通知类型的商户，该字段可以为空或者不上送该字段；但在签名数据中必须包含此项，取值可为空

续表

变量名称	变量命名	长度定义	说　明
	resultType	=1	选填，签名。取值“0”：无论支付成功或者失败，银行都向商户发送交易通知信息；取值“1”，银行只向商户发送交易成功的通知信息。只有通知方式为 HS 时此值有效，如果使用 AG 方式，可不上送此项，但签名数据中必须包含此项，取值可为空
商品编号	goodsID	MAX(30)	可选择输入
商品名称	goodsName	MAX(60)	可选择输入
商品数量	goodsNum	MAX(10)	可选择输入
已含运费金额	carriageAmt	MAX(10)	可选择输入
商城提示	merHint	MAX(120)	可选择输入
交易日期时间	orderDate	=14	必填，签名。格式为：YYYYMMDDHHmmss 要求在银行系统当前时间的前 1 小时和后 12 小时范围内，否则判定交易时间非法
订单签名数据	merSignMsg	无限制	必填，商户使用工行提供的签名 API 接口和商户证书将交易数据按一定格式进行签名，然后进行 BASE64 编码后得到的字符串（格式单独说明）
商城证书公钥	merCert	无限制	必填，商户用二进制方式读取证书公钥文件后，进行 BASE64 编码后产生的字符串
备注字段 1	Remark1	MAX(100)	可选择输入
备注字段 2	Renark2	MAX(100)	可选择输入

说　明

（1）数据中不能包含“|”“&”“=”，此字符为银行端程序保留字符；中文变量使用 GBK 编码，另请注意与 C2C 接口定义字段名称和大小写有区分。

（2）从商户 Post 过来的数据，参数名的名称必须与上表中完全相同，名称中的字母大小写均要相同，不能进行随意更改（在 form 中的提交按钮<input type= “submit” ...>中 submit 不能有 Name 属性）。此外，如果其他 input 项的 Name 中使用了双引号，如<input type=text name=" merURL " value="http://www.merchant.com/ICBCPay/">，则一定注意在引号内不要包含空格，不要写成“mer URL”，如果拼写错误或者多了空格，将造成数据无法识别，无法正常进行支付。

（3）接口名称和版本号一定要和上表中相同。

（4）商户提交数据中的空格将被认为是有效字符被接收，请商户开发时注意对多余空格的控制。

21.3.2　使用 DataList 控件显示商品数据

在实现在线银行支付前需要在商城中购买商品，以便将生成订单提交到银行在线银行支付中。这里商品信息的显示主要应用到了 DataList 控件，如图 21.2 所示就是使用 DataList 控件分类显示商品信息。

1. DataList 分页绑定数据

DataList 控件是一种数据绑定控件，其与绑定有关的属性及说明如下。

- DataKeyField 属性：获取或设置由 DataSource 属性指定的数据源中的键字段。
- DataKey 属性：获取存储数据列表控件中每个记录的键值。
- DataMember 属性：获取或设置多成员数据源中绑定到数据列表控件的特定数据成员。
- DataSource 属性：获取或设置数据源，该数据源中包含用于填充控件中的项的值列表。

图 21.2 使用 DataList 控件显示商品信息

DataList 控件的分页实现是借助 PagedDataSource 类实现的，该类封装了数据控件的分页属性，其常用属性及说明如表 21.2 所示。

表 21.2 PageDataSource 类中的常用属性

名　称	说　明
AllowPaging	获取是否启用分页
AllowCustomPaging	获取或设置是否启用自定义分页
CurrentPageIndex	获取或设置当前显示页的索引
DataSource	获取或设置用于填充控件中项的数据源
PageSize	获取或设置要在 GridView 控件的每页显示的项数
PageCount	获取显示 GridView 控件中各项所需的总页数
FirstIndexPage	获取页中的第一个索引
IsFirstPage	获取一个值，该值指示当前显示的页是否为首页
IsLastPage	获取一个值，该值指示当前显示的页是否为最后一页

商品信息展示页 goodsList.aspx 页面中实现了上述 DataList 分页绑定数据，具体实现过程如下。

在 goodsList.aspx 页面的 Page_Load 事件中，调用自定义 dlBind()和 deplayTitle()方法绑定 DataList 控件中数据，并实现分页显示数据。该页面的 Page_Load 事件代码如下。

例程 1 代码位置：光盘\mr\21\OnlineBank\ goodsList.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //显示浏览的商品信息
        dlBind();
        //显示当前页浏览商品的位置
        deplayTitle();
    }
}
```

在上述 Page_Load 事件中调用了自定义 deplayTitle()方法，该方法为自定义的无返回值类型方法，该方法主要用来从数据库中查询出符合指定条件的记录，并绑定到 DataList 控件中，然

后通过设置 PagedDataSource 类对象的 AllowPaging 属性为 True，来实现 DataList 控件的分页功能。deplayTitle()方法关键代码如下。

例程 2 代码位置：光盘\mr\21\OnlineBank\ goodsList.aspx.cs

```
public void dlBindPage(int IntDeplay,int IntClass)
{
    //获取数据源的数据表
    SqlCommand myCmd = dbObj.GetCommandProc("proc_GIList");
    //添加参数
    SqlParameter Deplay = new SqlParameter("@Deplay", SqlDbType.Int, 4);
    Deplay.Value = IntDeplay;
    myCmd.Parameters.Add(Deplay);
    //添加参数
    SqlParameter Class = new SqlParameter("@ClassID", SqlDbType.Int, 4);
    Class.Value = IntClass;
    myCmd.Parameters.Add(Class);
    dbObj.ExecNonQuery(myCmd);
    DataTable dsTable = dbObj.GetDataSet(myCmd, "tbGI");
    int curpage = Convert.ToInt32(this.labPage.Text);
    PagedDataSource ps = new PagedDataSource();
    ps.DataSource = dsTable.DefaultView;
    ps.AllowPaging = true; //是否可以分页
    ps.PageSize = 15; //显示的数量
    ps.CurrentPageIndex = curpage - 1; //取得当前页的页码
    this.lnkbtnUp.Enabled = true;
    this.lnkbtnNext.Enabled = true;
    this.lnkbtnBack.Enabled = true;
    this.lnkbtnOne.Enabled = true;
    if (curpage == 1)
    {
        this.lnkbtnOne.Enabled = false;//不显示第一页按钮
        this.lnkbtnUp.Enabled = false;//不显示上一页按钮
    }
    if (curpage == ps.PageCount)
    {
        this.lnkbtnNext.Enabled = false;//不显示下一页
        this.lnkbtnBack.Enabled = false;//不显示最后一页
    }
    this.labBackPage.Text = Convert.ToString(ps.PageCount);
    this.dLGoodsList.DataSource = ps;
    this.dLGoodsList.DataKeyField ="BookID";
    this.dLGoodsList.DataBind();

}
```

当用户单击用于操作分页的 LinkButton 控件时，程序根据当前页码执行指定操作。用于控制分页的 LinkButton 控件的 Click 事件代码如下。

例程 3 代码位置：光盘\mr\21\OnlineBank\ goodsList.aspx.cs

```
protected void lnkbtnOne_Click(object sender, EventArgs e)
{
    this.labPage.Text = "1";
    this.dlBind();
}
protected void lnkbtnUp_Click(object sender, EventArgs e)
{
    this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) - 1);
    this.dlBind();
}
protected void lnkbtnNext_Click(object sender, EventArgs e)
{
```

```
        this.labPage.Text = Convert.ToString(Convert.ToInt32(this.labPage.Text) + 1);
        this.dlBind();
    }
    protected void lnkbtnBack_Click(object sender, EventArgs e)
    {
        this.labPage.Text = this.labBackPage.Text;
        this.dlBind();
    }
```

2. DataList 模板列的应用

DataList 控件最大的特点就是一定要通过模板来定义数据的显示格式。如果想要设计出美观的界面，就需要花费一番心思。正因为如此，DataList 控件显示数据时更具灵活性，开发人员个人发挥的空间也比较大。DataList 控件支持的模板如下。

- AlternatingItemTemplate

如果已定义，则为 DataList 中的交替项提供内容和布局。如果未定义，则使用 ItemTemplate。

- EditItemTemplate

如果已定义，则为 DataList 中当前编辑项提供内容和布局。如果未定义，则使用 ItemTemplate。

- FooterTemplate

如果已定义，则为 DataList 的脚注部分提供内容和布局。如果未定义，将不显示脚注部分。

- HeaderTemplate

如果已定义，则为 DataList 的页眉节提供内容和布局。如果未定义，将不显示页眉。

- ItemTemplate

为 DataList 中的项提供内容和布局所要求的模板。

- SelectedItemTemplate

如果已定义，则为 DataList 中当前选定项提供内容和布局。如果未定义，则使用 ItemTemplate。

- SeparatorTemplate

如果已定义，则为 DataList 中各项之间的分隔符提供内容和布局。如果未定义，将不显示分隔符。

以显示商城“推荐商品”信息的 Default.aspx 页为例，主要应用了 DataList 控件的项目模板列 ItemTemplate 及其数据绑定技术，编写的前台代码如下。

例程 4 代码位置：光盘\mr\21\OnlineBank\ Default.aspx

```
<asp:DataList ID="dLRefine" runat="server" RepeatColumns="3"
    RepeatDirection="Horizontal" OnItemCommand="dLRefine_ItemCommand"
    onselectedindexchanged="dLRefine_SelectedIndexChanged" >
    <ItemTemplate>
        <table style=" height: 120px">
            <tr>
                <td rowspan="5" style="width: 29px">
                 <asp:Image ID="imageRefine" runat="server" ImageUrl ="<%#DataBinder.Eval (Container.DataItem,
"BookUrl")%>"/>
                    </td>
                    <td colspan="2">
                    <asp:LinkButton ID="lnkbtnRName" runat="server" CommandName="detailSee"  Font-Underline
=false CommandArgument ="<%#DataBinder.Eval(Container.DataItem, "BookID")%>">
            <%#DataBinder.Eval(Container.DataItem, "BookName")%>
            </asp:LinkButton></td>  </tr>
        <tr>
            <td>
```

```
                    市场价：</td>
                <td>
                 <%#GetVarMKP(DataBinder.Eval(Container.DataItem, "MarketPrice").ToString())%>￥
                  </td></tr>
               <tr>
                    <td>
                        热卖价：</td>
                    <td>
                      <%#GetVarHot(DataBinder.Eval(Container.DataItem, "HotPrice").ToString())%>￥
                      </td>
                  </tr> <tr>
              <td colspan="2">
             <asp:ImageButton ID="imagebtnRefine" runat="server" CommandName="buy"
                                                    CommandArgument ='<%# DataBinder.Eval(Container.DataItem,
"BookID") %>'   ImageUrl="~/images/mybuy.gif" OnClick="imagebtnRefine_Click" />
                    </td>
                </tr>
        </table>
        </ItemTemplate>
        </asp:DataList>
```

下面着重介绍下上述代码中应用的数据绑定表达式。

（1）在 ASP.NET1 中主要应用的是 DataBind.Eval 方法，该方法是一个完全成熟的方法，可以在程序中的任何地方使用。

DataBinder.Eval 方法的语法如下：

```
<%# DataBinder.Eval(Containter.DataItem, expression) %>
```


说 明

Containter.DataItem 表达式引用对该表达式进行计算的对象。该表达式通常是一个字符串，表示数据项对象上要访问的字段的名称。它可以是一个包括索引和属性名的表达式。DataItem 属性表示当前容器上下文中的对象。容器通常是即将生成的数据项对象的当前实例。

（2）在 ASP.NET 2.0 及以上版本中，只要是 ASP.NET1.x 中接受 DataBind.Eval 方法的地方，就可以使用如下表达式：

```
<%# Eval(expression) %>
```

从上述表达式中可以看出，ASP.NET 2.0 及以上版本也是完全支持 DataBinder 对象的。ASP.NET 2.0 及以上版本中的 Eval 方法是建立在 DataBind.Eval 方法之上的一个简单包装。该方法代表一种单向数据绑定，它实现了数据读取的自动化，但是没有实现数据写入自动化。如果要实现双向的数据绑定，可应用 ASP.NET 2.0 及以上版本中另一个新的数据绑定方法 Bind 方法读写数据项属性。

21.4 在线银行支付类的封装与设计

开发项目中以类的形式来组织、封装一些常用的方法和事件，不仅可以提高代码的重用率，也大大方便了代码的管理。

21.4.1 在线银行支付 BankPay 类的创建

在创建类时，用户可以直接在该项目中找到 App_Code 文件夹，然后单击鼠标右键，在弹出的快捷菜单中选择“添加新项”命令。在弹出的“添加新项”对话框中选择“类”，并为其命名（以创建 BankPay 为例），单击“添加”按钮即可创建一个新类，如图 21.3 所示。

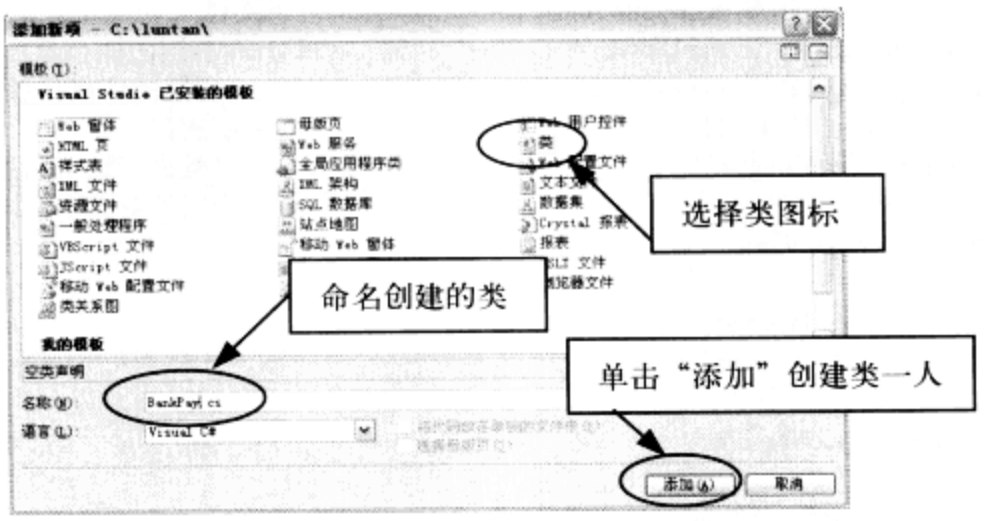


图 21.3　BankPay 类的创建

注　意

在 ASP.NET 2.0 及以上版本中，App_Code 文件夹专门用来存放一些应用于全局的代码（比如公共类），如果项目中没有该文件夹，可以在项目上单击鼠标右键，在弹出的快捷菜单中选择“添加 ASP.NET 文件夹”/“App_Code”选项，添加一个 App_Code 文件夹。

21.4.2　在线银行支付 BankPay 类的编写

在线银行支付 BankPay 类主要用于定义银行相关的在线银行支付变量，如银行接口名称、商户代码和商城账号等。定义完相应变量后设置变量的可读写（Read/Write）属性返回变量的值（对属性的值的读操作用 get 关键字，对属性的值写操作用 set 关键字）。具体实例代码如下。

例程 5　代码位置：光盘\mr\21\OnlineBank\ App_Code\ BankPay.cs

```
public class BankPay
{
    public BankPay()
    {
        //
        // TODO: 在此处添加构造函数逻辑
        //
    }
    private string interfaceName = "ICBC_PERBANK_B2C";   //接口名称
    private string interfaceVersion = "1.0.0.0";          //接口版本号
    private string merID = "0200EC20000012";              //商户代码
    private string merAcct = "02000291090000030106";      //商城账号
    private string merURL = "";                           //接收银行消息地址 (http://地址/Get.aspx)
    private string notifyType = "AG";                     //通知类型(在交易完成后不通知商户)
    private string orderid;                               //订单号
    private string amount;                                //订单金额
    private string curType = "001";                       //支付币种
    private string resultType = "0";                      //结果发送类型
    private string orderDate;                             //交易日期时间
    private string verifyJoinFlag = "0";                  //检验联名标志（不检验客户是否与商户联名，按金额
扣账）
    private string merCert;                               //商城证商品公钥
    private string goodsID = "";                          //商品编号
    private string goodsName = "";                        //商品名称
    private string goodsNum = "";                         //商品数量
    private string carriageAmt = "";                      //已含运费金额
    private string merHint = "";                          //商城提示
    private string comment1 = "";                         //备注字段1
    private string comment2 = "";                         //备注字段2
```

```
private string path1 = @"E:\bank\public.crt";                    //公钥路径
private string path2 = @"E:\bank\user.crt";                      //拆分pfx后缀的证商品后的公钥路径
private string path3 = @"E:\bank\user.key";                      //拆分pfx后缀的证商品后的私钥路径
private string key = "11111111";                                 //私钥保护密码
private string merSignMsg = "";                                  //订单签名数据（加密码后的字符串）
private string msg = "";                                         //需要加密码的明文字符串
public string InterfaceName
{
    get { return interfaceName; }
    set { interfaceName = value; }
}
public string InterfaceVersion
{
    get { return interfaceVersion; }
    set { interfaceVersion = value; }
}
public string MerID
{
    get { return merID; }
    set { merID = value; }
}
public string MerAcct
{
    get { return merAcct; }
    set { merAcct = value; }
}
public string MerURL
{
    get { return merURL; }
    set { merURL = value; }
}
public string NotifyType
{
    get { return notifyType; }
    set { notifyType = value; }
}
public string Orderid
{
    get { return orderid; }
    set { orderid = value; }
}
public string Amount
{
    get { return amount; }
    set { amount = value; }
}
public string CurType
{
    get { return curType; }
    set { curType = value; }
}
public string ResultType
{
    get { return resultType; }
    set { resultType = value; }
}
public string OrderDate
{
    get { return orderDate; }
    set { orderDate = value; }
```

```
}
public string VerifyJoinFlag
{
    get { return verifyJoinFlag; }
    set { verifyJoinFlag = value; }
}
public string MerSignMsg
{
    get { return merSignMsg; }
    set { merSignMsg = value; }
}
public string MerCert
{
    get { return merCert; }
    set { merCert = value; }
}
public string GoodsID
{
    get { return goodsID; }
    set { goodsID = value; }
}
public string GoodsName
{
    get { return goodsName; }
    set { goodsName = value; }
}
public string GoodsNum
{
    get { return goodsNum; }
    set { goodsNum = value; }
}
public string CarriageAmt
{
    get { return carriageAmt; }
    set { carriageAmt = value; }
}
public string MerHint
{
    get { return merHint; }
    set { merHint = value; }
}
public string Comment1
{
    get { return comment1; }
    set { comment1 = value; }
}
public string Comment2
{
    get { return comment2; }
    set { comment2 = value; }
}
public string Path1
{
    get { return path1; }
    set { path1 = value; }
}
public string Path2
{
    get { return path2; }
    set { path2 = value; }
```

```
        }
        public string Path3
        {
            get { return path3; }
            set { path3 = value; }
        }
        public string Key
        {
            get { return key; }
            set { Key = value; }
        }
        public string Msg
        {
            get { return msg; }
            set { msg = value; }
        }
    }
```

21.5 商城在线订单生成页

当商城会员确定购物车页中所购买的商品后，便可单击购物车页中的“前往服务台”按钮进入商城在线订单生成页（checkOut.aspx）。在填写完详细订单后就可以提交订单到网上银行，实现运行效果如图 21.4 所示。

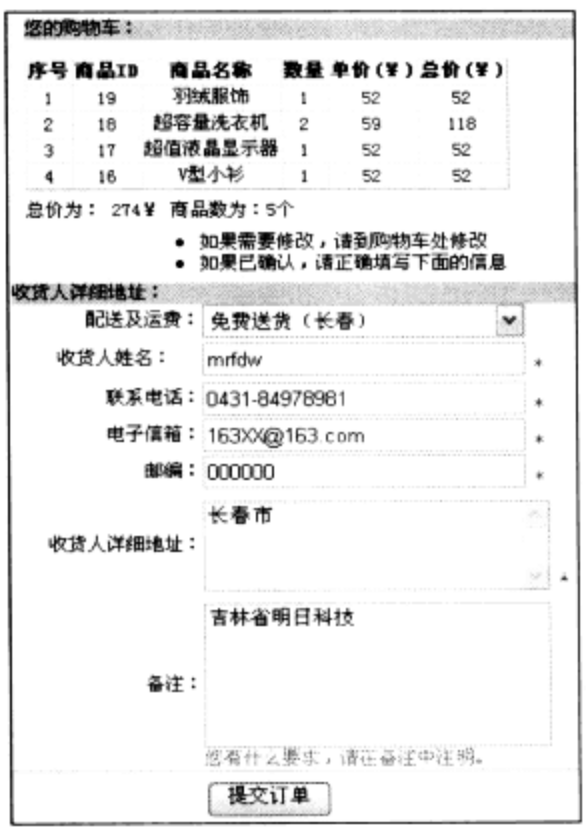


图 21.4 订单生成页

21.5.1 页面设计

在线订单生成页应用了母版页（MasterPage.master）技术，即为母版页的内容页。

（1）在应用程序中新创建 1 个 Web 窗体，命名为 checkOut.aspx，用于显示用户购物车中信息及收货人的详细地址信息。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 6 个 TextBox 控件、1 个 DropDownList 控件、1 个 GridView 控件和 1 个 Button 控件到窗体中，设置控件的属性。页面中各个控件的属性设置及其用途如表 21.3 所示。

表 21.3　　在线考试页面涉及的主要控件

控件类型	控件名称	主要属性设置	用　途
DropDownList	ddlShipType	AutoPostBack 属性设置为 True	用于选择配送方法及运输费用
Button	btnConfirm	Text 属性设置为"提交订单"	提交生成的订单
GridView	gvShopCart	AutoGenerateColumns 属性设置为 False	显示用户购物车中商品信息
TextBox	txtReciverName	无	用于显示收货人姓名
	txtReceiverPhone	无	用于显示收货人联系电话
	txtReceiverEmails	无	用于显示收货人的电子邮件
	txtReceiverPostCode	无	用于显示邮编
	txtReceiverAddress	TextMode 属性设置为 MultiLine	用于显示收货人的详细地址
	txtRemark	无	用于添写备注信息

21.5.2　代码实现

在页面的 Page_Load 事件中，首先判断用户是否已登录，如果登录则显示用户的基本信息，如收货人姓名、地址等；接着判断用户的购物车是否为空，如果不为空则通过创建 DataTable（内存中的数据表）对象来设置购物车内容的数据源，并最终将数据绑定到 GridView 控件中，具体实现的代码如下。

例程 6　代码位置：光盘\mr\21\OnlineBank\ checkOut.aspx .cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Session["Username"] == null)
        {
            Response.Write("<script lanuage=javascript>alert('对不起,您还不是会员！ ');location='kaoshi_timu.
aspx'</script>");
        }
        else
        {
            //如果用户已登录，则显示用户的基本信息
            DataTable dsTable = ucObj.GetUserInfo(Convert.ToInt32(Session["UserID"].ToString()));
            this.txtReciverName.Text = dsTable.Rows[0][1].ToString();        //收货人姓名
            this.txtReceiverPhone.Text = dsTable.Rows[0][5].ToString();    //收货人电话号码
            this.txtReceiverEmails.Text = dsTable.Rows[0][6].ToString();   //收货人E-mail
            this.txtReceiverPostCode.Text = dsTable.Rows[0][8].ToString();//收货人邮编
            this.txtReceiverAddress.Text =dsTable.Rows[0][7].ToString();   //收货人详细地址
        }
        if (Session["ShopCart"] == null)
        {
            //如果没有购物，则给出相应信息，并隐藏按钮
            this.labMessage.Text = "您还没有购物！ "; //显示提示信息
            this.btnConfirm.Visible = false;              //隐藏“确认”按钮
        }
        else
        {
            hashCar = (Hashtable)Session["ShopCart"];   //获取其购物车
            if (hashCar.Count == 0)
            {
```

```
        //如果没有购物，则给出相应信息，并隐藏按钮
        this.labMessage.Text = "您购物车中没有商品！";//显示提示信息
        this.btnConfirm.Visible = false;                    //隐藏“确认”按钮
    }
    else
    {
        //设置购物车内容的数据源
        dtTable = new DataTable();
        DataColumn column1 = new DataColumn("No");//序号列
        DataColumn column2 = new DataColumn("BookID");      //商品ID代号
        DataColumn column3 = new DataColumn("BookName");    //商品名称
        DataColumn column4 = new DataColumn("Num");         //数量
        DataColumn column5 = new DataColumn("price");       //单价
        DataColumn column6 = new DataColumn("totalPrice");  //总价
        dtTable.Columns.Add(column1);//添加新列
        dtTable.Columns.Add(column2);
        dtTable.Columns.Add(column3);
        dtTable.Columns.Add(column4);
        dtTable.Columns.Add(column5);
        dtTable.Columns.Add(column6);
        DataRow row;
        //对数据表中每一行进行遍历，给每一行的新列赋值
        foreach (object key in hashCar.Keys)
        {
        row = dtTable.NewRow();
        row["BookID"] = key.ToString();             //商品ID
        row["Num"] = hashCar[key].ToString();    //商品数量
        dtTable.Rows.Add(row);
        }
        //计算价格
        DataTable dstable;
        int i = 1;
        float price; //商品单价
        int num;       //商品数量
        float totalPrice = 0; //商品总价格
        int totailNum = 0;      //商品总数量
        foreach (DataRow drRow in dtTable.Rows)
        {
            strSql = "select BookName,HotPrice from tb_BookInfo where BookID=" + Convert.ToInt32 (drRow ["BookID"].
ToString());
                dstable = dbObj.GetDataSetStr(strSql, "tbGI");
                drRow["No"] = i;
                drRow["BookName"] = dstable.Rows[0][0].ToString();    //商品名称
                drRow["price"] = dstable.Rows[0][1].ToString();         //商品名称
                price = float.Parse(dstable.Rows[0][1].ToString());
                num = Int321.Parse(drRow["Num"].ToString());
                drRow["totalPrice"] =(price*num);                          //总价
                totalPrice += price * num;                                 //计算合价
                totailNum += num;                                              //计算商品总数
                i++;
            }
            this.labTotalPrice.Text = totalPrice.ToString();          //显示所有商品的价格
            this.labTotalNum.Text = totailNum.ToString();               //显示商品总数
            this.gvShopCart.DataSource = dtTable.DefaultView;        //绑定GridView控件
            this.gvShopCart.DataBind();
                }
            }
        }
```

双击页面中的“提交订单”按钮，触发其 btnConfirm_Click 事件完成对订单信息的提交操作。在“收货人详细地址”栏下面，系统首先会自动将数据库中绑定用户在注册会员时填写的姓名、电话和邮编等信息显示在指定的 TextBox 中，用户可对显示的信息进行核对并可进行更改操作，购买信息无误后，便可提交订单，进入“在线银行支付方式选择页”。实现的主要程序代码如下。

例程 7 代码位置：光盘\mr\21\OnlineBank\ checkOut.aspx.cs

```
protected void btnConfirm_Click(object sender, EventArgs e)
{
    if (Page.IsValid)
    {
        //得到用户输入的信息
        ... //省略判断用户邮编、电话正确与否等信息代码
        if (this.ddlShipType.SelectedIndex != 0)//获取邮递方式及其费用
        {
            fltShipFee = float.Parse(this.ddlShipType.SelectedValue.ToString());
        }
        else
        {
            Response.Write(ccObj.MessageBox("请选择运输方式！"));
            return;
        }
        string strName = this.txtReciverName.Text.Trim();          //收货人姓名
        string strAddress = this.txtReceiverAddress.Text.Trim();   //收货人详细地址
        string strRemark = this.txtRemark.Text.Trim();             //备注
        int IntTotalNum = int.Parse(this.labTotalNum.Text);        //商品总数
        float fltTotalShipFee = IntTotalNum * fltShipFee;          //运输总费用
        //将订单信息插入订单表中
        int IntOrderID = ocObj.AddOrder(float.Parse(this.labTotalPrice.Text), fltTotalShipFee, this.ddlShipType.
SelectedItem.Text, strName, strPhone, strZip, strAddress, strEmail);
        int IntBookID; //商品ID
        int IntNum;      //购买商品数量
        float fltTotalPrice;
        //对订单中的每一个货物插入订单详细表中
        foreach (GridViewRow gvr in this.gvShopCart.Rows)
        {
            IntBookID = int.Parse(gvr.Cells[1].Text);
            IntNum = int.Parse(gvr.Cells[3].Text);
            fltTotalPrice = float.Parse(gvr.Cells[5].Text);
            ocObj.AddDetail(IntBookID, IntNum, IntOrderID, fltTotalPrice, strRemark);
        }
        //设置Session
        Session["ShopCart"] = null;   //清空购物车
        Response.Redirect("PayWay.aspx?OrderID=" + IntOrderID);
    }
}
```

21.6 在线银行支付方式选择页

用户在“服务台”页填写完相关信息后，单击“提交按钮”即可进入“选择在线银行支付方式”页（PayWay.aspx），在该页用户可以选择在线银行支付方式（这里主要以选择工商银行在线支付为例），其运行结果如图 21.5 所示。

图 21.5　选择在线银行支付方式

实现该功能的具体步骤如下。

首先，将一个表格（Table）控件置于 PayWay.aspx 页中，为整个页面进行布局。

其次，从“工具箱”/“标准”选项卡中拖放 5 个 ImageButton 控件，设置各个控件的 ImageUrl 属性值，用于显示在线银行支付方式。

最后，在“中国工商银行”按钮的 Click 事件下添加代码，用于实现当用户单击该按钮后，跳转到“工商银行在线支付页”。其代码如下。

例程 8　代码位置：光盘\mr\21\OnlineBank\ PayWay.aspx.cs

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
   Response.Redirect("GoBank.aspx?OrderID=" + Request["OrderID"].ToString());
}
```

21.7　工商银行在线支付页

目前 ICBC 个人网上银行的 B2C 在线银行支付系统是 ICBC 专门为拥有工行牡丹信用卡账户，并开通网上支付功能的网上银行个人客户进行网上购物所开发的支付平台。下面详细地介绍一下开发工商银行在线支付页的全过程。

21.7.1　开发工商银行在线支付前期工作

首先，需要特约网站申请人到 ICBC 当地指定机构办理申请手续，并提交如下申请资料：

（1）营业执照副本及复印件；

（2）经办人员的有效身份证件；

（3）填妥的《特约网站注册申请表》；

（4）最近年度的资产负债表和损益表的复印件；

（5）《域名注册证》复印件或其他对所提供域名享有权利的证明；

（6）企业标识 LOGO 的电子文件；

（7）填妥的“牡丹卡单位申请表”。

其次，经工商银行审查合格后，工商银行将提供银行方的通信、数据接口和已有商户端程序及商户客户证书。

最后，特约网站可以根据工商银行提供的资料，开发工商银行在线支付功能。

21.7.2　开发工商银行在线支付的具体步骤

首先，按照工商银行提供的资料注册 com 组件，其步骤如下。

（1）将 ICBCEBankUtil.dll 和 LIB\windows\WIN32\infosecapi.dll 两个 dll 文件复制到系统 system32 目录下。

（2）打开 DOS 窗口，进行 system32 目录。

（3）运行“regsvr32 ICBCEBankUtil.dll”命令注册控件。

其次，将工商银行提供的 public 公钥、拆分 pfx 后缀证书的公钥和拆分 pfx 后缀证书的私钥放到本地磁盘（如 D 盘根目录下）。在本网站中，笔者将其放在了项目下的 bank 文件中。

然后，在项目的 Bin 文件中，单击鼠标右键，在弹出如图 21.6 所示的对话框中，添加引用 ICBCEBankUtil.dll 文件。

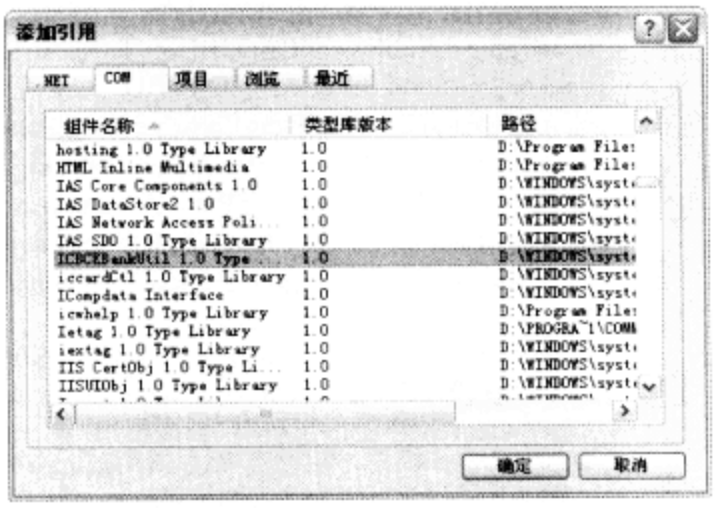


图 21.6　添加引用对话框

最后，设计提交表单页面（GoBank.aspx），其步骤如下。

（1）将提交表单页面（GoBank.aspx）切换到 HTML 视图中，添加如下代码，用于设计提交表单内容。

例程 9　代码位置：光盘\mr\21\OnlineBank\ GoBank.aspx.cs

```
<form id="form1"   name="order"    method="post"    action="银行地址">
<input    type="hidden"    name="interfaceName"    value="<%=bankpay.InterfaceName%>"    >
<input    type="hidden"    name="interfaceVersion"    value=<%=bankpay.InterfaceVersion%>    >
<input    type="hidden"    name="orderid"    value="<%=bankpay.Orderid%>">
<input    type="hidden"    name="amount"    value="<%=bankpay.Amount%>">
<input    type="hidden"    name="curType"    value="<%=bankpay.CurType%>">
<input    type="hidden"    name="merID"    value="<%=bankpay.MerID%>"    >
<input    type="hidden"    name="merAcct"    value="<%=bankpay.MerAcct%>"    >
<input    type="hidden"    name="verifyJoinFlag"    value="<%=bankpay.VerifyJoinFlag%>">
<input    type="hidden"    name="notifyType"    value="<%=bankpay.NotifyType%>">
<input    type="hidden"    name="merURL"    value="<%=bankpay.MerURL%>">
<input    type="hidden"    name="resultType"    value="<%=bankpay.ResultType%>">
<input    type="hidden"    name="orderDate"    value="<%=bankpay.OrderDate%>">
<input    type="hidden"    name="merSignMsg"    value="<%=bankpay.MerSignMsg%>">
<input    type="hidden"    name="merCert"    value="<%=bankpay.MerCert%>">
<input    type="hidden"    name="goodsID"    value="<%=bankpay.GoodsID%>">
<input    type="hidden"    name="goodsName"    value="<%=bankpay.GoodsName%>">
<input    type="hidden"    name="goodsNum"    value="<%=bankpay.GoodsNum%>">
<input    type="hidden"    name="carriageAmt"    value="<%=bankpay.CarriageAmt%>">
<input    type="hidden"    name="merHint"    value="<%=bankpay.MerHint%>">
<input    type="hidden"    name="comment1"    value="<%=bankpay.Comment1%>"    >
<input    type="hidden"    name="comment2"    value="<%=bankpay.Comment2%>"    >
<input    type="submit"    value="立即支付！" >
</form>
```

说　明

① 订单只能使用 POST 方式提交，使用 https 协议通信。

② 如果提交的表格含有中文，需要在<head></head>节点中，使用字符集 GBK 指定，其代码如下。

例程 10　代码位置：光盘\mr\21\OnlineBank\ GoBank.aspx

```
<meta http-equiv="content-type" content="text/html;charset=GBK">
```

（2）将提交表单页面切换到编辑器页（GoBank.aspx.cs）中，为提交表单赋值，其相关代码如下。

例程 11　代码位置：光盘\mr\21\OnlineBank\ GoBank.aspx.cs

```
public static BankPay bankpay = new BankPay();//实例化BankPay类对象
#region  初始化BankPay类
public BankPay  GetPayInfo()
{
```

```
    //从订单信息表中获取订单编号、订单金额
    string strSql = "select Round(TotalPrice,2) as TotalPrice from tb_OrderInfo
where OrderID=" + Convert.ToInt32(Page.Request["OrderID"].Trim());
    DataTable dsTable = dbObj.GetDataSetStr(strSql, "tbOI");
    bankpay.Orderid = Request["OrderID"].Trim();//订单编号
    bankpay.Amount = Convert.ToString(float.Parse(dsTable.Rows[0]["TotalPrice"].ToString())*100);  //订单金额
    bankpay.OrderDate = DateTime.Now.ToString("yyyyMMddhhmmss");                                   //交易日期时间
    bankpay.Path1 = Server.MapPath(@"bank\user.crt");                                               //公钥路径
    bankpay.Path2 = Server.MapPath(@"bank\user.crt");//拆分pfx后缀的证书后的公钥路径
    bankpay.Path3 = Server.MapPath(@"bank\user.key");//拆分pfx后缀的证书后的私钥路径
    //下面是需要加密的明文字符串
    bankpay.Msg = bankpay.InterfaceName + bankpay.InterfaceVersion + bankpay.MerID + bankpay.MerAcct
    + bankpay.MerURL + bankpay.NotifyType + bankpay.Orderid + bankpay.Amount + bankpay.CurType
    + bankpay.ResultType + bankpay.OrderDate + bankpay.VerifyJoinFlag;
    //项目中引用组件，以声明的方式创建com组件
    ICBCEBANKUTILLib.B2CUtil obj=new ICBCEBANKUTILLib.B2CUtil() ;
    //加载公钥、私钥、密码，如果返回0则初始化成功
    if (obj.init(bankpay.Path1, bankpay.Path2, bankpay.Path3, bankpay.Key) == 0)
     {
      bankpay.MerSignMsg = obj.signC(bankpay.Msg, bankpay.Msg.Length);                              //加密明文
      bankpay.MerCert = obj.getCert(1);                                                             //提取证书
     }
   else
     {
      //返回签名失败信息
      Response.Write(obj.getRC());
     }
  return (bankpay);
}
#endregion
```

21.8 程序错误与调试

本模块运行到在线银行支付选择页的时候，当单击“中国工商银行”图标会出现如图 21.7 所示的错误页，出现这种错误主要是由于在没有连接 Internet 互联网进行测试，而导致“检索 COM 类工厂中 CLSID 为{BE4D10EB-FD6B-474C-96B3-C5E70BC5EFB5}的组件时失败”。

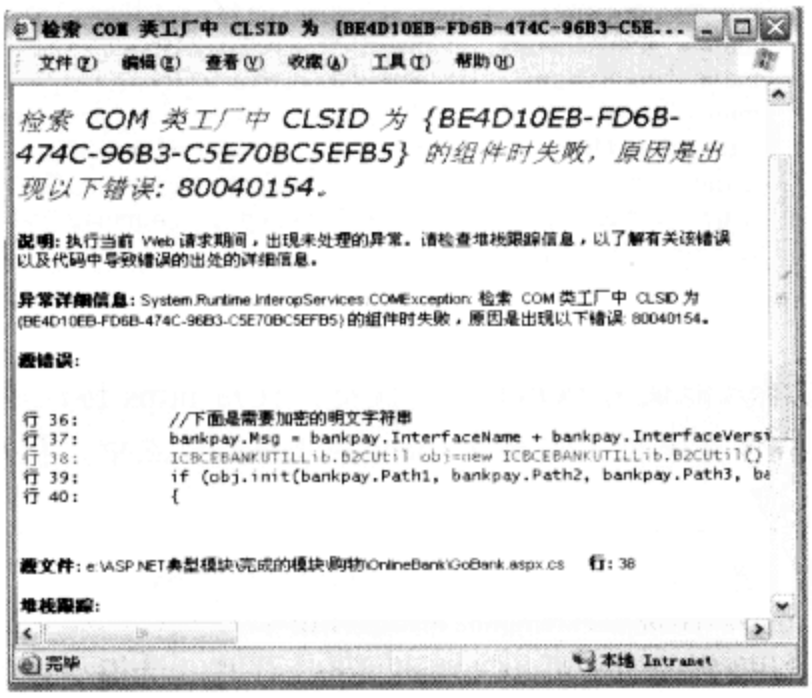


图 21.7 在线支付测试页

说 明

上述错误出现是由于没有连接 Internet 网络导致的，如果读者想测试此实例，需要与中国工商行申请一个在线支付的接口方可进行在线支付测试。

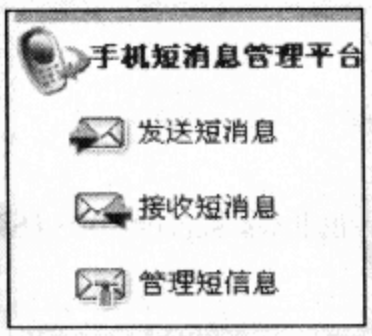
手机短消息管理平台

第22章

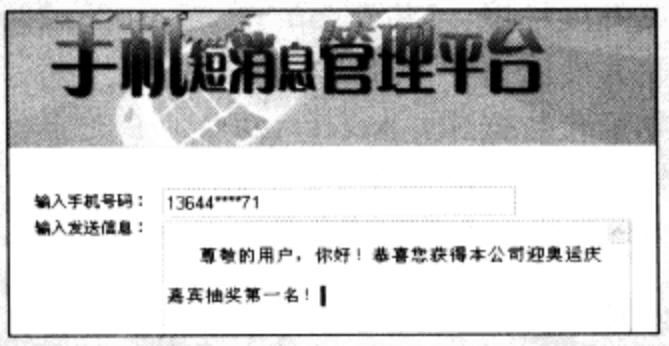
实例位置：光盘\mr\22\

如今手机已成为大众的交流工具。有关手机的程序开发越来越广泛，可以利用手机短信猫实现远程控制计算机、业务员销售数据的采集、短信息娱乐互动平台等功能，实现这些功能都是建立在手机短消息的收发功能之上。本章将设计、开发一个手机短消息管理平台模块，详细介绍搜索引擎的开发思想及过程。通过本章，读者能够学到以下内容。

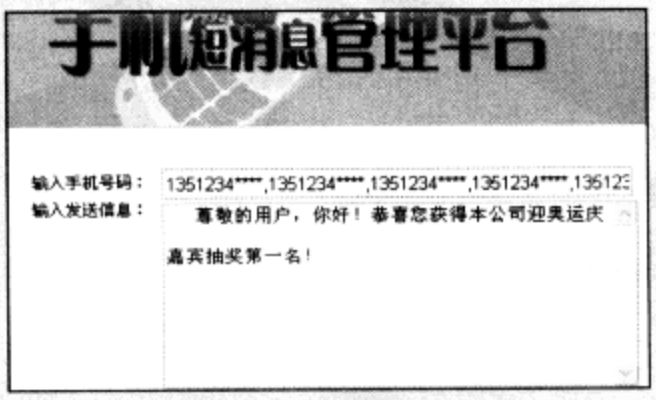
树状导航菜单



单用户发送短消息

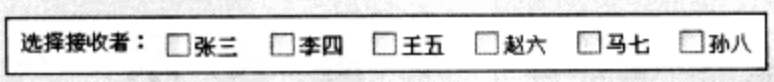


群发手机短消息



管理手机短消息

CheckBoxList 控件横向显示接收用户



22.1 手机短消息管理平台概述

22.1.1 概述

手机短消息管理平台的开发核心主要是对硬件短信猫开发，短信猫的应用非常广泛，例如在管理系统中短信的群发，大多数情况下都是通过短信猫来实现的。生产短信猫的厂商也特别多，但是基本道理大同小异。一般情况下都是生产厂商在销售硬件的同时，配带了已经编译好的中间组件，用户不必了解这个组件的具体编译方法，只需了解如何去调用其中的方法就可以，这样大大减少了技术方面的难度，使其能得到更好的普及。

本章所开发手机短消息管理平台，可以应用到客户关系管理系统中、网站后台在线客户、娱乐互动平台、手机群发广告系统等。

22.1.2 开发环境

1. 网站开发环境

- 网站开发环境：Microsoft Visual Studio 2008 集成开发环境。
- 网站开发语言： ASP.NET 3.5 + C#。
- 网站后台数据库：Access 2000。
- 开发架构：ASP.NET 3.5 + 硬件短信猫。
- 开发环境运行平台：Windows XP (SP2) / Windows 2000 (SP4) / Windows Server 2003 (SP1)。

注意

SP（Service Pack）为 Windows 操作系统补丁。

2. 服务器端

- 操作系统：Windows 2003 Server（SP1）。
- Web 服务器：IIS 6.0。
- 数据库服务器：Access 2000。
- 浏览器：IE 6.0。
- 网站服务器运行环境：Microsoft .NET Framework SDK v3.5。

3. 客户端

- 浏览器：Internet Explorer 6.0。
- 分辨率：最佳效果 1024×768 像素。

22.2 关 键 技 术

22.2.1 短信猫硬件接口介绍

短信猫是利用 SIM 卡发送短信的硬件设备，通过串口或 USB 接口（根据设备型号而定）与计算机相连。本例使用的是北京人大金仓信息技术有限公司的串口短信猫。在购买短信猫时会附带有 SDK 的开发包，其中提供了操作短信猫的函数（封装在 dllforvc.dll 动态库中）。下面介绍操作短信猫的主要函数。

注意

由于 dllforvc.dll 动态库属于非托管的类库，那么类库需要与 DllImport 属性一起使用；在这种情况下，方法必须声明为 static。

1. GSMModemGetSnInfoNew 函数

该函数获取短信猫注册需要的信息。

```
[DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetSnInfoNew",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
public static extern string GSMModemGetSnInfoNew(string device, string baudrate);
```

参数说明如下。

- device：通信端口，为 null 时系统会自动检测。
- baudrate：通信波特率，为 null 时系统会自动检测。

2. GSMModemGetDevice 函数

该函数获取当前的通信端口。

```
[DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetDevice",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
public static extern string GSMModemGetDevice();
```

3. GSMModemGetBaudrate 函数

该函数获取当前通信波特率。

```
[DllImport("dllforvc.dll",
        EntryPoint = "GSMModemGetBaudrate",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
public static extern string GSMModemGetBaudrate();
```

4. GSMModemInitNew 函数

该函数用于初始化短信猫。语法如下：

```
[DllImport("dllforvc.dll",
        EntryPoint = "GSMModemInitNew",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
public static extern bool GSMModemInitNew(
        string device,
        string baudrate,
        string initstring,
        string charset,
        bool swHandshake,
        string sn);
```

参数说明如下。

- device：标识通信端口，如果为 NULL，系统会自动检测。
- baudrate：标识通信波特率，如果为 NULL，系统自动检测。
- initstring：标识初始化命令，为 NULL 即可。
- charset：标识通信字符集，为 NULL 即可。
- swHandshake：标识是否进行软件握手，为 False 即可。
- sn：标识短信猫的授权号，需要根据实际情况填写。

5. GSMModemSMSsend 函数

该函数用于发送手机短信。语法如下：

```
[DllImport("dllforvc.dll",
        EntryPoint = "GSMModemSMSsend",
        CharSet = CharSet.Ansi,
        CallingConvention = CallingConvention.StdCall)]
```

```
public static extern bool GSMModemSMSsend(
    string serviceCenterAddress,
    int encodeval,
    string text,
    int textlen,
    string phonenumber,
    bool requestStatusReport);
```

参数说明如下。

- serviceCenterAddress：标识短信中心号码，为 Null 即可。
- encodeval：标识短信息编码格式，如果为 8，表示中文短信编码。
- text：标识短信内容。
- textlen：标识短信内容的长度。
- phonenumber：标识接收短信的电话号码。
- requestStatusReport：标识状态报告。

6. GSMModemSMSReadAll 函数

该函数取得所有短信息，包括 SIM 卡和手机中的。返回的短信内容格式为：电话号码 1|短信内容 1||电话号码 2|短信内容 2||。

```
//接收短信息返回字符串格式为:手机号码|短信内容||手机号码|短信内容||
//RD_opt为 1 接收短信息后不做任何处理， 0 为接收后删除信息
[DllImport("dllforvc.dll",
    EntryPoint = "GSMModemSMSReadAll",
    CharSet = CharSet.Ansi,
    CallingConvention = CallingConvention.StdCall)]
public static extern string GSMModemSMSReadAll(int RD_opt);
```

参数说明如下。

- RD_opt：对读取后短信息处理，为 0 表示删除，为 1 表示不做处理。

22.2.2 封装短信猫并生成 DLL 类库

在 ASP.NET 开发中，程序无法直接调用非托管的类库 dllforvc.dll，那么可以通过程序转换实现。主要步骤如下。

首先，在 Visual Studio 2008 中创建类库项目，设置名称为 SP。修改命名空间名称为 MobileSP，并在 MobileSP 空间中建立一个 Pulbic 类型的 GMS 类，在 GMS 类中调用非托管的类库 dllforvc.dll，并实现功能方法。实现代码如下。

例程 1 代码位置：光盘\mr\22\MobileTelephone\ SP\SP\Class1.cs

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Runtime.InteropServices;
namespace MobileSP
{
    public class GMS
    {
        //初始化gsm modem,并连接gsm modem
        [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemInitNew",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
        public static extern bool GSMModemInitNew(
            string device,
            string baudrate,
            string initstring,
            string charset,
```

```
        bool swHandshake,
        string sn);
    //获取短信猫新的标识号码
    [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemGetSnInfoNew",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetSnInfoNew(string device, string baudrate);
    //获取当前通信端口
    [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemGetDevice",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetDevice();
    //获取当前通信波特率
    [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemGetBaudrate",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetBaudrate();
    //断开连接并释放内存空间
    [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemRelease",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
    public static extern void GSMModemRelease();
    //取得错误信息
    [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemGetErrorMsg",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemGetErrorMsg();
    //发送短信息
    [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemSMSsend",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
    public static extern bool GSMModemSMSsend(
        string serviceCenterAddress,
        int encodeval,
        string text,
        int textlen,
        string phonenumber,
        bool requestStatusReport);
    //接收短信息返回字符串格式为：手机号码|短信内容||手机号码|短信内容||
    //RD_opt为 1 接收短信息后不做任何处理， 0 为接收后删除信息
    [DllImport("dllforvc.dll",
            EntryPoint = "GSMModemSMSReadAll",
            CharSet = CharSet.Ansi,
            CallingConvention = CallingConvention.StdCall)]
    public static extern string GSMModemSMSReadAll(int RD_opt);
  }
}
```

其次，将非托管的类库 dllforvc.dll 复制到类库项目中的 Bin 目录下。

最后，运行类库项目，生成 SP.dll 即可。

22.2.3 Web 中引用 DLL 类库

根据上一节的介绍可以使用非托管的类库 dllforvc.dll，在使用前先将 SP.dll 类库引用到网

站中，然后将其类库 dllforvc.dll 也复制到网站的 Bin 目录中即可，如图 22.1 所示。在程序开发中程序员可以自由调用 GMS 类中的成员方法并实现相应的功能。

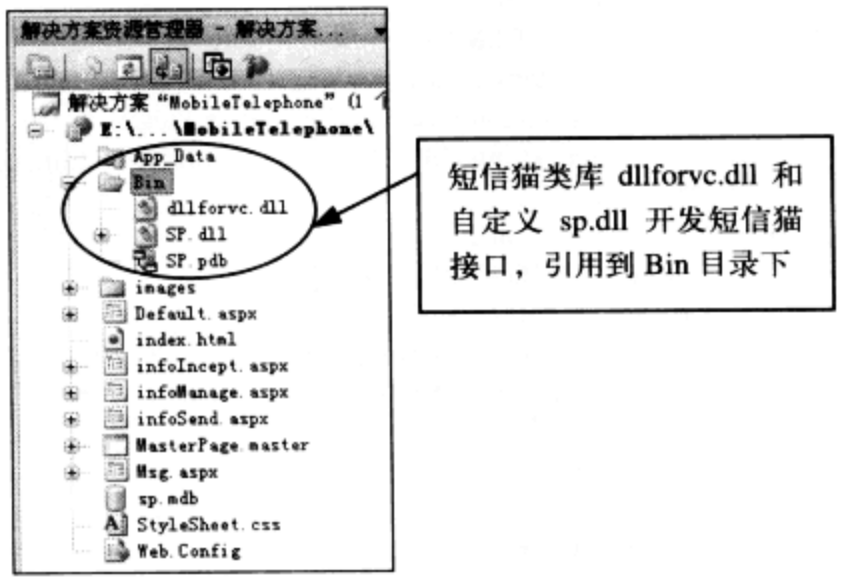


图 22.1　引用 DLL

22.3　手机短消息管理平台实现过程

22.3.1　发送手机短消息

发送手机短消息页面中，既可以单个用户发送短消息，也可以群发短消息。在页面的右侧选择接收短消息的用户名称，然后单击“提交”按钮，将接收者的手机号码提交到文本框内，最后，在内容文本框中输入内容，单击“发送”按钮，即可将消息发送到指定的手机号码中，如图 22.2 所示。

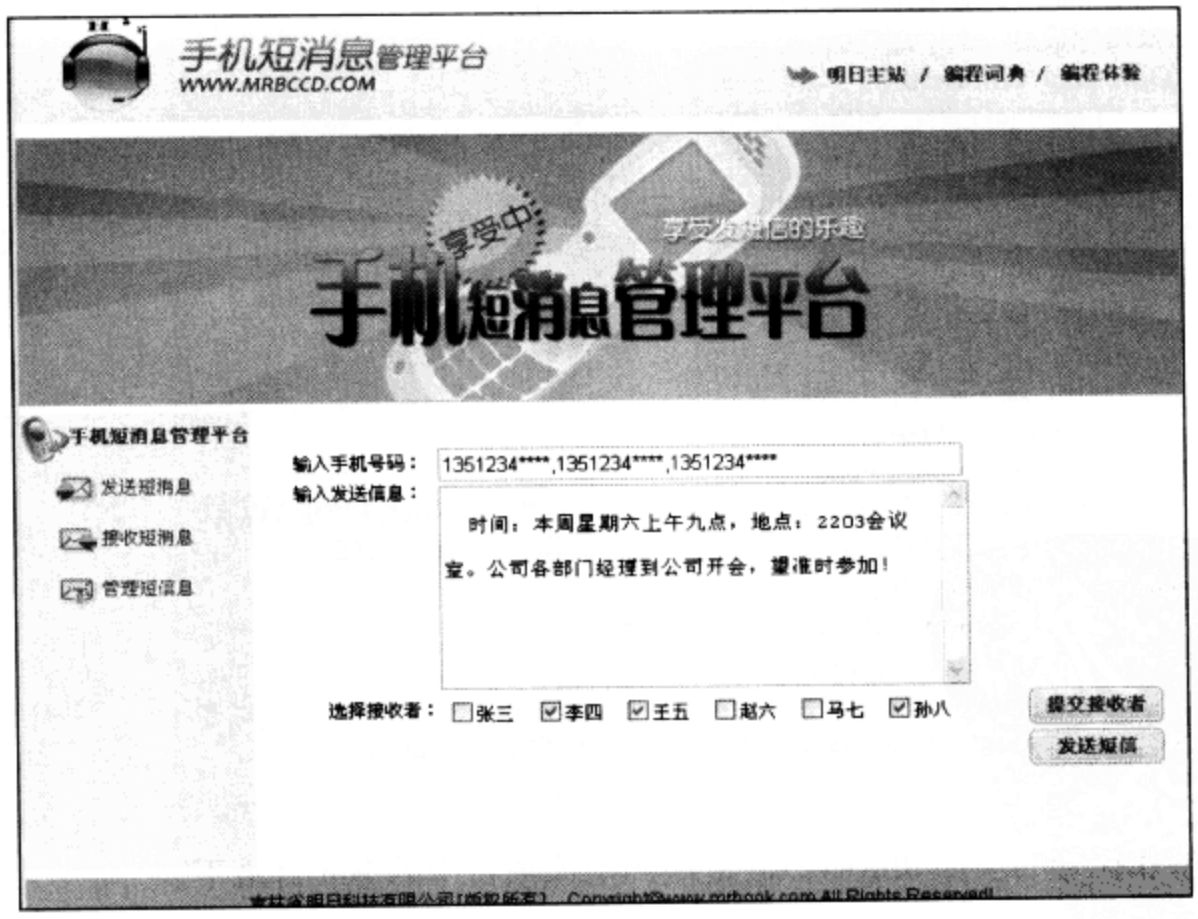


图 22.2　发送手机短消息页面

1. 页面设计

发送手机短消息页中涉及主要控件及控件用途如表 22.1 所示。

表 22.1　　发送手机短消息页中主要控件及用途

控　件	ID 属性	主 要 属 性	用　途
TextBox	txtAccepter	默认	输入接收者手机号码
TextBox	txtInfo	TextMode 属性值为 MultiLine	输入短消息
ImageButton	imgBtnSubmit	CausesValidation 属性值为 False	提交接收者手机号码
Button	btnSend	Text 属性值为“发送消息”	执行发送短消息命令
CheckBoxList	chkblEmployee	默认	显示接收短消息的用户名称
RequiredFieldValidator	RequiredFieldValidator1	ControlToValidate 属性值为 txtAccepter，ErrorMessage 属性值为“必填项”	验证手机号码文本框不能为空
RequiredFieldValidator	RequiredFieldValidator2	ControlToValidate 属性值为 txtInfo，ErrorMessage 属性值为“必填项”	验证短消息文本框不能为空

2. 代码设计

在页面的 Page_Load 事件下，将数据库中用户表中的用户名称和手机号码绑定显示在 CheckBoxList 控件中，当操作人员发送消息时，可以在 CheckBoxList 控件中选择接收消息用户，实现代码如下。

例程 2　代码位置：光盘\mr\22\MobileTelephone\ infoSend.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        DataSet ds = new DataSet();
        using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
+ HttpContext.Current.Server.MapPath("sp.mdb")))
        {
            OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_employee", conn);
            da.Fill(ds);
        }
        chkblEmployee.DataSource = ds;
        chkblEmployee.DataTextField = "eName";          //绑定显示文本
        chkblEmployee.DataValueField = "eTel";          //绑定显示文本值
        chkblEmployee.DataBind();
    }
}
```

单击“发送消息”按钮，通过代码控制短信猫硬件设备将短消息发送到指定的手机号码，并且支持单一用户发送和群发。实现代码如下。

例程 3　代码位置：光盘\mr\22\MobileTelephone\ infoSend.aspx.cs

```
protected void btnSend_Click(object sender, EventArgs e)
{
    //如果没有手机号码接收信息，则程序不继续执行
    if (txtAccepter.Text == "")
    {
        Response.Write("<script>alert('手机号码不能为空！');window.history(-1)</script>");
        return;
    }
    //获取短信猫的基本信息
    string strCOM = GMS.GSMModemGetDevice();                                    //COM1
    string strBTL = GMS.GSMModemGetBaudrate();                                  //波特率
    string strMobileNo = GMS.GSMModemGetSnInfoNew(strCOM, strBTL);  //机器号码
    string strWarranty = "YIWU-IJDD-CDQW-JDWG";                                 //设备授权号（设备上自带）
    string[] Accepters = txtAccepter.Text.Split(',');
    //连接设备
```

```
            if (GMS.GSMModemInitNew(strCOM, strBTL, null, null, false, strWarranty) == false)
            {
                Response.Write("<script>alert('设备连接失败！ " + GMS.GSMModemGetErrorMsg() + "');window. History
(-1)</script>");
                return;
            }
            // 发送短信
            if (Accepters.Length > 1)
            {
                for (int j = 0; j < Accepters.Length; j++)
                    if (GMS.GSMModemSMSsend(null, 8, txtInfo.Text, Encoding.Default.GetByteCount(txtInfo.Text),
Accepters[j].ToString(), false) == true)
                        Response.Write("<script>alert('短信发送成功！ ');window.history(-1)</script>");
                    else
                        Response.Write("<script>alert('短信发送失败！ ');window.history(-1)</script>");
            }
            else
            {
                if (GMS.GSMModemSMSsend(null, 8, txtInfo.Text, Encoding.Default.GetByteCount(txtInfo.Text), txtAccepter.
Text, false) == true)
                    Response.Write("<script>alert('短信发送成功！ ');window.history(-1)</script>");
                else
                    Response.Write("<script>alert('短信发送失败！ ');window.history(-1)</script>");
            }
        }
```

当计算机操作人员在 CheckBoxList 控件中选择接收短消息用户时（可单选或多选），然后单击“提交”按钮，将选择的接收人手机号码提交到指定文本框中。实现代码如下。

例程 4 代码位置：光盘\mr\22\MobileTelephone\ infoSend.aspx.cs

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
    //清空手机号码
    txtAccepter.Text = "";
    //将手机号码 循环添加到文本框中，如果是多个手机号码，则用逗号隔开
    for (int i = 0; i < chkblEmployee.Items.Count; i++)
    {
        if (chkblEmployee.Items[i].Selected)
            txtAccepter.Text = chkblEmployee.Items[i].Value + "," + txtAccepter.Text;
    }
    //去掉文本框中最后一个逗号
    if (txtAccepter.Text.Length != 0)
        txtAccepter.Text = txtAccepter.Text.Substring(0, txtAccepter.Text.Length - 1);
}
```

22.3.2 接收手机短消息

接收手机短消息页主要获取短信猫中的最新短消息、显示已有的短消息、阅读指定的短消息，另外，当鼠标移动某数据行时，该行高亮显示，如图 22.3 所示。

1. 页面设计

接收手机短消息的页面设计，主要通过 GridView 控件实现。GridView 控件可以通过列表的形式显示所有未读短消息和已读短消息，而且可以阅读指定的短消息。

下面开始编辑 GridView 控件，用于显示短消息和阅读指定的短消息。在 GridView 控件的属性对话框中选择 Columns 属性，在 Columns 属性值文本框中单击 按钮弹出如图 22.4 所示对话框。在该对话框中添加 BoundField 字段，用于显示指定列数据，前提必须设置 DataFile 属性（绑定数据库字段），设置 BoundField 字段后，还需要添加 CommandField 字段中的选择功能，用户可以阅读指定的短消息。

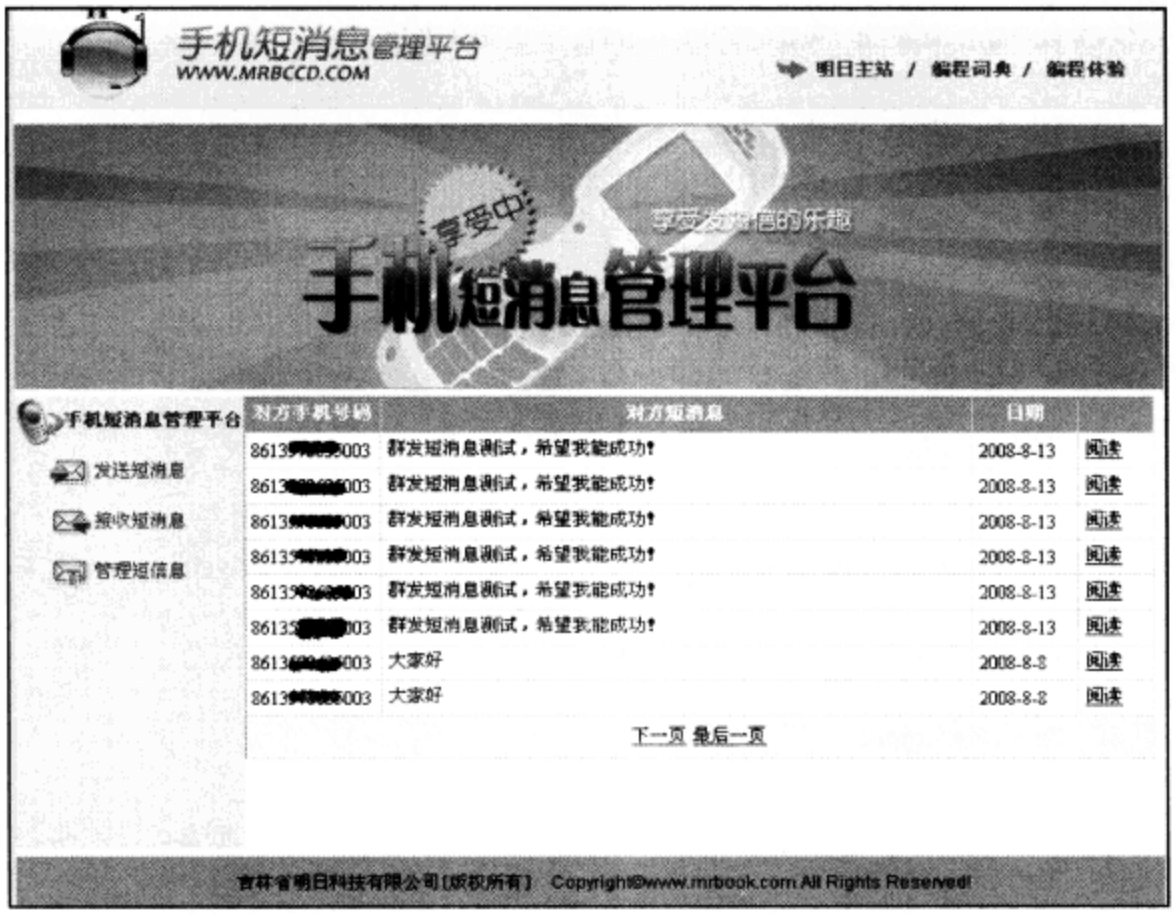


图 22.3　接收手机短消息

图 22.4　编辑 GridView 控件

2. 代码设计

在页面的 Page_Load 事件中，主要实现获取硬件短信猫设备的基本信息，利用获取短信猫设备的基本信息通过代码进行连接短信猫设备，最后获取短信猫设备中的短消息。实现代码如下。

例程 5　代码位置：光盘\mr\22\MobileTelephone\ infoIncept.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    //获取短信猫的基本信息
    string strCOM = GMS.GSMModemGetDevice();                              //COM1
    string strBTL = GMS.GSMModemGetBaudrate();                            //波特率
    string strMobileNo = GMS.GSMModemGetSnInfoNew(strCOM, strBTL);        //机器号码
    string strWarranty = "YIWU-IJDD-CDQW-JDWG";                           //设备授权号（设备上自带）
```

```
            //连接设备
            if (GMS.GSMModemInitNew(strCOM, strBTL, null, null, false, strWarranty) == false)
            {
                Response.Write("<script>alert('连接失败!'" + GMS.GSMModemGetErrorMsg() + ");window.history
(-1)</script>");
                return;
            }
            //接收短信
            string strContent = GMS.GSMModemSMSReadAll(0);
            if (strContent == null)
            {
                this.BindGridView();
                return;
            }
            //把手机号码或短消息分离出来
            string sc = "";
            foreach (string s in strContent.Split(new Char[] { '|' }))
            {
                if (s != string.Empty)
                {
                    sc = sc + "|" + s;
                }
            }
            sc = sc.Substring(1, sc.Length - 1);
            //创建数据源，将短信猫中消息保存到数据源中
            DataTable dt = new DataTable();
            dt.Columns.Add(new DataColumn("sender", typeof(string)));
            dt.Columns.Add(new DataColumn("incepter", typeof(string)));
            string[] r = sc.Split(new Char[] { '|' });
            //数据源中的数据保存到数据库中
            for (int j = 0; j < r.Length; j++)
            {
                using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
+ HttpContext.Current.Server.MapPath("sp.mdb")))
                {
                    OleDbCommand cmd = new OleDbCommand("insert into tb_sp (sender,incepter) values ('" + r.GetValue(j)
+ "','" + r.GetValue(j + 1) + "')", conn);
                    conn.Open();
                    cmd.ExecuteNonQuery();
                    conn.Close();
                }
                j++;
            }
            this.BindGridView();
        }
```

自定义 BindGridView 方法，获取数据库中所有的短消息，并显示到 GridView 控件中。实现代码如下。

例程 6 代码位置：光盘\mr\22\MobileTelephone\ infoIncept.aspx.cs

```
        private void BindGridView()
        {
            DataSet ds = new DataSet();
            using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="  +
HttpContext.Current.Server.MapPath("sp.mdb")))
            {
                OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_sp order by id desc", conn);
                da.Fill(ds);
```

```
        }
        GridView1.DataSource = ds;
        GridView1.DataKeyNames = new string[] { "id" };
        GridView1.DataBind();
    }
```

单击 GridView 控件上“阅读”链接，打开新窗口显示相关的短消息。实现代码如下。

例程 7 代码位置：光盘\mr\22\MobileTelephone\ infoIncept.aspx.cs

```
    protected void GridView1_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
    {
        string id = GridView1.DataKeys[e.NewSelectedIndex].Value.ToString();
        Response.Write("<script>window.open('Msg.aspx?id=" + id + "','','width=600 height=200 ,top=200 left=100');
window.go(-1); </script>");
    }
```

22.3.3 管理手机短消息

管理手机短消息页主要显示已接收的所有短消息，并可以删除指定的短消息，另外，当鼠标移动某数据行时，该行高亮显示，如图 22.5 所示。

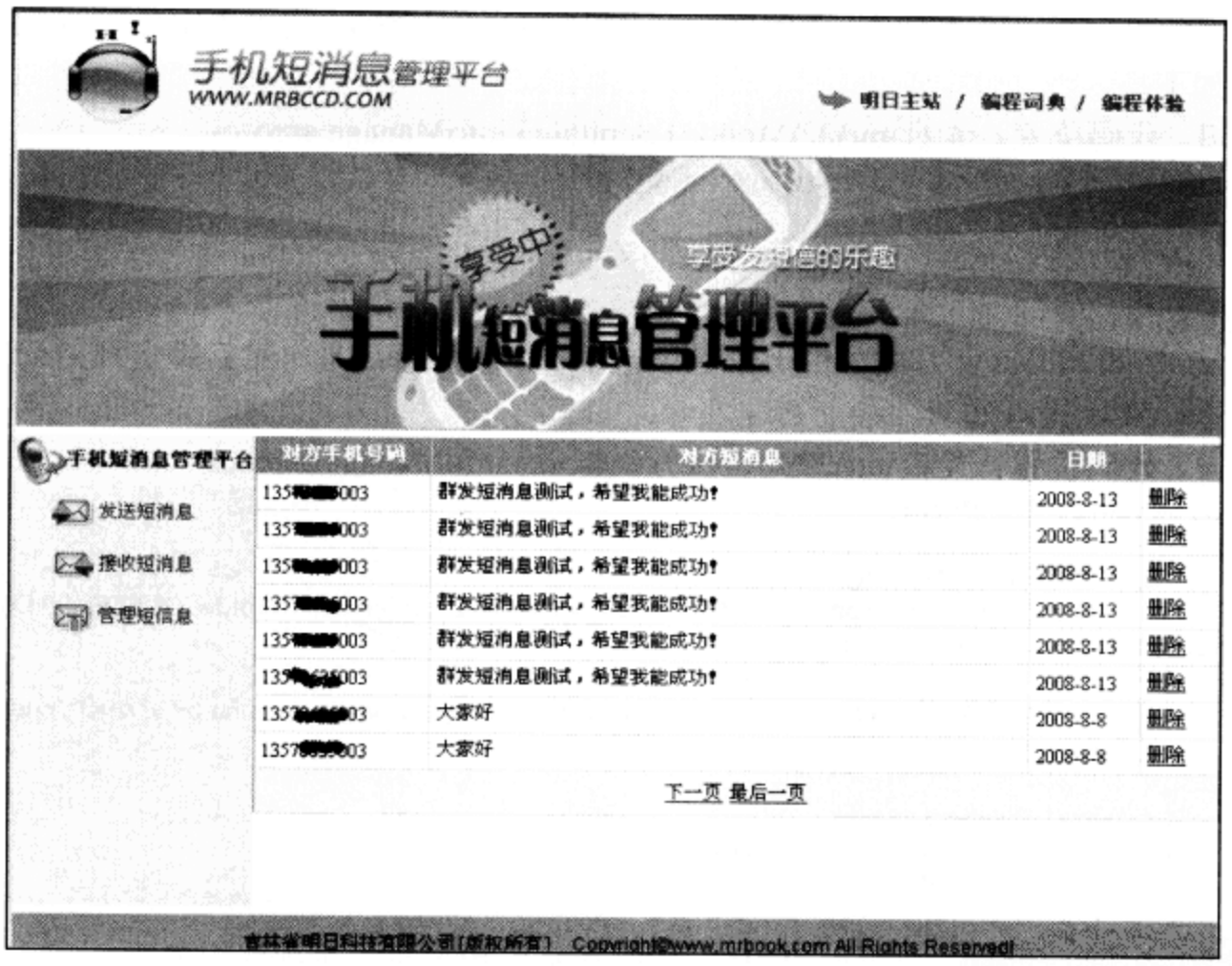


图 22.5 管理手机短消息页面

1. 页面设计

管理手机短消息的页面设计，主要通过 GridView 控件实现，GridView 控件可以通过列表的形式显示所有的短消息，而且可以删除指定的旧短消息。

下面开始编辑 GridView 控件，用于显示短消息和删除指定的短消息。在 GridView 控件的属性对话框中选择 Columns 属性，在 Columns 属性值文本框中单击 按钮弹出如图 22.6 所示对话框，在该对话框中添加 BoundField 字段，用于显示指定列数据，前提必须设置 DataFile 属性（绑定数据库字段），设置 BoundField 字段后，还需要添加 CommandField 字段中的选择功能，用户可以删除指定的短消息。

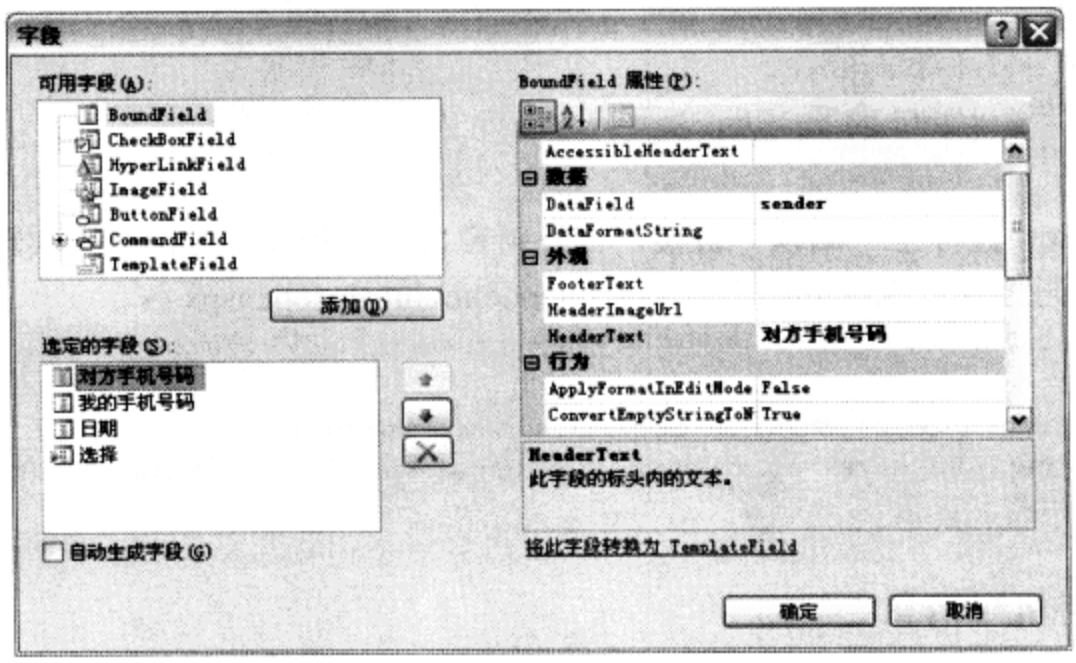


图 22.6　编辑 GridView 控件

2. 代码设计

在页面的 Page_Load 事件中，调用自定义 BindGridView 方法，用来绑定显示所有的短消息。实现代码如下。

例程 8　代码位置：光盘\mr\22\MobileTelephone\ infoManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
        this.BindGridView();
}
```

自定义 BindGridView 方法，用于获取数据库中所有的短消息，然后绑定到 GridView 控件上显示出来。实现代码如下。

例程 9　代码位置：光盘\mr\22\MobileTelephone\ infoManage.aspx.cs

```
private void BindGridView()
{
    DataSet ds = new DataSet();
    using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
+ HttpContext.Current.Server.MapPath("sp.mdb")))
    {
        OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_sp order by id desc", conn);
        da.Fill(ds);
    }
    GridView1.DataSource = ds;
    GridView1.DataKeyNames = new string[] { "id" };
    GridView1.DataBind();
}
```

单击 GridView 控件上“删除”链接，删除指定的短消息。实现代码如下。

例程 10　代码位置：光盘\mr\22\MobileTelephone\ infoManage.aspx.cs

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
    string id = GridView1.DataKeys[e.RowIndex].Value.ToString();
    using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source="
+ HttpContext.Current.Server.MapPath("sp.mdb")))
    {
        OleDbCommand cmd = new OleDbCommand("delete from tb_sp where id=" + id, conn);
        conn.Open();
        cmd.ExecuteNonQuery();
        conn.Close();
    }
    this.BindGridView();
}
```

在 GridView 控件的 RowDataBound 事件下，主要实现鼠标移动到某行时，则该行高亮显示；当单击“删除”链接时，弹出信息提示框。实现代码如下。

例程 11　代码位置：光盘\mr\22\MobileTelephone\ infoManage.aspx.cs

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        //高亮显示指定行
        e.Row.Attributes.Add("onMouseOver", "Color=this.style.backgroundColor;this.style.backgroundColor
='#FFF000'");
        e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=Color;");
        e.Row.Cells[0].Text = e.Row.Cells[0].Text.Substring(2, 11);
        //删除指定行数据时，弹出询问对话框
        ((LinkButton)(e.Row.Cells[3].Controls[0])).Attributes.Add("onclick", "return confirm('是否删除当前行
数据！')");
    }
}
```

22.4　疑难问题分析与解决

本节所讲解的疑难问题是利用母版页避免重复代码的编写。在网站开发过程中，总会碰到在外观和通用行为都相同的页，如果能够利用一个页定义所有页或者一组页面的相同外观和通用行为，将会减轻开发人员的编码负担。要达到这个目的，可以通过 iframe 或 Frameset 框架来实现，也可以通过 ASP.NET 2.0 及以上版本新增的母版页来完成。在手机短消息管理平台程序中功能菜单的导航是利用母版页实现的，它不仅能够利用一个页定义所有页或者一组页面的相同外观和通用行为，而且在需要修改页面布局时只修改母版页即可，如图 22.7 所示。

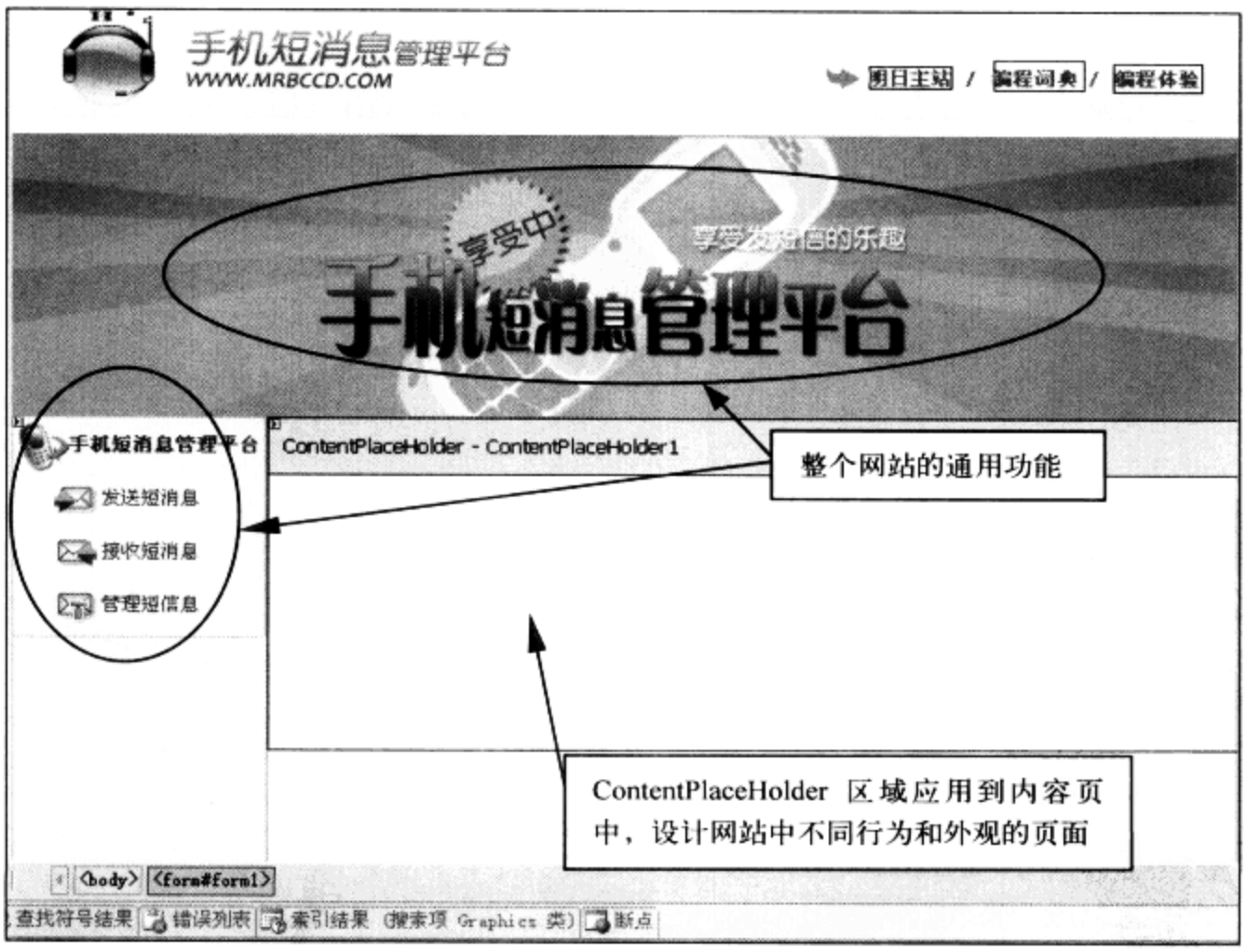
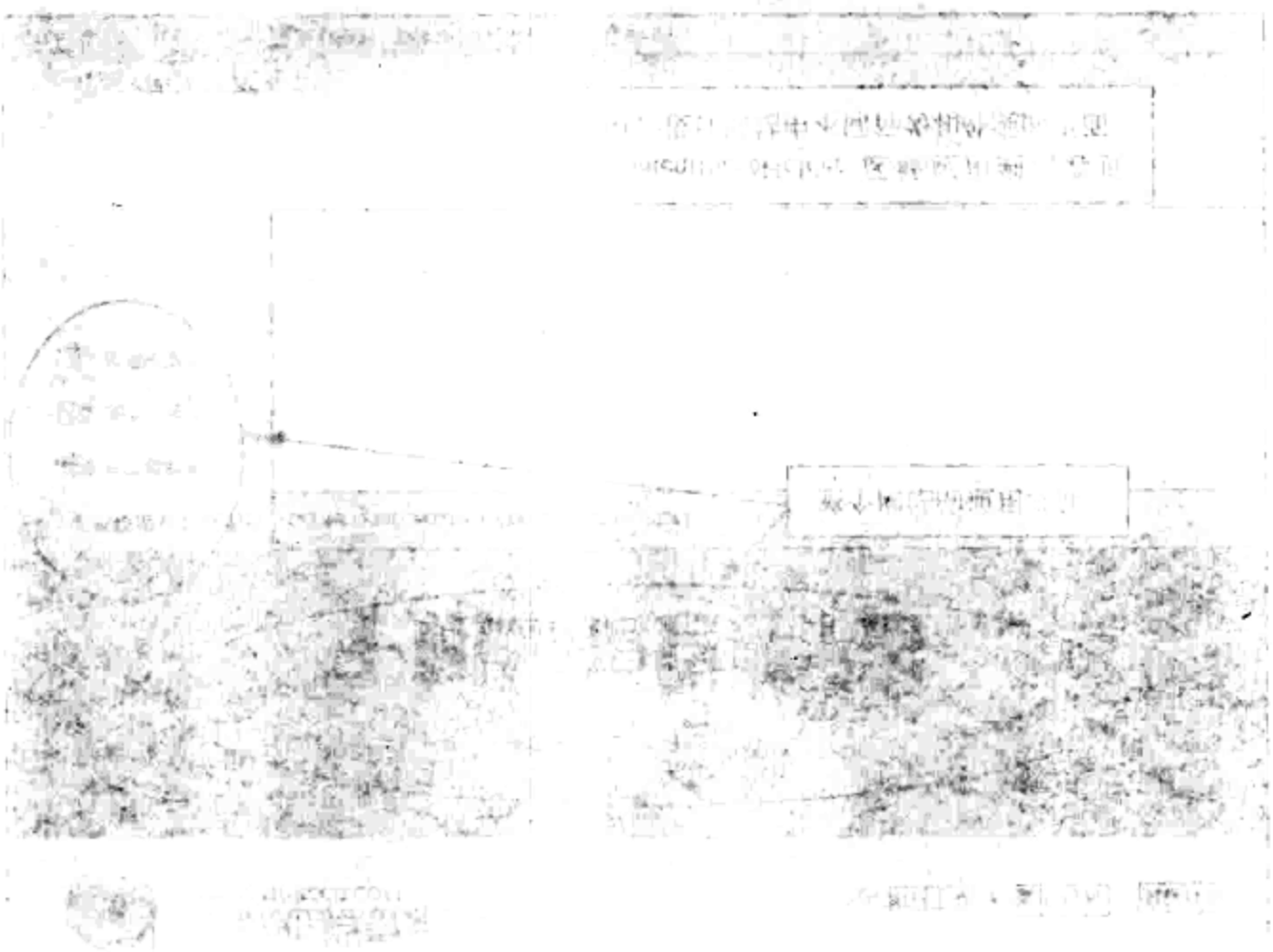


图 22.7　母版页面的设计视图

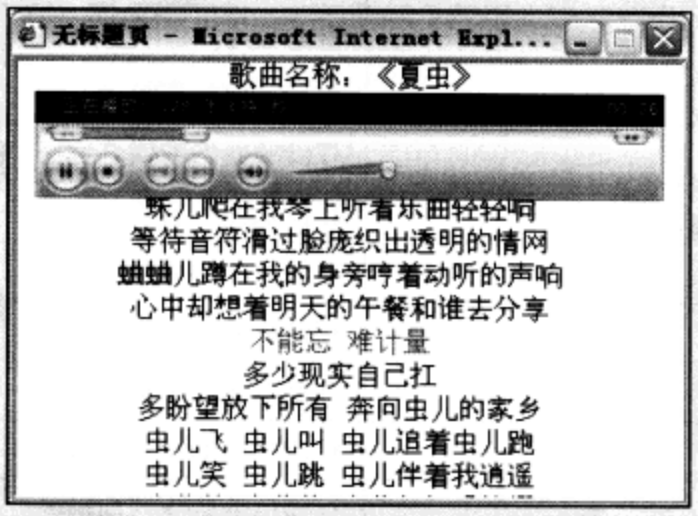


第23章 在线音乐模块

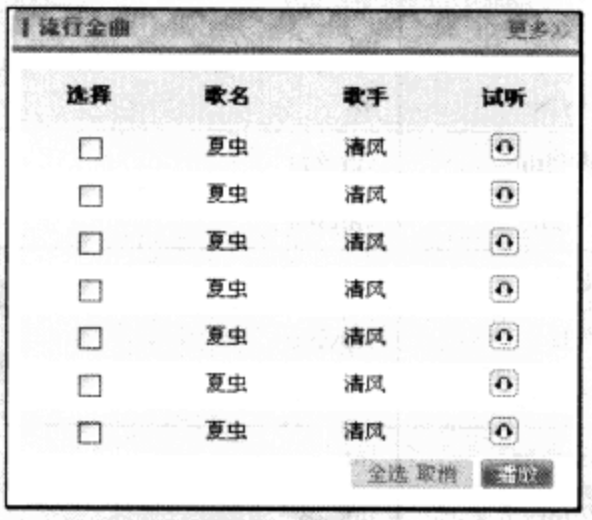
实例位置：光盘\mr\23\

随着人们的生活水平不断的进步，人们的生活压力和工作压力也不断增加。为了缓解人们的压力，现在的网络给人们提供了许多娱乐功能。例如，网络游戏、聊天室、网络电影、在线音乐等。在线音乐可以提供网友们在工作之余听听音乐放松一下自己的心情来减轻生活或工作带来的压力。本章将会介绍在线音乐模块的实现过程。通过本章，读者可以学到以下内容。

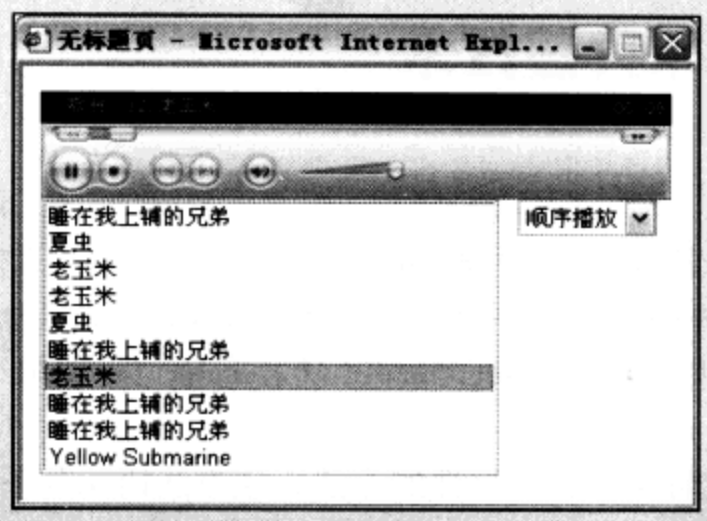
歌词同步显示



最新歌曲排行



歌曲选择播放



在线音乐首页

23.1 在线音乐概述

23.1.1 功能概述

在线音乐模块在网络上是比较流行模块，用户可以在网站中收听自己喜欢的歌曲来放松自己的心情。在本程序中的在线音乐模块提供给用户对某一首歌曲进行单一播放和歌词的同步显示的功能。用户也可以选择自己喜欢的多个歌曲进行顺序播放、随机播放和单曲播放。该模块还提供了下载功能，用户可以对自己喜欢的歌曲下载到本地进行收听。

23.1.2 数据库设计

本程序采用 SQL Server 2000 数据库，在 SQL Server 2000 数据库中创建一个名为 db_music 的数据库，在该数据库中创建一个表。表名为 tb_musicInfo，该表用来存储歌曲的详细信息。该表的表结构如表 23.1 所示。

表 23.1 tb_musicInfo 表的表结构

字　段	类　型	长　度	说　明
id	int	4	自动编号
musicType	int	4	存储歌曲类型编号
specialName	varchar	50	存储歌曲的专辑名称
musicName	varchar	50	存储歌曲的名称
musicPath	varchar	50	存储歌曲的路径
lyricPath	varchar	50	存储歌曲的歌词路径
singerName	varchar	50	存储歌手的名称
auditionSum	int	4	存储歌曲的试听次数
downSum	int	4	存储歌曲的下载次数
fileSize	char	10	存储歌曲文件的大小

23.2 在线音乐关键技术

23.2.1 根据播放模式播放歌曲

在播放歌曲的窗口中，会根据用户所选择的播放模式来播放列表中的所有歌曲。例如用户选择的播放模式是顺序播放，将会按照由上到下的播放顺序来播放列表中的歌曲，运行效果如图 23.1 所示。

实现这个功能主要是通过使用 JavaScript 中的 window.setTimeout 方法每隔 1s 调用一下 isPlay 函数。setTimeout 方法说明如下。

setTimeout 方法用来完成在 JavaScript 脚本定时执行各种事件的方法。该方法语法如下：

```
setTimeout（function,milliseconds）
```

function：要调用的 JavaScript 自定义函数名称。

milliseconds：设置超时时间（以毫秒为并单位）。

在 isPlay 函数中首先判断当前播放的歌曲是否已播放完，如果播放完将调用“类型播放”按钮的单击事件。判断歌曲是否已播放完成，可以通过 Windows Media Player 播放器的 PlayState 属性来实现。如果该属性的值为 1，表示歌曲已经播放完成。实现代码如下。

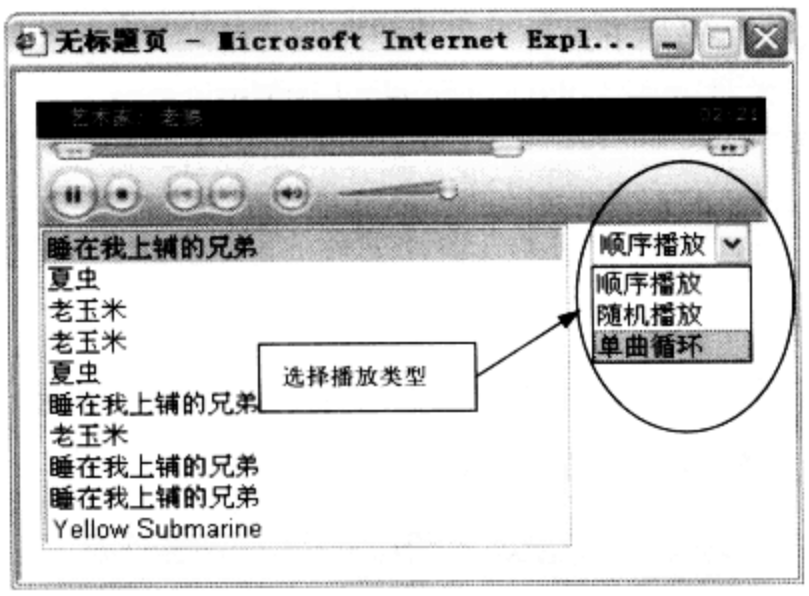


图 23.1　选择播放类型

例程 1　代码位置：光盘\mr\23\playMusic\playListMusic.aspx

```
function isPlay()
{
if(document.getElementById("mediaPlayer").PlayState==1){
            document.getElementById("btnIsPlay").click();
    }
      window.setTimeout(isPlay,1 000);
}
```

23.2.2　选择歌曲播放

选择歌曲播放，使用户可以在播放歌曲页的列表框中选择自己喜欢的歌曲进行播放。运行效果如图 23.2 所示。

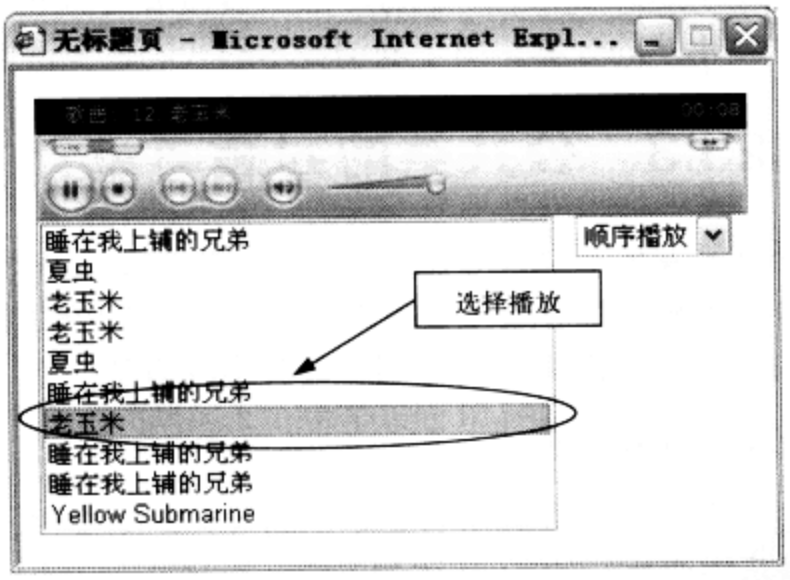


图 23.2　选择播放歌曲

该功能主要是通过 select 控件的 ondblclick 事件来实现的。当双击列表中的某个歌曲将会引发该事件，该事件将会调用使用 JavaScript 声明的 selectPlay 函数，该函数将调用“选择播放”按钮的单击事件。在 selectPlay 函数中获取“选择播放”按钮是通过使用 getElementById 方法来获取的，该方法说明如下。

getElementById 方法可以通过指定的 id 来获取 HTML 标记，并将其返回。语法如下：

```
sElement=document.getElementById(id)
```

sElement：用来接收该方法返回的一个对象。

id：用来设置需要获取 HTML 标记的 id 值。

SelectPlay 函数的实现代码如下。

例程 2 代码位置：光盘\mr\23\playMusic\playListMusic.aspx

```
function selectPlay()
{
    document.getElementById("btnSelectPlay").click();
}
```

23.2.3 歌词同步显示

为了方便用户学习某首歌曲，在本模块中提供了歌词同步显示功能，运行效果如图 23.3 所示。

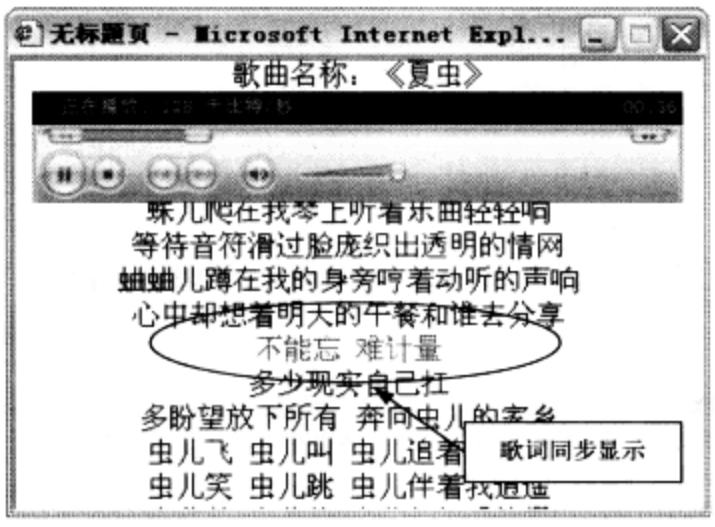


图 23.3 歌词同步显示

要实现显示歌词功能首先需要拥有歌曲的歌词文件，歌词文件是以.lrc 结尾的。该歌词文件中存储的都是某某时间段所演唱的歌词。拥有了歌词文件后就可以使用 JavaScript 代码来实现歌词的显示了。在 JavaScript 中首先使用正则表达式取出“[]”符号中的时间，并将所有歌词获取添加到表格中，再根据 Windows Media Player 播放器播放当前歌曲的时间来滚动歌词并设置所演唱歌词的颜色以方便用户的查看。由于歌词显示的代码较多请读者参见程序中的代码。

23.3 公共类的封装与设计

设计公共类，可以提高开发效率以及方便以后对程序的维护。对于一个好的程序来说，公共类是不可缺少的一部分。在本程序中编写了名为 dataOperate 的公共类，在该公共类中主要实现查询数据和对数据库的操作。下面将详细介绍公共类中的方法。

23.3.1 实现更新、插入、删除操作

自定义 execSql 方法用来实现对数据库的更新、插入和删除操作。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个布尔值变量，当执行成功时将返回 True，否则返回 False。实现代码如下。

例程 3 代码位置：光盘\mr\23\playMusic\dataOperate.cs

```
public static bool execSql(string sql)
{
    //创建数据库连接
    SqlConnection con = createCon();
    //打开数据库连接
    con.Open();
    //创建SqlCommand对象
    SqlCommand com = new SqlCommand(sql, con);
    //获取ExecuteNonQuery方法返回的值
    int i = com.ExecuteNonQuery();
```

```
        //关闭数据库连接
        con.Close();
        //判断返回的值是否大于1，大于1表示执行成功
        if (i > 0)
        {
            return true;
        }
        else
            return false;
    }
```

23.3.2 实现返回指定列的值

自定义 getTier 方法用来实现返回查询的指定列的值。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个字符串变量，该变量表示所查询的指定列的值。实现代码如下。

例程 4 代码位置：光盘\mr\23\playMusic\dataOperate.cs

```
    public static string getTier(string sql)
    {
        //创建数据库连接
        SqlConnection con = createCon();
        //打开数据库连接
        con.Open();
        //创建SqlCommand对象
        SqlCommand com = new SqlCommand(sql, con);
        //获取ExecuteReader方法返回的对象
        SqlDataReader sdr = com.ExecuteReader();
        //读取一条记录
        sdr.Read();
        //获取查询的指定列值
        string str = sdr[0].ToString();
        con.Close();
        sdr.Close();
        return str;
    }
```

23.3.3 实现查询数据返回 SqlDataReader 对象

自定义 getRow 方法用来实现查询数据并返回 SqlDataReader 对象。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个 SqlDataReader 对象，该对象中存储着所查询的数据。实现代码如下。

例程 5 代码位置：光盘\mr\23\playMusic\dataOperate.cs

```
    public static SqlDataReader getRow(string sql)
    {
        //创建数据库连接对象
        SqlConnection con = new SqlConnection("server=.;database=db_music;uid=sa;pwd=;");
        //打开数据库连接
        con.Open();
        //创建SqlCommdand对象
        SqlCommand com = new SqlCommand(sql, con);
        //获取ExecuteReader方法返回的SqlDataReader对象
        SqlDataReader sdr = com.ExecuteReader();
        return sdr;
    }
```

23.3.4 实现查询数据返回 DataSet 对象

自定义 getRows 方法用来实现查询数据并返回 DataSet 对象。调用该方法需要传入一个字符串变量，该变量表示需要执行的 SQL 语句。该方法将返回一个 DataSet 对象，该对象中存储

着所查询的数据。实现代码如下。

例程 6　代码位置：光盘\mr\23\playMusic\dataOperate.cs

```
public static DataSet getRows(string sql)
    {
        //创建数据库连接
        SqlConnection con = new SqlConnection("server=.;database=db_music;uid=sa;pwd=;");
        //打开数据库连接
        con.Open();
        //创建SqlDataAdapter对象
        SqlDataAdapter sda = new SqlDataAdapter(sql, con);
        //创建DataSet对象
        DataSet ds = new DataSet();
        //填充DataSet对象
        sda.Fill(ds);
        con.Close();
        return ds;
    }
```

23.4　在线音乐实现过程

23.4.1　在线音乐首页设计

在线音乐首页用来显示所有类型歌曲和最新发布的歌曲，在该页面中用户可以选择某一类型的歌曲进行查看。在线音乐首页如图 23.4 所示。

图 12.4　在线音乐首页

1. 前台页面设计

(1) 创建一个 Web 窗体，默认名为 Default.aspx

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 23.2 所示。

表 23.2 控件类型、控件名称及说明

控件类型	控件名称	主要属性	说明
数据/GridView 控件	gvFashion	均为默认值	显示流行金曲类歌曲信息
	gvOld	均为默认值	显示经典老歌类歌曲信息
	gvAudition	均为默认值	显示试听排行歌曲信息
	gvOccident	均为默认值	显示欧洲经典类歌曲信息
	gvCampus	均为默认值	显示校园民谣类歌曲信息
标准/ImageButton 控件	imgBtnFFull	均为默认值	全选或全部取消流行金曲操作
	imgBtnOldFull	均为默认值	全选或全部取消经典老歌操作
	imgBtnAFull	均为默认值	全选或全部取消试听排行操作
	imgBtnOcFull	均为默认值	全选或全部取消欧洲经典操作
	imgBtnCFull	均为默认值	全选或全部取消校园民谣操作
	imgBtnFPlay	均为默认值	播放所选择的流行金曲操作
	imgBtnOldPlay	均为默认值	播放所选择的经典老歌操作
	imgBtnAPlay	均为默认值	播放所选择的试听排行操作
	imgBtnOcPlay	均为默认值	播放所选择的欧洲经典操作
	imgBtnCPlay	均为默认值	播放所选择的校园民谣操作

2. 后台代码编写

在页面加载事件中将调用自定义方法，使 GridView 控件显示不同类型的歌曲信息。实现代码如下。

例程 7 代码位置：光盘\mr\23\playMusic\Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //调用自定义bindAudition方法，显示试听排行帮
        bindAudition();
        //调用自定义bindFashion方法，显示流行金曲
        bindFashion();
        //调用自定义bindOld方法，显示经典老歌
        bindOld();
        //调用自定义bindOccident方法，显示欧洲经典
        bindOccident();
        //调用自定义bindCampus方法，显示校园民谣
        bindCampus();
    }
}
```

自定义 bindAudition 方法用来绑定 GridView 控件显示试听排行音乐。在该方法中使用 SQL 语句查询出试听次数最多的 10 首歌曲，并将歌曲信息绑定到 GridView 控件上。实现代码如下。

例程 8 代码位置：光盘\mr\23\playMusic\Default.aspx.cs

```
protected void bindAudition()
{
 string sqlSel = "select top 10 * from tb_musicInfo order by auditionSum desc";
     gvAudition.DataSource = dataOperate.getRows(sqlSel);
```

```
        gvAudition.DataKeyNames = new string[] { "id" };
        gvAudition.DataBind();
    }
```

自定义 allSelect 方法用来实现全选歌曲或取消全选歌曲的操作。调用该方法需要传入两个参数，第一个参数是 GridView 对象是需要设置的 GridView 控件，第二个参数是布尔值，该值用来表示是选中操作还是取消操作。在该方法中首先将判断传入的布尔值是执行全选操作还是全部取消操作，根据不同的操作来设置 GridView 控件中的复选框状态。实现代码如下。

例程 9 代码位置：光盘\mr\23\playMusic\Default.aspx.cs

```
protected void allSelect(GridView gv, bool bl)
{
    //判断传入的复选框对象是否为选中状态
    if (bl)
    {
        //遍例传入的GridView对象
        foreach (GridViewRow gvr in gv.Rows)
        {
            //设置GridView控件中的所有复选框为选中状态
            ((CheckBox)gvr.FindControl("CheckBox1")).Checked = false;
        }
    }
    else
    {
        foreach (GridViewRow gvr in gv.Rows)
        {
            //设置GridView控件中的所有复选框为不选中状态
            ((CheckBox)gvr.FindControl("CheckBox1")).Checked = true;
        }
    }
}
```

自定义 playList 方法用来实现获取 GridView 控件中所有选中歌曲的 id，并将 id 传入歌曲播放窗口中。调用该方法需要传入两个参数，第一个参数是 GridView 对象，该对象表示要获取的 GridView 控件。第二个参数是布尔值，该布尔值表示 GridView 控件中是否有选中的复选框。在该方法中首先判断是否有选中的复选框，如果没有将给出提示信息。如果有将通过循环获取 GridView 控件中所有被选中歌曲的 id，并将歌曲的 id 传入到歌曲播放窗口中。实现代码如下。

例程 10 代码位置：光盘\mr\23\playMusic\Default.aspx.cs

```
protected void playList(GridView gv, bool bl)
    {
        //判断GridView控件中是否有选中的复选框
        if (bl)
        {
            //创建字符串变量
            string musicList = "";
            //循环GridView控件
            for (int i = 0; i < gv.Rows.Count; i++)
            {
                //判断复选框是否为选中状态
                if (((CheckBox)gv.Rows[i].FindControl("CheckBox1")).Checked)
                {
                    //获取GridView控件的主键
                    musicList += gv.DataKeys[i].Value.ToString() + ",";
                }
            }
            //使用javaScript打开播放音乐列表窗口
            Response.Write("<script>window.open('playListMusic.aspx?id=" + musicList + "','','width=380,height=260')</script>");
        }
        else
        {
```

```
            RegisterStartupScript("", "<script>alert('请选择需要播放的歌曲！')</script>");
        }
    }
```

自定义 isCheck 方法用来判断某个 GridView 控件中的复选框是否有选中的状态。如果有将返回布尔值 True，否则返回 False。调用该方法需要传入一个 GridView 对象。该对象表示需要查询的 GridView 控件。实现代码如下。

例程 11 代码位置：光盘\mr\23\playMusic\Default.aspx.cs

```
protected bool isCheck(GridView gv)
{
    //创建布尔值变量并将变量设置为False
    bool c = false;
    //遍例GridView控件
    foreach (GridViewRow gvr in gv.Rows)
    {
        //判断复选框状态是否选中
        if (((CheckBox)gvr.FindControl("CheckBox1")).Checked)
        {
            //设置布尔值为Treu
            c = true;
            break;
        }
    }
    return c;
}
```

23.4.2 歌曲详细信息页设计

歌曲详细信息页用来显示某一种类型的全部歌曲，用户可以在该页面中选择自己喜欢的歌曲进行试听或播放。歌曲详细信息页如图 23.5 所示。

图 23.5 歌曲详细信息页

1. 前台页面设计

(1) 创建一个 Web 窗体，命名为 Register.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 23.3 所示。

表 23.3 控件类型、控件名称及说明

控件类型	控件名称	主要属性	说明
标准/GridView 控件	gvMusic	均为默认值	显示歌曲的详细信息
	gvAudition	均为默认值	显示试听排行榜信息
标准/ImageButton 控件	imgBtnHaving	均为默认值	实现全选或全部取消歌曲的操作
	imgBtnReverse	均为默认值	实现反选歌曲的操作
	imgBtnSeries	均为默认值	实现播放歌曲的操作
	imgBtnAFull	均为默认值	实现全选或全部取消歌曲的操作
	imgBtnAPlay	均为默认值	实现播放歌曲的操作

2. 后台代码编写

在页面加载事件中将调用自定义 bindGV 方法和自定义 bindAudition 方法显示歌曲的详细信息，在自定义 bindGV 方法中使用 SQL 语句查询出歌曲的详细信息，并将查询出的歌曲信息绑定到 GridView 控件上显示出来。实现代码如下。

例程 12 代码位置：光盘\mr\23\playMusic\Register.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //调用自定义bindGV方法，显示歌曲的详细信息
        bindGV();
        bindAudition();
    }
    //根据歌曲类型显示图片
    string id = Request["id"].ToString();
    switch (id)
    {
        case "1": imgTb.ImageUrl = "~/images/lxjq.GIF"; break;
        case "2": imgTb.ImageUrl = "~/images/jdlg.GIF"; break;
        case "3": imgTb.ImageUrl = "~/images/omjd.GIF"; break;
        case "4": imgTb.ImageUrl = "~/images/xymy.GIF"; break;
    }
}
//自定义方法将歌曲的详细信息显示在GridView控件上
protected void bindGV()
{
    //创建SQL语句，该语句用来查询歌曲的详细信息
    string sqlSel = "select * from tb_musicInfo where musicType=" + Request["id"].ToString();
    //设置GridView控件的数据源
    gvMusic.DataSource = dataOperate.getRows(sqlSel);
    //设置GridView控件的主键
    gvMusic.DataKeyNames = new string[] { "id" };
    //绑定GridView控件
    gvMusic.DataBind();
}
```

歌曲的下载操作在 GridView 控件的 RowCommand 事件中实现。在该事件中首先将获取歌曲的 id，再根据 id 查询出歌曲的路径信息并调用自定义 downFile 方法实现歌曲的下载操作。实现代码如下。

例程 13　代码位置：光盘\mr\23\playMusic\Register.aspx.cs

```
protected void gvMusic_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        //判断是否为下载操作
        if (e.CommandName == "down")
        {
            //获取歌曲的id
            string id = e.CommandArgument.ToString();
            //创建SQL语句，查询歌曲的详细信息
            string sqlSel = "select musicPath from tb_musicInfo where id=" + id;
            //创建SQL语句，更新歌曲的下载次数
            string sqlUpdate = "update tb_musicInfo set downSum=downSum+1 where id=" + id; ;
            //执行SQL语句
            dataOperate.execSql(sqlUpdate);
            //调用自定义downFile方法实现歌曲的下载操作
            downFile(dataOperate.getTier(sqlSel));
        }
    }
```

自定义 downFile 方法用来实现歌曲的下载操作。调用该方法需要传入一字符串变量，该变量表示歌曲的路径。在该方法中使用 FileInfo 对象的 Exists 方法判断需要下载的文件是否存在，如果不存在将给出提示。如果存在将使用 Http 实现下载操作。实现代码如下。

例程 14　代码位置：光盘\mr\23\playMusic\Register.aspx.cs

```
protected void downFile(string musicPath)
    {
        string path = Server.MapPath("musicFile/") + musicPath;
        //初始化 FileInfo 类的实例，它作为文件路径的包装
        FileInfo fi = new FileInfo(path);
        //判断文件是否存在
        if (fi.Exists)
        {
            //将文件保存到本机上
            Response.Clear();
            Response.AddHeader("Content-Disposition", "attachment; filename=" + Server.UrlEncode(fi.Name));
            Response.AddHeader("Content-Length", fi.Length.ToString());
            Response.ContentType = "application/octet-stream";
            Response.Filter.Close();
            Response.WriteFile(fi.FullName);
            Response.End();
        }
    }
```

23.4.3　歌曲试听设计

在歌曲试听页面中播放用户所选择的歌曲并同步显示歌曲的歌词信息。歌曲试听页面如图 23.6 所示。

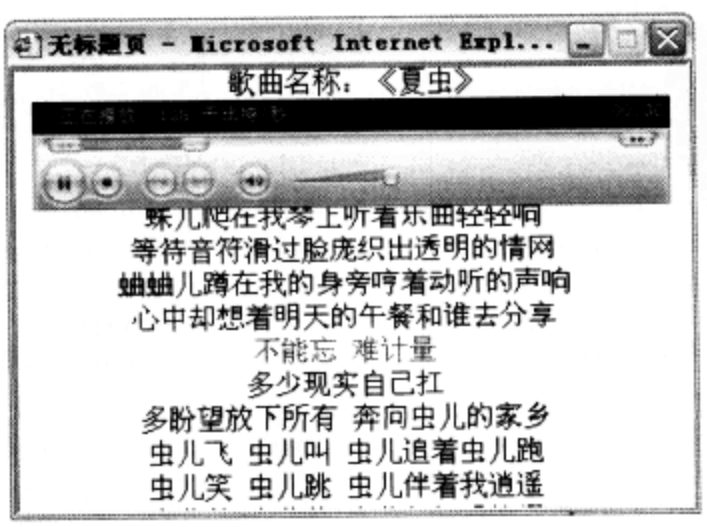


图 23.6　歌曲试听

1. 前台页面设计

（1）创建 1 个 Web 窗体，命名为 playMusic.aspx。

（2）在该页面中添加 1 个 Literal 控件用来显示歌词，还需要使用 HTML 代码添加 1 个播放器，该播放器用来播放歌曲。播放器的插入代码如下。

例程 15 代码位置：光盘\mr\23\playMusic\playMusic.aspx

```
<object classid='clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6'  id='mediaPlayer' height='64' style="width: 296px">
<param name='url' value=<%=fileUrl %>><param name='volume' value='100'><param name='playcount' value='100'><param
name='enablecontextmenu' value='0'><param name='enableerrordialogs' value='0'>
                <embed width='420' height='360' src=<%=fileUrl %> type="audio/x-pn-realaudio-plugin" controls="ImageWindow"
autostart="true"
                    ></embed></object>
```

2. 后台代码编写

在后台代码中首先创建几个全局变量来保存歌曲的信息，在页面的加载事件中首先更新歌曲的试听次数，再查询歌曲的信息并保存。最后使用 Literal 控件来设置显示的歌词。实现代码如下。

例程 16 代码位置：光盘\mr\23\playMusic\playMusic.aspx.cs

```
//设置全局变量用来存储歌曲路径
public string fileUrl;
//设置全局变量用来存储歌曲名称
public string fileName;
//设置全局变量用来存储歌曲的歌词
public string str;
//设置全局变量用来存储歌词中“]”总数
public int sysum;
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        //创建SQL语句，根据传入的歌曲id更新歌曲的试听次数
        string sqlUp = "update tb_musicInfo set auditionSum=auditionSum+1 where id=" + Request["id"];
        dataOperate.execSql(sqlUp);
        //创建SQL语句，根据歌曲的id查询歌曲信息
        string sqlSel = "select * from tb_musicInfo where id=" + Request["id"];
        //调用公共类中的getRow方法并接收该方法返回的对象
        SqlDataReader sdr = dataOperate.getRow(sqlSel);
        //读取一条记录
        sdr.Read();
        //获取歌曲的路径
        fileUrl = setUrl("musicFile/" + sdr["musicPath"]);
        //获取歌曲的名称
        fileName = sdr["musicName"].ToString();
        //获取歌词
        str = File.ReadAllText(Server.MapPath("musicFile//" + sdr["lyricPath"]), System.Text.Encoding.GetEncoding("gb2312"));
        //设置显示歌词的div
        Literal1.Text += "<div align='center'   id='lrcAreaDiv' style='   height:60px; width:300x;overflow:hidden;'>";
        Literal1.Text += "<table border='0' cellspacing='0' cellpadding='0' id='lrcArea' width='100%' style='po sition:
relative; top:120px;'>";
        Literal1.Text += "<tr><td nowrap height='20' align='center'>";
        Literal1.Text += "<table border='0' cellspacing='0' cellpadding='0'>";
        Literal1.Text += "<tr><td nowrap height='20'><span id='lrcLine1' style='height:20; color:#FF0000'>正在加载歌
词…</span></td></tr>";
        Literal1.Text += "<tr style='position:relative; top: -20px; z-index:6;'>";
        Literal1.Text += "<td nowrap height='20'><div id='lrcLine_will1' class='lrcLine_will'></div></td></tr></table>";
        Literal1.Text += "</td></tr>";
        //获取歌词中“[”总数
        sysum = getStr(sdr["lyricPath"].ToString());
        //循环添加表格
        for (int i = 0; i < getStr(sdr["lyricPath"].ToString()); i++)
```

```
            {
                Literal1.Text += "<tr style='position:relative; top: " + -20 * i + "px;'><td nowrap height='20' align='center'>";
                Literal1.Text += "<table border='0' cellspacing='0' cellpadding='0'>";
                Literal1.Text += "<tr><td nowrap height='20'><span id='lrcLine" + (i + 2) + "' style='height:20'></span></td></tr>";
                Literal1.Text += "<tr style='position:relative; top: -20px; z-index:6;'>";
                Literal1.Text += "<td nowrap height='20'><div id='lrcLine_will" + (i + 2) + "' class ='lrcLine_will'></div></td>";
                Literal1.Text += "</tr></table></td></tr>";
            }
            Literal1.Text += "</table></div>";
        }
    }
```

23.4.4 播放歌曲设计

在播放歌曲页面中，将播放用户所选择的所有歌曲，在该页面中用户还可以选择歌曲的播放模式，如顺序播放、随机播放和单曲播放等。播放歌曲页面如图 23.7 所示。

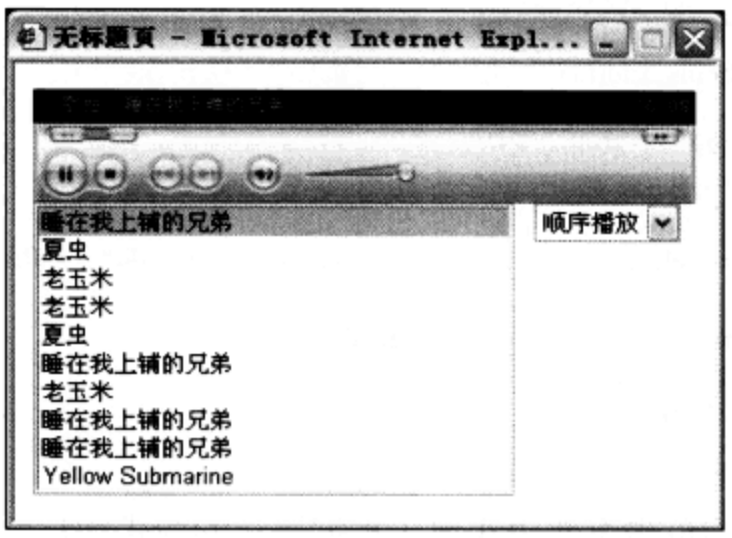


图 23.7 播放歌曲

1. 前台页面设计

(1) 创建 1 个 Web 窗体，命名为 playListMusic.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 23.4 所示。

表 23.4 控件类型、控件名称及说明

控 件 类 型	控 件 名 称	主 要 属 性	说 明
HTML/select 控件	Select1	均为默认值	用于显示播放歌曲的列表
标准/DropDownList 控件	ddlPlayType	均为默认值	用于选择歌曲的播放类型
标准/Button 控件	btnIsPlay	均为默认值	实现选择播放
	btnSelectPlay	均为默认值	根据播放类型实现播放

2. 后台代码编写

在后台代码中创建一个全局变量，用来存储歌曲的路径。在页面的加载事件中获取传入的歌曲，并将传入的歌曲以“,”分隔存储到数组中。通过循环将数组中歌曲 ID 设置为 SQL 语句的查询条件，查询出歌曲的名称。再将歌曲名称和歌曲 ID 添加到播放列表中。实现代码如下。

例程 17 代码位置：光盘\mr\23\playMusic\playListMusic.aspx.cs

```
//全局变量存储歌曲路径
public string fileUrl;
protected void Page_Load(object sender, EventArgs e)
```

```
{
    if (!IsPostBack)
    {
        //获取多个歌曲的id
        string[] sid = Request["id"].ToString().Split(',');
        //清空播放列表
        Select1.Items.Clear();
        //循环添加歌曲的列表
        for (int i = 0; i < sid.Length - 1; i++)
        {
            //创建SQL语句，查询歌曲的名称
            string sql = "select musicName from tb_musicInfo where id=" + sid[i];
            //创建ListItem对象
            ListItem lit = new ListItem();
            //添加歌曲的id值
            lit.Value = sid[i];
            //添加歌曲的文本
            lit.Text = dataOperate.getTier(sql);
            //将所有歌曲添加到列表中
            Select1.Items.Add(lit);
        }
        //设置列表中第一个歌曲为选中状态
        Select1.Items[0].Selected = true;
        //创建SQL语句获取第一个歌曲的路径
        string sqlSel = "select musicPath from tb_musicInfo where id=" + sid[0];
        //设置路径
        string url = "musicFile/" + dataOperate.getTier(sqlSel);
        //将路径保存到全局变量中
        fileUrl = setUrl(url);
    }
}
```

在“类型播放”按钮的单击事件中实现了根据播放的模式播放歌曲。歌曲的播放模式主要有 3 种，分别为顺序播放、随机播放和单曲播放。在该事件中将判断是哪种播放模式，再通过获取歌曲的 ID 来实现所选择的播放模式效果。实现代码如下。

例程 18 代码位置：光盘\mr\23\playMusic\playListMusic.aspx.cs

```
protected void btnIsPlay_Click(object sender, EventArgs e)
{
    string id = "";
    //获取列表中当前所选择的索引号
    int selectIx = Select1.SelectedIndex;
    //判断播放的模式0表示顺序播放
    if (ddlPlayType.SelectedValue == "0")
    {
        //获取列表项的总数
        int listCount = Select1.Items.Count;
        //获取当前选择的id
        int playId = Convert.ToInt32(Select1.Items[Select1.SelectedIndex].Value);
        //判读播放是否为列表中最后一首歌
        if ((selectIx+1) == listCount)
        {
            //取消当前歌曲的选择
            Select1.Items[selectIx].Selected = false;
            //设置第一首歌为选择状态
            Select1.Items[0].Selected = true;
            //获取第一首歌的id
            id = Select1.Items[0].Value;
        }
        else
        {
            //获取下一首个的id
```

```
            id = Select1.Items[Select1.SelectedIndex + 1].Value;
            //取消当前歌曲的选择状态
            Select1.Items[selectIx].Selected = false;
            //设置下一首歌为选择状态
            Select1.Items[selectIx + 1].Selected = true;
        }
    }
        //判断是否为随机播放
    else if (ddlPlayType.SelectedValue == "1")
    {
        //创建Random对象
        Random rad = new Random();
        //创建一个随机数
        int radIx = rad.Next(0, Select1.Items.Count);
        //获取歌曲id
        id = Select1.Items[radIx].Value;
        //取消当前歌曲的选择状态
        Select1.Items[selectIx].Selected = false;
        //设置下一首歌为选择状态
        Select1.Items[radIx].Selected = true;
    }
        //单曲播放
    else
    {
        //获取歌曲的id
        id = Select1.Items[selectIx].Value;
    }
    //创建SQL语句，获取歌曲的路径
    string sqlSel = "select musicPath from tb_musicInfo where id=" + id;
    //保存歌曲的路径
    fileUrl = setUrl("musicFile//" + dataOperate.getTier(sqlSel));
}
```

在“选择播放”按钮的单击事件通过获取当前所选择的歌曲 ID 来查询出歌曲的路径，并将该路径保存到全局变量中使播放器播放所选择的歌曲。实现代码如下。

例程 19 代码位置：光盘\mr\23\playMusic\playListMusic.aspx.cs

```
protected void btnSelectPlay_Click(object sender, EventArgs e)
{
    //获取当前在列表中选择的id
    string id = Select1.Items[Select1.SelectedIndex].Value;
    //创建SQL语句，根据id查询出歌曲的路径
    string sqlSel = "select musicPath from tb_musicInfo where id=" + id;
    //保存歌曲的路径
    fileUrl = setUrl("musicFile//" + dataOperate.getTier(sqlSel));
}
```

23.5 程序调试与错误处理

在歌曲详细信息页中的加载事件中，使用了 if 语句来判断 IsPostBack 属性。IsPostBack 属性获取一个布尔值，该值表示是正为响应客户端回发而加载页面还是第一次加载页面。这个属性是比较重要而常用的属性，很多初学者都会忽略这个属性的使用而造成程序的错误。下面笔者将结合本程序来演示一下忽略 IsPostBack 而引发的错误。

在歌曲详细信息页中要实现的功能是，在该页面中显示某种类型的所有歌曲。例如，在该页面中显示所有类型为流行金曲的歌曲。用户可以通过复选框选择几个自己喜欢的歌曲进行连续播放。实现这个功能首先在页面加载事件中调用自定义 bindGV 方法来显示所有歌曲。在页面加载事件中如果不使用 IsPostBack 属性就会达不到所要实现的效果。在页面加载事件中使用如下代码：

```
protected void Page_Load(object sender, EventArgs e)
{
        //调用自定义bindGV方法，显示歌曲的详细信息
        bindGV();
}
```

运行程序选择首页中流行金曲“更多”按钮，进入到歌曲详细信息页。在该页中选择“全选”按钮，运行效果如图 23.8 所示。

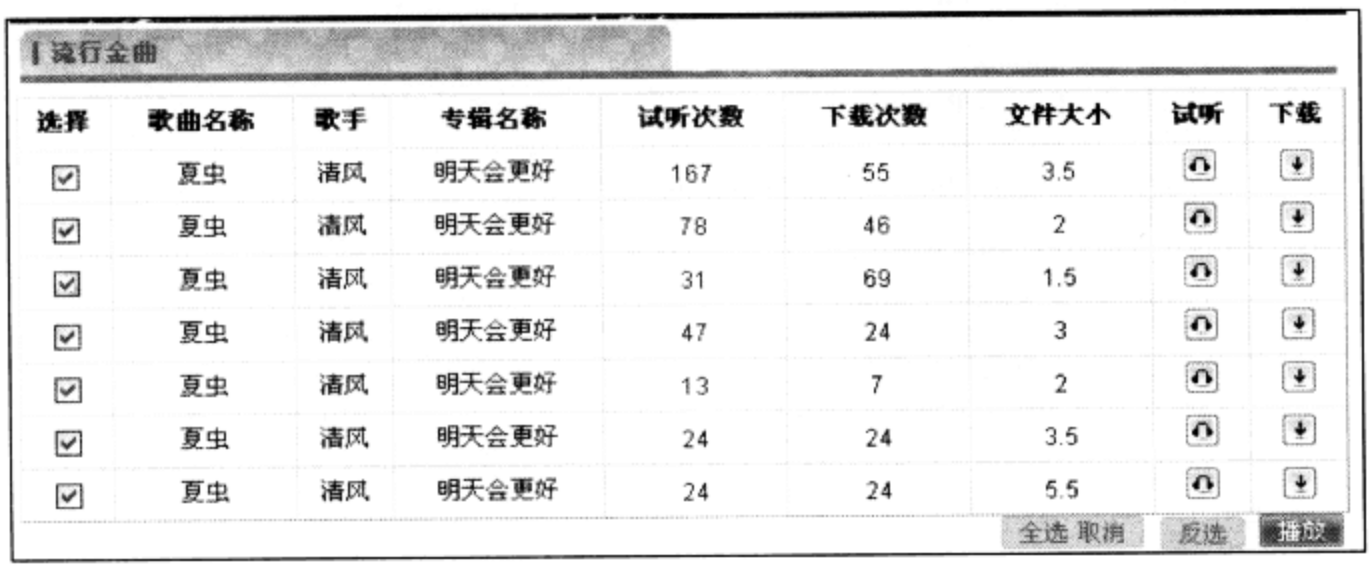


图 23.8　全选所有歌曲

所有歌曲被选择后单击“播放”按钮，会出现“请选择需要播放的歌曲”的提示。运行效果如图 23.9 所示。

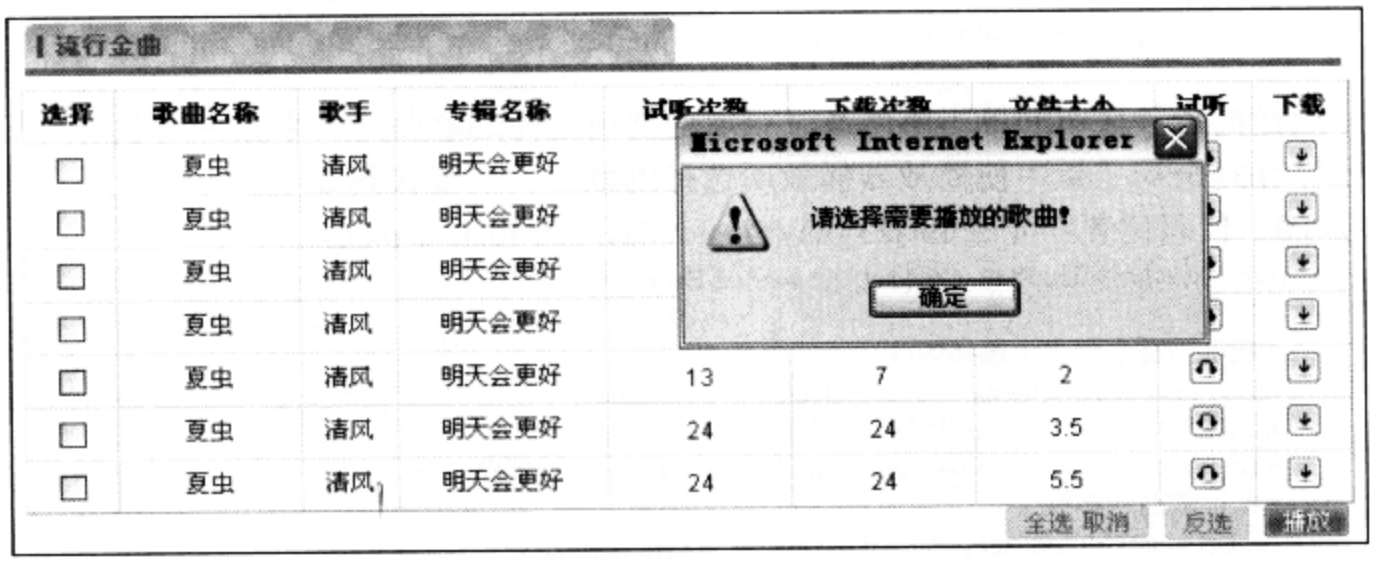


图 23.9　错误提示

此时错误已经体现出来了，用户已经全选了所有歌曲，但是当单击“播放”按钮还会给出“请选择需要播放的歌曲”的提示，这就是没有使用 IsPostBack 属性而引起的。当用户单击“播放”按钮后在页面加载事件中会调用自定义 bindGV 方法将歌曲的详细信息重新绑定到 GridView 控件上，而用户所选择的歌曲将会被取消掉，在“播放”按钮的单击事件中循环查询 GridView 控件中的复选框的状态也都是未选中状态，所以在页面加载事件中必须要判断 IsPostBack 属性，当页面加载为第一次时才会调用 bindGV 方法显示所有歌曲。

电子相册模块

第 24 章

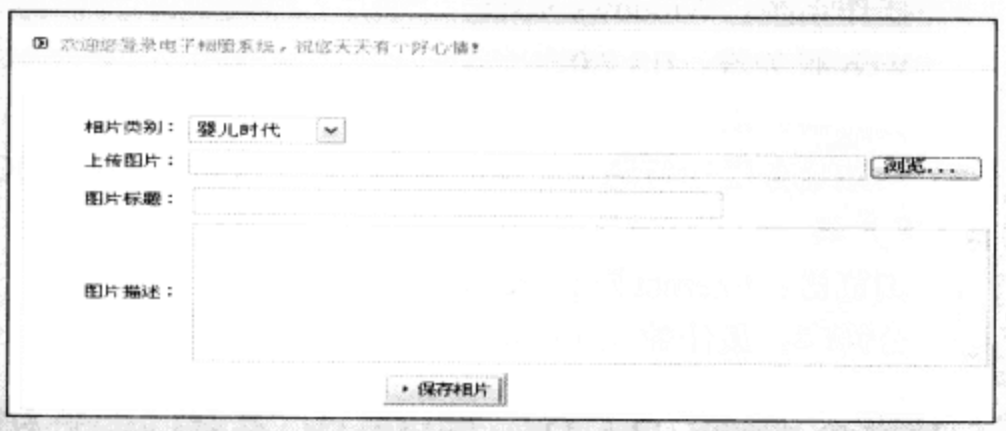
实例位置：光盘\mr\24\

电子相册，是以各类照片为基本素材，用计算机显示而成的作品。是一种现代新兴的影视艺术形式。与电影电视节目一样，它同样具有图文并茂的视觉冲击效果。富有极强的叙事性、观赏性和艺术表现力。电子相册，不仅能以艺术摄像的各种变换手法较完美地展现摄影（照片）画面的精彩瞬间，给家庭和亲友带来振奋和欢乐，并可通过文字说明，充分展示照片主题，发掘相册潜在的思想内涵，还能作为家庭或个人艺术档案资料。通过本章的学习，读者能够学到以下内容。

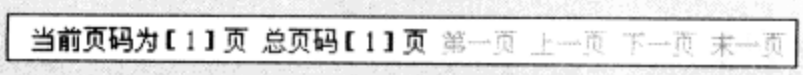
缩略图显示电子相册

树状菜单导航

上传照片



分页显示照片



相片显示控制

24.1 电子相册概述

随着计算机技术的发展和互联网时代的到来，人类已经进入了信息时代，也有人称为数字化时代。在数字化的网络环境下，网络用户希望得到个性化的满足，电子相册正是迎合这一时代需求而开发的，电子相册具有欣赏方便、容易保存、成本低廉、选择性强等优点。

24.1.1 需求分析

通过实际调查，要求电子相册具有以下功能。

- 用户管理：系统管理员能够对系统中的用户进行维护，包括用户注册、修改、删除、搜索和查看等操作。
- 相册分类管理：用户对自己的相册分类进行管理，包括增加、修改、删除和浏览类别。
- 相册管理：用户对自己的相片进行管理，包括更改相片的类别及相片存放位置、上传新相片、删除和浏览等操作。
- 相册浏览：按照用户选择的某一类别的相片进行有序显示，允许用户选择手动浏览和自动播放两种浏览方式。自动播放方式提供百叶窗和渐入/渐出等幻灯片播放效果，也能够设置相片切换时间参数。
- 修改密码：提供用户修改密码的功能，密码经加密后存入数据库。

24.1.2 开发环境

1. 网站开发环境

- 网站开发环境：Microsoft Visual Studio 2008 集成开发环境。
- 网站开发语言：ASP.NET 3.5 + C#。
- 网站后台数据库：Access 2000。
- 开发架构：ASP.NET 3.5 + Ajax。
- 开发环境运行平台：Windows XP（SP2）/ Windows 2000（SP4）/ Windows Vista/ Windows Server 2003（SP1）。

注意

SP（Service Pack）为 Windows 操作系统补丁。

2. 服务器端

- 操作系统：Windows Vista。
- Web 服务器：IIS 7.0。
- 浏览器：IE 7.0。
- 网站服务器运行环境：Microsoft .NET Framework SDK v3.5。

3. 客户端

- 浏览器：Internet Explorer 7.0。
- 分辨率：最佳效果 1 024 × 768 像素。

24.2 实现电子相册关键技术

24.2.1 在 ASP.NET 中搭建 Ajax 开发环境

1. 下载 ASP.NET 中的 Ajax 框架

下载 Ajax 具体步骤如下。

（1）访问 Http：//Ajax.asp.net 官方网站，单击首页下方的“Download ASP.NET Ajax V1.0”超级连接，如图 24.1 所示。进入下一个页面，如图 24.2 所示。

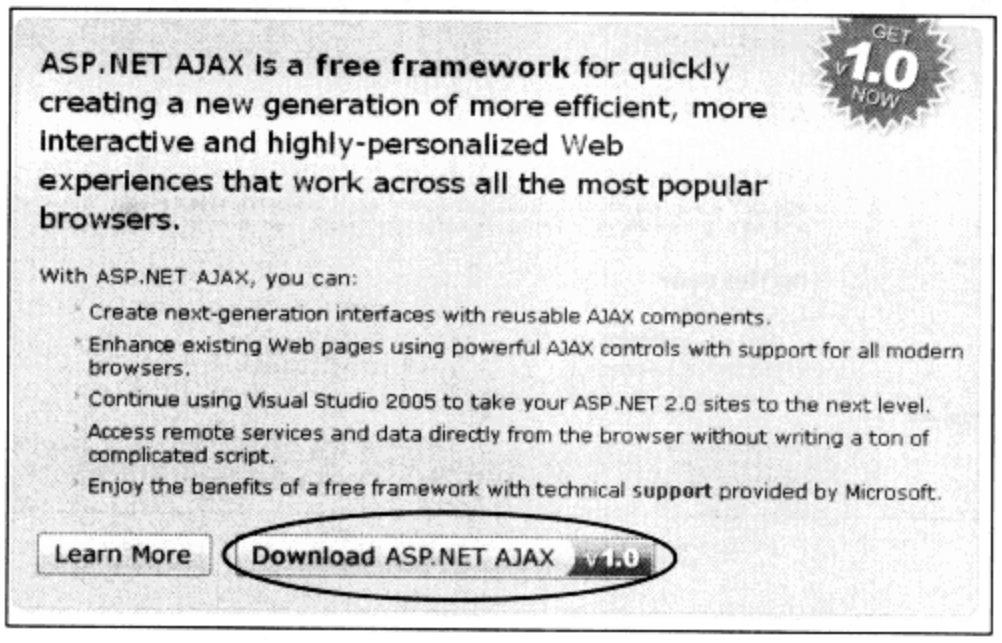


图 24.1　选择“Download ASP.NET Ajax V1.0”　超级连接

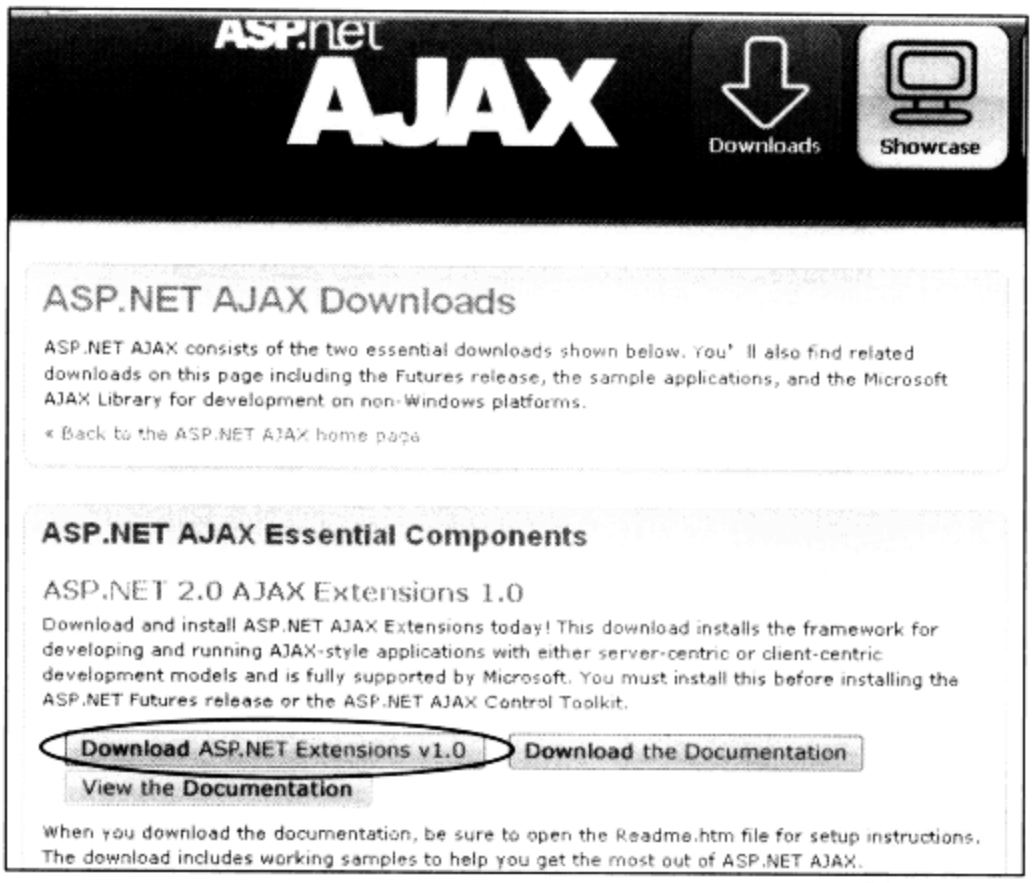


图 24.2　选择“Download ASP.NET Extensions v1.0”超级连接

（2）在图 24.2 所示的页面中，单击“Download ASP.NET Extensions v1.0”超级连接，进入微软公司网站页面，如图 24.3 所示。

（3）在图 24.3 所示的页面中，单击“Download”按钮，弹出“文件下载”对话框，如图 24.4 所示。单击“保存”按钮将 ASPAjaxExtSetup.msi 安装文件保存到相应的位置，并开始下载。

2. 安装 Ajax 框架到 ASP.NET 中

ASPAjaxExtSetup.msi 安装文件下载完成后，接下来的工作则是安装，具体步骤如下。

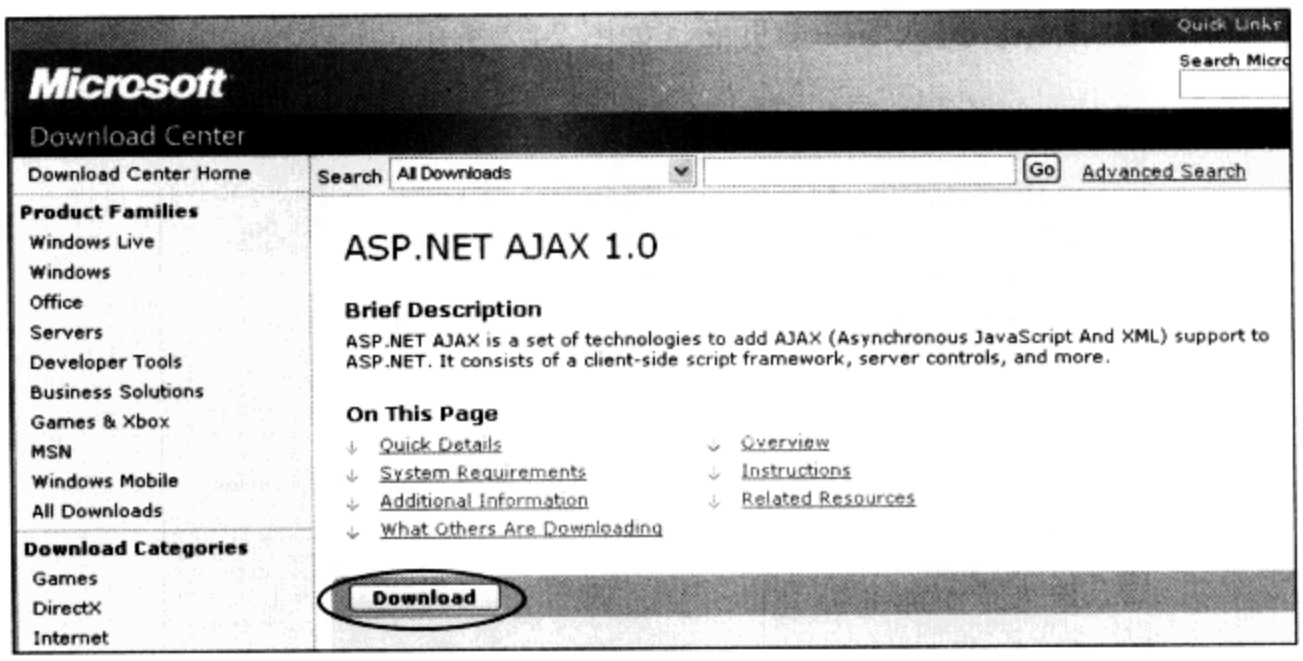


图 24.3　微软公司网站页面

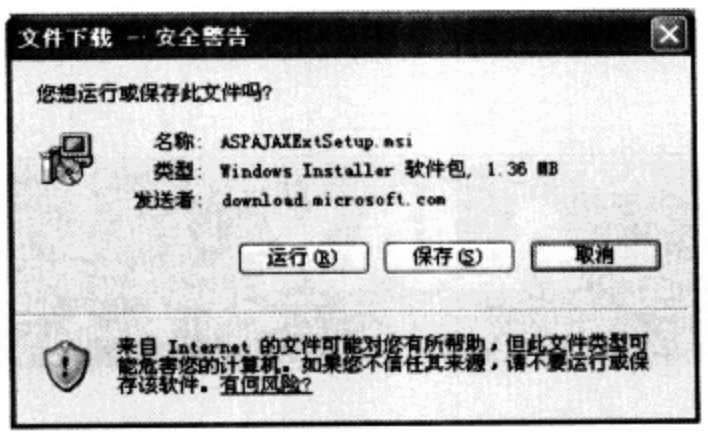


图 24.4　下载并保存到本地计算机

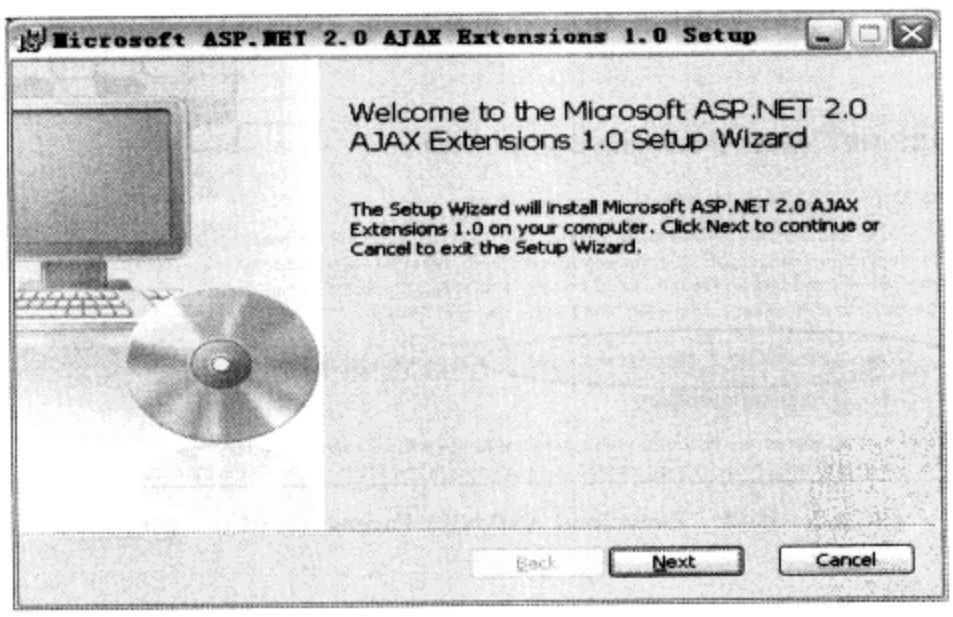


图 24.5　安装第一步界面

（1）执行 ASPAjaxExtSetup.msi 安装文件，弹出如图 24.5 所示的界面，单击“Next”按钮，进入 Atlas 安装协议界面，如图 24.6 所示。

（2）在图 24.6 所示的页面中，将“I accept the terms in the License Agreement”复选框选中，表示接受安装协议，然后单击“Next”按钮，开始安装。安装完毕后将会出现如图 24.7 所示的界面，单击“Finish”按钮完成操作。

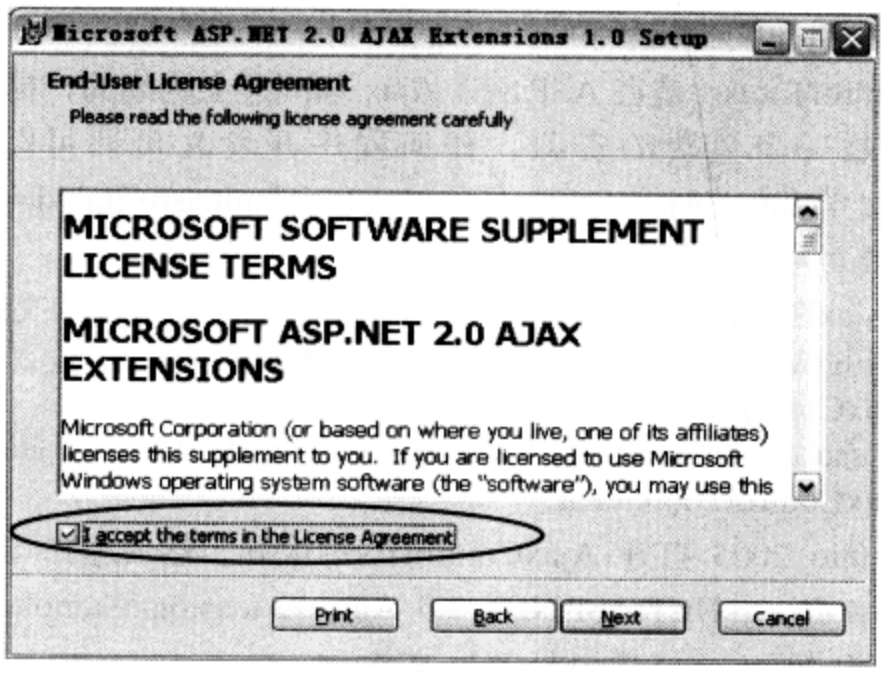


图 24.6　接受 Atlas 安装协议

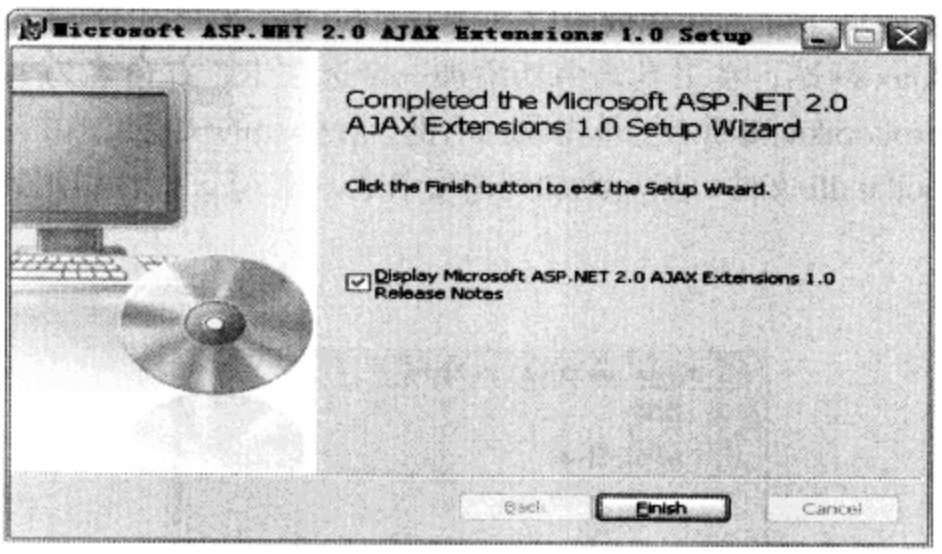


图 24.7　Atlas 安装完毕

（3）安装完毕后，启动 Visual Studio 2005 创建网站，如果模板项中存在“ASP.NET Ajax-Enabled Web Site 模板”，那么说明 Ajax 安装成功，如图 24.8 所示。

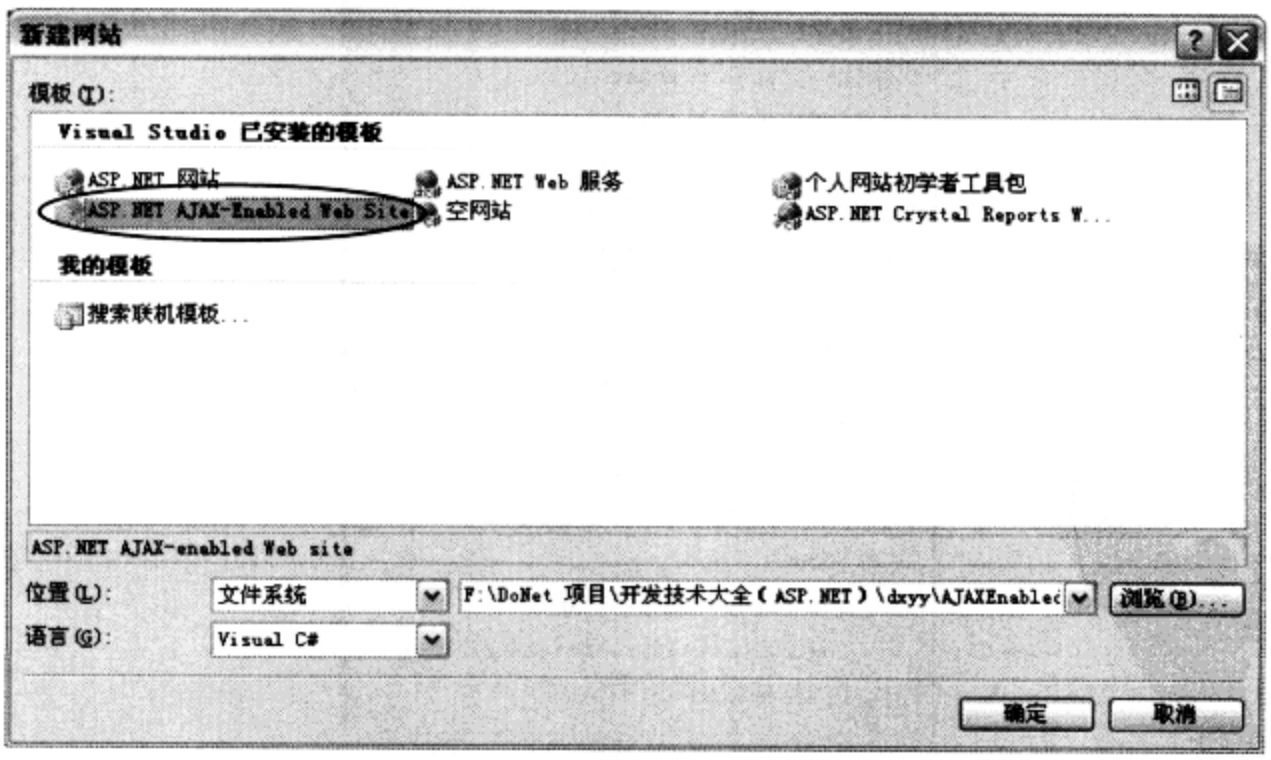


图 24.8　验证是否将 Ajax 安装到 Visual Studio 2005 开发环境中

3. 添加 AjaxControlToolkit 类控件

ASP.NET AjaxControlToolkit 是在 ASP.NET Ajax 基础之上构建的，提供了数十种 ASP.NET Ajax 控件，并且它是一个免费的资源，任何程序开发人员都可以使用该资源。下载 AjaxControlToolkit 地址为 http://Ajax.asp.net/downloads/default.aspx?tabid=47。

下载完成后，需要对环境进行如下设置。

（1）在安装 ASPAjaxExtSetup.msi 后，会在系统盘中生成如下文件夹路径：

```
C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 Ajax Extensions\v1.0.61025
```

（2）将下载的 AjaxControlToolkit 解压到如下路径位置：

```
C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 Ajax Extensions\v1.0.61025\AjaxContronlToolkit\
```

（3）双击运行 AjaxControlToolkit.sln。

（4）用 Visual Studio 2005 打开 AjaxControlToolkit.sln，然后编译 TemplateVSI 项目，在 C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 Ajax Extensions\SampleWebSite\Bin 下面生成 AjaxControlToolkit.dll 和 AjaxControlToolkit.pdb 文件。

（5）将这个两个文件复制到 C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 Ajax Extensions\Binaries 文件夹下面，这样就可以在 Visual Studio 2005 中使用 Ajax.NET 控件了。

（6）新建一个 Ajax 网站，在工具箱中新添加一个选项卡，并命名为 AjaxControltoolkit。

（7）在 AjaxControltoolkit 选项卡上单击鼠标右键，在弹出的快捷菜单中选择“选择项”，浏览并找到 AjaxControlToolkit.dll 文件，然后添加到选项卡中，故 Ajax 控件引用到 Visual Studio 2005，如图 24.9 所示。

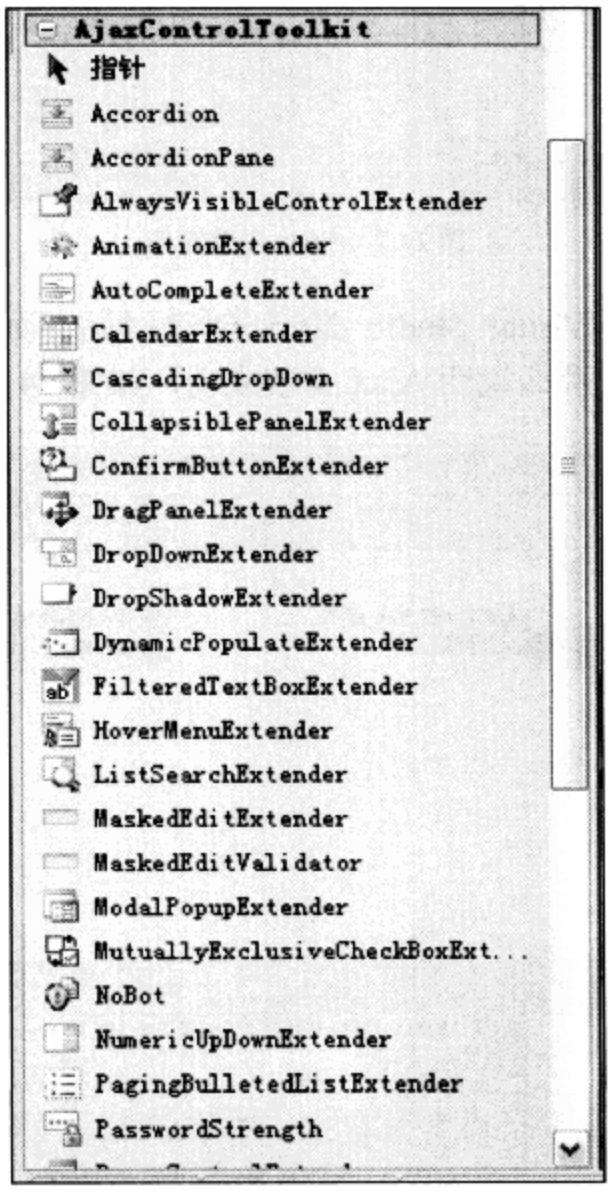


图 24.9　AjaxControltoolkit 控件

24.2.2 Ajax 框架中 SlideShowExtender 控件播放照片

1. SlideShowExtender 控件介绍

ASP.NET Ajax Control Toolkit 中的 SlideShowExtender 控件能够与 ASP.NET 中 Image 控件创建关联，并且实现出一个具备导航与自动播放功能的页面相册，SlideShowExtender 控件的属性及说明如表 24.1 所示。

表 24.1　　SlideShow 控件属性及说明

属　　性	说　　明
TargetControlID	要与“相册播放”扩展器创建起关联性，以便通过它来展示相片 Image 控件 ID
SlideShowServicePath	提供相片的 Web 服务的位置路径与文件名称。如果使用的是页面方法，则不需要
SlideShowServiceMethod	提供 Web 服务中的方法名称
AutoPlay	是否在页面加载完成之后，自动播放相片
ImageDescriptionLabelID	用来显示目前所显示之相片说明的 Label 控件 ID
NextButtonID	用来播放下一张相片的按钮控件 ID
PlayButtonText	播放按钮上的文字
StopButtonText	停止播放按钮上的文字
PreviousButtonID	用来播放上一张相片的按钮控件 ID
PlayButtonID	开始与停止播放相片按钮的控件 ID
Loop	是否允许循环播放
PlayInterval	自动播放时，每张相片的间隔时间（单位：毫秒）默认是 3 000

2. SlideShowExtender 控件播放照片

本程序实现的播放照片分为自动播放图片和上一张/下一张两种操作，如图 24.10 所示。

图 24.10　播放照片操作

实现自动播放照片主要通过 SlideShowExtender 控件的 PreviousButtonID 属性（上一张）、NextButtonID 属性（下一张）、PlayButtonID 属性（开始播放）、PlayButtonText 属性（开始播放提示信息）、StopButtonText 属性（停止播放提示信息）和 PlayInterval 属性共同来完成，属性的具体说明如表 24.1 所示。

实现关键代码如下。

例程 1　代码位置：光盘\mr\24\ Ealbum\ PlayPhoto.aspx

```
NextButtonID="btnNext"
PreviousButtonID="btnPrev"
PlayButtonID="btnPlay"
PlayInterval="5000"
layButtonText="自动播放"
StopButtonText="停止自动播放"
```

24.2.3 创建 Web 服务获取相册照片

本程序实现的架构是 Ajax+Web Service，那么在 Ajax 段主要用到了 SlideShowExtender 控件。开发人员需要设置 SlideShowExtender 控件的 SlideShowServicePath 属性和 SlideShowServiceMethod 属性，意思为指定 Web 服务的路径和服务中的方法。

为 SlideShowExtender 控件设置 Web 服务，实现关键代码如下：

```
SlideShowServicePath="PhotoService.asmx" SlideShowServiceMethod="GetPhoto"
```

通过 Web 服务中实现的 GetPhoto 方法来获取照片，并且将获取的照片返回到 SlideShowExtender 控件中，那么 GetPhoto 方法的返回值类型必须为 Slide 数组数据类型。声明

GetPhoto 方法数据类型的代码如下：

```
public Slide[] GetPhoto()
{
    ...
}
```

从上面介绍中读者可以了解到，GetPhoto 方法获取的照片需要存储到 Slide 数组中，读取数据中的相关数据通过 Foreach 循环存储到 Slide 数组中，实现代码如下。

例程 2 代码位置：光盘\mr\24\ Ealbum\ App_Code\PhotoService.cs

```
//创建数据库连接
OleDbConnection conn = new OleDbConnection(@"Provider = Microsoft.Jet.OLEDB.4.0;Data Source =" + Server. Map Path ("photo.mdb"));
//查询获取数据库中数据
OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_photo Where photoType='" + sType + "' and photo User='" + sUser + "' order by id desc", conn);
DataSet ds = new DataSet();
//获取数据库结果，填充到DataSet中
da.Fill(ds);
//创建Slide数据对象对象，用语存储将要播放的相片
AjaxControlToolkit.Slide[] AjaxPhoto = new Slide[ds.Tables[0].Rows.Count];
int j = 0;
foreach (DataRowView v in ds.Tables[0].DefaultView)
{
    //读取数据库中数据，并存储到Slide数组中
    AjaxPhoto.SetValue(new AjaxControlToolkit.Slide(v[3].ToString(), v[1].ToString(), v[2].ToString()), j);
    j = j + 1;
}
return AjaxPhoto;
```

24.2.4 DataList 控件实现分页

为了满足电子相册主页的个人相册显示形式，可使用 DataList 数据表格控件具有自定义布局显示方式，但该控件不具备分页功能，需要程序开发人员使用 PagedDataSource 类来完成分页功能，DataList 控件实现分页程序运行结果如图 24.11 所示。

图 24.11 DataList 控件实现分页

1. DataList 控件的使用

DataList Web 服务器控件通过自定义的格式显示数据库行的信息。显示数据的格式在创建

的模板中定义，可以为项、交替项、选定项和编辑项创建模板；标头、脚注和分隔符模板也用于自定义 DataList 的整体外观。

开发用到的 DataList 控件属性及说明如表 24.2 所示。

表 24.2　　DataList 控件相关属性及说明

属　性	说　明
DataKeyField	获取或设置由 DataSource 属性指定的数据源中的键字段
DataKeys	获取 DataKeyCollection 对象，该对象存储数据列表控件中每个记录的键值
DataSource	获取或设置源，该源包含用于填充控件中的项的值列表
EditItemIndex	获取或设置 DataList 控件中要编辑的选定项的索引号
Items	获取表示控件内单独项的 DataListItem 对象的集合
ItemTemplate	获取或设置 DataList 控件中项的模板
RepeatColumns	获取或设置要在 DataList 控件中显示的列数
RepeatDirection	获取或设置 DataList 控件是垂直显示还是水平显示
SelectedIndex	获取或设置 DataList 控件中的选定项的索引
SelectedItem	获取 DataList 控件中的选定项
SelectedItemTemplate	获取或设置 DataList 控件中选定项的模板
SelectedValue	获取所选择的数据列表项的键字段的值

2. PagedDataSource 类的使用

PagedDataSource 类封装那些允许数据表格控件（如 DataList 控件）执行分页操作的属性。如果控件开发人员需对自定义数据绑定控件提供分页支持，即可使用此类。

开发用到的 PagedDataSource 类的属性及说明如表 24.3 所示。

表 24.3　　PagedDataSource 类的相关属性及说明

名　称	说　明
AllowCustomPaging	获取或设置一个值，指示是否在数据绑定控件中启用自定义分页
AllowPaging	获取或设置一个值，指示是否在数据绑定控件中启用分页
AllowServerPaging	获取或设置一个值，指示是否启用服务器端分页
Count	获取要从数据源使用的项数
CurrentPageIndex	获取或设置当前页的索引
DataSource	获取或设置数据源
FirstIndexInPage	获取页面中显示的首条记录的索引
IsCustomPagingEnabled	获取一个值，该值指示是否启用自定义分页
IsFirstPage	获取一个值，该值指示当前页是否是首页
IsLastPage	获取一个值，该值指示当前页是否是最后一页
IsPagingEnabled	获取一个值，该值指示是否启用分页
IsSynchronized	获取一个值，该值指示是否同步对数据源的访问（线程安全）
PageCount	获取显示数据源中的所有项所需要的总页数
PageSize	获取或设置要在单页上显示的项数

3. DataList 控件的分页实现

根据以上两小节读者已经对 DataList 控件和 PagedDataSource 类有一定的认识，接下来给出 DataList 控件实现分页功能的关键代码，代码如下：

```
PagedDataSource pds = new PagedDataSource();
pds.DataSource = dt.DefaultView;                    //将查询结果绑定到分页数据源上
pds.AllowPaging = true;                             //允许分页
pds.PageSize = 12;                                  //设置每页显示多少张图片
pds.CurrentPageIndex = Convert.ToInt32(lblCurrentPage.Text) - 1;      //设置当前页
lnkBtnFirst.Enabled = true;                         //控件翻页控件都设置为可用
lnkBtnLast.Enabled = true;
lnkBtnNext.Enabled = true;
lnkBtnPrevious.Enabled = true;
if (lblCurrentPage.Text == "1")                     //如果当前显示第一页，“第一页”和“上一页”按钮不可用
{
    lnkBtnPrevious.Enabled = false;
    lnkBtnFirst.Enabled = false;
}
//如果显示最后一页，“末一页”和“下一页”按钮不可用
if (lblCurrentPage.Text == pds.PageCount.ToString())
{
    lnkBtnNext.Enabled = false;
    lnkBtnLast.Enabled = false;
}
lblSumPage.Text = pds.PageCount.ToString();   //实现总页数
//将分页结果绑定到DataList控件上
dlPictrue.DataSource = pds;                         //绑定数据源
dlPictrue.DataKeyField = "photoUser";
dlPictrue.DataBind();
```

上面只给出分页功能的关键设置。关于 DataList 控件的翻页设置，请参见 24.3.2 实现过程，那里将详细介绍。

24.2.5 DataList 控件事件冒泡浏览个人相册

当网站浏览者进入电子相册网站时，网站主页显示所有用户的个人电子相册，当网站浏览者要进入个人电子相册时，需要单击“点击浏览”按钮进入个人电子相册。刚才所描述的显示个人电子相册和进入电子相册都是通过 DataList 控件实现的，显示个人电子相册比较容易实现，通过单击按钮进入个人电子相册，主要通过 DataList 控件的事件冒泡来实现，如图 24.12 所示。

图 24.12 使用 DataList 事件冒泡

1. 理解事件冒泡

在 ASP.NET 框架中包含 3 个支持事件冒泡的 Web 服务器控件（如 Repeater、DataList 和

DataGrid 控件），这些控件可以捕获其子控件的事件。当子控件产生一个事件时，事件就“冒泡”传给包含该子控件的容器控件，并且容器控件就可以执行一个子程序来处理该事件。

2. 捕获 DataList 控件中产生的事件

DataList 控件支持事件冒泡，可以捕获 DataList 内包含的控件产生的事件，并且通过普通的子程序处理这些事件。没有经验的程序人员可能不理解事件冒泡的好处。如果没有事件冒泡，那么对于 DataList 内包含的每一个控件产生的事件都需要定义一个相应的处理函数，如果 DataList 中包含成百上千个控件，那么程序员得写多少个事件处理程序呢？所以事件冒泡，不管 DataList 中包含多少个控件，程序人员只需要一个处理程序就足够了。DataList 控件支持 5 个冒泡事件。

- EditCommand 事件。由带有 CommandName = "edit"的子控件产生。
- CancelCommand 事件。由带有 CommandName = "cancel"的子控件产生。
- UpdateCommand 事件。由带有 CommandName = "update"的子控件产生。
- DeleteCommand 事件。由带有 CommandName = "delete"的子控件产生。
- ItemCommand 事件。DataList 的默认事件。

在 ASP.NET 中有 3 个控件带有 CommandName 属性，分别是 Button、LinkButton 和 ImageButton，均可以设置 CommandName 属性来表示容器控件内产生的事件类型。例如，如果设置 DataList 中的一个 ImageButton 控件的 CommandName 属性为“Delete”，那么在程序运行的时候，单击该按钮将会触发 DataList 的 DeleteCommand 事件，可以将相关功能代码写到 DeleteCommand 事件中即可，如图 24.13 所示。

本程序就是通过 ImageButton 控件的 CommandName 属性和 DataList 控件的 DeleteCommand 事件来实现。

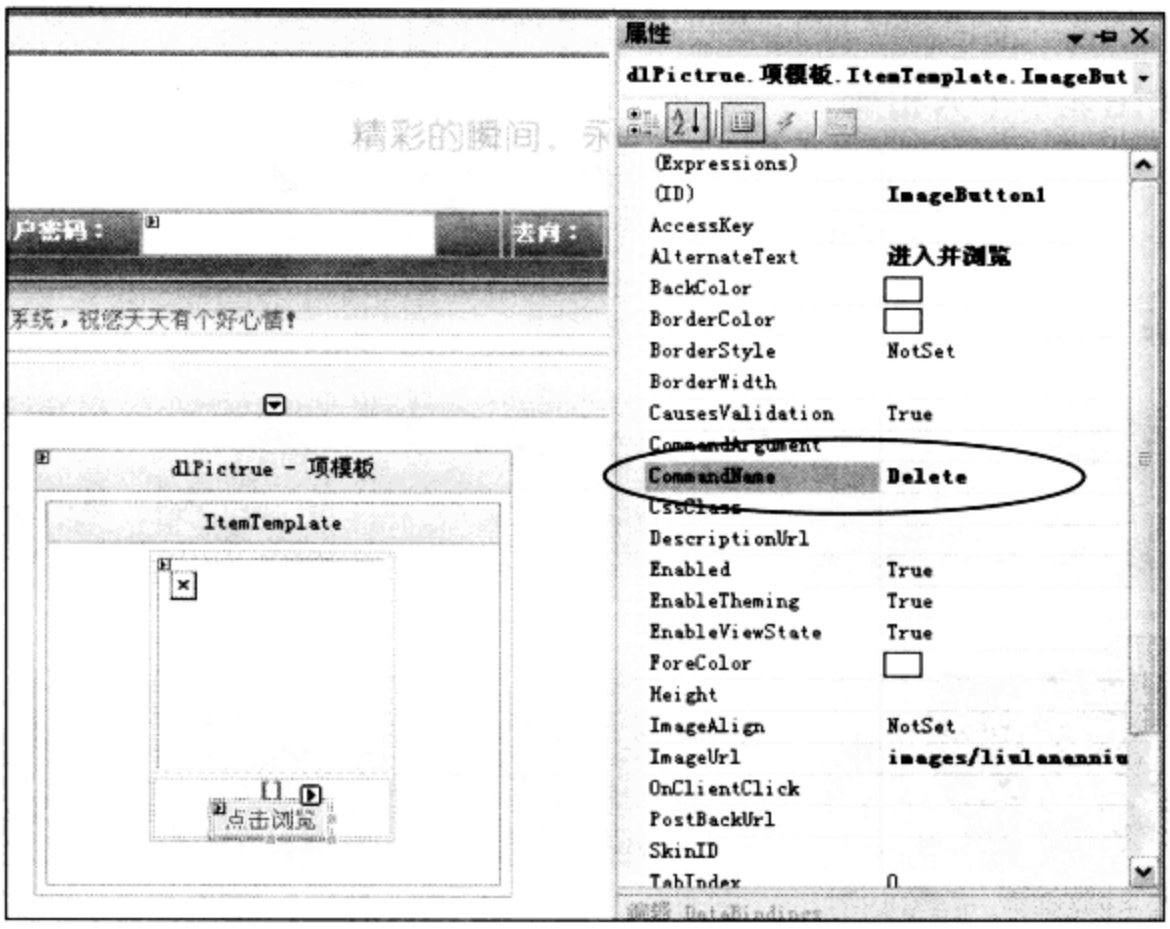


图 24.13 设置 ImageButton 控件的事件冒泡

24.3 电子相册主页设计

电子相册主页分为 3 部分：缩略图显示个人相册、分页显示相册缩略图和电子相册用户登

录，如图 24.14 所示。下面分别对这 3 部分进行详细讲解。

图 24.14　电子相册主页

24.3.1　缩略图显示个人相册

当开发电子相册类网站时，用户的电子相册不仅仅是面向自己，而且还面向广大网络用户。在设计主页时，可以通过缩略图的形式显示个人的电子相册，如图 24.15 所示。

图 24.15　缩略图显示个人相册

下面开始设计缩略图显示个人相册的页面，缩略图的显示主要通过 DataList 控件编辑模板实现的，如图 24.16 所示，在 DataList 控件编辑模板中添加控件如下。

- 添加 1 个 Table 表格，主要用来布局编辑模板。

- 添加 1 个 Image 服务器控件，显示个人相册缩略图。
- 添加 1 个 ImageButton 服务器控件，设置 CommandName 属性值为 Delete，当单击 ImageButton 控件时，则触发 DataList 控件的 DeleteCommand 事件，在这里主要通过此方法实现进入并浏览个人的电子相册。

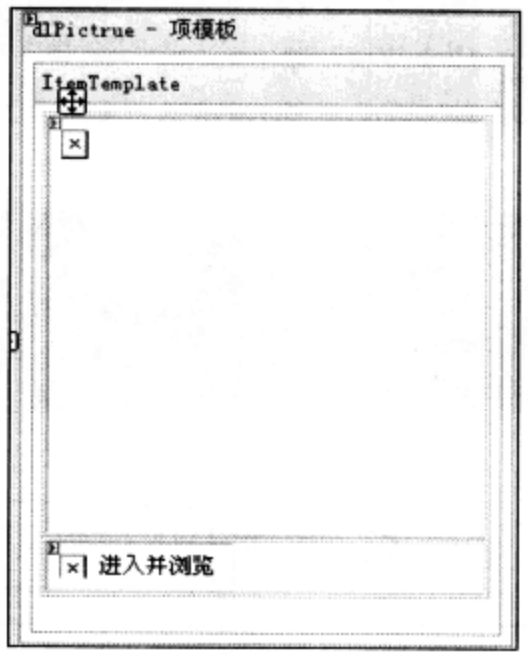


图 24.16　实现缩略图的设计结果

在 DataList 控件编辑模板中使用 DataBinder.Eval 方法绑定照片路径数据和个人相册的用户名称。

例程 3　代码位置：光盘\mr\24\ Ealbum\ Logon.aspx

```
    <asp:DataList ID="dlPictrue" runat="server" RepeatColumns="4" OnDeleteCommand="dlPictrue_DeleteCommand"
RepeatDirection="Horizontal">
            <ItemTemplate>
                <table style="width: 116px">
                    <tr>
                        <td>
    <asp:Image ID="Image1" runat="server" Width="227px" ImageUrl='<%# DataBinder.Eval(Container.DataItem,"photoPath")
%>' Height="228px" /></td>
                    </tr>
                    <tr>
                        <td>
        <%# DataBinder.Eval(Container.DataItem,"photoUser") %>
    <asp:ImageButton ID="ImageButton1" runat="server" AlternateText="进入并浏览" CommandName="Delete"
                                                    Height="27px" Width="98px" /></td>
                    </tr>
                </table>
            </ItemTemplate>
    </asp:DataList>
```

当单击 ImageButton 按钮时，触发 DataList 控件的 DeleteCommand 事件，用于进入个人的电子相册。

例程 4　代码位置：光盘\mr\24\ Ealbum\ Logon.aspx.cs

```
    protected void dlPictrue_DeleteCommand(object source, DataListCommandEventArgs e)
    {
        Session["User"] = dlPictrue.DataKeys[e.Item.ItemIndex].ToString();
        Response.Redirect("Default.aspx");
    }
```

24.3.2　分页显示相册缩略图

当通过缩略图显示个人相册时，如果相册的用户非常多时，用一个页面显示所有的数据，

既不符合程序开发规则，又不符合程序的人性化，这时需要分页显示数据，如图 24.17 所示。

图 24.17　分页显示设置

实现本程序所用的控件如表 24.4 所示。

表 24.4　　控件属性设置及说明

控件名/ID	属 性 设 置	用　　途
Label/lblCurrentPage	均默认	显示当前页
Label/lblSumPage	均默认	显示总页数
LinkButton/lnkBtnFirst	Text 属性为“第一页”	将页数定位第一页
LinkButton/ lnkBtnPrevious	Text 属性为“上一页”	上一页翻页操作
LinkButton/ lnkBtnNext	Text 属性为“下一页”	下一页翻页操作
LinkButton/ lnkBtnLast	Text 属性为“末一页”	将页数定位到最后一页

自定义 DataListBind 方法用于将数据绑定 DataList 控件并显示缩略图相片。

例程 5　代码位置：光盘\mr\24\ Ealbum\ Logon.aspx.cs

```
        /// <summary>
        /// 将数据绑定到DataList控件
        /// </summary>
        public void DataListBind()
        {
            DataTable dt = new DataTable();
            dt.Columns.Add(new DataColumn("id", typeof(int)));
            dt.Columns.Add(new DataColumn("photoUser", typeof(string)));
            dt.Columns.Add(new DataColumn("photoPath", typeof(string)));
            dt.PrimaryKey = new DataColumn[] { dt.Columns["photoUser"] };
            using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + Server.
MapPath("photo.mdb")))
            {
                //查询数据库中数据
                OleDbDataAdapter da = new OleDbDataAdapter("select userName from tb_User", conn);
                DataSet ds = new DataSet();
                da.Fill(ds, "Table");
                for (int i = 0; i < ds.Tables[0].Rows.Count; i++)
                {
```

```
                    OleDbDataAdapter dap = new OleDbDataAdapter("select id,photoUser, photoPath from tb_Photo where
photoUser='" + ds.Tables[0].Rows[i][0].ToString() + "'", conn);
                    DataSet dss = new DataSet();
                    dap.Fill(dss);
                    if (dss.Tables[0].Rows.Count > 0)
                    {
                        DataRow dr = dt.NewRow();
                        dr[0] = dss.Tables[0].Rows[0][0].ToString();
                        dr[1] = dss.Tables[0].Rows[0][1].ToString();
                        dr[2] = dss.Tables[0].Rows[0][2].ToString();
                        dt.Rows.Add(dr);
                    }
                }
            }
            PagedDataSource pds = new PagedDataSource();
            pds.DataSource = dt.DefaultView;                    //将查询结果绑定到分页数据源上
            pds.AllowPaging = true;                             //允许分页
            pds.PageSize = 8;                                   //设置每页显示多少张图片
            pds.CurrentPageIndex = Convert.ToInt32(lblCurrentPage.Text) - 1;      //设置当前页
            lnkBtnFirst.Enabled = true;                         //控件翻页控件都设置为可用
            lnkBtnLast.Enabled = true;
            lnkBtnNext.Enabled = true;
            lnkBtnPrevious.Enabled = true;
            if (lblCurrentPage.Text == "1")  //如果当前显示第一页，“第一页”和“上一页”按钮不可用
            {
                lnkBtnPrevious.Enabled = false;
                lnkBtnFirst.Enabled = false;
            }
            //如果显示最后一页，“末一页”和“下一页”按钮不可用
            if (lblCurrentPage.Text == pds.PageCount.ToString())
            {
                lnkBtnNext.Enabled = false;
                lnkBtnLast.Enabled = false;
            }
            lblSumPage.Text = pds.PageCount.ToString();             //实现总页数
            dlPictrue.DataSource = pds;                         //绑定数据源
            dlPictrue.DataKeyField = "photoUser";
            dlPictrue.DataBind();
        }
```

单击“第一页”链接按钮，将照片缩略图定位到第一页。

例程 6 代码位置：光盘\mr\24\ Ealbum\ Logon.aspx.cs

```
    protected void lnkBtnFirst_Click(object sender, EventArgs e)
    {
        lblCurrentPage.Text = "1";
        DataListBind();
    }
```

单击“上一页”链接按钮，将照片缩略图定位到上一页。

例程 7 代码位置：光盘\mr\24\ Ealbum\ Logon.aspx.cs

```
    protected void lnkBtnPrevious_Click(object sender, EventArgs e)
    {
        lblCurrentPage.Text = (Convert.ToInt32(lblCurrentPage.Text) - 1).ToString();
        DataListBind();
    }
```

单击“下一页”链接按钮，将照片缩略图定位到下一页。

例程 8 代码位置：光盘\mr\24\ Ealbum\ Logon.aspx.cs

```
    protected void lnkBtnNext_Click(object sender, EventArgs e)
    {
        lblCurrentPage.Text = (Convert.ToInt32(lblCurrentPage.Text) + 1).ToString();
        DataListBind();
    }
```

单击“末一页”链接按钮，将照片缩略图定位到最后一页。

例程 9　代码位置：光盘\mr\24\ Ealbum\ Logon.aspx.cs

```
protected void lnkBtnLast_Click(object sender, EventArgs e)
{
    lblCurrentPage.Text = lblSumPage.Text;
    DataListBind();
}
```

24.3.3　电子相册用户登录

在电子相册中除了缩略图显示个人相册和分页显示缩略图外，还包括电子相册用户登录功能，用户登录成功后，可以进入自己的电子相册空间，也可以进入管理页面进行管理自己的照片。电子相册用户登录功能如图 24.18 所示。

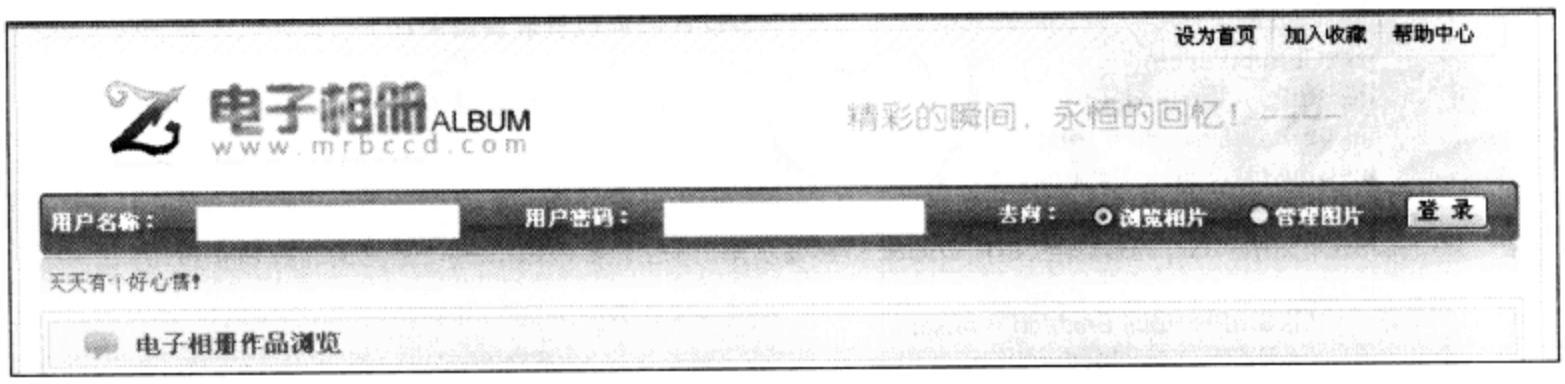


图 24.18　电子相册用户登录

实现电子相册登录功能主要用到控件有：2 个 TextBox 控件，用于用户输入用户名称和用户密码，其中密码文本框需要设置 TextMode 属性值为 Password；2 个 RadioButton 控件，分别用来选择进入浏览页面还是管理页面，其中 2 个 RadioButton 控件的 Text 属性分别为“浏览照片”和“管理照片”，设置 2 个 RadioButton 控件的 GroupName 属性为“direction”字符串，意思为分为同一“direction”组；1 个 Button 控件，用来执行用户登录功能。

电子相册登录功能主要代码都集中在“登录”按钮的 Click（单击）事件下，实现电子相册用户登录功能代码如下。

例程 10　代码位置：光盘\mr\24\ Ealbum\ Logon.aspx.cs

```
protected void btnLogon_Click(object sender, EventArgs e)
{
    using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + Server.
MapPath("photo.mdb")))
    {
        //查询数据库中数据
        OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_User where userName='" + txtUserName. Text
+ "'and userPwd='" + txtPwd.Text + "'", conn);
        DataSet ds = new DataSet();
        da.Fill(ds, "Table");
        //如果查询结果记录数大于0 则用户名称和密码正确。否则，用户名称或密码不正确
        if (ds.Tables[0].Rows.Count > 0)
        {
            Session["User"] = txtUserName.Text;
            //判断选中的单选按钮
            if (rdoBtnIndex.Checked)
                Response.Redirect("Default.aspx");                    //浏览相片
            else
                Response.Redirect(@"BackGround\Default.aspx");   //管理图片
        }
        else
```

```
                WebMessageBox.Show("用户名称或密码不正确！");
            }
        }
```

24.4 浏览电子相册页设计

浏览电子相册中照片是本模块最核心的功能之一，在如图 24.19 所示界面中浏览者可以选择左侧的树状菜单导航栏，然后在页面的右侧显示照片，并且显示每张照片的标题和说明。另外，照片可以自动播放，也可以上一张或下一张有选择性地播放照片。

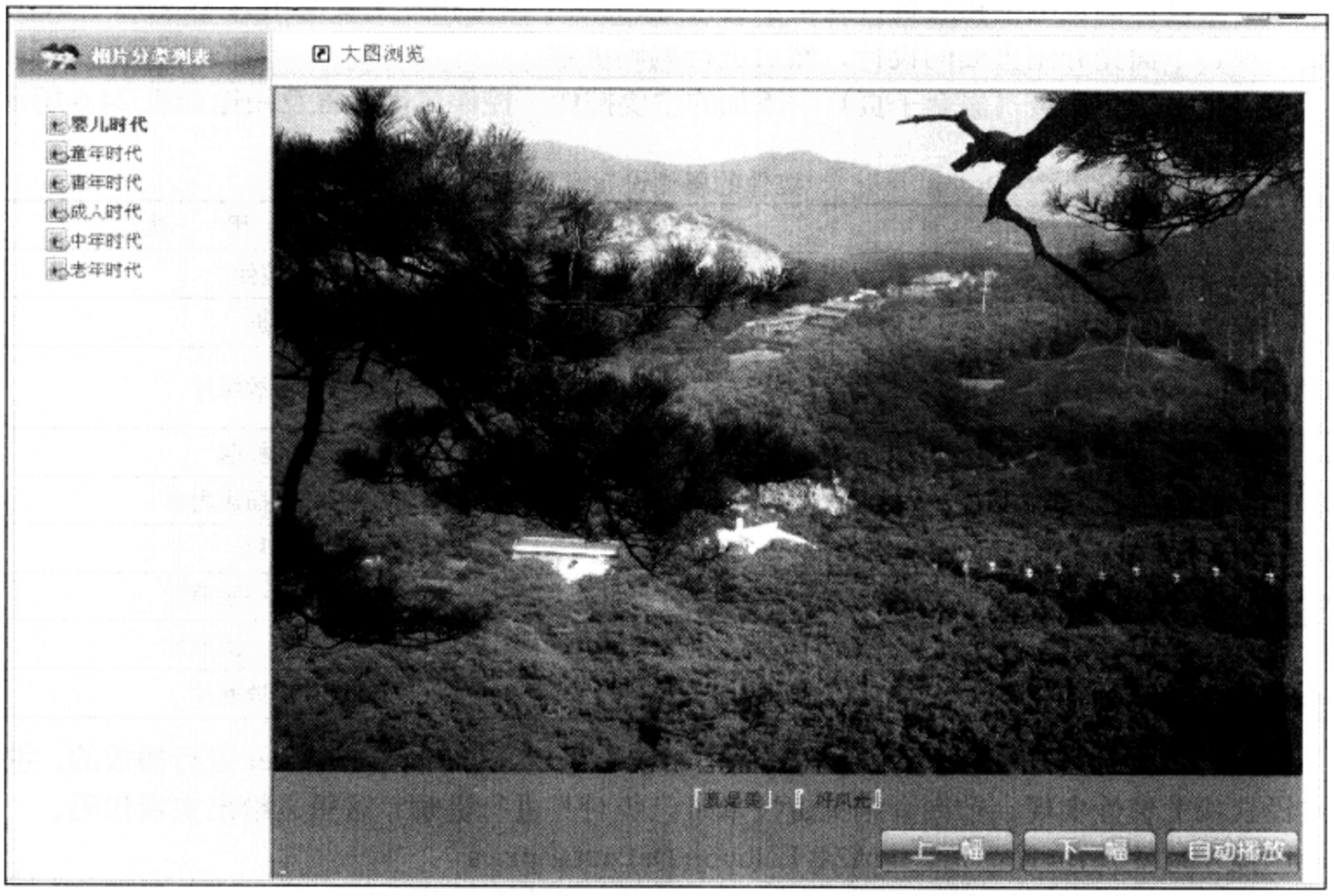


图 24.19 浏览电子相册中照片

下面开始设计浏览电子相册页，该页面由主页和嵌套子页完成。其中主页主要实现导航功能，嵌套子页面主要用来浏览相册中的相片，并且是嵌套在 Iframe 框架中显示。下面分别对这两个页介绍。在主页中添加主要控件、控件属性设置及用途如表 24.5 所示。

表 24.5 主要控件属性设置及说明

控 件 名 称	属 性 设 置	用 途
Html / IFrame	设置 Iframe 为服务器控件，ID 属性为 mainFrame	用于链接照片播放页
标准控件 / TreeView	设置 Nodes 节点属性值为“婴儿时代”、“童年时代”、“青年时代”、“成人时代”、“中年时代”、“老年时代”	用于导航浏览照片

控件添加完毕后，开始编写代码。这里主要代码用于实现导航链接，实现代码如下。

例程 11 代码位置：光盘\mr\24\ Ealbum\ Default.aspx.cs

```
        protected void TreeView1_SelectedNodeChanged(object sender, EventArgs e)
        {
            using (OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Server.MapPath("photo.mdb")))
```

```
            {
                //初始化数据添加操作
                OleDbCommand cmd = new OleDbCommand("update tb_show set showUser='" + Session["User"].ToString()
+ "', showType='" + TreeView1.SelectedValue + "' where id=0", conn);
                //打开数据库连接
                conn.Open();
                int i = cmd.ExecuteNonQuery();
                conn.Close();
            }
            //定向链接
            mainFrame.Attributes.Add("src", "PlayPhoto.aspx");
        }
```

接下来设计浏览电子相册页（嵌套子页），该页面中浏览相片主要使用 Ajax + Web Service 框架实现。下面先介绍页面的设计，然后进行编码讲解。

在浏览电子相册页（嵌套子页）中添加的主要控件、控件属性设置及用途如表 24.6 所示。

表 24.6　　控件的属性设置及用途

控 件 类 型	属 性 设 置	用　　途
ScriptManager(Ajax 控件)	均默认	管理 Ajax 控件
UpdatePanel(Ajax 控件)	均默认	局部更新数据
SlideShowExtender(Ajax 控件)	设置 TargetControlID 值为 Image1(播放照片的 Image 控件 ID)	控制循环播放照片
Label（服务器控件）	ID 值为 lblTitle	显示照片的标题
Label（服务器控件）	ID 值为 lblShow	显示照片的描述内容
Image（服务器控件）	均默认	显示播放照片
ImageButton（服务器控件）	ID 值为 btnPrev	控制显示上一张照片
ImageButton（服务器控件）	ID 值为 btnNext	控制显示下一张照片
ImageButton（服务器控件）	ID 值为 btnPlay	控制循环播放照片

浏览电子相册页（嵌套子页）中实现播放的核心是用 SlideShowExtender 进行播放的。主要通过设置相关属性实现，前面介绍关键技术时已经对其进行讲解，这里只给出实现代码。

例程 12　代码位置：光盘\mr\24\ Ealbum\ Default.aspx.cs

```
        <cc1:SlideShowExtender ID="SlideShowExtender1" runat="server" TargetControlID="Image1"
            AutoPlay="false" ImageDescriptionLabelID="lblShow" Loop="true" NextButtonID="btnNext"
            PreviousButtonID="btnPrev" PlayButtonID="btnPlay" PlayInterval="5000" PlayButtonText="自动播放"
            StopButtonText="停止自动播放" SlideShowServicePath="PhotoService.asmx" SlideShowServiceMethod="GetPhoto"
ImageTitleLabelID="lblTitle" Enabled="True">
        </cc1:SlideShowExtender>
```

在上述代码中设置 SlideShowServicePath 属性和 SlideShowServiceMethod 属性时，需要设计 Web Service，Web Service 主要用来获取循环播放的照片。

例程 13　代码位置：光盘\mr\24\ Ealbum\ Default.aspx.cs

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;
//引入新的命名空间
using AjaxControlToolkit;
using System.Data;
using System.Data.OleDb;
/// <summary>
/// PhotoService 的摘要说明
/// </summary>
```

```
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
    //添加服务脚本
    [System.Web.Script.Services.ScriptService()]
    public class PhotoService : System.Web.Services.WebService
    {
        public PhotoService()
        {
            //如果使用设计的组件，请取消注释以下行
        }
        [WebMethod]
        public Slide[] GetPhoto()
        {
            string sType = "";          //相册类型
            string sUser = "";          //用户
            //创建数据库连接
            OleDbConnection con = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + Server.
MapPath("photo.mdb"));
            //查询获取数据库中数据
            OleDbDataAdapter dap = new OleDbDataAdapter("Select * from tb_Show Where id=0", con);
            DataSet dss = new DataSet();
            dap.Fill(dss);
            if (dss.Tables[0].Rows.Count > 0)
            {
                sType = dss.Tables[0].Rows[0][2].ToString();
                sUser = dss.Tables[0].Rows[0][1].ToString();
            }
            //创建数据库连接
            OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + Server.
MapPath("photo.mdb"));
            //查询获取数据库中数据
            OleDbDataAdapter da = new OleDbDataAdapter("select * from tb_photo Where photoType='" + sType + "' and photoUser =
'" + sUser + "' order by id desc", conn);
            DataSet ds = new DataSet();
            //获取数据库结果，填充到DataSet中
            da.Fill(ds);
            //创建Slide数据对象对象，用语存储将要播放的相片
            AjaxControlToolkit.Slide[] AjaxPhoto = new Slide[ds.Tables[0].Rows.Count];
            int j = 0;
            foreach (DataRowView v in ds.Tables[0].DefaultView)
            {
                //读取数据库中数据，并存储到Slide数组中
                AjaxPhoto.SetValue(new AjaxControlToolkit.Slide(v[3].ToString(), v[1].ToString(), v[2].ToString()), j);
                j = j + 1;
            }
            return AjaxPhoto;
        }
    }
```

24.5 常见开发技术问题总结

开发电子相册模块程序中，笔者遇到很多开发常见问题，笔者都一一记录下来，希望对读者在程序开发过程当中具有一定的参考价值，开发常见技术问题如下。

- Panel 横向滚动，纵向自动扩展

```
<asp:panel   style="overflow-x:scroll;overflow-y:auto;"></asp:panel>
```

- 回车转换成 Tab（切换光标焦点）

```
//定义实现方法
<script language="javascript" for="document" event="onkeydown">
if(event.keyCode==13 && event.srcElement.type!='button' &&
event.srcElement.type!='submit' && event.srcElement.type!='reset'
```

```
&& event.srcElement.type!="&& event.srcElement.type!='textarea');
event.keyCode=9;
</script>
//调用
onkeydown="if(event.keyCode==13) event.keyCode=9"
```

- GridView 行随鼠标变色（高量显示行）

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        //高亮显示指定行
        e.Row.Attributes.Add("onMouseOver", "Color = this.style.backgroundColor;this.style.backgroundColor = '#FFF000'");
        e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=Color;");
        e.Row.Cells[0].Text = e.Row.Cells[0].Text.Substring(2, 11);
    }
}
```

- 读取 GridView 控件 TextBox 控件中值

```
foreach(GridView dgi in yourGridViewObject.Items)
{
    TextBox txt = (TextBox)dgi.FindControl("yourTextBoxId");
}
```

- DataList 选定比较底下的行时，页面会刷新一下，然后页面滚动到最上面，刚才选定的行因为刷屏的原因就看不到了，下面给出解决代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    Page.SmartNavigation = true;
}
```

- 在 GridView 控件中修改数据，当单击编辑按钮时，行数据将显示在文本框中，如何控制文本框的大小？

```
protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
{
    for (int i = 0; i < e.Item.Cells.Count - 1; i++)
        if (e.Item.ItemType == ListItemType.EditType)
        {
            e.Item.Cells[i].Attributes.Add("Width", "80px");
        }
}
```

- 自定义网页对话框

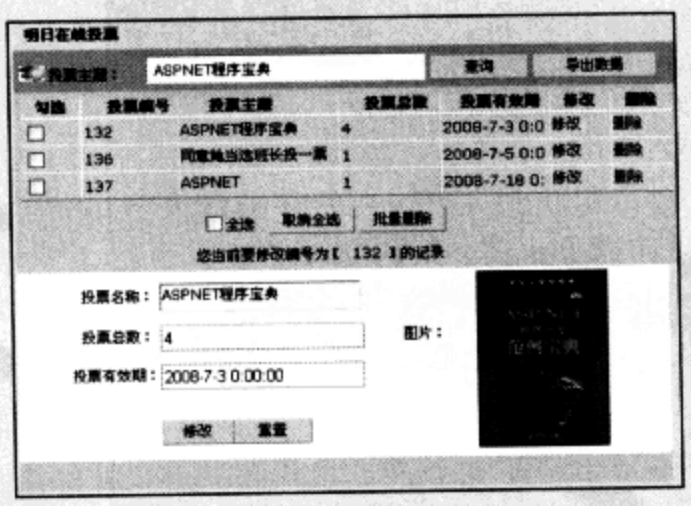
```
//开始标记
private static string ScriptBegin = "<script language=\"JavaScript\">";
//结束标记
private static string ScriptEnd = "</script>";
//自定义方法
public static void MessageBox(string pageTarget, string Content)
{
    string ConfirmContent = "var retValue =window.confirm('" + Content + "');" + "if(retValue){window.location='" + pageTarget + "';}";
    ConfirmContent = ScriptBegin + ConfirmContent + ScriptEnd;
    Page ParameterPage = (Page)System.Web.HttpContext.Current.Handler;
    ParameterPage.RegisterStartupScript("confirm", ConfirmContent);
}
```

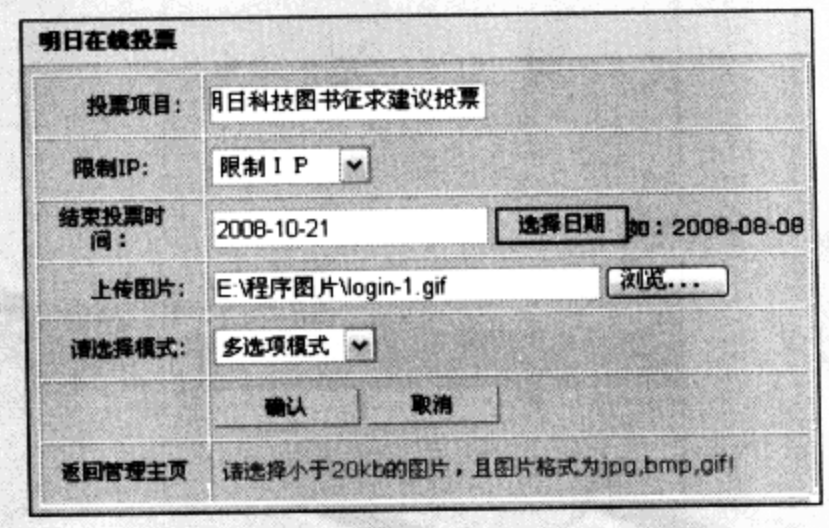
第25章 投票系统模块

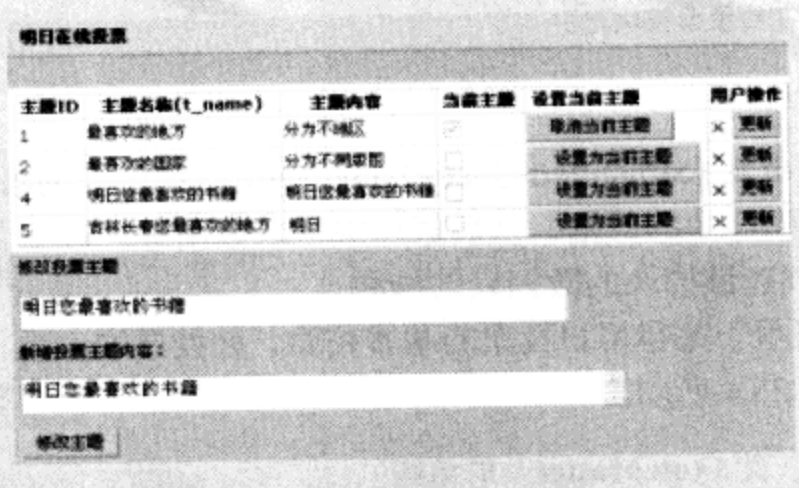
实例位置：光盘\mr\25\

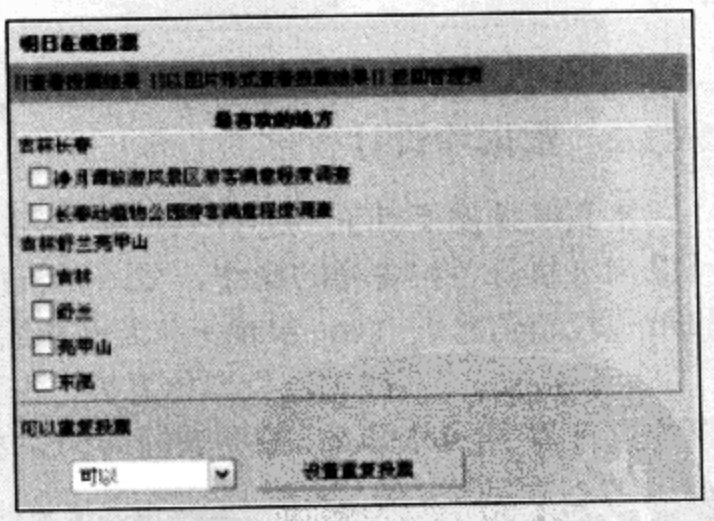
在线投票是 Web 应用中使用非常频繁的功能之一。在一些具有调查性质的网站中，经常会遇见到各种各样的在线投票，例如一些购物网站会根据用户的投票评选用户最喜欢的商品、最信任的商家等。本章将通过实例介绍开发在线投票系统的详细过程，并且读者能够学到：

- 单选投票管理
- IP 限制投票
- 多选投票主题管理
- 多选投票属性设置









25.1 在线投票模块功能概述

本模块主要包括在线单选投票和在线多选投票，实现更为人性化的操作。

25.1.1 功能简介

本章主要介绍一个在线投票的具体实现过程，包括在线单选投票及在线多选投票，在后台管理中，单选投票管理主要包括单选投票主题的查询、修改、删除（可实现批量删除）、数据导出等功能，单选投票模式又分为“一投一”和“多投一”两种模式；多选投票管理包括多选投票主题管理、投票项内容管理和属性设置（是否可重复投票）。

模块首页运行结果如图 25.1 所示。

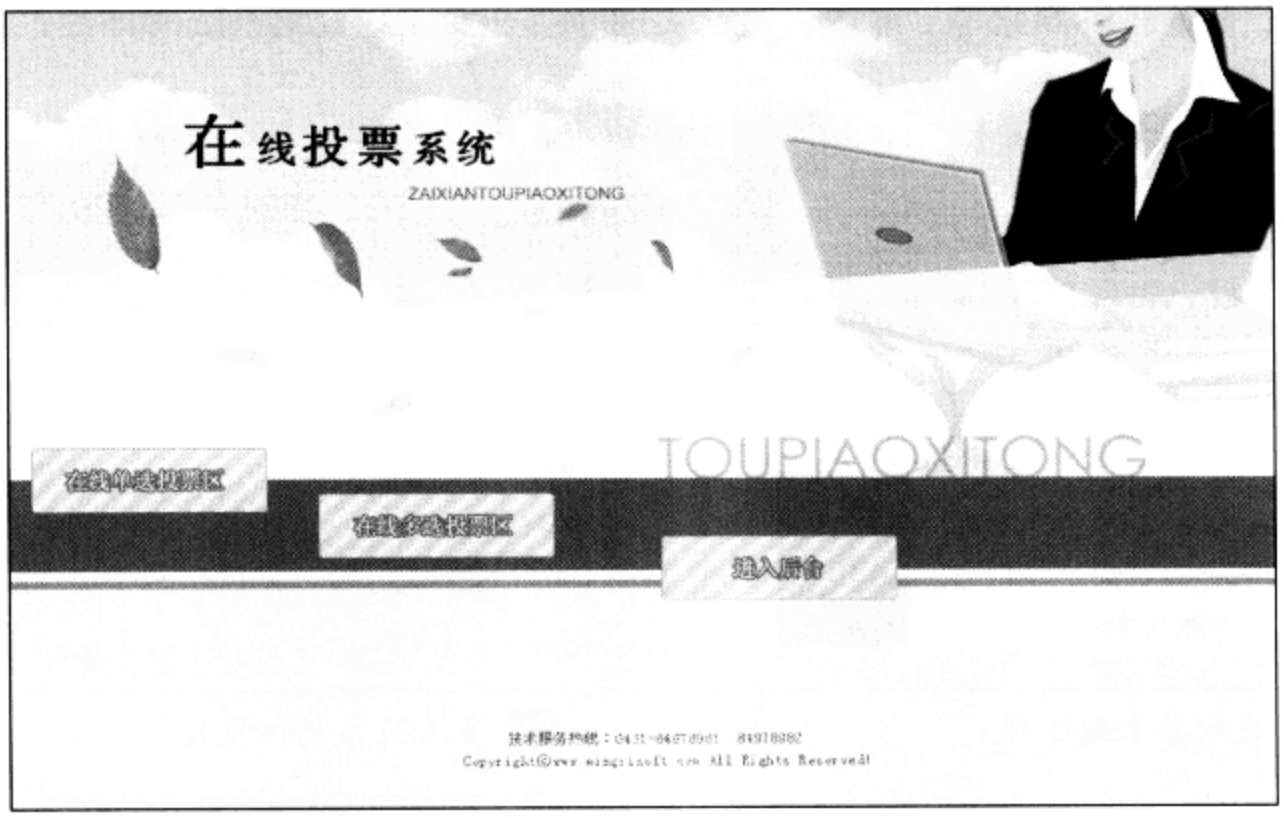


图 25.1 在线投票模块首页

25.1.2 数据库设计

本系统采用了 SQL Server 2000 数据库，SQL（Structured Query Language，结构化查询语言）可以用来执行各种各样的操作，可以更新数据库中的数据、从数据库中提取数据等。这里以在线单选投票信息表（voteMaster）为例介绍创建该模块的数据表的结构。

voteMaster（在线单选投票信息表）主要用于保存用户投票的基本信息，如投票的主题、投票的数目等，voteMaster 数据表的结构如表 25.1 所示。

表 25.1 在线单选投票信息表（voteMaster）的结构

字　段	类　型	长　度	是否可为空	说　明
id	int	4	否	投票编号
voteTitle	varchar	200	否	投票主题
voteSum	int	4	否	投票数目
image	varchar	500	是	投票主题图片
endTime	datetime	50	是	投票有效期
mode	datetime	8	是	创建投票主题日期

25.2 关键技术详解

25.2.1 通过 IP 限制投票

在线投票系投中最主要的一个功能就是禁止用户对某一投票主题进行重复投票，即一个用户只能对一个主题投票一次，投票后将不能继续投票。

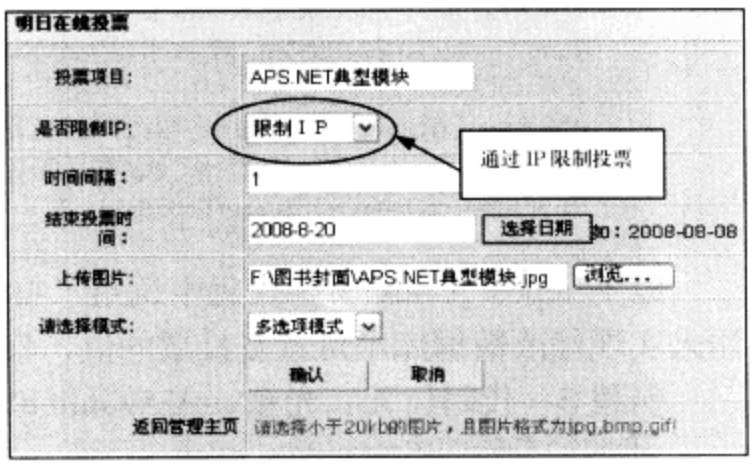


图 25.2 通过 IP 限制投票

实现这一功能通常有两种方法，一种方法是使用 Cookie 对象确认用户的行为，另一种方法是当用户进行投票时，系统首先获取该用户的本地 IP 地址，然后将它与数据库中存在的 IP 地址比较，如果能检索出相同的 IP 地址，就弹出错误信息，如果没有，则会完成用户的投票执行票数增加的操作，并将该用户的本地 IP 存入数据库。实际运行效果如图 25.2 所示。

本模块中采用的是第二种方法。由于系统中存在多个投票主题，为了区别各个选项，需要与先前存入数据库中的 IP 值进行比较来防止重复投票。这里存入数据库中的 IP 地址主要为获取主机名对应的 IP 地址。实现这一功能的代码如下：

例程 1 代码位置：光盘\mr\25\OnlineVotes\ Admin \ AddModule.aspx.cs

```
//获取主机名对应的IP地址
string a = Dns.GetHostName();
IPAddress[] b=Dns.GetHostAddresses(a);
for (int i = 0; i < b.Length; i++)
{
   GetIP = b[i].ToString();
}
```

注意 这里需要引入 using System.Net 命名空间。获取主机名对应的 IP 地址主要使用了该空间中的 Dns 类的 GetHostEntry()方法，该方法有多种重载形式。

25.2.2 多选投票属性设置

多选投票属性设置由 MoreVote 文件夹下的 Votes.aspx 页面实现，在该页面中可通过单击“设置重复投票”按钮来设置多选投票中是否允许重复投票的属性，实际运行效果如图 25.3 所示。

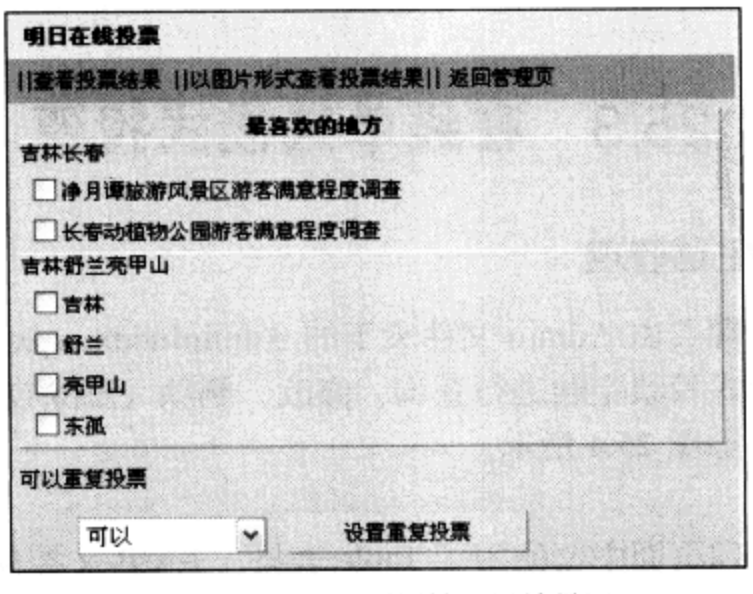


图 25.3 多选投票属性设置效果图

双击 Votes.aspx 页面中的“设置重复投票”按钮，触发其 repeat_Click 事件实现投票主题的绑定，实现的代码如下。

例程 2 代码位置：光盘\mr\25\OnlineVotes\ MoreVote \ Votes.aspx.cs

```
protected void repeat_Click(object sender, EventArgs e)
 {///设置是否重复投票
      //定义公共类Items
      IItems item = new Items();
      //调用公共类中的UpdateIsRepeat来修改是否可重复投票
      item.UpdateIsRepeat(RepeatList.SelectedValue);
      //弹出提示对话框
      Response.Write("<script>alert('修改是否可重复投票成功！');</script>");
      //调用自定义方法BindSubjectDatat重新绑定GridView控件中的数据
      BindSubjectData(GetCurrentTopic());
 }
```

以上 repeat_Click 事件代码中调用了 IItems 公共类中的 UpdateIsRepeat 方法，该方法主要用来实现修改数据库中可重复投票的值，代码如下。

例程 3 代码位置：光盘\mr\25\OnlineVotes\ MoreVote \ Votes.aspx.cs

```
public int UpdateIsRepeat(string isRepeat)
{
  //建议与数据库的连接
  SqlConnection myConnection = new SqlConnection(ConfigurationManager.AppSettings["conStr"].ToString());
    //定义更新数据库是否可重复投票的SQL语句
    string cmd = "update Profiles set p_IsRepeat='" + isRepeat + "'";
    //创建SqlCommad命令对象
    SqlCommand myCommand = new SqlCommand(cmd, myConnection);
    int nResult;
    try
    {
        //打开数据库连接
        myConnection.Open();
        //执行SQL语句操作
        nResult = myCommand.ExecuteNonQuery();
    }
    catch (SqlException ex)
    {
        throw new Exception(ex.Message, ex);
    }
    finally
    {
        myConnection.Close();
    }
    return nResult;
}
```

25.3 在线单选模式投票

25.3.1 单选模式投票主题管理

单选模式投票主题管理页由 Admin 文件夹下的 AdminIndex.aspx 页实现，管理员登录后，在该页面中可以对单选模式投票主题进行查询、修改、删除（删除包括批量删除）和导出数据等操作。页面的运行效果如图 25.4 所示。

1. 页面设计

单选模式投票主题管理页面主要使用了 Table 表格、TextBox 控件、Image 控件、ChekBox 控件、GridView 控件和 Button 控件设计完成，其主要控件设置如表 25.2 所示。

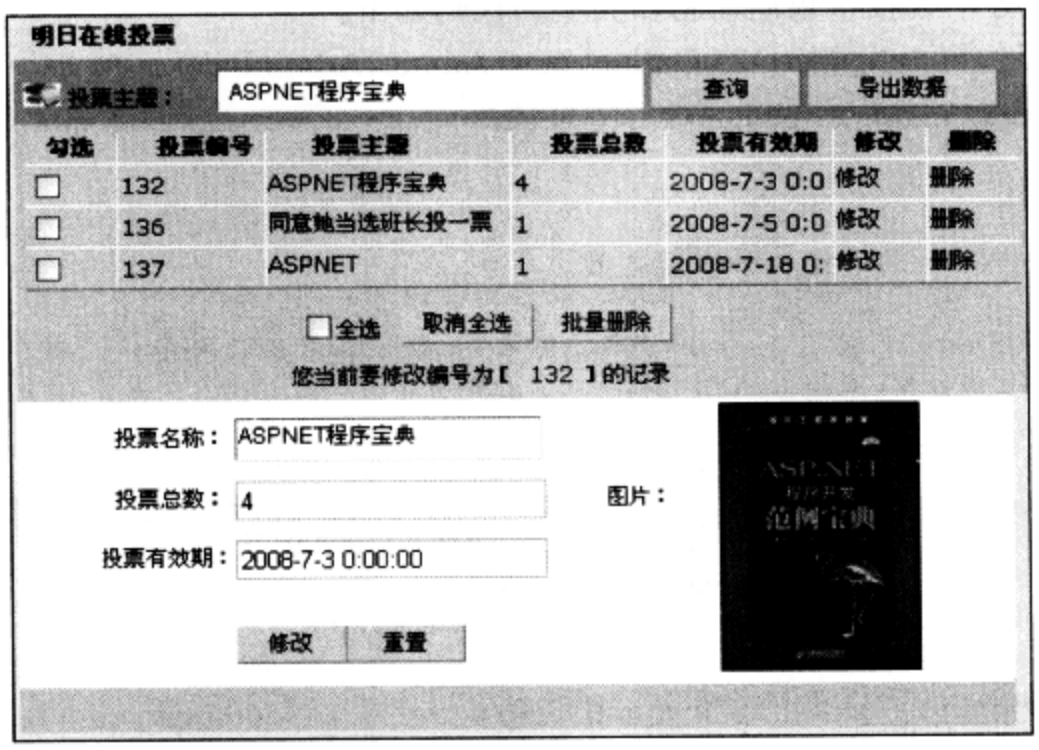


图 25.4　在线单选投票模式管理图

表 25.2　　**单选模式投票控件设置**

控 件 类 型	控 件 名 称	主要属性设置	用　　途
Table	无	无	页面布局
Image	imgPhoto	无	用来显示上传的投票主题图片
TextBox	控件的名称分别为“txtName”、“txtSum”、“txtEndTime”、“txtCondition”	这 4 个控件的 CssClass 值都设置为“login”	txtCondition 用于输入查询信息，其他 3 个控件用于输入修改信息
CheckBox	控件的名称分别为“CheckBox1”、“CheckBox2”	无	都用于批量删除操作
GridView	控件的名称为“gvVoteMaster”	AutoGenerateColumns 属性设置为“False”、AllowPaging 属性设置为“True”和 AutoPostBack 属性设置为“True”	用来绑定单选模式投票主题信息

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 4　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化一个类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下。

例程 5　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
DBClass dbobj = new DBClass();
SqlData sdobj = new SqlData();
```

在 Page_Load 事件中，首先判断是否可以获取到投票主题的 ID 值，如果获取到 ID 值则显示在指定的 Lable 控件中，然后定义一个 DataSet 对象，并调用公共类中自定义方法 ExceDS 来执行指定的 SQL 语句，最后根据定义的 DataSet 对象来创建一个 DataRowView 对象，从数据库中分别读出投票主题、投票数、投票的有效日期和投票主题图片。另外，调用一个自定义方法

OutBindData 绑定导出的投票数据信息。实现的代码如下。

例程 6　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
   if (!IsPostBack)
   {
       //调用自定义方法BindData()绑定投票信息
BindData();
       if (Request.QueryString["id"] != null)//判断，如果可以获取到id的值，则执行以下操作
       {
           //获取修改的投票主题编号
           lbID.Text = Request.QueryString["id"];
           string sqlstr = "select * from voteMaster where id=" + Request.QueryString["id"] + "";
           //定义DataSet对象，并调用自定义方法ExceDS来执行SQL语句
           DataSet ds = dbobj.ExceDS(sqlstr,"voteMaster");
           DataRowView drv = ds.Tables["voteMaster"].DefaultView[0];
           //显示投票主题
           this.txtName.Text = drv["voteTitle"].ToString();
           //显示投票的数量
           this.txtSum.Text = drv["voteSum"].ToString();
           //显示投票的有效日期
           this.txtEndTime.Text = drv["endTime"].ToString();
           //显示投票主题的图片
           this.imgPhoto.ImageUrl = drv["image"].ToString();
           }
           //调用自定义 OutBindData方法绑定导出数据信息
           OutBindData();
           this.Panel1.Visible = false;
       }
   }
```

自定义 BindData()方法用来从数据库中绑定投票主题信息，实现的代码如下。

例程 7　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
 public void BindData()
{
    string sqlstr = "select * from voteMaster";
    this.gvVoteMaster.DataSource = dbobj.ExceDS(sqlstr,"voteMaster");
    gvVoteMaster.DataKeyNames = new string[] { "id" };
    gvVoteMaster.DataBind();
}
```

双击页面中的“修改”按钮，触发其 btnEdit_Click 事件，对获取的投票信息进行修改操作，具体实现的代码如下。

例程 8　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
protected void btnEdit_Click(object sender, EventArgs e)
{
    //自定义一个UPDATE的SQL语句，更新数据中投票信息
      string sqlstr = "update voteMaster set voteTitle='" + this.txtName.Text + "',voteSum='" + Convert.ToInt32 (this.txtSum.Text).To
String() + "',endTime='" + Convert.ToDateTime(this.txtEndTime.Text).ToString() + "' where id=" + Request["id"];
    //调用公共中的ExceSQL方法执行UPDATESQL语句
    dbobj.ExceSQL(sqlstr);
    //调用自定义方法绑定数据库信息
    BindData();
    //调用公共类中的MessageBoxAndUrl方法弹出提示对话框
    Response.Write(sdobj.MessageBoxAndUrl("修改数据成功！", "AdminIndex.aspx"));
    //清空投票信息
    this.txtName.Text = this.txtSum.Text = this.txtEndTime.Text = this.imgPhoto.ImageUrl = "";
}
```

在该页面中可以对 GridView 控件中的数据进行批量删除。在单击“批量删除”按钮，程序

先判断 GridView 控件中哪些行被选中，并对选中行执行删除操作，然后对 GridView 控件重新绑定，以显示最新信息，具体实现代码如下。

例程 9　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
protected void Button2_Click(object sender, EventArgs e)
 {
  for (int i = 0; i <= gvVoteMaster.Rows.Count - 1; i++)
  {
      //应用FindControl方法查找GridView控件中id值为CheckBox1的CheckBox控件
      CheckBox cbox = (CheckBox)gvVoteMaster.Rows[i].FindControl("CheckBox1");
      //判断GridView控件中CheckBox控件是否被选中
      if (cbox.Checked == true)
      {
        //定义一个delete类型的SQL语句
        string sqlstr = "delete from voteMaster where id='" + gvVoteMaster.DataKeys[i].Value + "'";
        //调用公共类中的ExceSQL执行删除语句
            dbobj.ExceSQL(sqlstr);
            //调用公共类中的MessageBoxAndUrl弹出提示对话框
            Response.Write(sdobj.MessageBoxAndUrl("批量删除成功！", "AdminIndex.aspx"));
       }
   }
}
```

另外，在应用程序中还可以将投票主题信息导出到 Excel 中，双击页面中的“导出数据”按钮，触发其 Button1_Click 事件，具体实现代码如下。

例程 10　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
    {
        Response.Clear();
        Response.Buffer = true;
        Response.Charset = "GB2312";
        Response.AppendHeader("Content-Disposition", "attachment;filename=Message.xls");
        Response.ContentEncoding = System.Text.Encoding.UTF7;
        Response.ContentType = "application/ms-excel";//设置输出文件类型为excel文件
        System.IO.StringWriter oStringWriter = new System.IO.StringWriter();
        System.Web.UI.HtmlTextWriter oHtmlTextWriter = new System.Web.UI.HtmlTextWriter(oStringWriter);
        this.GridView2.RenderControl(oHtmlTextWriter);
        Response.Output.Write(oStringWriter.ToString());
        Response.Flush();
        Response.End();
    }
```

在导出到 Excel 中后，程序需要重写 VerifyRenderingInServerForm(Control control)方法，以确保在程序运行时，指定的 GridView 控件总是位于“<form runat="server">”标记内。重写 VerifyRenderingInServerForm(Control control)方法的代码如下。

例程 11　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
public override void VerifyRenderingInServerForm(Control control)
{
    //无须填写任何代码
}
```

注　意

在该页面的前台页面头代码中还要加上 EnableEventValidation="false"，否则可能会出现“只能在执行 Render()的过程中调用 RegisterForEventValidation “错误”

在该页面中还应用了为删除 GridView 控件行信息和单击“批量删除”按钮时弹出确认提示框的技巧，它们触发的事件分别为 GridView 控件的 RowDataBind 事件和 Button 控件 Load 事件，具体实现代码如下。

例程 12　代码位置：光盘\mr\25\OnlineVotes\ Admin\ AdminIndex.aspx.cs

```
//删除GridView控件行信息弹出确认提示框
    protected void gvVoteMaster_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            ((LinkButton)(e.Row.Cells[6].Controls[0])).Attributes.Add("onclick", "return confirm('确定要删除吗？')");
        }
    }
  //为“批量删除”按钮弹出确认提示框
    protected void DeleteAll_Load(object sender, EventArgs e)
    {
        ((Button)sender).Attributes["onclick"] = "return confirm('您确定要删除选中的这些主题吗？');";
    }
```

25.3.2 多选一投票主题模式

多选模式投票主题由 Admin 文件夹下的 AddModule.aspx 页面实现，该页主要包括添加投票项目名称、是否限制 IP、投票有效期、投票主题图片上传等。用户登录页的运行效果如图 25.5 所示。

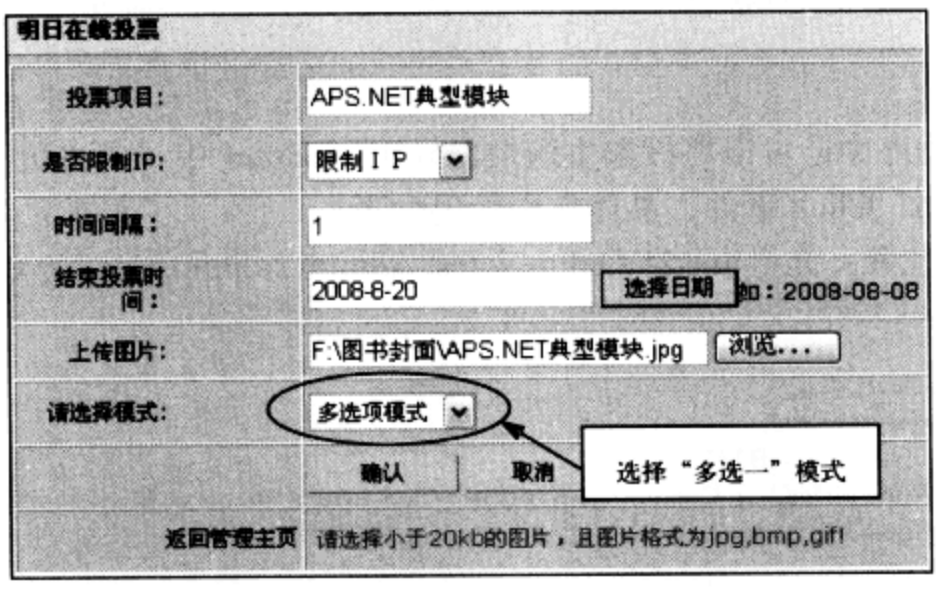


图 25.5 多选一投票主题添加效果图

1. 页面设计

多选一投票主题主要使用 Table 表格、Button 控件、TextBox 控件、FileUpload 控件和 DropDowList 控件设计完成，其主要控件属性设置如表 25.3 所示。

表 25.3 登录页控件列表

控件类型	控件名称	主要属性设置	用途
Table	无	无	页面布局
TextBox	控件的名称分别为“ItemsName”、“TextBox3”、	两控件的 SkinID 属性都设置为 tbSkin	用于输入投票项目名称、投票时间间隔
DropDownList	控件的名称分别为“ddlModule”和“ddlIP”	两控件的 SkinID 属性都设置为 tbSkin	分别用于显示“是否限制 IP”信息和单选投票模式选择
Button	控件的名称分别设置为“btOK”和“btCancle”	将 Text 属性值分别设置为“确认”和“取消”	这两个按钮的功能分别为“审核填写的投票信息是否正确”和“取消添写的内容”
FileUpload	FileUpload1	该控件的 CssClass 属性设置为 Button	用来上传投票主题图片

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 13 代码位置：光盘\mr\25\OnlineVotes\ Admin\AddModule.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件外声明并且实例化一个类对象。声明类对象的目的是利用声明的类对象来调用类中的方法。实现的代码如下。

例程 14 代码位置：光盘\mr\25\OnlineVotes\ Admin\AddModule.aspx.cs

```
DBClass dbobj = new DBClass();
SqlData sdobj = new SqlData();
```

双击页面中的“确认”按钮，触发其 btOK_Click 事件，实现投票项目名称的添加、投票主题图片的上传等操作，在图片进行上传时限制了上传大小，实现的代码如下。

例程 15 代码位置：光盘\mr\25\OnlineVotes\ Admin\AddModule.aspx.cs

```
protected void btOK_Click(object sender, EventArgs e)
{
    //客户添加投票项目，可选择是否上传图片
    int ip = Convert.ToInt32(this.ddlIP.SelectedValue);
    if (FileUpload1.HasFile)
    {
        if (FileUpload1.PostedFile.ContentLength < 40960)
        {
            //获取上传图片的路径
            string filepath = FileUpload1.PostedFile.FileName;
            // 获取上传图片名称
            string filename = filepath.Substring(filepath.LastIndexOf("\\") + 1, filepath.Length - (filepath.LastIndexOf("\\")) - 1);
          //设置上传图片的格式
          string fileEx = filepath.Substring(filepath.LastIndexOf(".") + 1);
          //保存到服务器的路径
          string serverpath = Server.MapPath("~/File/") + filename;
          //判断图片格式
          if (fileEx.ToLower() == "jpg" || fileEx.ToLower() == "bmp" || fileEx.ToLower() == "gif")
          {
              //保存上传路径
              FileUpload1.PostedFile.SaveAs(serverpath);
          }
          else
          {
              //调用公共类中的MessageBoxAndUrl方法弹出提示对话框
           Response.Write(sdobj.MessageBoxAndUrl("上传的图片扩展名错误!","AddModule.aspx"));
        }
           //以下操作是将上传文件信息保存到数据库中
           string strmaster1 = "insert into voteMaster(voteTitle,image,endTime) values('" + this.ItemsName.Text.ToStr
ing() + "','" + @"..\File\" + filename + "','" + Convert.ToDateTime(this.EndTime.Text.ToString()) + "')";
           //调用公共类中的ExceSQL方法执行SQL语句
           dbobj.ExceSQL(strmaster1);
           //定义一个insertItem方法，用来添加选项表和系统设置
           insertItem();
           //调用公共类中的MessageBox方法弹出提示对话框
           Response.Write(sdobj.MessageBox("添加图片投票成功!"));
        }
            else
            {
                //调用公共类中的MessageBox方法弹出提示对话框
                Response.Write(sdobj.MessageBox("你上传的图片超过了40KB!"));
            }
        }
        else
```

```
            {///如果用户没有上传图片，则执行以下代码
                string strmaster2 = "insert into voteMaster(voteTitle,endTime) values('" + this.ItemsName.Text.ToString (). Trim() +
"','" + this.EndTime.Text.ToString().Trim() + "')";
                //调用公共类中的ExceSQL方法执行SQL语句
                dbobj.ExceSQL(strmaster2);
                //定义一个insertItem方法，用来添加选项表和系统设置
                insertItem();
                //调用公共类中的MessageBox方法弹出提示对话框
                Response.Write(sdobj.MessageBox("添加文字投票成功!"));
            }
        }
    }
```

自定义一个 insertItem()方法，用来实现在单选模式投票中实现添加选项表、系统设置、单选条件项和多选条件项模式，实现的代码如下。

例程 16 代码位置：光盘\mr\25\OnlineVotes\ Admin\AddModule.aspx.cs

```
//定义一个方法，用来添加选项表和系统设置，模式
public void insertItem()
{
    int ip = Convert.ToInt32(this.ddlIP.SelectedValue);
    string strid = "select id from voteMaster where voteTitle='" + this.ItemsName.Text.ToString() + "'";
    int GetID = Convert.ToInt32(dbobj.ExeGetReturn(strid));
    string striconfig = "insert into voteConfig(id,checkIP,checkTime) values(" + GetID + "," + ip + "," + this.TextBox3.Text + ")";
    //判断模式：单选模式和多选模式
    if (this.ddlModule.Text == "单选项模式")
    {
            //投票标题添加到选项表并储存模式记录
            string stritem = "insert into voetDetails(voteItem,id) values('" + this.ItemsName.Text + "'," + GetID + ")";
            string strmode = "update voteMaster set mode ="+ 0 +"where id=" + GetID;
            string[] arq = new string[3];
            arq[0] = stritem;
            arq[1] = strmode;
        arq[2] = striconfig;
        if (sdobj.trantion(arq))
        {
            Response.Write(sdobj.MessageBoxAndUrl("添加单选项模式成功!","../Index.aspx"));
        }
        else
        {
            Response.Write(sdobj.MessageBox("添加单选项模式失败!"));
        }
    }
    if (this.ddlModule.Text == "多选项模式")
    {
        //添加系统设置和模式记录并跳转到添加选项页面
        string strmode2 = "update voteMaster set mode ="+ 1 +" where id=" + GetID;
        string[] arq1 = new string[2];
        arq1[0] = strmode2;
        arq1[1] = striconfig;
        if (sdobj.trantion(arq1))
        {
            //跳转到添加选项页面并把投票标题传过去
            Response.Redirect("addItems.aspx?voteTitle=" + this.ItemsName.Text.ToString());
        }
        else
        {
            Response.Write(sdobj.MessageBox("添加多选项模式失败!"));
        }
    }
}
```

25.4 在线多选模式投票

25.4.1 多选模式投票主题管理

多选模式投票主题管理由 MoreVote 文件夹下的 TopicManage.aspx 页实现，在该页面中可以对多选模式投票主题进行添加、更新、删除和设置当前主题等操作。

该页面实际运行的效果如图 25.6 所示。

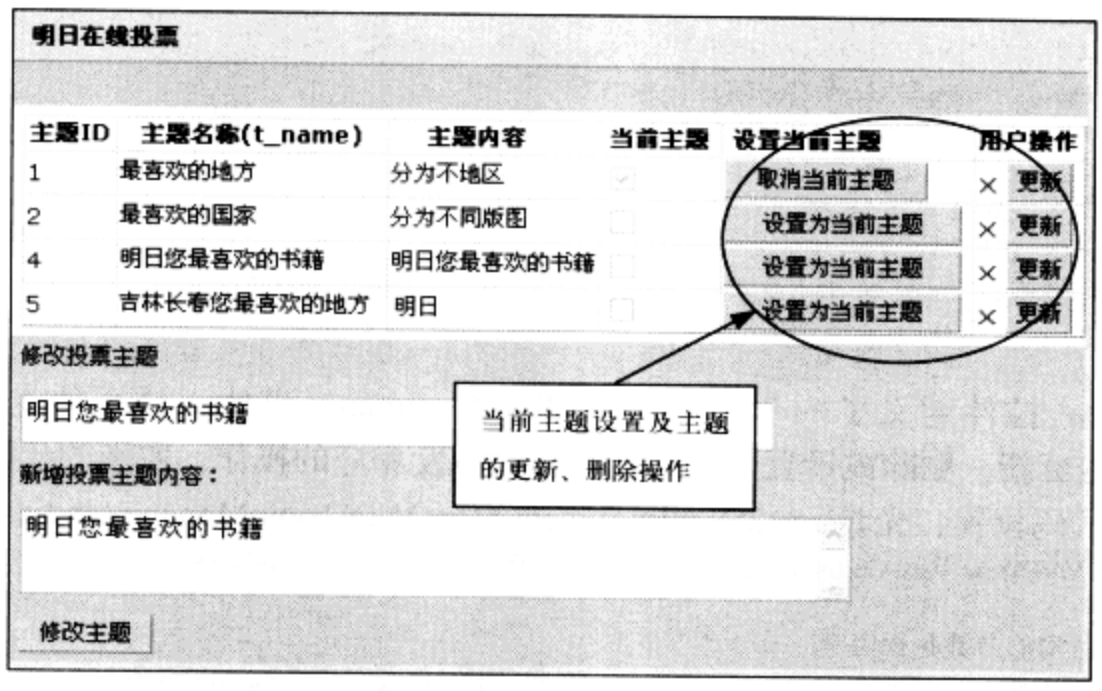


图 25.6 运行效果图

1. 页面设计

多选模式投票主题管理页面主要使用 Table 表格、TextBox 控件和 Button 控件设计完成，其主要控件属性设置如表 25.4 所示。

表 25.4 多选模式投票控制设置

控件类型	控件名称	主要属性设置	用途
Table	无	无	页面布局
GridView	TopicView	AutoGenerateColumns 属性设置为“False”	显示多选投票主题信息
TextBox	控件的名称为“name”和“content”	控件名为“name”无属性设置，控件名为“content”的 TextMode 属性设置为“MultiLine”	这两个控件分别用来实现新增投票主题和内容的添置
Button	控件的名称为“AddUpdatebtn”	CommandName 属性设置为“add”、CssClass 属性设置为“login”，Text 属性设置为“新增主题”	用于对新增主题的添加

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 17 代码位置：光盘\mr\25\OnlineVotes\ MoreVote\TopicManage.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

首先，在 Page_Load 事件中调用一个自定义方法 BindTopicData()用来实现从数据库中绑定多选模式投票主题内容。实现的代码如下。

例程 18　代码位置：光盘\mr\25\OnlineVotes\ MoreVote\TopicManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack) BindTopicData();
}
```

自定义一个 BindTopicData()方法从数据库绑定多模式投票主题的内容，具体实现代码如下。

例程 19　代码位置：光盘\mr\25\OnlineVotes\ MoreVote\TopicManage.aspx.cs

```
public void BindTopicData()
{
    //实例化公共类ITopics
    ITopics topic = new Topics();
    //调用公共类中的GetAllTopic()方法创建一个SqlDataReader对象
    SqlDataReader dr = topic.GetAllTopic();
    //设置TopicView控件数据源
    TopicView.DataSource = dr;
    //绑定数据库中数据
    TopicView.DataBind();
    //关闭数据阅读器
    dr.Close();
}
```

在 GridView 控件定义了一个 VoteView_RowCommand 事件，该事件主要用来实现在 GridView 中单击更新、删除或设置当前主题按钮时触发相应的操作，实现的代码如下。

例程 20　代码位置：光盘\mr\25\OnlineVotes\ MoreVote\TopicManage.aspx.cs

```
protected void VoteView_RowCommand(object sender,GridViewCommandEventArgs e)
{
    //判断传过来的参数是否为空
    if(Int32.TryParse(e.CommandArgument.ToString(),out sid)==falselle.CommandName=="")
    {
        Response.Write("<script>alert('没有接收到参数！ ');</script>");
        return;
    }
    //实例化公共类ITopics
    ITopics topic = new Topics();
    //设置按钮属性值
    switch (e.CommandName)
    {
    //单击删除按钮时执行以下条件中代码
    case "delete":
        //调用公共类中的DeleteTopic方法接受传递过来的参数
        topic.DeleteTopic(sid);
        Response.Write("<script>alert('删除成功！    ');</script>");
        //调用自定义方法BindTopicData()重新绑定数据库中数据
        BindTopicData();
        break;
    case "update":
        SqlDataReader dr = topic.GetTopic(sid);
        if (dr.Read())
        {
            name.Text=dr["t_name"].ToString();
            content.Text=dr["t_content"].ToString();
            AddUpdate.Text = "修改投票主题";
            AddUpdatebtn.Text = "修改主题";
            ///将CommandName、CommandArgument传给ID=AddUpdatebtn的按钮
            AddUpdatebtn.CommandName = "update";
            AddUpdatebtn.CommandArgument = sid.ToString();
        }
        dr.Close();
        break;
```

```
        case "iscurrent":
            topic.UpdateCurrentTopicByProc(sid);
            Response.Write("<script>alert('更新前台主题成功！    ');</script>");
            BindTopicData();
            break;
    }
    }
```

双击“增加主题”按钮，触发其 AddUpdatebtn_Click 事件，实现多选模式投票主题的添加操作，实现代码如下。

例程 21　代码位置：光盘\mr\25\OnlineVotes\ MoreVote\TopicManage.aspx.cs

```
protected void AddUpdatebtn_Click(object sender, EventArgs e)
{
    if (name.Text.Trim().Length < 2 || content.Text.Trim().Length < 2)
    {
        Response.Write("<script>alert('字数不够');</script>");
        return;
    }
    ITopics topic=new Topics();
    ///根据AddUpdatebtn.CommandName分为添加和更新，因为不是RowDataBound所以不能用e.CommandName
    switch (AddUpdatebtn.CommandName)
    {
      case "add":
            try
            {
                topic.AddTopic(name.Text.Trim().ToString(), content.Text.Trim().ToString());
                Response.Write("<script>alert('新增成功！    ');</script>");
                BindTopicData();
            }
            catch (Exception ex)
            {
                Response.Redirect("ErrorPage.aspx?ErrorUrl=" + Request.Url.ToString().Replace("<br>", "").Replace("\n", "") +
"&ErrorMsg=" + ex.Message.Replace("<br>", "").Replace("\n", ""));
                            }
            break;
        case "update":
            try
            {
                topic.UpdateTopic(name.Text.Trim().ToString(), content.Text.Trim().ToString(), Int32.Parse (AddUpdatebtn.
CommandArgument));
            Response.Write("<script>alert('修改主题成功！');</script>");
            BindTopicData();
        }
        catch (Exception ex)
        {
            Response.Redirect("ErrorPage.aspx?ErrorUrl=" + Request.Url.ToString().Replace("<br>", "").Replace("\n","") +
"&ErrorMsg=" + ex.Message.Replace("<br>", "").Replace("\n", ""));
                            }
                            break;
                }
            }
```

25.4.2　多选模式投票项管理

多选模式投票项管理页由 MoreVote 文件夹下的 SubjectManage.aspx 页实现，在该页面中首先通过一个下拉列表框选择一个投票主题，接着可以对投票项内容进行添加，对于添加的投票项可以进行更新和删除操作，运行的效果如图 25.7 所示。

1．页面设计

多选模式投票项管理页面主要使用 Table 表格、Button 控件、TextBox 控件、DropDownList

控件和 GridView 控件设计完成，其主要控件设置如表 25.5 所示。

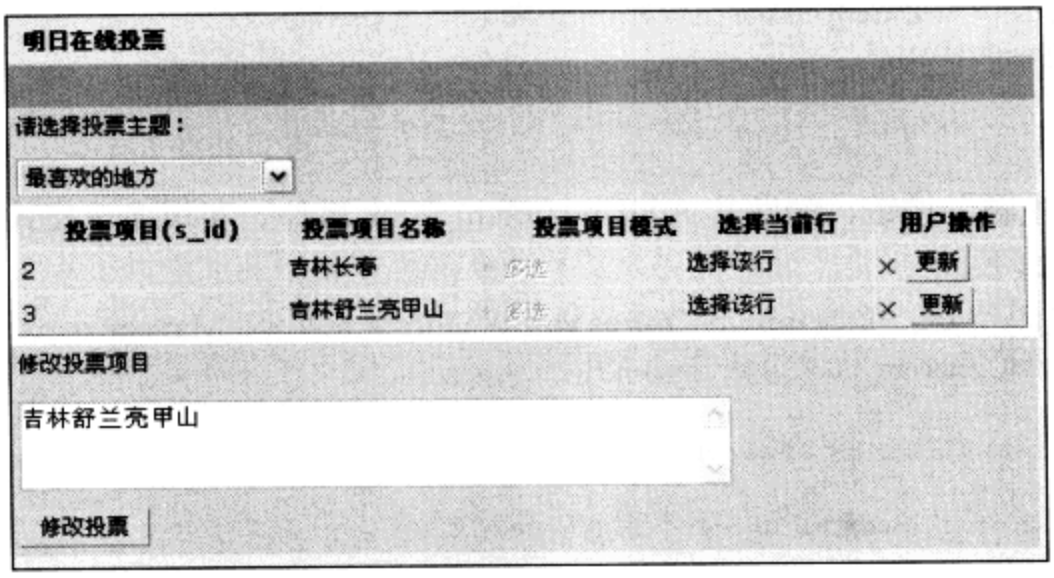


图 25.7　多选模式投票项管理运行效果图

表 25.5　　多选模式投票项控件设置

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
TextBox	控件名称为“name”	TextMode 属性设置为 MultiLine	用于添加新增投票项目和在修改投票项目时显示项目信息
Button	控件的名称“AddUpdateBtn”	将 CommandName 属性值设置为“add”、Css Class 属性设置为“login”	该按钮根据 CommandName 属性传值来实现投票项目的添加和修改
GridView	控件的名称“SubjectView”	AutoGenerateColumns 属性设置为"False"，CssClass 属性设置为“login”、Data Key Names 属性设置为“s_id”	该控件用来绑定多选投票的投票项目信息
DropDownList	将控件的名称设置为“TopicList”	将该控件的 AutoPostBack 属性设置为"Truie" 、将 CssClass 属性设置为 "login"	该控件用来选择投票主题

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 22　代码位置：光盘\mr\25\OnlineVotes\ MoreVote\SubjectManage.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 页面加载的事件当中，首先判断页面是否首次加载，然后调用自定义方法 BindTopicData()绑定下拉列表中投票主题名称，调用 BindSubjectData(int tid)方法绑定 GridView 控件中投票项目信息，实现代码如下。

例程 23　代码位置：光盘\mr\25\OnlineVotes\ MoreVote\SubjectManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        //调用自定义方法BindTopicData()绑定下拉列表框中投票主题内容
        BindTopicData();
        if (TopicList.Items.Count > 0)
        {///调用自定义方法BindSubjectData使SubjectLIst绑定TopicList的第一项
```

```
                TopicList.SelectedIndex = 0;
                BindSubjectData(Int32.Parse(TopicList.SelectedValue));
            }
        }
    }
    //自定义方法BindTopicData()绑定下拉列表框中投票主题内容
    private void BindTopicData()
    {
        //实例化公共类中的ITopics
        ITopics topic = new Topics();
        SqlDataReader dr = topic.GetAllTopic();
        //绑定下拉列表框中数据
        TopicList.DataSource = dr;
        TopicList.DataTextField = "t_name";
        TopicList.DataValueField = "t_id";
        TopicList.DataBind();
        dr.Close();
    }
    //自定义BindSubjectData(int tid)方法绑定GridView控件中数据
    private void BindSubjectData(int tid)
    {
        ISubjects subject = new Subjects();
        //创建一个数据阅读器对象
        SqlDataReader dr = subject.GetSubjectByTopic(tid);
        //绑定GridView控件中的数据
        SubjectView.DataSource = dr;
        SubjectView.DataBind();
        dr.Close();
    }
```

双击页面中的“新增投票”按钮，触发其 AddUpdateBtn_Click 事件实现对投票项目内容的添加和更新操作，实现的代码如下。

例程 24 代码位置：光盘\mr\25\OnlineVotes\ MoreVote\SubjectManage.aspx.cs

```
protected void AddUpdateBtn_Click(object sender, EventArgs e)
{
    //实例化公共类ISubjects
    ISubjects subject = new Subjects();
    if (AddUpdateBtn.CommandName != "")
    {///根据AddUpdatebtn.CommandName分为添加和更新，因为不是RowDataBound所以不能用e.CommandName
        switch (AddUpdateBtn.CommandName)
        {
            //执行添加操作代码
            case "add":
                subject.AddSubject(name.Text.Trim().ToString(), Int32.Parse(TopicList.SelectedValue));
                    Response.Write("<script>alert('添加投票项目成功！');</script>");
                    BindSubjectData(Int32.Parse(TopicList.SelectedValue));
                    break;
                //执行更新操作代码
                case "update":
                    subject.UpdateSubject(name.Text.Trim().ToString(), Int32.Parse (AddUpdateBtn.CommandArgument));
                    Response.Write("<script>alert('更新投票项目成功！');</script>");
                    BindSubjectData(Int32.Parse(TopicList.SelectedValue));
                    break;
            }
        }
    }
```

25.4.3 多选模式投票内容管理

多选模式投票内容管理由 MoreVote 文件夹下的 ItemManage.aspx 页实现，在该页面中首先选择一个投票主题，之后再选择一个投票项目，然后可以对投票内容进行添加操作，添加投票

内容后可以对添加的信息进行删除和更新操作，实际运行效果如图 25.8 所示。

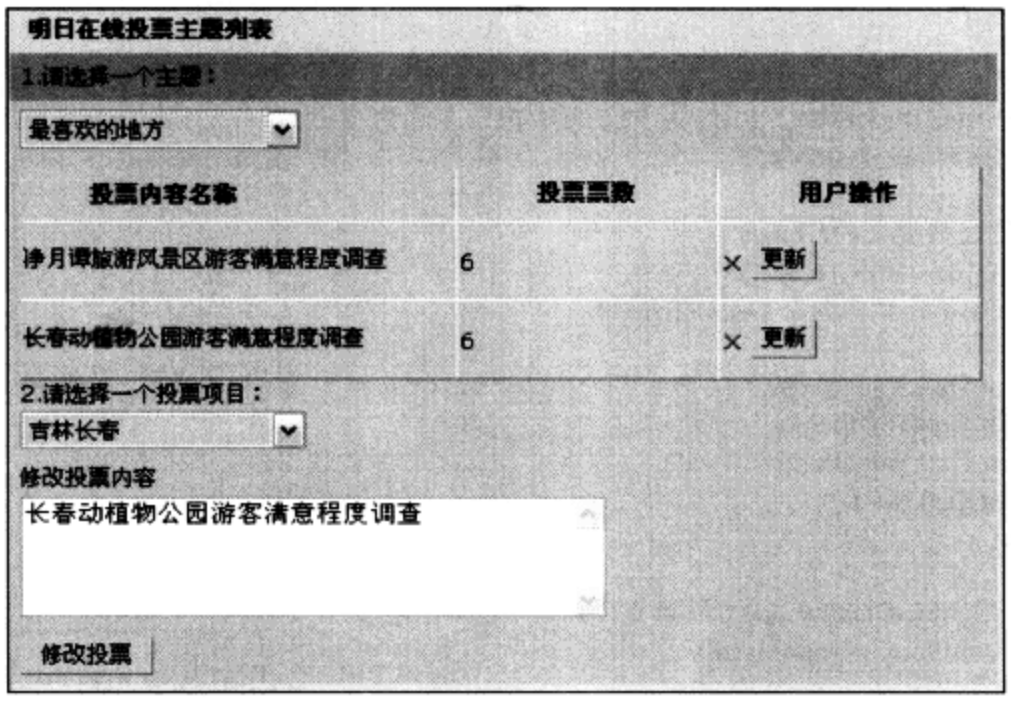


图 25.8　多选模式投票内容管理效果图

1. 页面设计

多选模式投票内容管理页面主要使用 Table 表格、Button 控件、TextBox 控件和 DropDownList 控件设计完成，其主要控件设置如表 25.6 所示。

表 25.6　　多选模式投票内容控件设置

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
TextBox	控件名称为“name”	TextMode 属性设置为“MultiLine”	用于添加新增投票内容和在修改投票内容时显示投票内容
Button	控件的名称“AddUpdateBtn”	将 CommandName 属性值设置为“add”、CssClass 属性设置为“login”	该按钮根据 CommandName 属性传值来实现多选投票内容的添加和修改
GridView	控件的名称“ItemView”	AutoGenerateColumns 属性设置为“False”，CssClass 属性设置为“login”	该控件用来绑定多选投票的投票内容信息
DropDownList	将控件的名称设置为“TopicList”和“SubjectList”	这两个控件的 AutoPost Back 属性设置为“Truie”、将 CssClass 属性设置为“login”	这两个控件分别用来选择投票主题和选择一个投票项目

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下：

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 页面加载的事件当中，首先判断页面是否首次加载，然后调用自定义方法加载页面，代码如下。

例程 25　代码位置：光盘\mr\25\OnlineVotes\ MoreVote\ItemManage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
```

```
            //调用自定义方法绑定投票主题信息
            BindTopicData();
            if (TopicList.Items.Count > 0)
            {///使SubjectLIst绑定TopicList的第一项
                TopicList.SelectedIndex = 0;
                //调用自定义方法绑定投票项目名称
                BindSubjectData(Int32.Parse(TopicList.SelectedValue));
                if (SubjectList.Items.Count > 0)
                {
                    SubjectList.SelectedIndex = 0;
                    BindItemData(Int32.Parse(SubjectList.SelectedValue));
                }
            }
        }
    }
    //自定义方法BindTopicData()绑定下拉列表框中投票主题
    private void BindTopicData()
    {
        ITopics topic = new Topics();
        SqlDataReader dr = topic.GetAllTopic();
        TopicList.DataSource = dr;
        TopicList.DataTextField = "t_name";
        TopicList.DataValueField = "t_id";
        TopicList.DataBind();
        dr.Close();
    }
    //自定义方法BindSubjectData(int tid)绑定下拉列表框中投票项目
    private void BindSubjectData(int tid)
    {
        ISubjects subject = new Subjects();
        SqlDataReader dr=subject.GetSubjectByTopic(tid);
        SubjectList.DataSource=dr;
        SubjectList.DataTextField = "s_name";
        SubjectList.DataValueField = "s_id";
        SubjectList.DataBind();
        dr.Close();
    }
    //自定义方法绑定GridView控件中信息
        private void BindItemData(int sid)
        {
            IItems item = new Items();
            SqlDataReader dr = item.GetItemBySubject(sid);
            ItemView.DataSource = dr;
            ItemView.DataBind();
            dr.Close();
        }
```

定义GridView控件的一个ItemView_RowCommand事件，该事件主要在对GridView控件绑定的数据进行修改和删除操作时触发，具体代码如下。

例程 26 代码位置：光盘\mr\25\OnlineVotes\ MoreVote\ItemManage.aspx.cs

```
    protected void ItemView_RowCommand(object sender, GridViewCommandEventArgs e)
    {
        if (Int32.TryParse(e.CommandArgument.ToString(), out iid) == false || e.CommandName == "")
        {
            //调出“删除成功” 对话框
    Response.Write("<script>alert('没有接收到参数！');</script>");
            return;
        }
        //实例化公共类IItems
```

```
        IItems item=new Items();
        switch (e.CommandName)
        {
            //当触发该控件的删除操作按钮时执行以下代码
            case "delete":
                item.DeleteItem(iid);
                Response.Write("<script>alert('删除成功！');</script>");
                //调用自定义方法重新绑GridView中数据
                BindItemData(Int32.Parse(SubjectList.SelectedValue));
                break;
            //当触发该控件的修改操作按钮时执行以下代码
            case "update":
                //创建一个数据阅读器对象
                SqlDataReader dr = item.GetSingleItem(iid);
                //如果读取到数据库中数据
                if (dr.Read())
                {
                    title.Text = "修改投票内容";
                    name.Text=dr["i_name"].ToString();
                    AddUpdatebtn.Text = "修改投票";
                    AddUpdatebtn.CommandName = "update";
                    AddUpdatebtn.CommandArgument = dr["i_id"].ToString();
                }
                break;
        }
    }
```

双击页面中的“增加投票”按钮，触发其 AddUpdatebtn_Click 事件，实现对投票内容进行添加和更新添加的操作，实现代码如下。

例程 27 代码位置：光盘\mr\25\OnlineVotes\ MoreVote\ItemManage.aspx.cs

```
protected void AddUpdatebtn_Click(object sender, EventArgs e)
    {
        if (AddUpdatebtn.CommandName != "")
        {
            //实例化公共类IItems
            IItems item = new Items();
            ///根据AddUpdatebtn.CommandName分为添加和更新，因为不是RowDataBound所以不能用e.CommandName
            switch (AddUpdatebtn.CommandName)
            {
                //当选择添加投票内容时执行的代码
                case "add" :
                    try
                    {
item.AddItem(name.Text.Trim().ToString(),Int32.Parse(SubjectList.SelectedValue));
                        Response.Write("<script>alert('新增成功！');</script>");
                        BindItemData(Int32.Parse(SubjectList.SelectedValue));
                    }
                    catch (Exception ex)
                    {
                        Response.Redirect("ErrorPage.aspx?ErrorUrl=" + Request.Url.ToString().Replace("<br>",
"").Replace("\n", "") + "&ErrorMsg=" + ex.Message.Replace("<br>", "").Replace("\n", ""));
                    }
                    break;
                //当选择更新投票内容时执行的代码
                case "update":
                    try
                    {

item.UpdateItem(name.Text.Trim().ToString(),Int32.Parse(AddUpdatebtn.CommandArgument.ToString()));
```

```
                    Response.Write("<script>alert('更新成功！');</script>");
                    BindItemData(Int32.Parse(SubjectList.SelectedValue));
                }
                catch (Exception ex)
                {
                    Response.Redirect("ErrorPage.aspx?ErrorUrl=" + Request.Url.ToString().Replace("<br>",
"").Replace("\n", "") + "&ErrorMsg=" + ex.Message.Replace("<br>", "").Replace("\n", ""));
                }
                break;
            }
        }
    }
```

25.5 程序调试与错误处理

在单选投票主题管理页 AdminIndex.aspx 中，可实现将单选投票主题导入到 Excel 中，在该页的“导出数据”按钮的 Click 事件中编写代码如下。

例程 28 代码位置：光盘\mr\25\OnlineVotes\ Admin \ AdminIndex.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Clear();
    Response.Buffer = true;
    Response.Charset = "GB2312";
    Response.AppendHeader("Content-Disposition", "attachment;filename=Message.xls")
    Response.ContentEncoding = System.Text.Encoding.UTF7;
    Response.ContentType = "application/ms-excel";//设置输出文件类型为excel文件
    System.IO.StringWriter oStringWriter = new System.IO.StringWriter();
    System.Web.UI.HtmlTextWriter oHtmlTextWriter = new System.Web.UI.HtmlTextWriter(oStringWriter);
    this.GridView2.RenderControl(oHtmlTextWriter);
    Response.Output.Write(oStringWriter.ToString());
    Response.Flush();
    Response.End();
}
```

运行上述导出数据到 Excel 文件中的代码，提示出“类型‘GridView’的控件‘GridView2’必须放在具有 runat = server 的窗体标记内”，如图 25.9 所示。

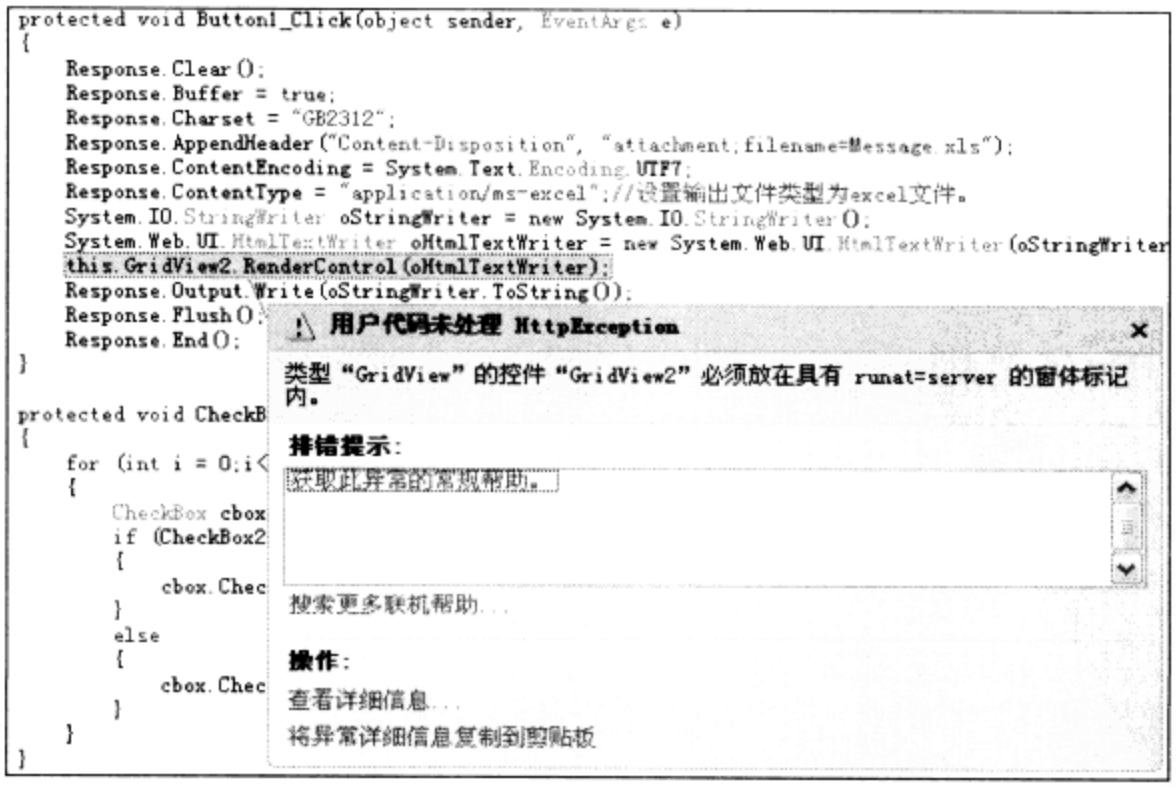


图 25.9 程序错误与调试

出现如图 25.9 所示的错误主要是由于 AdminIndex.aspx 页中用到了 GridView 控件，所以需要重写 VerifyRenderingInServerForm(Control control)方法。如果没有重写该方法，程序会报错。

在该页面的后台代码中重写 VerifyRenderingInServerForm(Control control)方法的代码如下。

例程 29 代码位置：光盘\mr\25\OnlineVotes\ Admin \ AdminIndex.aspx.cs

```
public override void VerifyRenderingInServerForm(Control control)
 {
     //无须填写任何代码
 }
```

重写 VerifyRenderingInServerForm(Control control)方法，以确保在程序运行时，指定的 GridView 控件总位于“<form runat=server>”标记内。

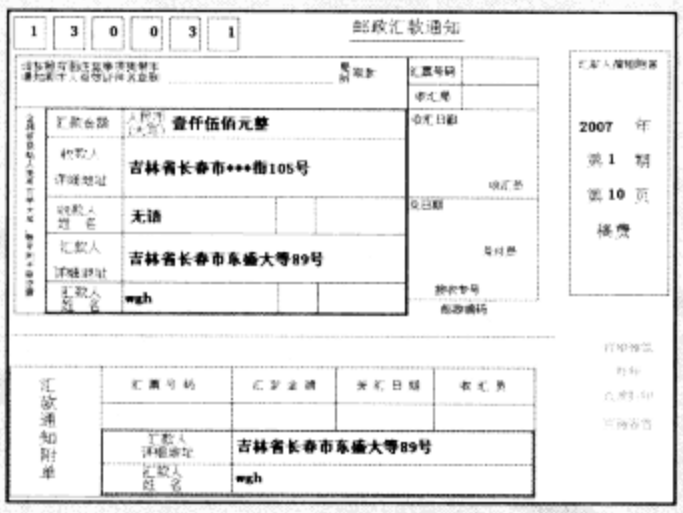
万能打印模块

第 26 章

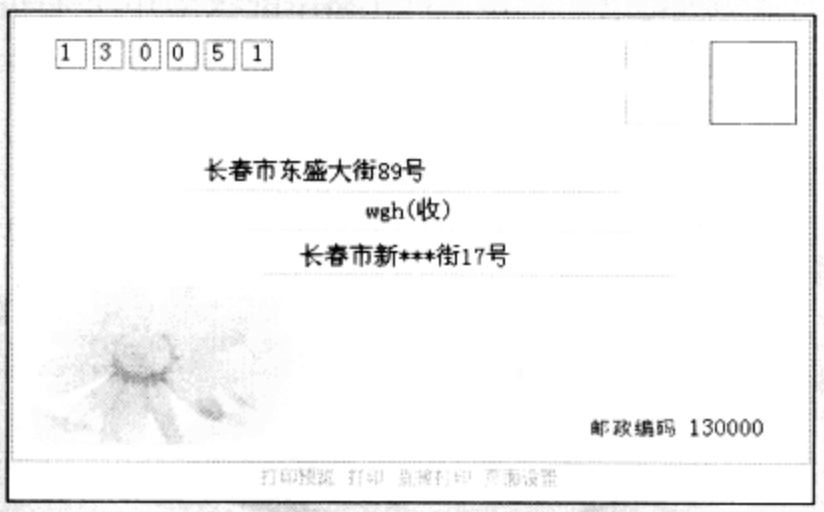
实例位置：光盘\mr\26\

在开发应用程序时，许多用户都要求程序具有打印功能，以便保存数据，或是保存修改后的数据等。为了便于对不同种类的信息进行打印，笔者制作了一个万能打印模块，该模块可以执行“套打邮寄产品单”、“打印图片”、“操作表格并打印”、“调用 office 进行打印”和“打印购物车中的订单”等操作。通过本章的学习，读者能够学到：

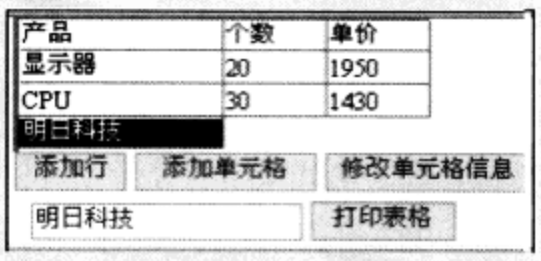
打印汇款单



打印信封

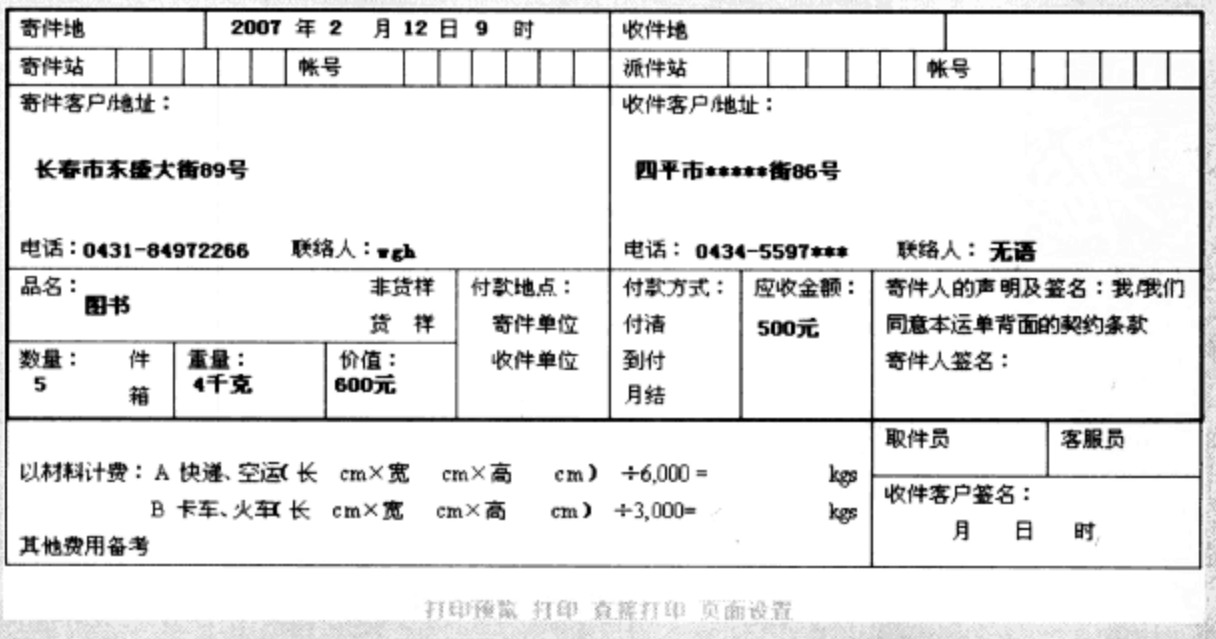


智能添加行和单元格并打印



修改表格中的数据并打印

打印快递单



26.1 万能打印模块设计思路

万能打印模块，其主要实现以下目标。

- 设置页眉、页脚或者清空页眉、页脚。
- 对汇款单、快递单和信封进行打印。
- 获取焦点对指定框架中的内容进行打印。
- 利用 WebBrowser 进行打印。
- 智能放大或者缩小图片后进行打印。
- 对简历进行打印。
- 将数据导出到 Word 中并且打印。
- 将数据导出到 Excel 中并且打印。
- 操作表格中的数据时能够修改表格中的数据并打印。
- 可以左右移动单元格中的数据并且打印。
- 在表格中添加行或者添加单元格并打印。
- 动态生成行或者单元格并打印。
- 对购物车中的订单进行打印。

万能打印模块首页的运行效果如图 26.1 所示。

图 26.1 万能打印首页运行效果图

26.2 万能打印模块关键技术

26.2.1 获取焦点并且打印框架中的内容

在万能打印模块中通过 JavaScript 调用 IE 自身的打印功能实现打印，但是该方法将打印页面中的全部内容，包括“打印”超级链接，有时这是不需要的。下面介绍如何通过打印指定框架中的内容实现页面部分内容的打印。打印指定框架中的内容运行效果如图 26.2 所示，单击“打印”超级链接，会弹出“打印”对话框，进行相应的设置后，单击“打印”即可对表格内容进行打印。

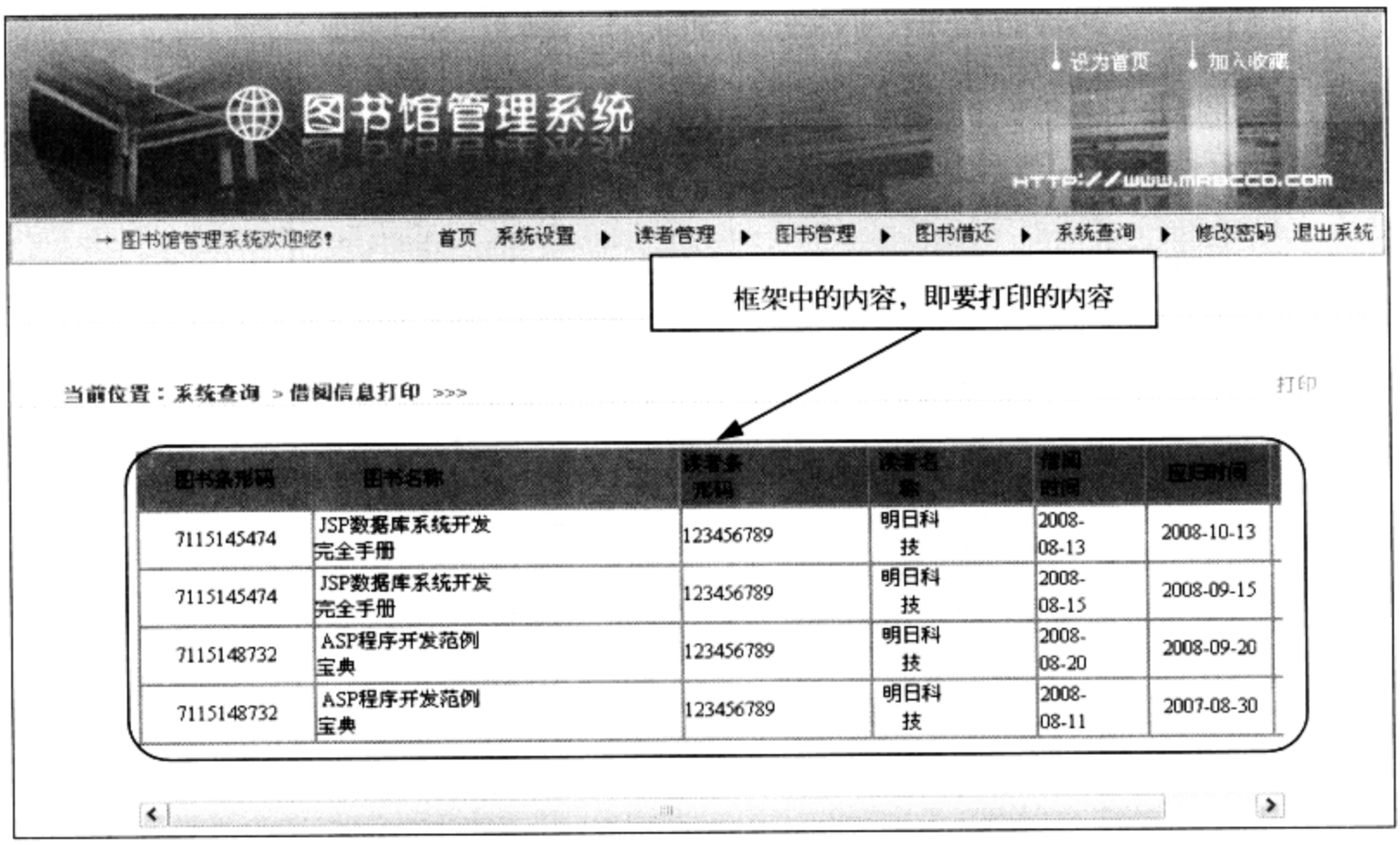


图 26.2 打印指定框架中的内容

在实现打印指定框架中内容时，首先需要让打印的框架获得焦点，然后再调用 window 对象的 print()方法实现打印。

语法格式如下：

```
parent.mainframe.focus();
window.print();
```

参数说明如下。

- mainFrame：表示框架的名称。

实现打印指定框架中的内容步骤如下。

（1）首先在设置 content.aspx 页面中，将要打印的内容放置在该页面中。

（2）然后设置“printContent.aspx”页面，在该页面的适当位置添加浮动框架（应用<iframe>标记创建）。并且将该浮动框架的 src 属性指定为（1）中创建的 content.aspx 文件，代码如下：

```
<iframe name="contentFrame" src="content.aspx"  frameborder="0" width="100%" height="100%"></iframe>
```

（3）在 printContent.aspx 页面中添加“打印”超级链接，打印指定浮动框架中的内容，实现的代码如下：

```
<a href="#" onClick="parent.contentFrame.focus();window.print();">打印</a>
```

26.2.2 利用 WebBrowser 打印

在万能打印模块中用到了 WebBrowser 来进行打印，WebBrowser 是 IE 内置的浏览器控件，无须用户自行下载。它的优点是客户端独立完成打印目标文档，减轻服务器负荷；缺点是原文档的分析操作复杂，并且要对源文档中要打印的内容进行约束。在利用 WebBrowser 实现打印中，单击“打印预览”超级链接，即可打开“打印预览”对话框，如图 26.3 所示，单击“打印”按钮，即可打开“打印”对话框进行打印。

图 26.3 利用 WebBrowser 打印

万能打印中利用 WebBrowser 实现打印主要应用 IE 内置的 WebBrowser 控件实现，该控件的具体参数如下。

- document.all.WebBrowser.Execwb(7,1)：表示打印预览。
- document.all.WebBrowser.Execwb(6,1)：表示打印。
- document.all.WebBrowser.Execwb(6,6)：表示直接打印。
- document.all.WebBrowser.Execwb(8,1)：表示页面设置。

另外，WebBrowser 组件中还有其他一些方法，列举如下。

- document.all.WebBrowser.Execwb(1,1)：表示打开。
- document.all.WebBrowser.Execwb(2,1)：表示关闭现在所有的 IE 窗口，并打开一个新窗口。
- document.all.WebBrowser.Execwb(4,1)：表示保存网页。
- document.all.WebBrowser.Execwb(0,1)：表示查看页面属性。
- document.all.WebBrowser.Execwb(15,1)：表示撤销。
- document.all.WebBrowser.Execwb(17,1)：表示全选。
- document.all.WebBrowser.Execwb(22,1)：表示刷新。
- document.all.WebBrowser.Execwb(45,1)：表示关闭窗体无提示。

利用 WebBrowser 实现打印的步骤如下。

（1）建立 HTML 的 Object 标签，调用 WebBrowser 控件，实现的代码如下：

```
<object id="WebBrowser" classid="ClSID:8856F961-340A-11D0-A96B-00C04Fd705A2" width="0" height="0">
```

（2）建立相关的打印超级链接，并调用 WebBrowser 控件的相应参数实现“打印预览”、“打印”等功能。实现的代码如下：

```
<a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a> <a href="#"
onClick="document.all.WebBrowser.Execwb(6,1)">打印</a> <a href="#"
onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a> <a href="#"
onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a>
```

26.2.3 使用 JavaScript 脚本清空页眉、页脚和恢复页眉、页脚

在万能打印模块中，无论是利用 IE 自身的打印功能进行打印，还是利用 IE 自带的 WebBrowser 控件进行打印，在打印的文稿顶部和底部都会包括页眉、页脚，有时并不需要打印默认的页眉和页脚。下面将介绍在万能打印模块中如何清空页眉、页脚和恢复页眉、页脚。清空页眉、页脚和恢复页眉、页脚是由“PageSetUp.aspx”页来实现的，运行该页，单击页面中设置的“清空页眉页脚”超级链接，即可清空 IE 默认的页眉页脚，这时再单击“打印预览”超级链接，在打开的“打印预览”窗口中将不显示 IE 默认的页眉、页脚，如图 26.4 所示，另外单击“恢复页眉页脚”超级链接还可恢复页眉、页脚的显示。

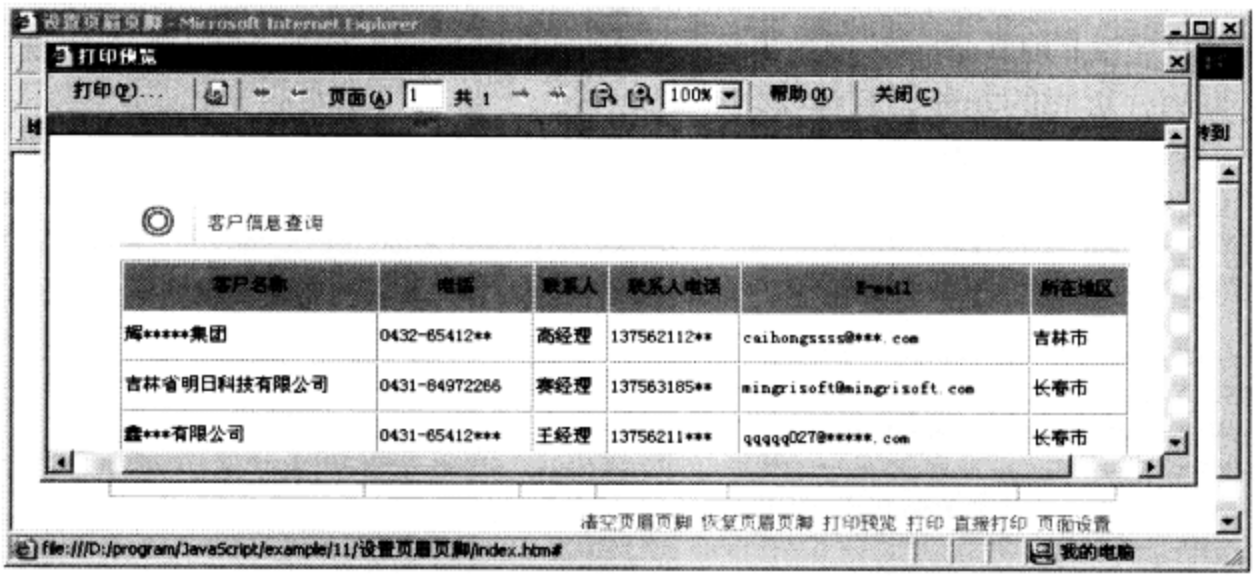


图 26.4　设置页眉、页脚

在实现清空页眉、页脚和恢复页眉、页脚主要通过 WshShell 对象的相关方法实现。WshShell 对象是 WSH（Windows Script Host，内嵌于 Windows 操作系统中的脚本语言工作环境）的内建对象，主要负责程序的本地运行、处理注册表、创建快捷方式、获取系统文件夹信息及处理环境变量等工作。WshShell 对象的相关方法如表 26.1 所示。

表 26.1　　WshShell 对象的相关方法

方　法	说　明
CreateShortcut	创建并返回 WshShortcut 对象
ExpandEnvironmentStrings	扩展 PROCESS 环境变量并返回结果字符串
Popup	显示包含指定信息的消息窗口
RegDelete	从注册表中删除指定的键或值
RegRead	从注册表中返回指定的键或值
RegWrite	在注册表中设置指定的键或值
Run	创建新的进程，该进程用指定的窗口样式执行指定的命令

在清空页眉、页脚和恢复页眉、页脚页面中主要应用了 RegWrite 方法，下面将对该方法进行详细的介绍。

RegWrite 方法用于在注册表中设置指定的键或值，其语法格式如下：

WshShell.RegWrite strName,anyValue,[strType]

参数说明如下。

- strName：用于指定注册表的键或值，若 strName 以一个反斜杠（在 JavaScript 中为\\）结束，则该方法设置键，否则设置值。StrName 参数必须以根键名“HKEY_CURRENT_USER”、“HKEY_LOCAL_MACHINE”、“HKEY_CLASSES_ROOT”、“HKEY_USERS”或“HKEY_CURRENT_CONFIG”开头。
- AnyValue：用于指定注册表的键或值。当 strType 为 REG_SZ 或 REG_EXPAND_SZ 时，RegWrite 方法自动将 anyValue 转换为字符串。若 strType 为 REG_DWORD，则 anyValue 被转换为整数。若 strType 为 REG_BINARY，则 anyValue 必须是一个整数。
- StrType：用于指定注册表的键或值的数据类型。RegWrite 方法支持的数据类型为 REG_SZ、REG_EXPAND_SZ、REG_DWORD 和 REG_BINARY。其他的数据类型被作为 strType 传递，RegWrite 返回 E_INVALLDARG。

实现清空页眉、页脚和恢复页眉、页脚实现的步骤如下。

（1）编写自定义的 JavaScript 函数 PageSetup_del()和 PageSetup_set()，用于实现清空页眉、页脚和恢复页眉、页脚的功能，具体实现代码如下。

例程 1 代码位置：光盘\mr\26\printWeb\PageSetup.aspx

```
<script  language="JavaScript">
var HKEY_RootPath="HKEY_CURRENT_USER\\Software\\Microsoft\\Internet Explorer\\PageSetup\\";
function PageSetup_del(){ //清空页眉页脚
  try{
      var WSc=new ActiveXObject("WScript.Shell");
      HKEY_Key="header";
      WSc.RegWrite(HKEY_RootPath+HKEY_Key,"");
      HKEY_Key="footer";
      WSc.RegWrite(HKEY_RootPath+HKEY_Key,"");
  }catch(e){}
}
function  PageSetup_set(){    //恢复页眉页脚
  try{
      var WSc=new ActiveXObject("WScript.Shell");
      HKEY_Key="header";
      WSc.RegWrite(HKEY_RootPath+HKEY_Key,"&w&b页码,&p/&P");
      HKEY_Key="footer";
      WSc.RegWrite(HKEY_RootPath+HKEY_Key,"&u&b&d");
  }catch(e){}
}
</script>
```

（2）建立 HTML 的 Object 标签，调用 WebBrowser 控件，实现的代码如下。

例程 2 代码位置：光盘\mr/26\print Web\Page Setup. aspx

```
<object id="WebBrowser" classid="ClSID:8856F961-340A-11D0-A96B-00C04Fd705A2" width="0" height="0">
</object>
```

（3）创建“清空页眉页脚”和“恢复页眉页脚”的超级链接，并调用自定义函数 PageSetup_del()和 PageSetup_set()实现相应功能。实现的代码如下。

例程 3 代码位置：光盘\mr\26\print Web\PageSetup. aspx

```
<a href="#" onClick="PageSetup_del()">清空页眉页脚</a><a href="#" onClick="PageSetup_set()"> 恢复页眉页脚</a>
```

（4）建立相关的打印超级链接，并调用 WebBrowser 控件的相应参数实现“打印预览”、“打印”等功能。实现的代码如下。

例程 4　代码位置：光盘\mr\26\print Web\PageSetup. aspx

```
<a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a> <a href="#"
onClick="document.all.WebBrowser.Execwb(6,1)">打印</a> <a href="#"
onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a> <a href="#"
onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a>
```

26.2.4　调用 IE 自身的打印功能实现打印

通过 JavaScript 调用 IE 自身的打印功能实现打印，这种方法比较简单，也是常用的打印方式。使用该方法只需将要打印的页面设计好，再通过 JavaScript 的 window 对象的 print 方法调用 IE 的打印功能即可。运行本实例，单击“打印”超级链接，会弹出“打印”对话框，如图 26.5 所示，然后进行相应的设置，并进行打印。

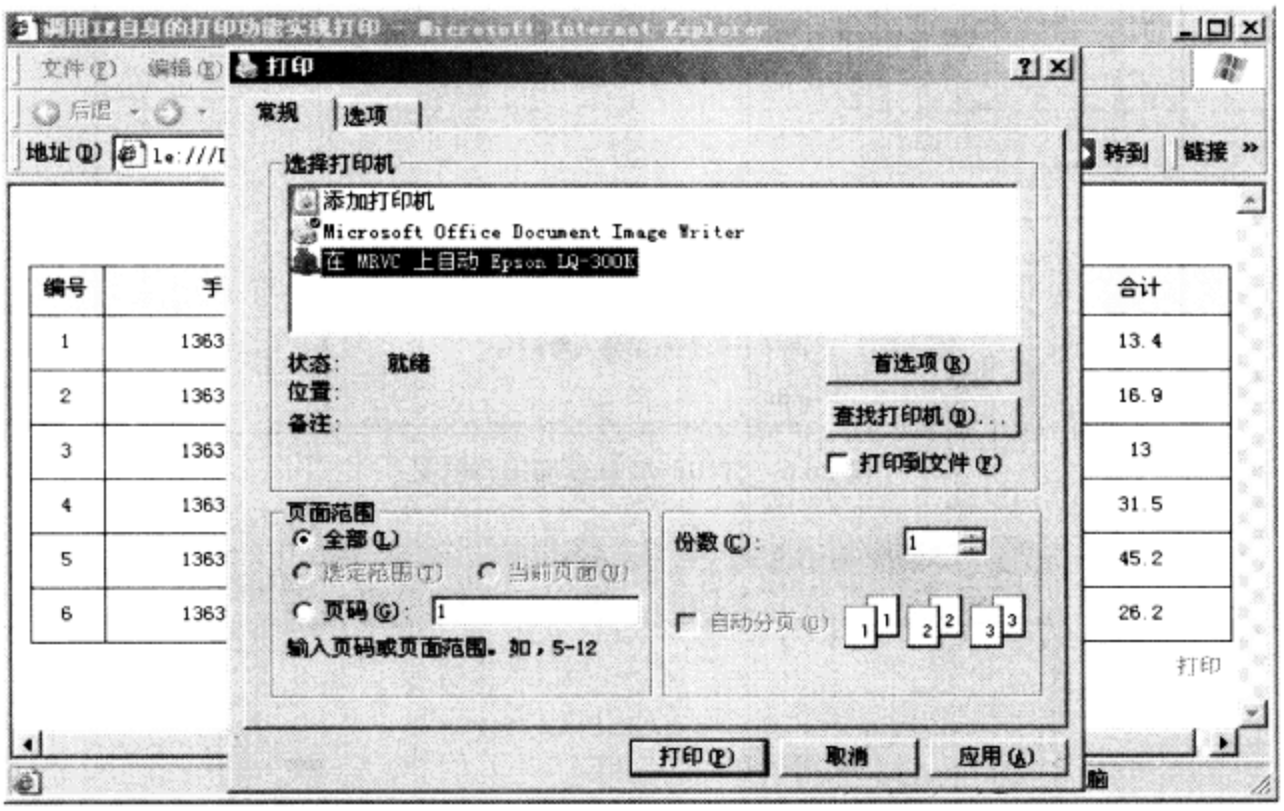


图 26.5　调用 IE 自身的打印功能实现打印

window 对象的 print()方法的语法格式如下。

```
window.print();
```

例如：

```
<a href="#" onClick="window.print()">打印</a>
```

调用 IE 自身的打印功能实现打印的关键步骤如下。

（1）设计要打印的页面。

（2）调用 window 对象的打印方法实现打印功能，代码如下：

```
<a href="#" onClick="window.print()">打印</a>
```

26.3　万能打印模块实现过程

26.3.1　套打邮寄产品单（打印汇款单）

1. 功能概述

在程序中加入打印邮寄产品单的功能，不但可以使用户非常方便地操作程序、提高工作效率，而且更能够使程序适应人性化的潮流。

在开发办公自动化等系统时，可能会遇到汇款单打印的情况，这时就需要在系统中加入该

功能。运行程序，在页面中将显示要打印的汇款单及与打印相关的超级链接，如图 26.6 所示，单击“打印”超级链接，可以打印该汇款单，单击“打印预览”超级链接，可以预览打印结果，如图 26.7 所示。

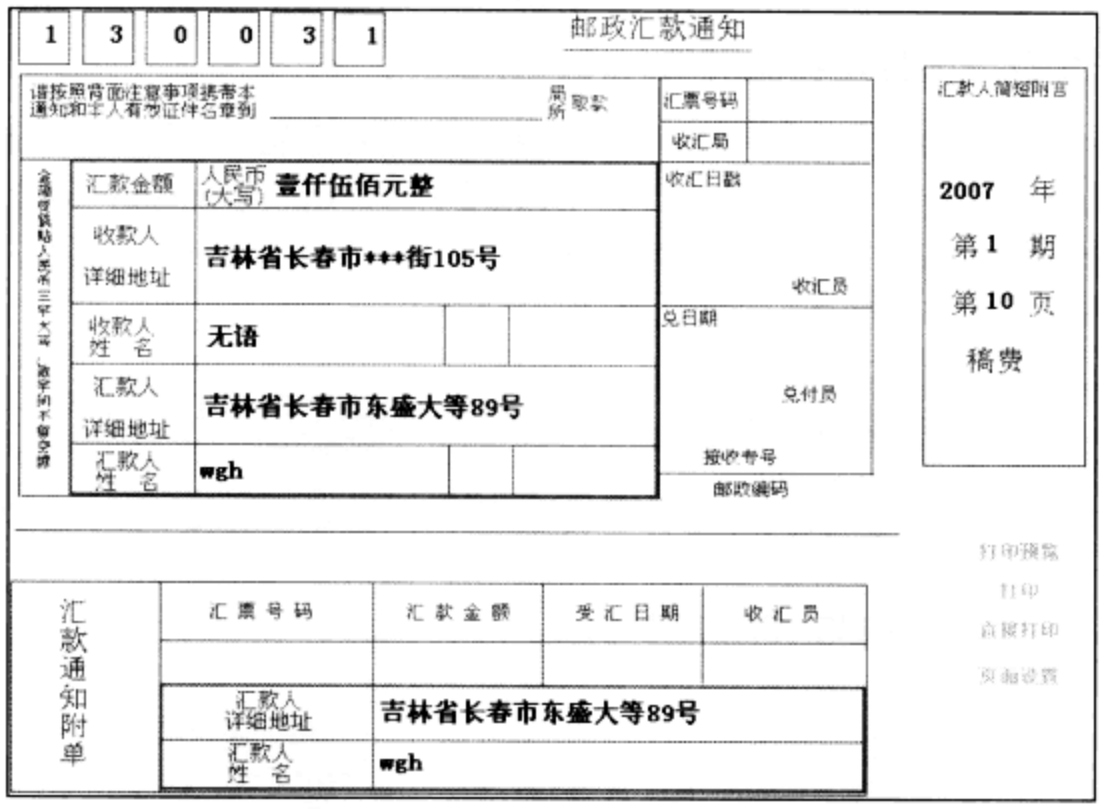


图 26.6　打印汇款单页面运行结果

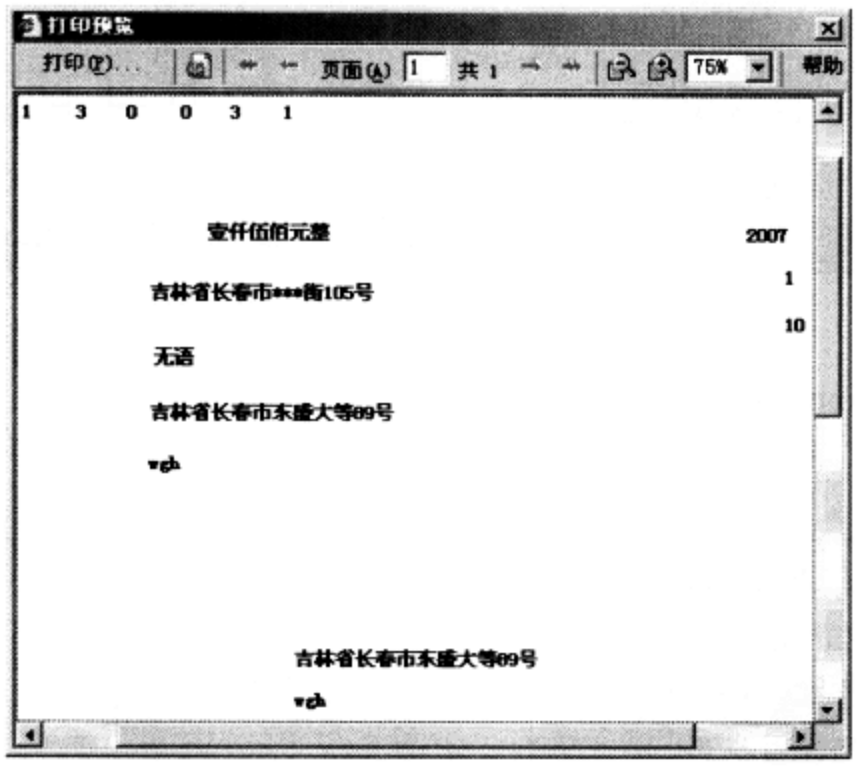


图 26.7　打印预览效果

注　意

套打印中模板是一样的，只是根据实际需要，修改套打印模板中的内容。

2. 实现过程

（1）在页面中插入一个表格，将该表格的背景设置为空的汇款单图片，并在表格中插入新的表格用于在指定位置显示汇款信息。

（2）在页面的指定位置填写汇款信息。

（3）在页面的相应位置加入“打印预览”、“打印”、“直接打印”、“页面设置”等超级链接。

关键代码如下。

例程 5　代码位置：光盘\mr\26\print Web\huikuan.aspx

```
<div>
<table width="81" height="111" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a></td>
  </tr>
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(6,1)">打印</a></td>
  </tr>
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a></td>
  </tr>
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a> </td>
  </tr>
</table>
</div>
```

（4）应用 CSS 样式控制表格背景及打印等超级链接在打印时不显示，关键代码如下。

例程 6　代码位置：光盘\mr\26\print Web\huikuan.aspx

```
<style>
@media print{
div{display:none}
td,table{
      background:display:none;
}
}
</style>
```

26.3.2　利用 CSS 样式分页打印

1. 功能概述

在制作数据打印程序时，对于多页数据（指的是一页纸不能全部打印完毕的数据）通常采用分页打印。这里的分页打印是指在每一页数据的顶端都打印表头信息。下面将通过具体实例介绍如何利用 CSS 样式实现分页打印。运行 Css.apsx 页，实现的效果如图 26.8 所示，单击页面中的“打印预览”超级链接，可以在打印前事先查看打印效果，如图 26.9 所示，然后单击“打印”超级链接即可进行分页打印。

客户信息查询

客户名称	电话	联系人	联系人电话	E-mail	所在地区
采虹**集团	0432-65412**	高经理	137562112**	caihongssss@***.com	吉林市
吉林省明日科技有限公司	0431-84972266	赛经理	137563185**	mingrisoft@mingrisoft.com	长春市
鑫***有限公司	0431-65412***	王经理	13756211***	qqqqq027@*****.com	长春市
东西南北***通讯公司	0434-56167***	李经理	1375631***4	mingrisoft@sina.com	四平市
明*有限责任公司	0431-849722**	张经理	1305531***6	mingxing@ming**.com	长春市

打印预览 打印

图 26.8　利用 CSS 样式分页打印

2. 实现过程

（1）在要打印的页面中添加用于显示客户信息的表格，并设置好表头、表尾及打印分页，关键代码如下。

图 26.9　打印预览效果图

例程 7　代码位置：光盘\mr\26\printWeb\Css.aspx

```
<table width="650" border="1" cellpadding="0" align="center" cellspacing="0"
bgcolor="#FE7529" id="pay" bordercolor="#FE7529" bordercolordark="#FE7529"
bordercolorlight="#FFFFFF" style="border-bottom-style:none;">
<!--设置表头-->
      <thead style="display:table-header-group;font-weight:bold">
  <tr align="center" bgcolor="#FE7529">
    <td width="155" height="30">客户名称</td>
    <td width="99">电话</td>
    <td width="59" >联系人</td>
    <td width="84">联系人电话</td>
    <td width="175">E-mail</td>
    <td width="64">所在地区</td>
  </tr>
  </thead>
  <tr>
    <td height="30" bgcolor="#FFFFFF">采虹**集团</td>
    <td width="99">电话</td>
    <td width="59" >联系人</td>
    <td width="84">联系人电话</td>
    <td width="175">E-mail</td>
    <td width="64">所在地区</td>
  </tr>
  <tr>
    <td height="30" bgcolor="#FFFFFF">吉林省明日科技有限公司</td>
    ……        //此片省略了显示客户其他信息的HTML代码
  </tr>
  <tr>
<!--控制分页-->
    <td height="30" bgcolor="#FFFFFF" style="page-break-after:always">鑫***有限公司</td>
        ……        //此片省略了显示客户其他信息的HTML代码
  </tr>
  <tr>
    <td height="30" bgcolor="#FFFFFF">东西南北***通讯公司</td>
    ……        //此片省略了显示客户其他信息的HTML代码
  </tr>
  <tr>
    <td height="30" bgcolor="#FFFFFF">明*有限责任公司</td>
```

```
      ……      //此片省略了显示客户其他信息的HTML代码
    </tr>
<!--设置表尾-->
<tfoot style="display:table-footer-group; border:none;"><tr><td></td></tr></tfoot>
</table>
```

(2) 控制“打印”和“打印预览”超级链接，在打印时不显示。关键代码如下。

例程 8 代码位置：光盘\mr\26\printWeb\Css.aspx

```
<style>
@media print{
.noprint{display:none}
}
</style>
  <table width="647" align="center" class="noprint">
       <tr align="center" bgcolor="#FFFFFF">
     <td height="27" colspan="3" align="right">
      <a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a>
      <a href="#" onClick="document.all.WebBrowser.Execwb(6,1)">打印</a>
      <a href="#" onClick="document.all.WebBrowser.Execwb(8,1)"></a> </td>
  </tr>
</table>
```

26.3.3 利用 Excel 打印报表

1. 功能概述

Microsoft Excel 是微软公司提供的用于办公管理的应用软件，它具有强大的报表统计等功能，该功能在“ExcelPrint.aspx”页面实现，通过将数据导入到 Excel 文件中进行打印。执行“ExcelPrint.aspx”页面，当用户单击“输出 Excel 报表”按钮时，打开“文件下载”对话框，如图 26.10 所示，单击对话框中的“打开”按钮，Web 页面中的数据便以 Excel 文件方式打开，如图 26.11 所示，用户便可以用 Excel 自带的打印功能对数据信息进行打印。

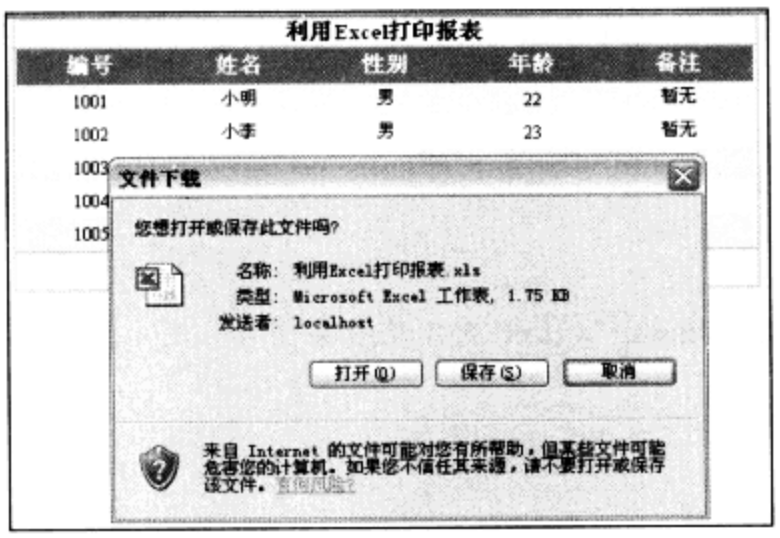


图 26.10 利用 Excel 打印报表

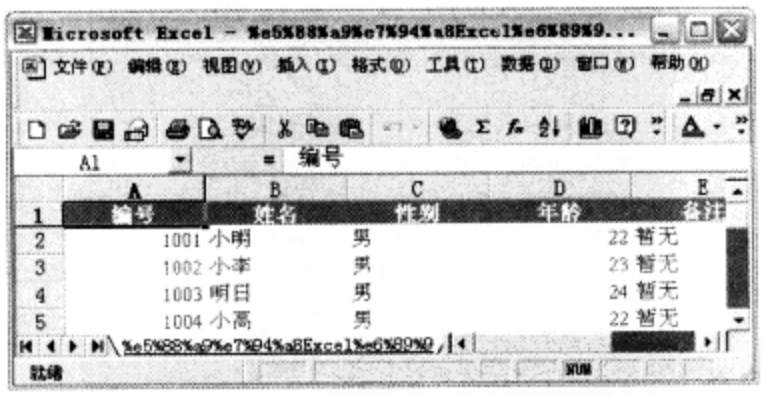


图 26.11 输出的 Excel 报表

2. 实现过程

(1) 创建 1 个 Web 窗体，命名为 printExcel.aspx，用来实现利用 Excel 打印报表功能。

(2) 在 printExcel.aspx 页面中添加 1 个 Table 表格，用于布局页面，然后在该 Table 表格中分别添加 1 个 GridView 控件和 1 个 Button 控件，其中，GridView 控件用来显示数据表中的信息，Button 控件用来执行将 GridView 控件中的数据导出为 Excel 格式操作。

在编写 Page_Load 事件代码前，首先需要引入 3 个必要的命名空间，引用如下：

```
using System.Data.SqlClient;
using System.IO;
using System.Text;
```

在 Page_Load 事件中首先判断页面是否是首次加载，然后调用用户自定义 bind 方法，实现将数据库中数据绑定到 GridView 列表控件中，实现的代码如下。

例程 9 代码位置：光盘\mr\26\printWeb\ExcelPrint.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
 if (!IsPostBack)
  {
      bind();//调用用户自定义的bind方法
  }
}
```

用户自定义的 bind 方法主要用来实现将数据绑定到 GridView 控件中，在该方法中首先创建一个数据阅读器对象，接着创建一个 DataSet 对象，然后应用数据阅读器对象中的 Fill 方法填充创建的数据集 myds，最后绑定列表控件中的数据，实现的代码如下。

例程 10 代码位置：光盘\mr\26\printWeb\ExcelPrint.aspx.cs

```
 private void bind()
 {
    SqlDataAdapter myda = new SqlDataAdapter("select * from tb_Student", sqlcon);
    DataSet myds = new DataSet();
    sqlcon.Open();
    myda.Fill(myds);
    sqlcon.Close();
    GridView1.DataSource = myds;
    GridView1.DataBind();
 }
```

双击页面中的“输出 Excel 报表”按钮，触发其 Button1_Click 事件，实现的代码如下。

例程 11 代码位置：光盘\mr\26\printWeb\ExcelPrint.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
    Export("application/ms-excel", "利用Excel打印报表.xls");
}
```

上述 Button1_Click 事件代码中，调用了用户自定义的 Export 方法，实现将 GridView 控件中的数据导出到 Excel 中，实现的代码如下。

例程 12 代码位置：光盘\mr\26\printWeb\ExcelPrint.aspx.cs

```
private void Export(string FileType, string FileName)
{
    Response.Charset = "GB2312";
    Response.ContentEncoding = System.Text.Encoding.UTF7;
    Response.AppendHeader("Content-Disposition", "attachment;filename=" + HttpUtility.UrlEncode
    (FileName, Encoding.UTF8). ToString());
    Response.ContentType = FileType;
    this.EnableViewState = false;
    StringWriter tw = new StringWriter();
    HtmlTextWriter hw = new HtmlTextWriter(tw);
    GridView1.RenderControl(hw);
    Response.Write(tw.ToString());
    Response.End();
}
```

这里主要使用了 Response 对象中的 Write()方法输出打印 Excel 报表信息。

26.3.4 打印快递单

1. 功能概述

有时在开发物流网站时，经常会遇到打印快递单的情况。本实例将介绍如何实现打印快递单。运行本实例，在页面中将显示要打印的快递单及与打印相关的超级链接，如图 26.12 所示。

单击“打印”超级链接，可以打印该快递单，单击“打印预览”超级链接，可以预览打印结果，如图 26.13 所示。

寄件地	2007 年 2 月 12 日 9 时											收件地											
寄件站						帐号						派件站						帐号					
寄件客户/地址： 长春市东盛大街89号 电话：0431-84972266 联络人：wgh												收件客户/地址： 四平市*****街86号 电话： 0434-5597*** 联络人： 无语											
品名： 图书					非货样 货 样		付款地点： 寄件单位 收件单位		付款方式： 付清 到付 月结		应收金额： 500元		寄件人的声明及签名：我/我们同意本运单背面的契约条款 寄件人签名：										
数量： 5	件 箱		重量： 4千克			价值： 600元																	
以材料计费： A 快递、空运(长 cm×宽 cm×高 cm) ÷6,000 = kgs B 卡车、火车(长 cm×宽 cm×高 cm) ÷3,000= kgs 其他费用备考																			取件员			客服员	
																			收件客户签名： 月 日 时				

打印预览 打印 直接打印 页面设置

图 26.12 打印快递单页面运行结果

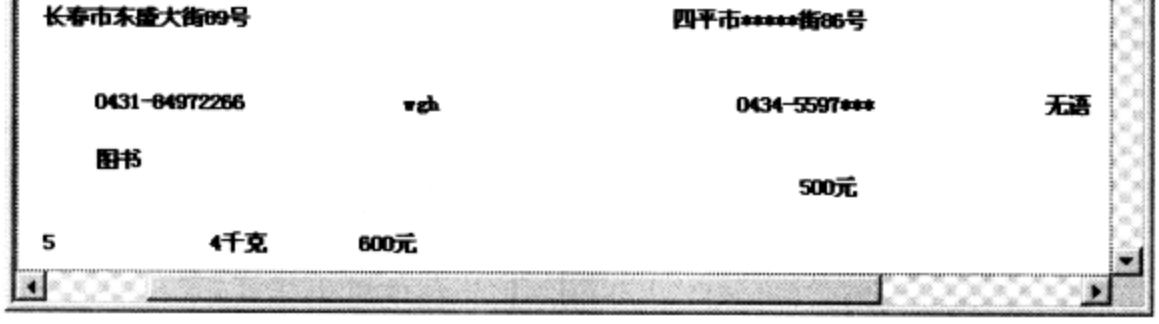


图 26.13 打印预览效果

2. 实现过程

(1) 在页面中插入一个表格，将该表格的背景设置为空的快递单图片，并在表格中插入新的表格用于在指定位置显示快递信息。

(2) 在页面的指定位置填写快递信息。

(3) 在页面的相应位置加入“打印预览”、“打印”、“直接打印”、“页面设置”等超级链接。具体实现的代码如下。

例程 13 代码位置：光盘\mr\26\printWeb\huikuan.aspx

```
<div>
<table width="81" height="111" border="0" align="center" cellpadding="0" cellspacing="0">
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a></td>
  </tr>
  <tr align="center" bgcolor="#FFFFFF">
    <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(6,1)">打印</a></td>
  </tr>
```

```
    <tr align="center" bgcolor="#FFFFFF">
      <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a></td>
    </tr>
    <tr align="center" bgcolor="#FFFFFF">
      <td colspan="3"><a href="#" onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a> </td>
    </tr>
  </table>
  </div>
```

（4）应用 CSS 样式控制表格背景及打印等超级链接在打印时不显示，具体实现的代码如下。

例程 14 代码位置：光盘\mr\26\printWeb\huikuan.aspx

```
<style>
@media print{
div{display:none}
td,table{
      background:display:none;
}
}
</style>
```

26.3.5 打印信封

1. 功能概述

在报社或电台日常业务中，最频繁的工作就是给客户或听众发信，如果每一封信都手工填写，不仅会降低工作效率，而且容易出现错误，为了解决该问题，可以在网页中加入打印信封的功能。运行本实例，在页面中将显示要打印的信封及与打印相关的超级链接，如图 26.14 所示，单击“打印”超级链接，可以在信封的指定位置打印相关内容，单击“打印预览”超级链接，可以预览打印结果，如图 26.15 所示。

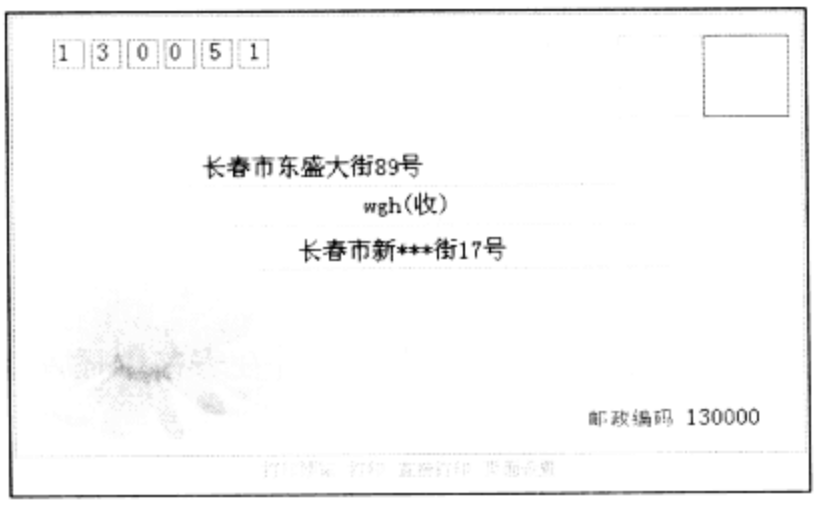


图 26.14 打印信封页面运行结果

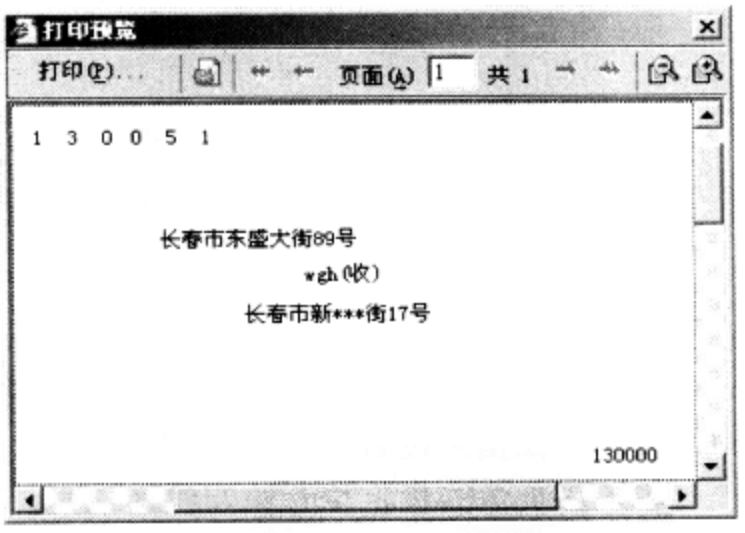


图 26.15 页面预览效果

2. 实现过程

（1）在页面中插入一个表格，该表格的尺寸和一个标准的信封大小相同，在表格中插入新的表格用于在指定位置显示收信人和发信人信息。

（2）在页面的指定位置填写收信人和发信人信息。

（3）在页面的底部加入“打印预览”、“打印”、“直接打印”、“页面设置”等超级链接。实现的代码如下。

例程 15 代码位置：光盘\mr\26\printWeb\envelop.aspx

```
<div align="center">
  <table width="542" border="0" align="center" cellpadding="0" cellspacing="0">
    <tr>
      <td><div align="center"><a href="#" onClick="document.all.WebBrowser.Execwb(7,1)">打印预览</a>
          <a href="#" onClick="document.all.WebBrowser.Execwb(6,1)">打印</a>
          <a href="#" onClick="document.all.WebBrowser.Execwb(6,6)">直接打印</a>
        <a href="#" onClick="document.all.WebBrowser.Execwb(8,1)">页面设置</a></div></td>
    </tr>
  </table>
</div>
```

（4）应用 CSS 样式控制表格背景及打印等超级链接在打印时不显示，实现的具体代码如下。

例程 16 代码位置：光盘\mr\26\printWeb\envelop.aspx

```
<style>
@media print{
div{display:none}
td,table{
    background:display:none;
}
}
</style>
```

26.4 程序调试与错误处理

在对程序进行调试时，可能出现如图 26.16 所示的错误。

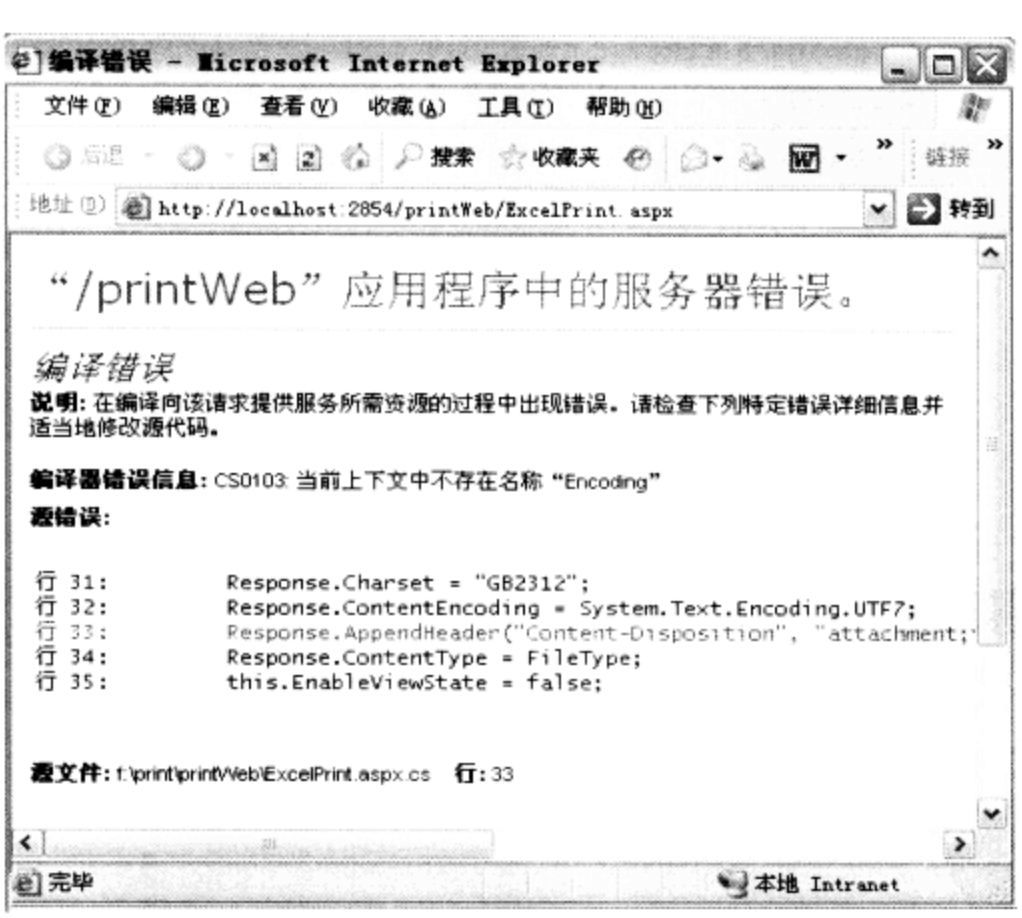


图 26.16 ASP.NET 版本错误的出错页

原因就是在编写程序时没有引入相应的命名空间，使 ASP.NET 程序无法在 IIS 中正常运行，将会出现如图 26.16 所示错误下面将详细介绍解决这一问题的方法。

解决这一问题非常简单，只需将相应的命名空间写到命名空间处即可，在万能打印模块中

的 ExcelPrint.aspx.cs 页面中用到的两个命名空间的代码如下：

```
//引入命名空间
using System.IO;
using System.Text;
```

这时命名空间已经完成，然后测试一下程序在 IIS 中是否可用，运行该程序，程序运行的效果如图 26.17 所示。

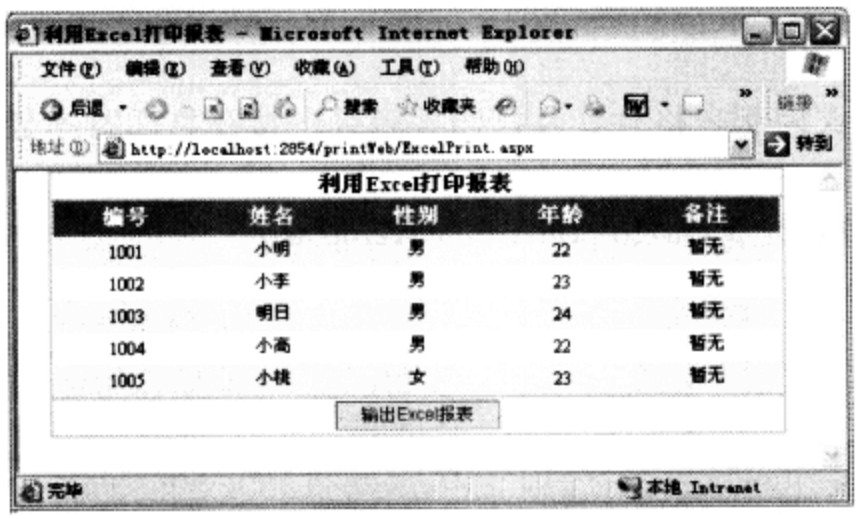


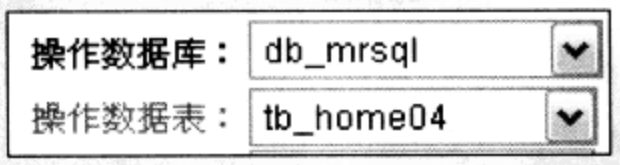
图 26.17　页面运行效果图

第 27 章 数据备份与恢复模块

实例位置：光盘\mr\27\

随着数字化时代的到来，数据这一名词在生活中越来越有分量，数据丢失问题也越来越令人头痛，为了避免由于意外而引发的数据丢失，常常需要对数据进行备份，以便在日后能够对相关数据进行恢复，减少由于意外而造成的损失。本模块利用 ASP.NET 实现对 SQL Server 2000 数据库和 Access 数据库中的数据进行数据备份与恢复操作等。通过本章，读者可以学习到以下内容。

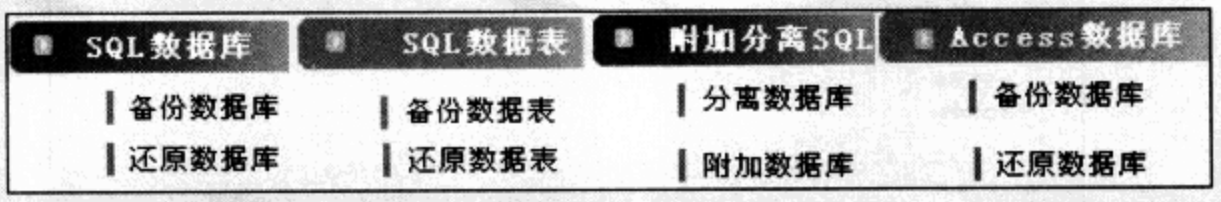
级联更新数据表



浏览选择要还原的数据库

智能生成数据库备份名称

网站导航



27.1 数据备份与恢复功能概述

为了避免因某些原因使数据库中的数据遭到破坏或删除带来的极大损失，数据库的备份对大型的网站来说是至关重要的。本模块将详细介绍如何在 ASP.NET 中实现对 SQL Server 2000 和 Access 这两种比较流行的数据库进行数据库备份和恢复操作。

本模块的首页中首先通过登录页面进入到对 Sql Server 2000 数据库进行备份页中，在该页中通过网站左侧的导航功能，可分别进入到“备份 SQL 数据库”、“还原 SQL 数据库”、“备份 SQL 数据表”、“还原 SQL 数据表”、“分离 SQL 数据库”、“附加 SQL 数据库”、“备份 Access 数据库”和“还原 Access 数据库”的详细页面中。数据自动备份与恢复模块首页运行结果如图 27.1 所示。

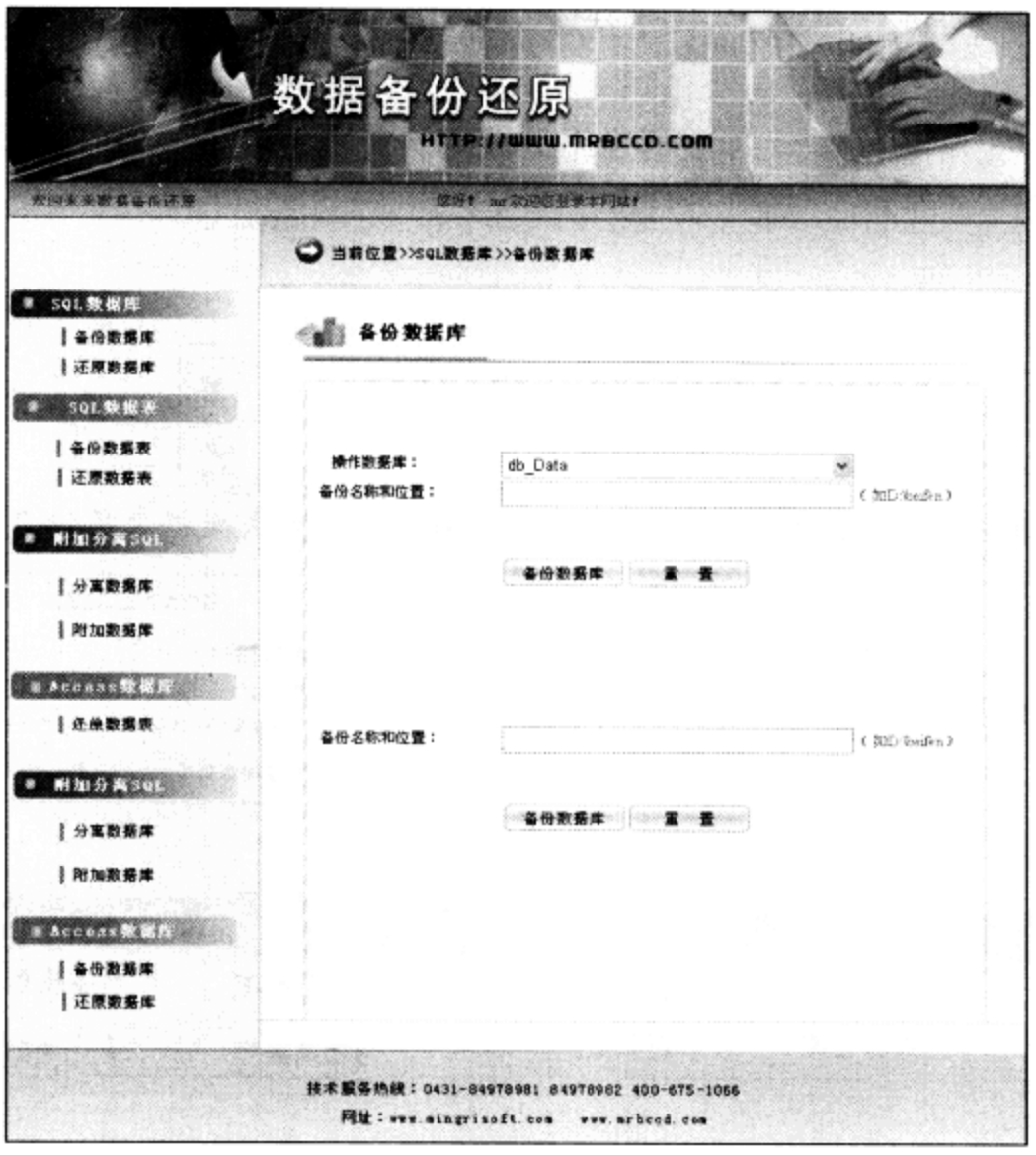


图 27.1 数据备份与恢复模块运行效果图

为了方便读者阅读和有效地利用本书附赠光盘中的实例，这里将网站页面即图 27.1 中标记的各部分说明以列表形式给出，如表 27.1 所示。

表 27.1 网站首页解析

区 域	名 称	说 明	对 应 文 件
1	网站头	主要用于网站的旗帜广告	BeiFenKSql.aspx
2	显示当前用户名信息	主要用于将用户名等信息	BeiFenKSql.aspx
3	显示当前位置	主要用户显示当前位置	BeiFenKSql.aspx
4	显示当航区	主要用于显示当航信息	BeiFenKSql.aspx
5	备份数据库操作区	主要用于执行备份数据库操作	BeiFenKSql.aspx
6	网站脚标	用于显示客服的电话等信息内容	BeiFenKSql.aspx

27.2 数据备份与恢复关键技术

27.2.1 数据库备份技术

数据库的备份为了保证用户数据的安全性，避免计算机在操作过程中出现意外，以及种种原因而引发的数据的丢失。

数据库的备份主要是对 SQL Server 数据库或事务日志进行拷贝，数据库备份记录了在进行备份这一操作时数据库中所有数据的状态。在 SQL Server 中提供了 BACKUP DATABASE 语句用于实现备份数据库。数据库的备份主要是对数据库的完全备份，包括所有的数据以及数据库对象。

由于是对数据库的完全备份，所以这种备份类型不仅速度较慢，而且将占用大量磁盘空间，正因为如此，在进行数据库备份时常将其安排在晚间，因为此时整个数据库系统几乎不进行其他事务操作，从而可以提高数据库备份的速度。

在 SQL Server 中数据库备份有 4 种类型。

（1）数据库备份（Database Backups）。

数据库备份是指对数据库的完整备份，包括所有的数据以及数据库的对象。但它需要花费更多的时间和空间。

（2）事务日志备份（Transaction Log Backup）。

事务日志备份是指对数据库发生的事务进行备份，包括从上次进行事务日志备份、差异备份和数据库备份之后所有已经完成的事务。它的特点是花费时间少、速度快。

（3）差异备份（Differential Database Backups）。

差异备份是指将最近一次数据库备份以来发生的数据变化备份起来，因此差异备份实际上是一种增量数据库备份。它的优点是存储和恢复速度快。推荐每天做一次差异备份。

（4）文件和文件组备份（File and File Group Backup）。

文件或文件组备份是指对数据库文件或文件夹进行备份，但其不像完整的数据库备份那样，同时也进行事务日志备份。数据库一般是由硬盘上的许多文件构成。如果这个数据库非常大，并且一个晚上也不能将它备份完，那么就可以使用文件和文件组备份，每个晚上备份数据库的一部分。一般情况下数据库不会大到必须使用多个文件存储，所以此备份并不常用。

在对数据库进行完全备份时，所有未完成的事务或者发生在备份过程中的事务，都不会被备份。通常情况下，可以一边进行备份一边进行其他操作，但是，在备份过程中不允许执行以下操作：

- 创建或删除数据库文件；
- 创建索引；
- 执行非日志操作；
- 自动或手工缩小数据库或数据库文件大小。

本模块在数据备份方面应用了两个技术要点：应用 SQL 存储过程查找企业管理器中所有数据库和 bcp 实用工具。

1. 应用 SQL 存储过程查找企业管理器中所有数据库

为了使软件更加智能化，利用 SQL 中自带的存储过程查找企业管理器中所有数据表，并将数据填充到下拉列表框中，在本实例中主要用到的存储过程是 sp_helpdb。下面介绍 sp_helpdb

的语法结构和用法。

Sp_helpdb：报告有关指定数据库或所有数据库的信息。

语法格式如下：

```
sp_helpdb[[@dbname=]'name']
```

[@dbname=]'name'：是要为其提供信息的数据库名称。Name 的数据类型 sysname，无默认值。如果没有指定 name，则 sp_helpdb 报告 master.dbo.sysdatabases 中的所有数据库。

- 权限：执行权限默认授予 public 角色。

Sp_helpdb 必须访问服务器上的数据库以确定要显示的有关数据库的信息。因此，服务器上的每个数据库都必须满足下列条件之一。

- 指定 sp_helpdb 的用户必须拥有访问数据库的权限。
- Guest 用户账户必须存在于数据库中。
- 如果无法访问数据库，那么 sp_helpdb 将显示错误信息 15622 和有关数据库信息。

例如：下面的示例将显示有关 master 数据库的信息，实现的代码如下：

```
Exec sp_helpdb master
```

例如：下面的示例显示运行在 SQL Server 服务器上的所有数据库的信息，实现的代码如下：

```
Exec sp_helpdb
```

2. bcp 实用工具

bcp 实用工具用于 SQL Server 2000 实例和数据文件之间指定的格式复制数据。以下介绍 bcp 实用工具中几个重要的参数说明。

In|out|queryout|format：指定大容量复制的方向。In 是从文件复制到数据库表或视图，out 是指从数据库表或视图复制到文件。

参数说明如下。

- -c：使用字符数据类型执行大容量复制操作。
- -q：在 bcp 实用工具和 SQL Server 实例的连接中执行 SET QUOTED_IDENTIFLERSON 语句。
- -S server_name[/instance_name]：指定要连接到的 SQL Server 实例。
- -U login_id：指定用于连接到的 SQL Server 实例。
- -P password：指定登录 ID 的密码。

以下代码为本实例即备份数据表实例中应用的代码，实现的代码如下：

```
cmdtxt2 += " out "+this.txtPath.Text.Trim()+".xls -c -q -S. -Usa -P"+this.txtPwd.Text.Trim()+"";
```

上述语句主要应用了 bcp 实用工具中几个重要的参数，读者可根据以上内容的介绍来进一步理解该代码的意义。

27.2.2 数据库恢复技术

对于一个大型网站来说，具有数据恢复功能尤为重要。因为数据恢复功能可以在数据遭到破坏时，将数据恢复到系统中，保证系统重新正常运转，从而避免因数据遭到异常丢失所带来的损失。

恢复数据库是使用数据库的备份文件对数据库进行恢复操作。由于病毒的破坏、磁盘损坏或操作员操作失误等原因会导致数据丢失、不完整或数据错误，此时，需要对数据库进行恢复。

在实际应用中选用怎样的数据库备份方案将直接对数据库的恢复产生影响，而且也决定了数据库在遭到破坏前后的一致性水平，所以在做出选用什么样的数据库备份方案时必须认识到以下几个问题。

（1）如果进行数据库的完全备份，那么在恢复数据库时将无法恢复自最近一次数据库备份

以来的数据库中所发生的所有事务，这种方案的优点是简单，而且在进行数据库恢复时操作也很方便。

（2）如果在进行数据库备份时也进行事务日志备份，可以将数据库恢复到失败点。

在 SQL Server 中数据库恢复有 3 种模式。

1. 简单恢复（SimpleRecovery）

所谓简单恢复就是指在进行数据库恢复时，仅用了数据库备份或差异备份，而不涉及事务日志备份。简单恢复模式可使数据库恢复到上一次备份的状态，但由于不使用事务日志备份来进行恢复，所以无法恢复到失败点状态。选择简单恢复模式时常使用的备份策略是首先进行数据库备份，然后进行差异备份。

2. 完全恢复（FullRecovery）

完全数据库恢复模式是指通过使用数据库备份和事务日志备份将数据库恢复到操作失败的时刻，几乎不造成任何数据丢失，这是解决因为存储介质损坏而数据丢失的最佳方法。为了保证数据库的这种恢复能力，所有的批数据操作都被写入日志文件。

3. 批日志恢复（Bulk-loggedRecovery）

批日志恢复在性能上要优于简单恢复和完全恢复模式，它能尽最大努力减少批操作所需要的存储空间。这些批操作主要是查询语句 SELECT INTO、批装载操作、创建索引和针对大文本或图像的操作。选择批日志恢复模式所采用的备份策略与完全恢复所采用的备份策略基本相同。

注意

恢复数据库备份将重新创建数据库和备份完成时数据库中存在的所有相关文件。但是，自创建备份后所做的任何数据库修改都将丢失。若要恢复创建数据库备份后所发生的事务，必须使用事务日志备份或差异备份。

恢复数据库的语法如下：

```
RESTORE DATABASE { database_name | @database_name_var }
[ FROM < backup_device > [ ,...n ] ]
[ WITH
    [ RESTRICTED_USER ]
    [ [ , ] FILE = { file_number | @file_number } ]
    [ [ , ] PASSWORD = { password | @password_variable } ]
    [ [ , ] MEDIANAME = { media_name | @media_name_variable } ]
    [ [ , ] MEDIAPASSWORD = { mediapassword | @mediapassword_variable } ]
    [ [ , ] MOVE 'logical_file_name' TO 'operating_system_file_name' ]
            [ ,...n ]
    [ [ , ] KEEP_REPLICATION ]
    [ [ , ] { NORECOVERY | RECOVERY | STANDBY = undo_file_name } ]
    [ [ , ] { NOREWIND | REWIND } ]
    [ [ , ] { NOUNLOAD | UNLOAD } ]
    [ [ , ] REPLACE ]
    [ [ , ] RESTART ]
    [ [ , ] STATS [ = percentage ] ]
]
```

主要参数选项的含义说明如下。

- DATABASE：指定从备份还原整个数据库。如果指定了文件和文件组列表，则只还原那些文件和文件组。
- {database_name | @database_name_var}：将日志或整个数据库还原到的数据库。如果将其作为变量(@database_name_var)提供，则可将该名称指定为字符串常量(@database_name_var = database name)或字符串数据类型（ntext 或 text 数据类型除外）的变量。
- FROM：指定从中还原备份的备份设备。如果没有指定 FROM 子句，则不会发生备份

还原，而是恢复数据库。可用省略 FROM 子句的办法尝试恢复通过 NORECOVERY 选项还原的数据库，或切换到一台备用服务器上。如果省略 FROM 子句，则必须指定 NORECOVERY、RECOVERY 或 STANDBY。

- < backup_device >：指定还原操作要使用的逻辑或物理备份设备。可以是下列一种或多种形式。
- {'logical_backup_device_name' | @logical_backup_device_name_var}：是由 sp_addumpdevice 创建的备份设备（数据库将从该备份设备还原）的逻辑名称，该名称必须符合标识符规则。如果作为变量(@logical_backup_device_name_var)提供，则可以指定字符串常量(@logical_backup_device_name_var = logical_backup_device_name)或字符串数据类型（ntext 或 text 数据类型除外）的变量作为备份设备名。
- {DISK|TAPE}='physical_backup_device_name' | @physical_backup_device_name_var：允许从命名磁盘或磁带设备还原备份。磁盘或磁带的设备类型应该用设备的真实名称（例如：完整的路径和文件名）来指定，如 DISK = 'C:\Program Files\Microsoft SQL Server\MSSQL\BACKUP\Mybackup.dat' 或 TAPE = '\\.\TAPE0'。如果指定为变量(@physical_backup_device_name_var)，则设备名称可以是字符串常量（@physical_backup_device_name_var = 'physical_backup_device_name')或字符串数据类型（ntext 或 text 数据类型除外）的变量。

27.2.3 实现将数据绑定到 DropDownList 控件中

在本数据备份与恢复模块中，数据库以及数据库中的相关数据表的信息都绑定到了 DropDownList 控件中。DropDownList 控件实际上是列表项的容器，使用 DropDownList 服务器控件，用户可以很方便地选择要备份或恢复的数据库或数据库中的表，如图 27.2 所示。

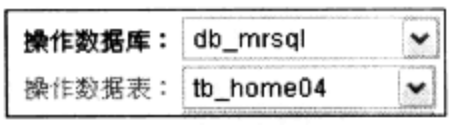


图 27.2 实现将数据绑定到 DropDownList 控件中

本模块用到的在 DropDownList 控件下绑定的 SQL Server 2000 数据库中所有数据库的名称及指定数据库下的所有数据表的名称的核心代码分别如下。

- 在 DropDownList 控件中绑定 SQL Server 2000 数据库中所有数据库的名称关键代码如下：

```
string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
string cmdtxt2 = "Exec sp_helpdb";
SqlConnection Con = new SqlConnection(cmdtxt1);
Con.Open();
SqlCommand mycommand = new SqlCommand(cmdtxt2, Con);
SqlDataReader dr = mycommand.ExecuteReader();
this.dropDatabase.DataSource = dr;
this.dropDatabase.DataTextField = "name";
this.dropDatabase.DataBind();
dr.Close();
```

- 在 DropDownList 控件中绑定 SQL Server 2000 数据库中指定的数据库下的所有数据表的关键代码如下：

```
string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
string cmdtxt2 = "Exec sp_helpdb";
SqlConnection Con = new SqlConnection(cmdtxt1);
Con.Open();
string cmdtxt3 = "use " + this.dropDatabase.SelectedValue + " SELECT * FROM " + this.dropDatabase.SelectedValue +
".dbo.sysobjects";
cmdtxt3 += " WHERE xtype='U' AND STATUS>=0";
SqlCommand mycommand1 = new SqlCommand(cmdtxt3, Con);
SqlDataReader dr1 = mycommand1.ExecuteReader();
this.dropTable.DataSource = dr1;
```

```
this.dropTable.DataTextField = "name";
this.dropTable.DataBind();
dr1.Close();
Con.Close();
```

本模块中主要使用 DropDownList 控件的“DataSource”和“DataTextField”两个属性，下面详细介绍一下这两个属性。

- DataSource 属性

通过 DataSource 属性可以从数组或集合中获取列表项，并将其添加到控件中。当编程人员希望从数组或集合中填充列表时，可以使用此属性。

语法格式如下：

```
public virtual Object DataSource { get; set; }
```

- DataTextField 属性

设置为列表项提供文本内容的数据源字段。

语法格式如下：

```
public virtual string DataTextField { get; set; }
```

- 属性值：指定为列表项提供文本内容的数据源字段，默认为空。

请读者参照下面的代码理解此属性的含义：

```
DropDownList1.DataTextField = "UserName";
```

注意

UserName 是数据库中的字段名。

在将数据绑定到 DropDownList 控件中还用到一个比较重要的 DataBind 方法。下面将详细介绍该方法。

- DataBind 方法用于 DropDownList 控件使用 DataSource 属性附加了数据源后，将数据源绑定到被调用的 DropDownList 控件。

语法格式如下：

```
public override void DataBind ()
```

27.3 数据备份与恢复实现过程

27.3.1 数据库的备份操作

数据库的备份主要由 BeiFenKSql.aspx 页面实现，在该页面中主要完成对 SQL Server 2000 数据库的备份操作，页面实际运行的效果如图 27.3 所示。

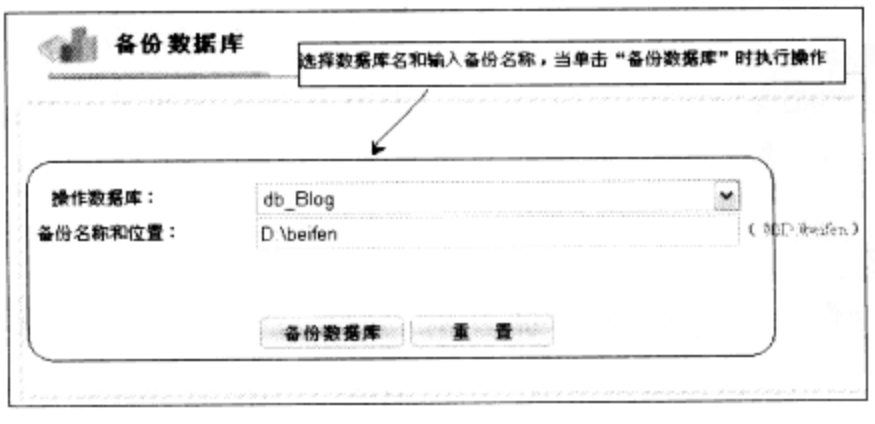


图 27.3 数据库的备份页运行效果图

1. 页面设计

(1) 在应用程序中创建一个 Web 窗体，默认名为“BeiFenKSql.aspx”，作为数据库的备份

页。页面 BeiFenKSql.aspx 的设计界面如图 27.4 所示。

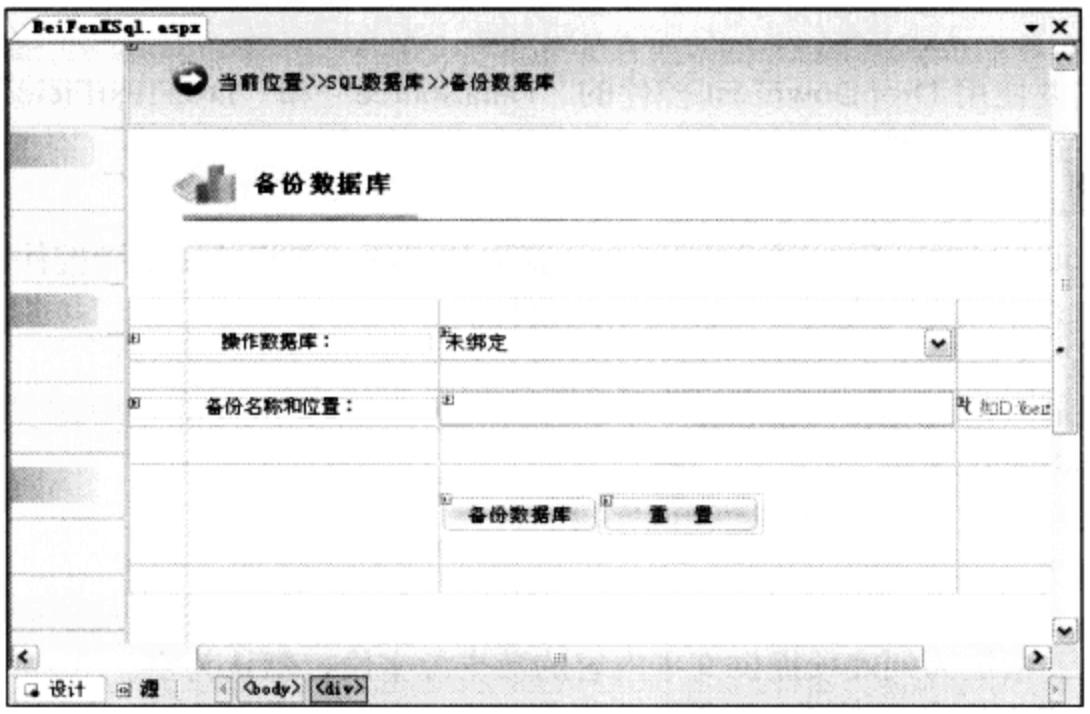


图 27.4　BeiFenKSql.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 DropDownList 控件和 1 个 ImageButton 控件放置在界面中，设置控件的属性。页面中各个控件的属性设置及其用途如表 27.2 所示。

表 27.2　　数据库备份页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
DropDownList	控件的名称为“DDLDataBase”	无	将 SQL Server 2000 中的数据库的名称绑定到该控件中
ImageButton	控件的名称为“ImgBtnOK”	将该控件的 ID 属性设置为 "ImgBtnOK"，将 ImageUrl 设置为 "～/image/btnbf.JPG" ，将 OnClick 属性设置为 "ImgBtnOK_Click"	执行备份数据库操作

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 1　代码位置：光盘\mr\27\Revert\BeiFenKSql.aspx.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在该页面的 Page_load 事件中，首先判断页面是否是首次加载，然后调用一个用户自定义的 binddropdownList 方法，目的是当页面加载时就将数据绑定到 DropDownList 控件当中，实现的代码如下。

例程 2　代码位置：光盘\mr\27\Revert\BeiFenKSql.aspx.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!this.IsPostBack)
    {
        //调用binddropdownList()方法
this.binddropdownList();
    }
}
```

在用户自定义的 binddropdownList 方法中，主要是从数据库中检索出相应的数据，并将检索到的数据库的名称绑定到 DropDownList 控件中，实现的代码如下。

例程 3 代码位置：光盘\mr\27\Revert\BeiFenKSql.aspx.aspx.cs

```
public void binddropdownList()
{
    //定义连接数据库字符串
    string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
    //定义查询企业管理器中的所有数据库
    string cmdtxt2 = "Exec sp_helpdb";
    //创建数据库连接对象
    SqlConnection Con = new SqlConnection(cmdtxt1);
    //打开数据库连接
    Con.Open()
    //创建命令对象
    SqlCommand mycommand = new SqlCommand(cmdtxt2, Con);
    //创建数据阅读器读取数据库中数据
    SqlDataReader dr = mycommand.ExecuteReader();
    //设置下拉列表控件的数据源、绑定的主键字段
    this.DDLDataBase.DataSource = dr;
    this.DDLDataBase.DataTextField = "name";
    //从数据库中绑定数据到下拉列表框中
    this.DDLDataBase.DataBind();
    //关闭数据阅读器
    dr.Close();
    //关闭数据库连接
    Con.Close();
}
```

双击页面中的“备份数据库”按钮，触发其 ImgBtnOK_Click 事件，在该事件中，首先从下拉列表框中选择需要备份的数据库，并在 TextBox 文本框中输入备份数据库存放的位置和名称，实现的代码如下。

例程 4 代码位置：光盘\mr\27\Revert\BeiFenKSql.aspx.aspx.cs

```
protected void ImgBtnOK_Click(object sender, ImageClickEventArgs e)
{
        string cmdtxt1 = "Server=(local);database='" + this.DDLDataBase.SelectedValue + "';Uid=sa;Pwd=";
        string cmdtxt2 = "backup database " + this.DDLDataBase.SelectedValue + " to disk='" + this.TextBox1.Text.Trim() + ".bak'";
        SqlConnection Con = new SqlConnection(cmdtxt1);
        Con.Open();//打开数据库的链接
        try
        {
            SqlCommand Com = new SqlCommand(cmdtxt2, Con);
            Com.ExecuteNonQuery();
            Response.Write("<script language=javascript>alert('备份数据成功！');location='javascript:history.go(-1)'</script>");
        }
        catch (Exception ms)
        {
            Response.Write(ms.Message);
            Response.Write("<script language=javascript>alert('备份数据失败！');location='javascript:history.go(-1)'</script>");
        }
        finally
        {
            Con.Close();
        }
}
```

27.3.2 数据库的还原操作

数据库的还原由“hyKSql.aspx”页面实现，该页主要完成对 SQL Server 2000 数据库的还原操作，运行的效果如图 27.5 所示。

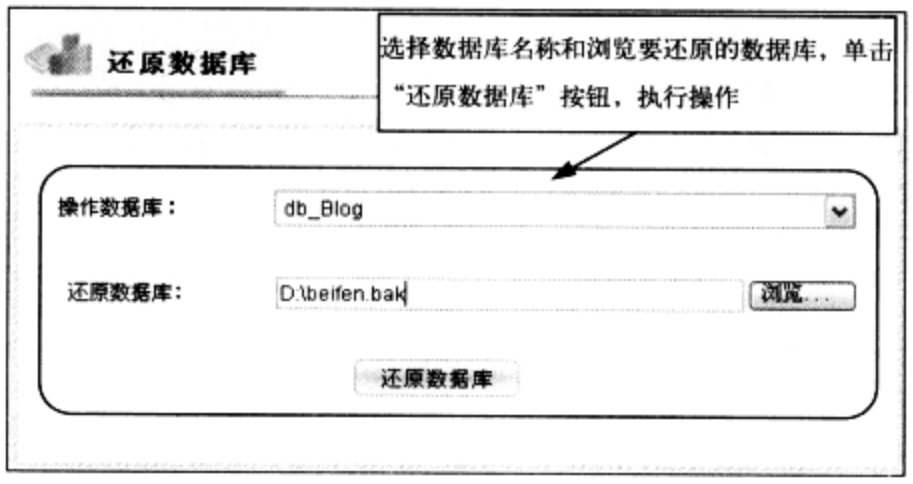


图 27.5　运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，默认名为“hyKSql.aspx”，作为数据库的还原页。hyKSql.aspx 的设计界面如图 27.6 所示。

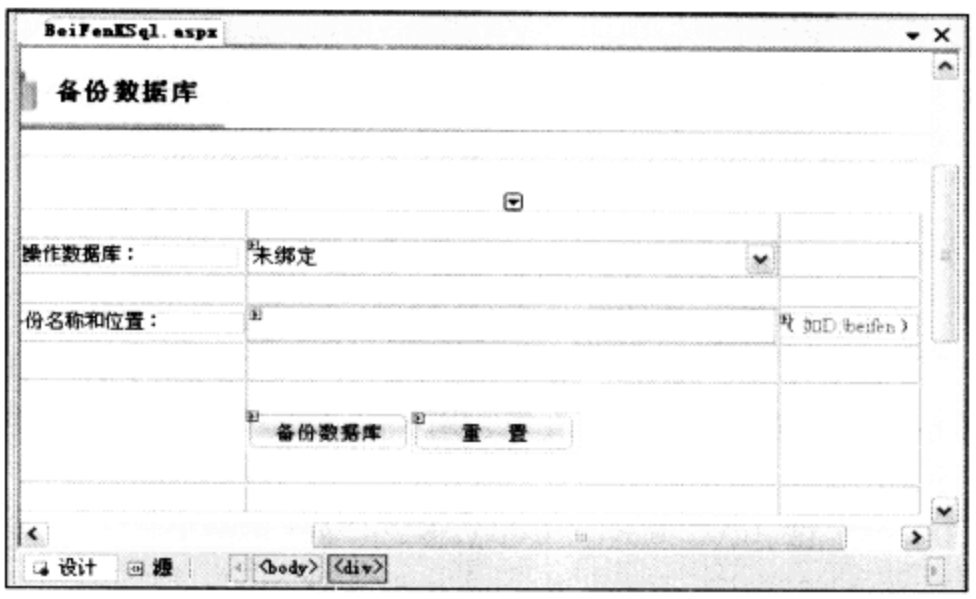


图 27.6　hyKSql.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 DropDownList 控件、1 个 FileUpload 控件和 ImageButton 控件，设置控件的属性。页面中各个控件的属性设置及其用途如表 27.3 所示。

表 27.3　　数据库还原页控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　途
Table	无	无	页面布局
DropDownList	控件的名称为 dropSqlName	无	将 SQL Server 2000 中的数据库的名称绑定到该控件中
FileUpload	控件的名称设置为 fileShow	无	用于选择数据库
ImageButton	控件的名称为 ImgBtnOK	将该控件的 ID 属性设置为 ImgBtnOK，将 ImageUrl 设置为～/image/btnbf.JPG，将 OnClick 属性设置为 ImgBtnOK_Click	执行还原数据库操作

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 5 代码位置：光盘\mr\27\Revert\hyKSql.aspx.aspx.cs

```
//引入命名空间
using System.Data.SqlClient;
```

在 Page_Load 事件中，主要是从数据库中检索出相应的数据，并将检索到的数据库的名称绑定到 DropDownList 控件中，实现的代码如下。

例程 6 代码位置：光盘\mr\27\Revert\hyKSql.aspx.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)//判断页面是否是首次加载
    {
        //定义执行数据库连接字符串
        string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
        //定义查询SQL企业管理器中所有数据库
        string cmdtxt2 = "Exec sp_helpdb";
       //创建数据库连接对象
        SqlConnection Con = new SqlConnection(cmdtxt1);
       //打开数据库连接
        Con.Open();
       //创建命令对象
        SqlCommand mycommand = new SqlCommand(cmdtxt2, Con);
       //创建数据阅读器
        SqlDataReader dr = mycommand.ExecuteReader();
       //设置下拉列表框的数据源及主键字段值
        this.dropSqlName.DataSource = dr;
        this.dropSqlName.DataTextField = "name";
       //从数据库中绑定数据
        this.dropSqlName.DataBind();
        dr.Close();//关闭阅读器
        Con.Close();//关闭数据库连接
    }
}
```

双击页面中的“还原数据库”按钮，触发其 ImgBtnk_Click 事件，在该事件中，首先从下拉列表框中选择需要还原的数据库，并通过 FileUpLoad 控件指定此数据库备份文件路径，实现的代码如下。

例程 7 代码位置：光盘\mr\27\Revert\hyKSql.aspx.aspx.cs

```
protected void ImgBtnk_Click(object sender, ImageClickEventArgs e)
{
     //获得备份路径及数据库名称
    string path = this.fileShow.PostedFile.FileName;
    string dbname = this.dropSqlName.SelectedValue;
    string cmdtxt1 = "Server=(local);database='" + this.dropSqlName.SelectedValue + "';Uid=sa;Pwd=";
    string cmdtxt2 = "use master restore database " + dbname + " from disk='" + path + "'";
   //创建数据库连接对象
    SqlConnection Con = new SqlConnection(cmdtxt1);
    Con.Open();//打开数据库连接
    try
    {
        //创建命令对象
        SqlCommand Com = new SqlCommand(cmdtxt2, Con);
        //执行指定的SQL语句，并返回受影响的行数
        Com.ExecuteNonQuery();
        Response.Write("<script language=javascript>alert('还原数据成功！');location='javascript:history.go(-1)'</script>");
    }
    catch (Exception ms)
    {
        //如果还原数据库失败，则给出错误提示
        Response.Write(ms.Message);
        Response.Write("<script language=javascript>alert('还原数据失败！');location='javascript:history.go(-1)'</script>");
```

```
        }
        finally
        {
            Con.Close();//关闭数据库连接
        }
    }
```

27.3.3 备份数据表的操作

备份数据库中数据表由“BeiFenBSql.aspx”页面实现，该页主要完成对 SQL Server 2000 数据库中的数据表的备份操作，运行的效果如图 27.7 所示。

图 27.7 运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，默认名为“BeiFenBSql.aspx”，作为数据库的备份页。BeiFenBSql.aspx 的设计界面如图 27.8 所示。

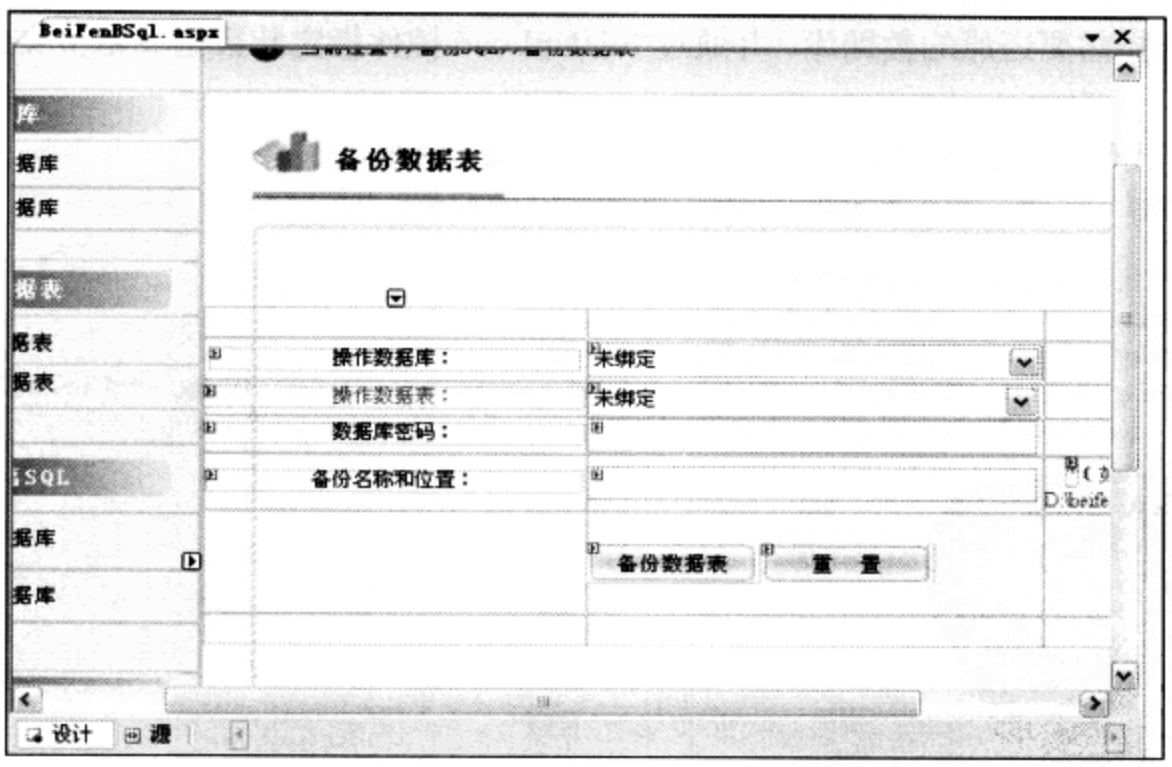


图 27.8 BeiFenBSql.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 DropDownList 控件、1 个 FileUpload 控件和 ImageButton 控件放入页面中，设置控件的属性。页面中各个控件的属性设置及其用途如表 27.4 所示。

表 27.4　　备份数据表页中的控件列表

控 件 类 型	控 件 名 称	主要属性设置	用　　途
Table	无	无	页面布局
TextBox	控件的名称分别为 txtPwd 和 txtPath	将 txtPwd 控件的 TextMode 属性设置为 Password	分别用于输入“数据库的密码”和“数据库的名称和位置”
DropDownList	控件的名称分别为 dropDatabase 和 dropTable	例如：将“dropDatabase”控件的 AutoPostBack 属性设置为 True，将 OnSelectedIndexChanged 属性设置为 dropDatabase_SelectedIndexChanged	分别用于显示“操作的数据库”和“操作的数据表”
ImageButton	控件的名称设置为 ImgBtnBiao	将控件的 ID 属性设置为 ImgBtnBiao，将控件的 ImageUrl 属性设置为～/image/bfsjb.JPG，将控件的 OnClick 属性设置为 ImgBtnBiao_Click	执行备份数据表操作

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 8　代码位置：光盘\mr\27\Revert\hyKSql.aspx.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
```

在 Page_Load 事件中，从数据库中检索出相应的数据，并将检索到的数据库的名称绑定到 DropDownList 控件中，实现的代码如下。

例程 9　代码位置：光盘\mr\27\Revert\hyKSql.aspx.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
    string cmdtxt2 = "Exec sp_helpdb";
    SqlConnection Con = new SqlConnection(cmdtxt1);
    Con.Open();
    if (!Page.IsPostBack)
    {
        SqlCommand mycommand = new SqlCommand(cmdtxt2, Con);
        SqlDataReader dr = mycommand.ExecuteReader();
        this.dropDatabase.DataSource = dr;
        this.dropDatabase.DataTextField = "name";
        this.dropDatabase.DataBind();
        dr.Close();
        string cmdtxt3 = "use " + this.dropDatabase.SelectedValue + " SELECT * FROM " + this.dropDatabase.Selected
Value + ".dbo.sysobjects";
        cmdtxt3 += " WHERE xtype='U' AND STATUS>=0";
        SqlCommand mycommand1 = new SqlCommand(cmdtxt3, Con);
        SqlDataReader dr1 = mycommand1.ExecuteReader();
        this.dropTable.DataSource = dr1;
        this.dropTable.DataTextField = "name";
        this.dropTable.DataBind();
        dr1.Close();
        Con.Close();
    }
}
```

双击 dropDatabase 控件，触发“dropDatabase_SelectedIndexChanged”事件，将数据表的名称绑定到 dropTable 控件中，实现的代码如下。

例程 10　代码位置：光盘\mr\27\Revert\hyKSql.aspx.aspx.cs

```
protected void dropDatabase_SelectedIndexChanged(object sender, EventArgs e)
{
    string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
    SqlConnection Con = new SqlConnection(cmdtxt1);
```

```
            Con.Open();
            string cmdtxt3 = "use " + this.dropDatabase.SelectedValue + " SELECT * FROM " + this.dropDatabase.SelectedValue +
".dbo.sysobjects";
            cmdtxt3 += " WHERE xtype='U' AND STATUS>=0";
            SqlCommand mycommand1 = new SqlCommand(cmdtxt3, Con);
            SqlDataReader dr1 = mycommand1.ExecuteReader();
            this.dropTable.DataSource = dr1;
            this.dropTable.DataTextField = "name";
            this.dropTable.DataBind();
            dr1.Close();
            Con.Close();
        }
```

双击“备份数据表”按钮，触发“ImgBtnBiao_Click”事件，从下拉列表框中选择需要备份的数据表，在 txtPwd 文本框中输入数据库密码，并在 txtPath 文本框中输入备份数据库存放的位置和名称，实现的代码如下。

例程 11　代码位置：光盘\mr\27\Revert\hyKSql.aspx.aspx.cs

```
    protected void ImgBtnBiao_Click(object sender, ImageClickEventArgs e)
    {
        string cmdtxt1 = "Server=(local);Database=master;Uid=sa;Pwd='" + this.txtPwd.Text.Trim() + "'";
        string cmdtxt2 = "USE master EXEC xp_cmdshell 'bcp " + this.dropDatabase.SelectedValue + ".dbo." + this.drop
Table.SelectedValue + "";
        cmdtxt2 += " out " + this.txtPath.Text.Trim() + ".xls -c -q -S. -Usa -P" + this.txtPwd.Text.Trim() + "'";
        SqlConnection Con;
        try
        {
            Con = new SqlConnection(cmdtxt1);
            Con.Open();
            SqlCommand Com = new SqlCommand(cmdtxt2, Con);
            Com.ExecuteNonQuery();
            Con.Close();
            Response.Write("<script language=javascript>alert('备份数据成功！');location='javascript:history.go(-1)'</script>");
        }
        catch (Exception ms)
        {
            Response.Write(ms.Message);
            Response.Write("<script language=javascript>alert('备份数据失败！');location='javascript:history.go(-1)'</script>");
        }
    }
```

27.3.4　还原数据表的操作

还原数据表实现在“HYBSql.aspx”页面中，该页面主要完成对 SQL Server 2000 数据库中的数据表的还原操作，运行的效果如图 27.9 所示。

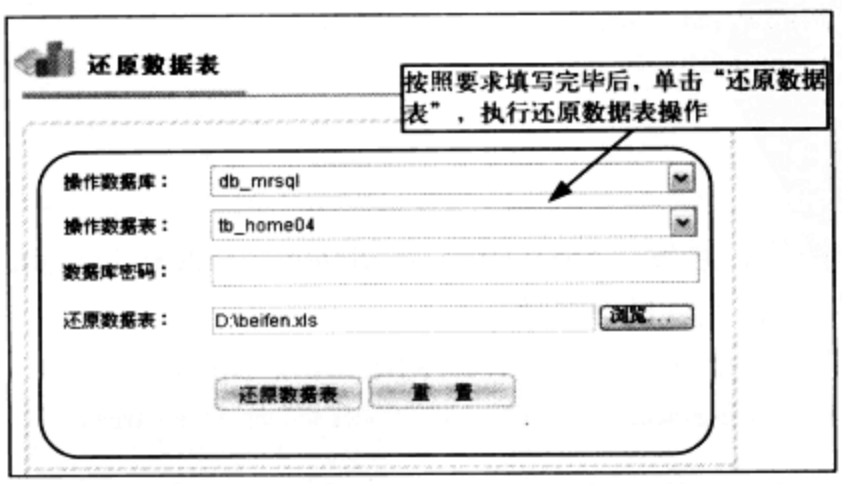


图 27.9　还原数据表运行效果图

1. 页面设计

（1）在应用程序中创建一个 Web 窗体，默认名为“HYBSql.aspx”，作为还原数据表页。HYBSql.aspx 的设计界面如图 27.10 所示。

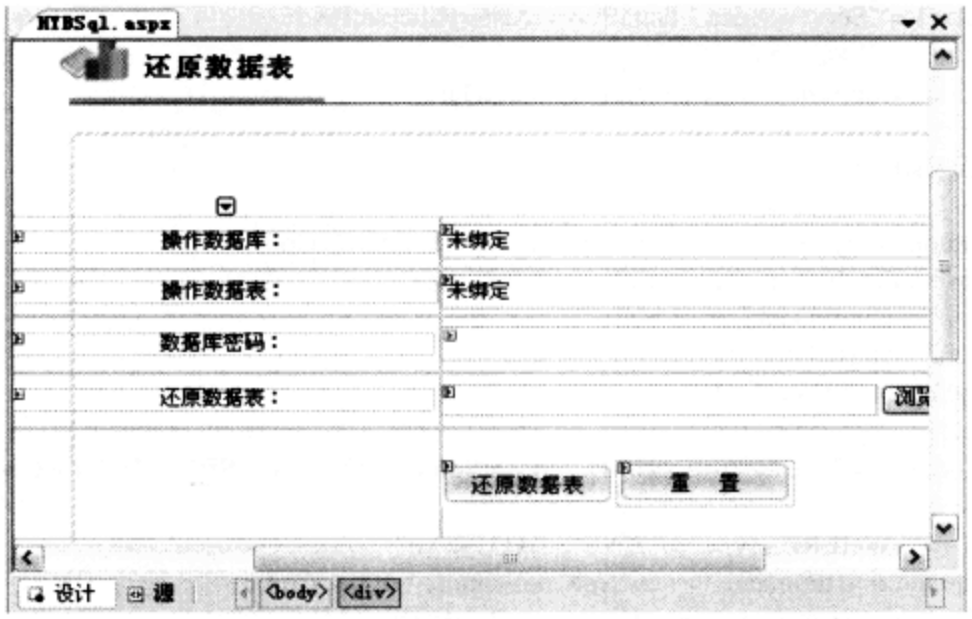


图 27.10 HYBSql.aspx 的设计界面

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 2 个 DropDownList 控件、1 个 TextBox 控件、1 个 FileUpload 控件和 2 个 ImageButton 控件放到界面中，设置控件的属性。页面中各个控件的属性设置及其用途如表 27.5 所示。

表 27.5 备份数据表页中的控件列表

控件类型	控件名称	主要属性设置	用途
Table	无	无	页面布局
TextBox	控件的名称为 txtPwd	将 txtPwd 控件的 TextMode 属性设置为 Password	用于输入“数据库的密码”
DropDownList	控件的名称分别为 dropDatabase 和 dropTable	例如，将“dropDatabase”控件的 AutoPostBack 属性设置为 True，将 OnSelectedIndexChanged 属性设置为 dropDatabase_SelectedIndexChanged	分别用于显示“操作的数据库”和“操作的数据表”
ImageButton	控件的名称分别为 ImgBtnbiao 和 ImgBtnCZ	将控件的 ImageUrl 属性分别设置为～/image /btnhyb.JPG 和～/image/CZ.JPG	分别执行还原数据表操作和执行清空文本框操作

2. 代码实现

在编写代码之前，首先引入命名空间，引入命名空间的代码如下。

例程 12 代码位置：光盘\mr\27\Revert\HYBSql.aspx.aspx.cs

```
using System.Data.SqlClient;//引入命名空间
using System.IO;
```

在 Page_Load 事件加载事件中，首先判断用户是否登录，如果没有登录则将页面跳转到 tishi.aspx 页面中，否则将用户的名称显示到 lblMessage 控件中。然后从数据库中检索出相应的数据，最后将检索到的数据库的名称绑定到 DropDownList 控件中，实现的代码如下。

例程 13 代码位置：光盘\mr\27\Revert\HYBSql.aspx.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["UserName"] == null && Session["Pwd"] == null)
    {
```

```
            Response.Redirect("tishi.aspx");//跳转到tishi.aspx页面中
        }
        else
        {
            this.LblMessage.Text = "您好！  " + Session["UserName"].ToString() + "  欢迎您登录本网站！ ";
            string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
            string cmdtxt2 = "Exec sp_helpdb";
            SqlConnection Con = new SqlConnection(cmdtxt1);
            Con.Open();
            if (!Page.IsPostBack)
            {
                SqlCommand mycommand = new SqlCommand(cmdtxt2, Con);
                SqlDataReader dr = mycommand.ExecuteReader();
                this.dropDatabase.DataSource = dr;
                this.dropDatabase.DataTextField = "name";
              this.dropDatabase.DataBind();
              dr.Close();
             string cmdtxt3 = "use " + this.dropDatabase.SelectedValue + " SELECT * FROM " + this.dropDatabase.
SelectedValue + ".dbo.sysobjects";
              cmdtxt3 += " WHERE xtype='U' AND STATUS>=0";
              SqlCommand mycommand1 = new SqlCommand(cmdtxt3, Con);
              SqlDataReader dr1 = mycommand1.ExecuteReader();
              this.dropTable.DataSource = dr1;
              this.dropTable.DataTextField = "name";
              this.dropTable.DataBind();
              dr1.Close();
              Con.Close();
            }
          }
        }
```

在“dropDatabase_SelectedIndexChanged”事件中，从数据库中检索出相应的数据，并将检索到的数据表的名称绑定到 DropDownList 控件中，实现的代码如下。

例程 14 代码位置：光盘\mr\27\Revert\HYBSql.aspx.aspx.cs

```
    protected void dropDatabase_SelectedIndexChanged(object sender, EventArgs e)
    {
        string cmdtxt1 = "Server=(local);DataBase=master;Uid=sa;Pwd=";
        SqlConnection Con = new SqlConnection(cmdtxt1);
        Con.Open();
        string cmdtxt3 = "use " + this.dropDatabase.SelectedValue + " SELECT * FROM " +
this.dropDatabase.SelectedValue + ".dbo.sysobjects";
        cmdtxt3 += " WHERE xtype='U' AND STATUS>=0";
        SqlCommand mycommand1 = new SqlCommand(cmdtxt3, Con);
        SqlDataReader dr1 = mycommand1.ExecuteReader();
        this.dropTable.DataSource = dr1;
        this.dropTable.DataTextField = "name";
        this.dropTable.DataBind();
        dr1.Close();
        Con.Close();
    }
```

双击“还原数据表”按钮，触发其“ImgBtnbiao_Click”事件，实现的是执行还原数据表的操作，实现的代码如下。

例程 15 代码位置：光盘\mr\27\Revert\HYBSql.aspx.aspx.cs

```
    protected void ImgBtnbiao_Click(object sender, ImageClickEventArgs e)
    {
      string cmdtxt1 = "Server=(local);Database=master;Uid=sa;Pwd='" + this.txtPwd.Text.Trim() + "'";
        string cmdtxt2 = "USE master EXEC xp_cmdshell 'bcp " + this.dropDatabase.SelectedValue + ".dbo." + this.drop
Table.SelectedValue + "'";
            cmdtxt2 += " in " + this.fileShow.PostedFile.FileName + " -c -q -S. -Usa -P" + this.txtPwd.Text.Trim() + "'";
            SqlConnection Con;
```

```
try
{
    Con = new SqlConnection(cmdtxt1);
    Con.Open();
    SqlCommand Com = new SqlCommand(cmdtxt2, Con);
    Com.ExecuteNonQuery();
    Con.Close();
    Response.Write("<script language=javascript>alert('还原数据成功！')</script>");
}
catch (Exception ms)
{
    Response.Write(ms.Message);
    Response.Write("<script language=javascript>alert('还原数据失败！');location='javascript:history.go(-1)'</script>");
    }
}
```

27.4 程序错误与调试

在对程序进行调试时，可能出现如图 27.11 所示的错误。

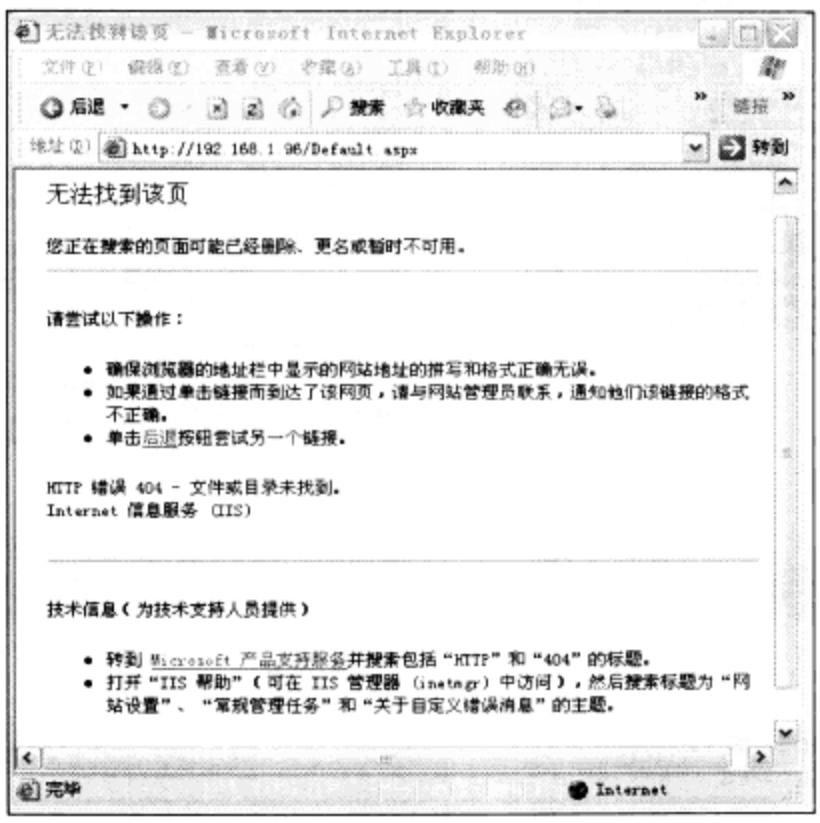


图 27.11 ASP.NET 版本错误的出错页

原因：在使用 IIS 运行程序前，首先看一下是否启动 ASP.NET 服务，如果没有启动该服务，那么将会使 ASP.NET 程序无法在 IIS 中正常运行，将会出现如图 27.10 所示错误，下面将详细介绍解决这一问题的方法。

解决方法步骤如下。

（1）首先选择“开始”→“程序”→“管理工具”→“Internet 信息服务（IIS）管理器”，展开左侧的导航菜单，单击“Web 服务扩展”，如图 27.12 所示。

（2）然后选择“ASP.NET v2.0.50727”项，单击鼠标右键，在弹出的下拉菜单中选择“允许”选项，如图 27.13 所示。这时将会看到状态由原来的“禁止”状态改为现在的“允许”状态。这时设置就已经完成。

（3）最后测试一下程序在 IIS 中是否可用，运行该程序，程序运行的效果如图 27.14 所示。

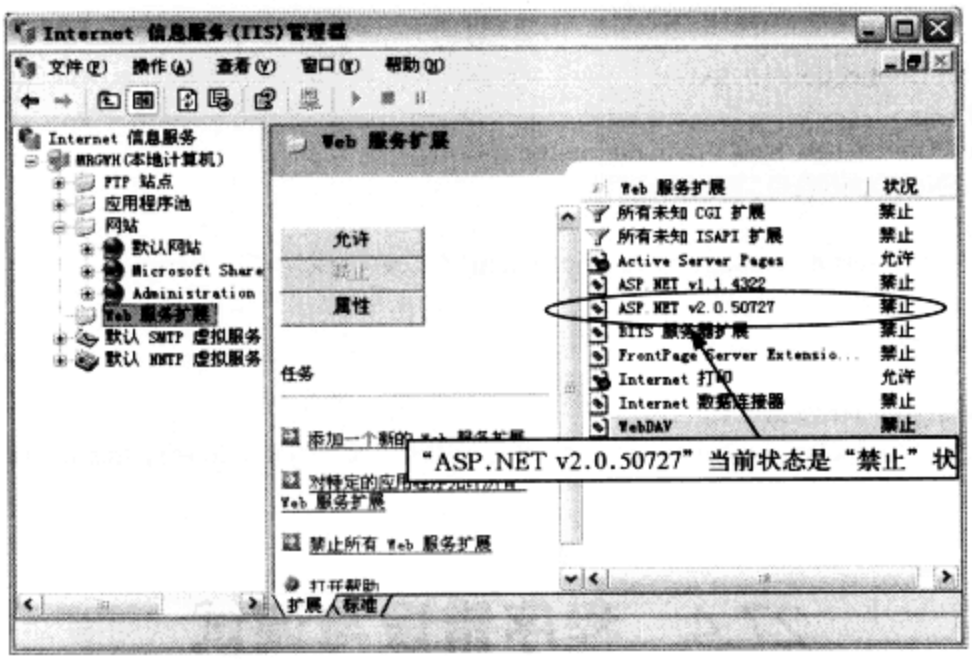


图 27.12　IIS 设置

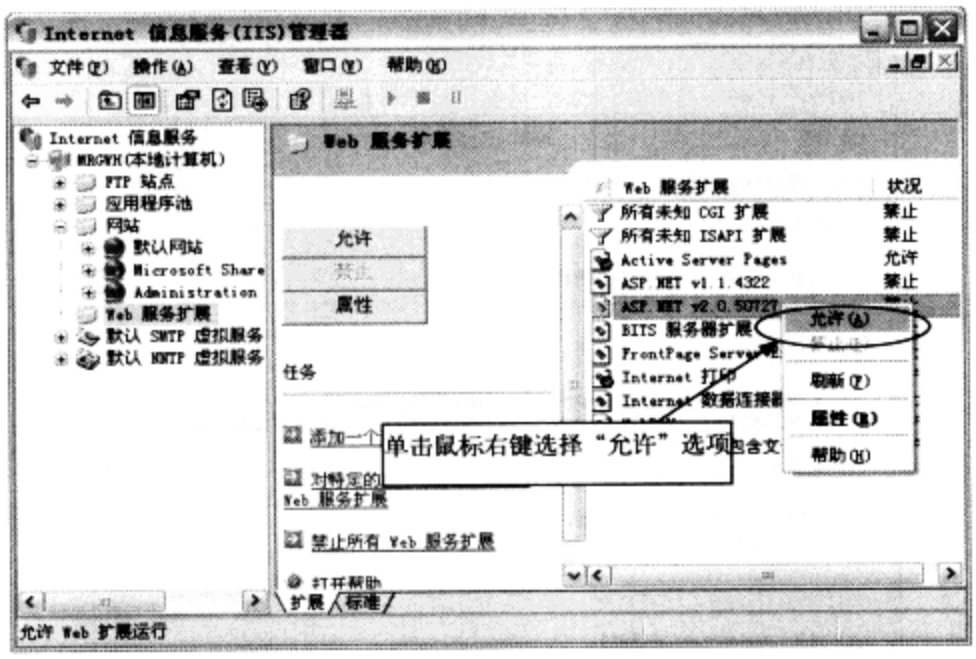


图 27.13　运行效果图

图 27.14　页面运行效果

LINQ 数据库访问技术

第 28 章

实例位置：光盘\mr\28\

28.1 LINQ 技术概述

LINQ（Language-Integrated Query，语言集成查询）是微软公司提供的一项新技术，它能够将查询直接引入到.Net FrameWork3.5 所支持的编程语言（如 C#和 VB.Net 等）中。LINQ 查询操作可以通过编程语言自身传达，而不是以字符串方式嵌入到应用程序代码中。

28.1.1 查询与 LINQ 的区别

查询是一种从给定的数据源中检索满足指定条件的数据表达式的功能。通常情况下，查询数据往往使用字符串来表示查询操作，如查询关系数据库的 SQL 语句、查询 XML 结构数据的 Xquery 等，而在这些查询操作中，一般不会有检查被查询数据的类型。同时，这些查询操作往往与编程语言处于一种相对孤立的状态。

语言集成查询是 Visual Studio 2008 中的一组功能，可为 C#和 Visual Basic 语言语法提供强大的查询功能。LINQ 引入了标准的、易于学习的查询和更新数据模式，可以对其技术进行扩展以支持几乎任何类型的数据存储。Visual Studio 2008 包含 LINQ 提供程序的程序集，这些程序集支持将 LINQ 与.NET Framework 集合、SQL Server 数据库、ADO.NET 数据集和 XML 文档一起使用。

28.1.2 LINQ 基本组成

LINQ 是 Visual Studio 2008 和.Net Framework 3.5 版中一项突破性的创新，它在对象领域和数据领域之间架起了一座桥梁。LINQ 主要由 3 部分组成：LINQ to ADO.NET、LINQ to Objects、LINQ to XML。其中，LINQ to ADO.NET 可以分为两部分：LINQ to SQL 和 LINQ to DataSet。LINQ 可以查询或操作任何存储形式的数据，LINQ 的组成说明如下。

- LINQ to SQL 组件，可以查询基于关系数据库的数据，并对这些数据进行检索、插入、修改、删除、排序、聚合、分区等操作。
- LINQ to DataSet 组件，可以查询 DataSet 对象中的数据，并对这些数据进行检索、过滤、排序等操作。
- LINQ to Objects 组件，可以查询 Ienumerable 或 Ienumerable<T>集合，也就是说可以查询任何可枚举的集合，如数据（Array 和 ArrayList）、泛型列表 List<T>、泛型字典 Dictionary<T>等，以及用户自定义的集合，而不需要使用 LINQ 提供程序或 API。
- LINQ to XML 组件，可以查询或操作 XML 结构的数据（如 XML 文档、XML 片段、XML 格式的字符串等），并提供了修改文档对象模型的内存文档和支持 LINQ 查询表达式等功能，以及处理 XML 文档的全新的编程接口。

28.1.3 LINQ 与 ADO.NET 的关系

在通常情况下，ADO.NET 提供了大量的读取、查询、检索、添加、修改、删除和过滤数据库中数据的方法。程序开发人员使用这些方法，需要编程查询或操作数据库的每一个步骤（如创建数据库连接、打开数据库连接、执行相关数据库操作等），这样显得很繁琐。

使用 LINQ 可以把数据从数据库的表中传递到内存的对象中，并将数据源转换为基于 IEnumerable 的对象集合。这样，传统的操作数据库方法转换为使用 LINQ 查询和处理基于 IEnumerable 的对象集合。由于 LINQ 查询被嵌入到.NET Framework 3.5 支持的编程语言中，因此，在创建 LINQ 查询表达式时，可以使用 Visual Studio 2008 的智能支持功能。

LINQ 提供了名称为 LINQ to ADO.NET 的技术专门用来处理关系数据。其中，LINQ to ADO.NET 包括两种独立的技术：LINQ to DataSet 和 LINQ to SQL。使用 LINQ to DataSet 可以查询或处理 DataSet 对象中的数据，使用 LINQ to SQL 可以直接查询或处理关系数据。

LINQ to DataSet 基于 ADO.NET，并为 ADO.NET 提供了更加高级的、简单的查询技术，它和 ADO.NET 之间的关系如图 28.1 所示。

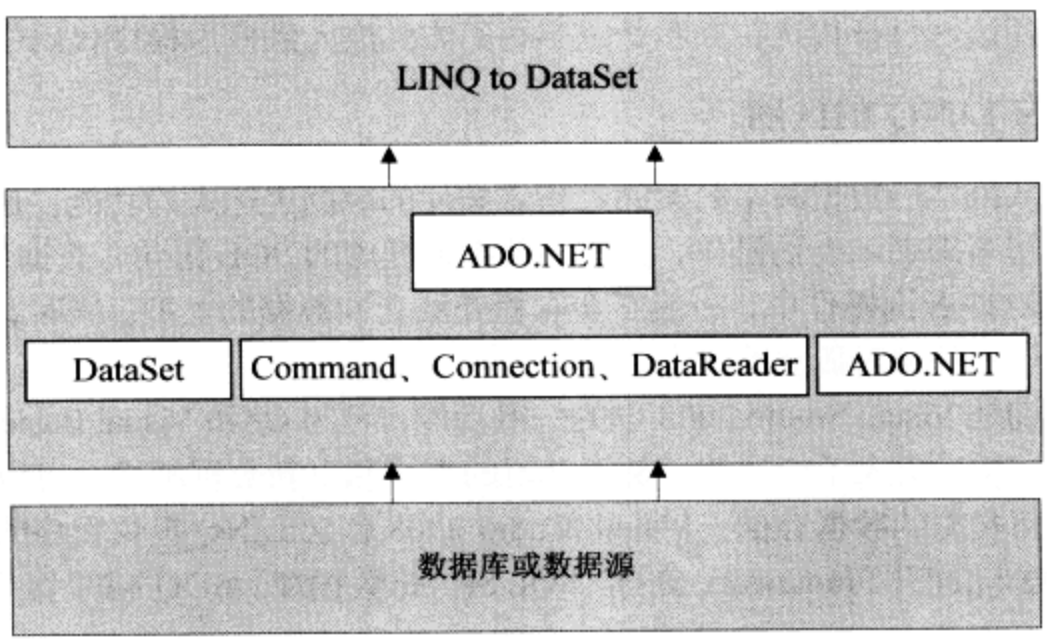


图 28.1 LINQ to DataSet 与 ADO.NET 的关系

使用 LINQ to SQL 可以创建 LINQ 编程模型，并直接映射到关系型数据库（如 SQL Server 数据库等）之上。LINQ to SQL 可以直接创建表示数据的.NET Framework 的类，并将这些类映射到数据库中的表、视图、存储过程、函数等对象。因此，使用 LINQ to SQL 可以直接操作表示数据的.NET Framework 的类来操作与之映射的关系数据库（如 SQL Server 数据库等）中的数据。

28.2 LINQ 查询常用子句

LINQ 查询表达式是 LINQ 中非常重要的一部分内容，它可以从一个或多个给定的数据源中检索数据，并指定检索结果的数据类型和表现形式。LINQ 查询表达式由一个或多个 LINQ 查询子句按照一定的规则组成。LINQ 查询表达式包括 from 子句、where 子句、select 子句、orderby 子句、group 子句、into 子句、join 子句、let 子句。这些子句的具体说明如表 28.1 所示。

表 28.1 LINQ 查询子句说明

查 询 子 句	说　明
form 子句	指定查询操作的数据源和范围变量
where 子句	筛选元素的逻辑条件，一般由逻辑运算符组成

续表

查 询 子 句	说　　明
select 子句	指定查询结果的类型和表现形式
orderby 子句	对查询结果进行排序（降序或升序）
group 子句	对查询结果进行分组
into 子句	提供一个临时的标识符。该标识可以引用 join、group 和 select 子句的结果
join 子句	连接多个查询操作的数据源
Let 子句	引入用于存储查询表达式中子表达式结果的范围变量

下面对一些常用的 LINQ 子查询进行详细讲解。

28.2.1　from 子句

LINQ 查询表达式必须包括 form 子句，且以 from 子句开头。如果该查询表达式还包括子查询，那么子查询表达式也必须以 form 子句开头。form 子句指定查询操作的数据源和范围变量。其中，数据源不但包括查询本身的数据源，而且还包括子查询的数据源。范围变量一般用来表示源序列中的每一个元素。

下面的代码演示一个简单的 LINQ 查询操作，该查询操作从 values 数组中查询小于 3 的元素，其中，v 为范围变量，values 是数据源。

```
int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
var value = from v in values
            where v < 5
            select v;
Response.Write("查询结果：<br>");
foreach (var v in value)
{
    Response.Write(v.ToString() + "<br>");
}
```

程序运行结果如图 28.2 所示。

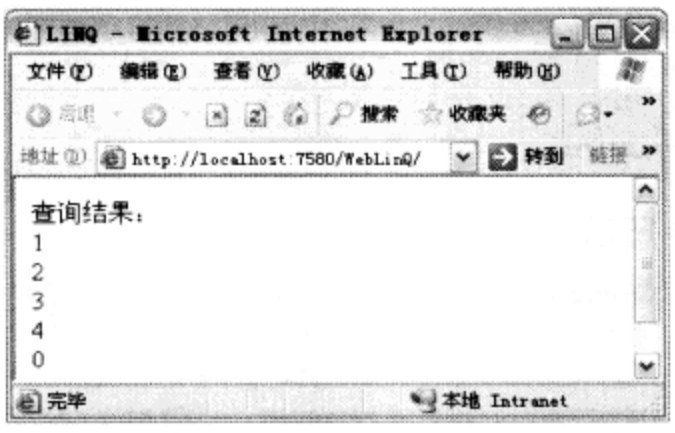


图 28.2　from 子句查询结果

28.2.2　where 子句

在 LINQ 查询表达式中，where 子句指定筛选元素的逻辑条件，一般由逻辑运算符（例如，逻辑与和逻辑或）组成。一个查询表达式可以不包含 where 子句，也可以包含 1 个或多个 where 子句。每一个 where 子句可以包含 1 个或多个布尔条件表达式。

对于一个 LINQ 查询表达式而言，where 子句不是必须的。如果 where 子句在查询表达式中出现，那么 where 子句不能作为查询表达式的第一个子句或最后一个子句。在上一节 form 子句中演示的代码就使用 where 子句。

下面演示一个示例，在查询表达式中使用 where 子句，并且 where 子句由两个布尔表达式

和逻辑与&&组成。实现代码如下：

```
int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
var value = from v in values
            where v < 9 && v > 5
            select v;
Response.Write("查询结果：<br>");
foreach (var v in value)
{
    Response.Write(v.ToString() + "<br>");
}
```

程序运行结果如图 28.3 所示。

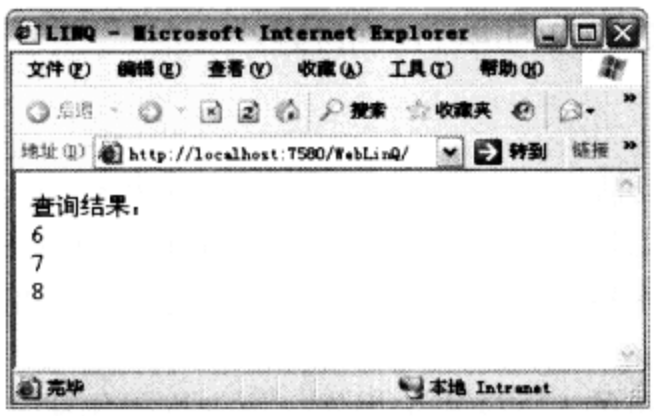


图 28.3　where 子句查询结果

28.2.3　select 子句

在 LINQ 查询表达式中，select 子句指定查询结果的类型和表现形式。LINQ 查询表达式必须以 select 子句或 group 子句结束。

下面示例代码演示了包含最简单 select 子句的查询操作。代码如下：

```
int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
var value = from v in values
            where v < 5
            select v;
Response.Write("查询结果：<br>");
foreach (var v in value)
{
    Response.Write(v.ToString() + "<br>");
}
```

程序运行结果如图 28.4 所示。

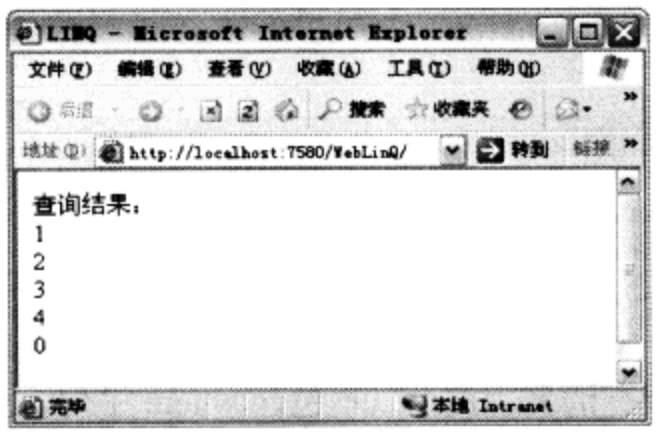


图 28.4　select 子句查询结果

28.2.4　group by 子句

在查询表达式中，group by 子句对查询的结果进行分组，并返回元素类型为 IGrouping<Tkey, TElement>的对象序列。

下面通过一个示例来演示 group by 子句对查询的结果进行分组。本示例实现的是将数据源中的数字按奇偶分组，然后使用 foreach 嵌套输出查询结果。实现具体代码如下：

```
protected void Page_Load(object sender, EventArgs e)
    {
        int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
        var value = from v in values
                    group v by v % 2 == 0; ;
        //输出查询结果
        foreach (var i in value)
        {
            foreach (int j in i)
            {
                Response.Write(j + "<br>");
            }
        }
    }
```

程序运行结果如图 28.5 所示。

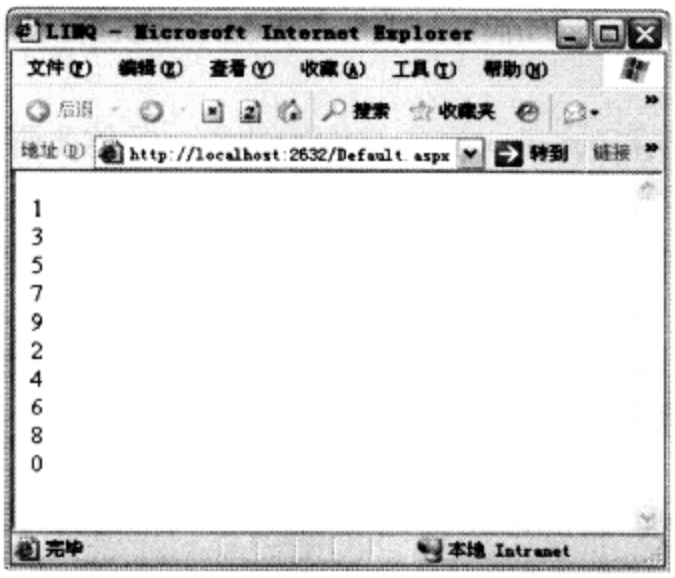


图 28.5　group by 子句查询结果

28.2.5　orderby 子句

在 LINQ 查询表达式中，orderby 子句可以对查询结果排序。排序方式可以为“升序”或“降序”，且排序的主键可以是一个或多个。在这里，读者值得注意的是，LINQ 查询表达式对查询结果的默认排序方式为“升序”。

提示：在 LINQ 查询表达式中，orderby 子句升序使用 ascending 关键字，降序使用 descending 关键字。

下面通过一个示例来演示 orderby 子句对查询结果进行排序。本示例实现的是将数据源中的数字按降序排序，然后使用 foreach 输出查询结果。实现具体代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    int[] values = { 5, 8, 3, 4, 1, 6, 7, 2, 9, 0 };
    var value = from v in values
                where v < 3 || v > 6
                orderby v descending
                select v;
    //输出查询结果
    foreach (var i in value)
    {
        Response.Write(i + "<br>");
    }
}
```

程序运行结果如图 28.6 所示。

图 28.6　orderby 子句查询结果排序

28.2.6　into 子句

在 LINQ 查询表达式中，into 子句可以创建一个临时标识符，使用该标识符可以存储 group、join 及 select 子句的结果。

下面通过一个示例来演示 into 子句操作查询的结果，具体步骤如下。

（1）创建数据源，int 型数据，并设置初始值“1, 2, 3, 4, 5, 6, 7, 8, 9, 0”。

（2）使用 group 子句对结果进行分组，分为两组：奇数组和偶数组。

（3）使用 into 子句创建临时标识符 g 存储查询结果。

（4）使用 where 子句筛选查询结果元素大于 8 的奇数组或偶数组。

（5）使用嵌套 foreach 语句输出查询结果。

实现具体代码如下：

```
protected void Page_Load(object sender, EventArgs e)
    {
        int[] values = { 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 };
        var value = from v in values
                    group v by v % 2 == 0 into g
                    where g.Max() > 8        //分组后   查找组中大于8的组
                    select g;
        //输出查询结果
        foreach (var i in value)
        {
            foreach (int j in i)
            {
                Response.Write(j + "<br>");
            }
        }
    }
```

程序运行结果如图 28.7 所示。

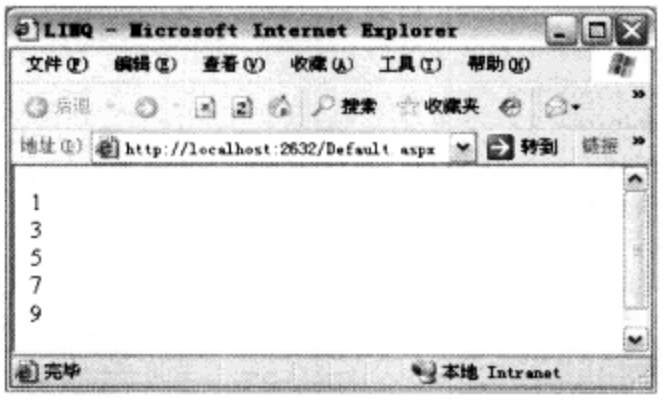


图 28.7　into 子句查询结果

28.3 使用 LINQ 查询和操作数据库

使用 LINQ 查询或操作数据库，需要建立 LINQ 数据源，LINQ 数据源使用 DBML 文件作为数据源。下面以 SQL Server 2005 数据库 db_LinQ 为例，建立一个 LINQ 数据源，详细步骤如下。

（1）启动 Visual Studio 2008 开发环境，建立一个目标框架为 Framework SDK v 3.5 的 ASP.NET 空网站。

（2）在“解决方案资源管理器”中，右键单击“App_Code”文件夹，在弹出的快捷菜单中选择“添加新项”，弹出“添加新项”对话框，如图 28.8 所示。

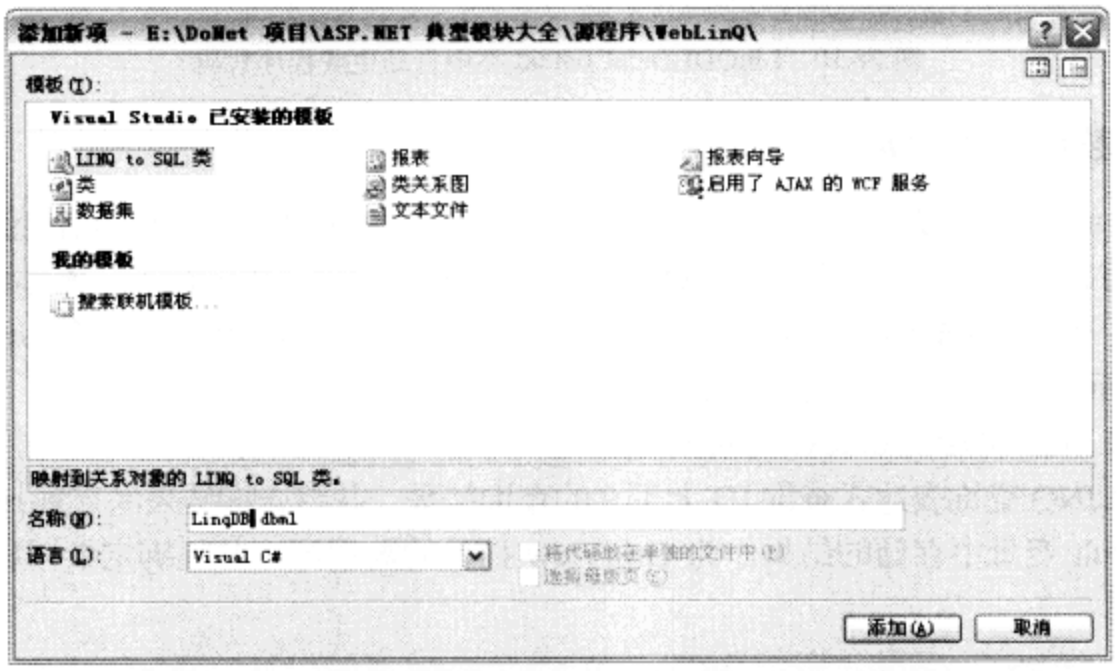


图 28.8 添加新项

（3）在模板列表中选择“LINQ to SQL 类”，并为其命名为 LinqDB.dbml。

（4）服务资源管理器中连接 SQL Server 2005 数据库中的 db_LinQ 数据库，然后将表 tb_info 映射到 LinqDB.dbml 中（可以将 tb_info 表拖曳到设计视图中），如图 28.9 所示。

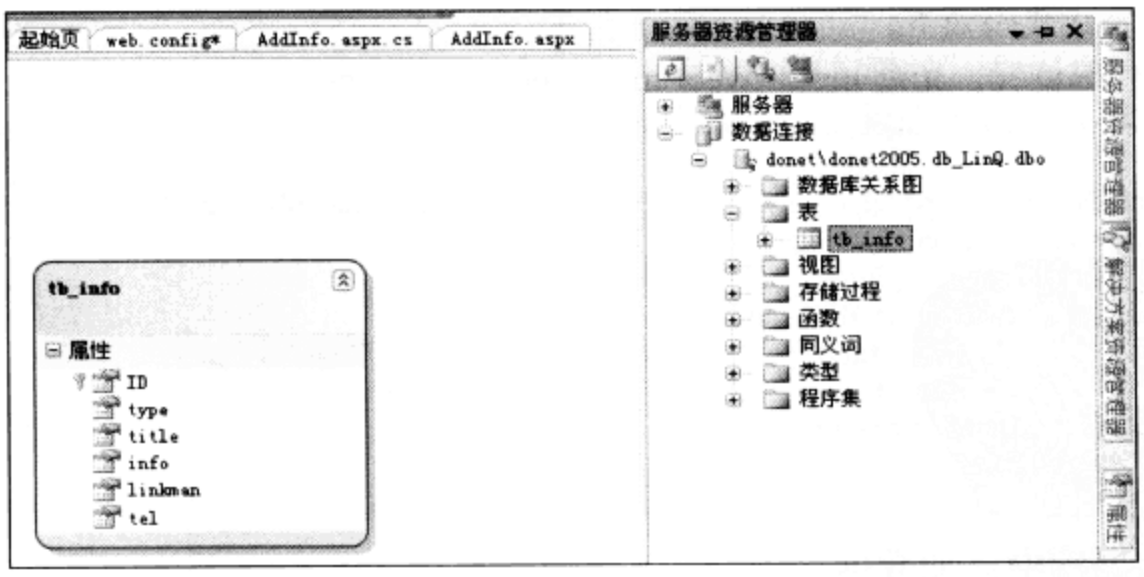


图 28.9 映射 tb_info 数据表

（5）LinqDB.dbml 文件创建一个名称为 LinqDBDataContext 的数据上下文类，为 db_LinQ 数据库提供查询或操作数据库的方法，LINQ 数据源创建完毕。LinqDBDataContext 类中的程序代码均自动生成，如图 28.10 所示。

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.Linq;
using System.Data.Linq.Mapping;
using System.Linq;
using System.Linq.Expressions;
using System.Reflection;


[System.Data.Linq.Mapping.DatabaseAttribute(Name="db_LinQ")]
public partial class LinqDBDataContext : System.Data.Linq.DataContext
{

    private static System.Data.Linq.Mapping.MappingSource mappingSource = new AttributeMappingSource();

  Extensibility Method Definitions

    public LinqDBDataContext() :
            base(global::System.Configuration.ConfigurationManager.ConnectionStrings["db_LinQConnectionSt
    public LinqDBDataContext(string connection) :
            base(connection, mappingSource)
    public LinqDBDataContext(System.Data.IDbConnection connection) :
            base(connection, mappingSource)
```

图 28.10　LinQDBDataContext 类中自动生成程序代码

28.3.1　查询数据库中数据

使用 LINQ to SQL 可以查询数据库中的数据，与传统的 SQL 语句或存储过程相比这种方法更加简捷。下面使用 LINQ to SQL 查询 db_LinQ 数据库中 tb_info 数据表中的数据，操作详细步骤如下。

（1）创建一个 Web 窗体，命名为 FindInfo.aspx，在 Web 窗体中添加 1 个 GridView 控件。

（2）在页面的 Page_Load 事件下编写 LINQ 查询代码。

（3）声明 LinqDBDataContext 类对象 lqDB。

（4）使用 LINQ 查询表达式查询 ID 大于 0 的查询结果，并将查询结果保存到 result 变量中。

（5）将 result 变量中存储的结果作为 GridView 控件的数据源，并且绑定数据显示查询结果，如图 28.11 所示。

ID	type	title	info	linkman	tel
1	类型	求职信息	内容	联系人	电话
478	企业广告	AD	ASADADADADADADADADADADADAD	AD	0431-12345678
479	招聘信息	招聘 ASP.NET软件工程师	具体要求： 2年以上工作经验。	张经理	0000-0000000

图 28.11　LINQ 查询结果

（6）实现完成代码如下。

例程 1　代码位置：光盘\mr\28\WebLinQ\ FindInfo.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    LinqDBDataContext lqDB = new LinqDBDataContext(ConfigurationManager.Connection Strings["db_LinQ
ConnectionString"].ConnectionString.ToString());
    tb_info info = new tb_info();
    var result = from f in lqDB.tb_info
                 where f.ID > 0
                 select f;
    GridView1.DataSource = result;
    GridView1.DataBind();
}
```

28.3.2　向数据库中添加数据

使用 LINQ to SQL 不仅可以实现查询数据库中的数据，而且还能够实现向数据库中添加数据。下面使用 LINQ to SQL 向 db_LinQ 数据库 tb_info 数据表中添加数据。实现该功能主要通过 Table<T>类的 InsertOnSubmit()方法和 SubmitChanges ()方法实现，其中，InsertOnSubmit 方法将单个实体的集合添加到 Tabel<T>类的实例中，SubmitChanges 方法计算要插入、更新或删

除的已修改对象的集，并执行相应命令以实现对数据库的更改。

实现使用 LINQ 向数据库中添加数据详细步骤如下。

（1）创建一个 Web 窗体，命名为 AddInfo.aspx，在 Web 窗体中添加相应数量的 TextBox 控件、DropDownList 控件及按钮控件。

（2）在页面中按钮控件的 Click 事件下编写 LINQ 添加数据代码。

（3）声明 LinqDBDataContext 类对象 lqDB。

（4）声明实体类对象 info，并设置该类对象中实体属性，为实体属性赋值。

（5）调用 InsertOnSubmit()方法将实体类对象 info 添加到 lqDB 对象的 tb_Info 表中，其次再调用 SubmitChanges 方法将实体类中数据添加到数据库中，如图 28.12 所示。

图 28.12 LINQ 查询结果

（6）实现完成代码如下。

例程 2 代码位置：光盘\mr\28\WebLinQ\AddInfo.aspx.cs

```
protected void imgBtnAdd_Click(object sender, ImageClickEventArgs e)
{
    LinqDBDataContext lqDB = new LinqDBDataContext(ConfigurationManager.ConnectionStrings["db_LinQ
ConnectionString"].ConnectionString.ToString());
    tb_info info = new tb_info();
    //要添加的内容
    info.type = DropDownList1.Text;
    info.title = txtTitle.Text;
    info.info = txtInfo.Text;
    info.linkman = txtLinkMan.Text;
    info.tel = txtTel.Text;
    //执行添加
    lqDB.tb_info.InsertOnSubmit(info);
    lqDB.SubmitChanges();
}
```

28.3.3 修改数据库中数据

在前两小节中讲解了使用 LINQ to SQL 实现查询数据库中的数据和向数据库中添加数据。下面使用 LINQ to SQL 修改 db_LinQ 数据库 tb_info 数据表中数据。

使用 LINQ 修改数据库中数据的设计思路和实现步骤如下。

（1）创建一个 Web 窗体，命名为 UpdateInfo.aspx。

（2）在页面的 Load 事件下编写 LINQ 修改数据代码。

（3）声明 LinqDBDataContext 类对象 lqDB。

（4）使用 LINQ 查询将要修改的数据，并设置要修改的值。

（5）调用 SubmitChanges 方法将修改的数据保存到数据库中。数据修改前后如图 28.13 和图 28.14 所示。

（6）实现代码如下。

例程 3 代码位置：光盘\mr\28\WebLinQ\UpdateInfo.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    LinqDBDataContext lqDB = new LinqDBDataContext(ConfigurationManager.ConnectionStrings["db_LinQ
ConnectionString"].ConnectionString.ToString());
    //查询要修改的数据
    var result = from u in lqDB.tb_info
                 where u.ID == 478
                 select u;
    //设置修改该数据
    foreach (tb_info info in result)
    {
        info.info = "招聘若干名多媒体广告业务人员，有经验者优先录用！";
    }
    //将修改的数据保存到数据库中
    lqDB.SubmitChanges();
}
```

表 - dbo.tb_info

	ID	type	title	info	linkman	tel
	1	类型	求职信息	内容	联系人	电话
	478	企业广告	AD	ASADADADAD...	AD	0431-12345678
	479	招聘信息	招聘 ASP.NET...	具体要求： ...	张经理	0000-0000000
▶*	NULL	NULL	NULL	NULL	NULL	NULL

图 28.13 数据修改前

表 - dbo.tb_info

	ID	type	title	info	linkman	tel
	1	类型	求职信息	内容	联系人	电话
	478	企业广告	AD	招聘若干名多媒体广告业务人员...	AD	0431-12345678
	479	招聘信息	招聘 ASP.NET...	具体要求： 2年以上工作经验。	张经理	0000-0000000
▶*	NULL	NULL	NULL	NULL	NULL	NULL

图 28.14 数据修改后

28.3.4 删除数据库中数据

在前几节中已经讲解了使用 LINQ to SQL 实现查询数据库中的数据、添加数据库中数据和修改数据库中数据，那么本节将讲解使用 LINQ to SQL 实现删除数据库中的数据。

实现具体步骤如下。

（1）创建一个 Web 窗体，命名为 DeleteInfo.aspx。

（2）在页面的 Load 事件下编写 LINQ 修改数据代码。

（3）声明 LinqDBDataContext 类对象 lqDB。

（4）使用 LINQ 查询将要删除的数据，将查询结果保存到 result 变量中。

（5）调用 DeleteAllOnSubmit 方法和 SubmitChanges 方法，删除指定的数据并提交到数据库中。删除数据前后如图 28.15 和图 28.16 所示。

（6）实现代码如下。

例程 4 代码位置：光盘\mr\28\WebLinQ\DeleteInfo.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
```

```
            LinqDBDataContext lqDB = new LinqDBDataContext(ConfigurationManager.ConnectionStrings["db_LinQ
ConnectionString"].ConnectionString.ToString());
            //查询要删除的数据
            var result = from u in lqDB.tb_info
                         where u.ID == 1
                         select u;
            //删除数据，并提交数据库中
            lqDB.tb_info.DeleteAllOnSubmit(result);
            lqDB.SubmitChanges();
        }
```

表 - dbo.tb_info

	ID	type	title	info	linkman	tel
▶	1	类型	求职信息	内容	联系人	电话
	478	企业广告	AD	招聘若干名多媒体广告业务人员...	AD	0431-12345678
	479	招聘信息	招聘 ASP.NET...	具体要求： 2年以上工作经验。	张经理	0000-0000000
*	NULL	NULL	NULL	NULL	NULL	NULL

图 28.15　数据删除前

表 - dbo.tb_info

	ID	type	title	info	linkman	tel
▶	478	企业广告	AD	招聘若干名多媒体广告业务人员...	AD	0431-12345678
	479	招聘信息	招聘 ASP.NET...	具体要求： 2年以上工作经验。	张经理	0000-0000000
*	NULL	NULL	NULL	NULL	NULL	NULL

图 28.16　数据删除后

28.4　LINQ 查询结果绑定到 DropDownList 控件

在 28.3 节中讲解了使用 LINQ 查询和操作数据库，那么本节介绍将 LINQ 查询的结果绑定到 DropDownList 控件中。

LINQ 查询结果绑定到 DropDownList 控件的设计思路和实现步骤如下。

（1）创建 LINQ 数据源文件，在这里使用 28.3 节中的 LinqDB.dbml 数据源文件。

（2）创建一个 Web 窗体，命名为 DropDownList.aspx，在 Web 窗体中主要添加一个 DropDownList 控件。

（3）在页面的 Page_Load 事件下编写实现该功能的代码。

（4）声明 LinqDBDataContext 类对象 lqDB。

（5）创建 LINQ 查询表达式，并将查询结果保存到 result 变量中。

（6）将 result 变量中存储的结果设置 DropDownList 控件的数据源，并且显示查询结果，如图 28.17 所示。

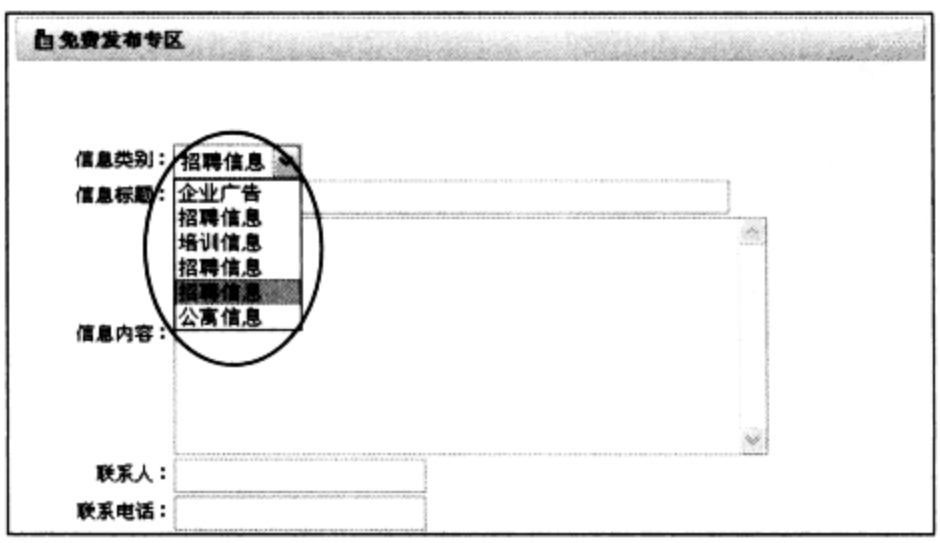


图 28.17　LINQ 查询结果绑定到 DropDownList 控件

（7）实现代码如下。

例程 5 代码位置：光盘\mr\28\WebLinQ\DropDownList.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    LinqDBDataContext lqDB = new LinqDBDataContext(ConfigurationManager.ConnectionStrings["db_LinQ
ConnectionString"].ConnectionString.ToString());
    //查询结果
    var result = from u in lqDB.tb_info
                 where u.ID > 1
                 select new
                 {
                     Type = u.type
                 };
    //设置绑定字段
    DropDownList1.DataTextField = "Type";
    //绑定查询结果
    DropDownList1.DataSource = result;
    DropDownList1.DataBind();
}
```

28.5 LINQ 查询结果绑定 GridView 控件

在 28.3 节中讲解了使用 LINQ 查询和操作数据库，那么本节介绍将 LINQ 查询的结果绑定到 GridView 控件中。

LINQ 查询结果绑定到 GridView 控件的设计思路和实现步骤如下。

（1）创建 LINQ 数据源文件，在这里使用 28.3 节中的 LinqDB.dbml 数据源文件。

（2）创建一个 Web 窗体，命名为 GridView.aspx，在 Web 窗体中主要添加一个 GridView 控件。

（3）在页面的 Page_Load 事件下编写实现该功能的代码。

（4）声明 LinqDBDataContext 类对象 lqDB。

（5）创建 LINQ 查询表达式，在查询表达式中，既可以创建中文字段，也可以创建英文字段，创建的字段类型将会在绑定 GridView 控件中，如图 28.18 所示，最后将查询结果保存到 result 变量中。

（6）将 result 变量中存储的结果设置 GridView 控件的数据源，并且显示查询结果，如图 28.18 所示。

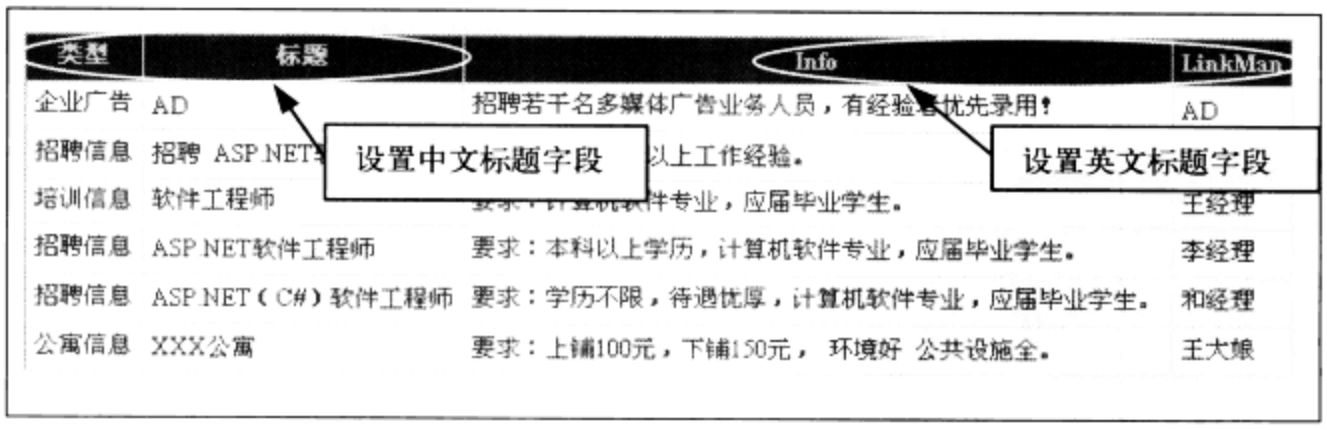


图 28.18 LINQ 查询结果绑定到 GridView 控件

（7）实现代码如下。

例程 6 代码位置：光盘\mr\28\WebLinQ\GridView.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    LinqDBDataContext lqDB = new LinqDBDataContext(ConfigurationManager.ConnectionStrings["db_LinQ
```

```
ConnectionString"].ConnectionString.ToString());
            //创建查询表达式
            var result = from u in lqDB.tb_info
                         where u.ID > 1
                         select new
                         {
                             类型=u.type,
                             标题=u.title,
                             Info=u.info,
                             LinkMan=u.linkman
                         };
            //绑定查询结果
            GridView1.DataSource = result;
            GridView1.DataBind();
        }
```

28.6 LINQ 查询结果绑定 DataList 控件

在 28.3 节中讲解了使用 LINQ 查询和操作数据库，那么本节介绍将 LINQ 查询的结果绑定到 DataList 控件中。

LINQ 查询结果绑定到 DataList 控件的设计思路和操作步骤如下。

（1）创建 LINQ 数据源文件，在这里使用 28.3 节中的 LinqDB.dbml 数据源文件。

（2）创建一个 Web 窗体，命名为 DataList.aspx，在 Web 窗体中主要添加一个 DataList 控件，并且编辑 DataList 控件模板，在编辑模板项中使用 DataBinder.Eval 方法绑定数据显示字段，实现代码如下。

例程 7 代码位置：光盘\mr\28\WebLinQ\DataList.aspx

```
<asp:DataList ID="DataList1" runat="server">
    <ItemTemplate>
        <table align="center" cellpadding="0" cellspacing="0" width="543">
        <tr>
        <td class="huise">
        <span class="shenlancu">『<%# DataBinder.Eval(Container.DataItem,"type") %>』</span><span
          class="banhui"><%# DataBinder.Eval(Container.DataItem,"title") %>
        <br />
</span><span class="banhui">   <%# DataBinder.Eval(Container.DataItem,"info") %>
        <br />
</span><span class="chengse">联系人：<%#DataBinder.Eval(Container.DataItem,"linkMan") %>
   联系电话：<%#DataBinder.Eval(Container.DataItem,"tel") %></span></td>
        </tr>
        <tr style="color: #000000">
        <td align="center">
        <img height="1" src="images/longline.gif" width="525" /></td>
        </tr>
        <tr style="color: #000000">
        <td height="10">
        </td>
        </tr>
    </table>
    </ItemTemplate>
</asp:DataList>
```

（3）在页面的 Page_Load 事件下编写实现 LINQ 查询结果绑定 GridView 控件功能的代码。

（4）声明 LinqDBDataContext 类对象 lqDB。

（5）创建 LINQ 查询表达式，将查询结果保存到 result 变量中。

（6）将 result 变量中存储的结果设置 DataList 控件的数据源，并且显示查询结果，如

图 28.19 所示。

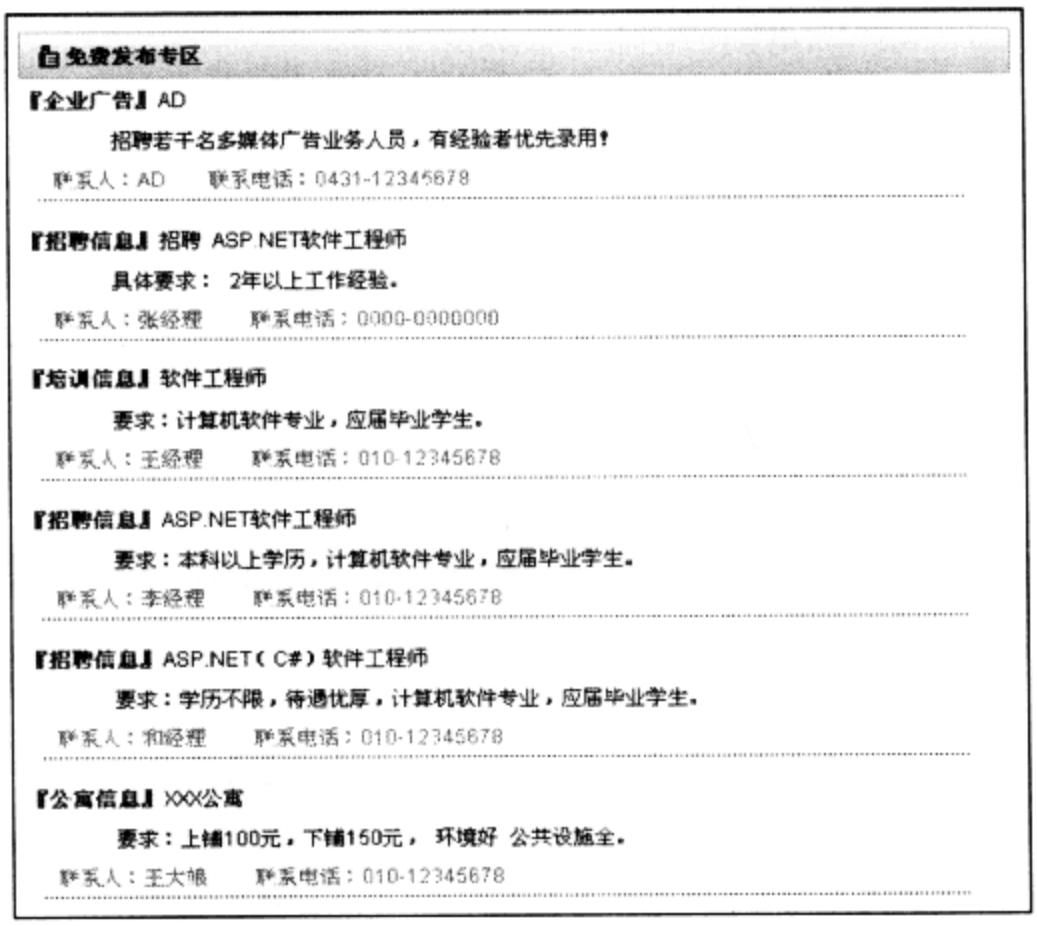


图 28.19 LINQ 查询结果绑定到 GridView 控件

（7）实现代码如下。

例程 8 代码位置：光盘\mr\28\WebLinQ\DataList.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    LinqDBDataContext lqDB = new LinqDBDataContext(ConfigurationManager.ConnectionStrings["db_LinQ
ConnectionString"].ConnectionString.ToString());
    //创建查询表达式
    var result = from u in lqDB.tb_info
                 where u.ID > 1
                 select new
                 {
                     Type = u.type,
                     Title = u.title,
                     Info = u.info,
                     LinkMan = u.linkman,
                     Tel=u.tel
                 };
    //绑定查询结果
    DataList1.DataSource = result;
    DataList1.DataBind();
}
```

第29章 安全技术

实例位置：光盘\mr\29\

29.1 Web.config 加密与解密

29.1.1 认识 Web.config 配置文件

1. 什么是 Web.config 文件

Web.config 文件是一个 XML 文本文件，用来储存 ASP.NET 中 Web 应用程序的配置信息（如最常用的设置 ASP.NET Web 应用程序的身份验证方式），还可以出现在应用程序的每一个目录中。默认情况下会在根目录自动创建一个默认的 Web.config 文件，包括默认的设置，所有的子目录都继承它。如果想修改子目录的设置，可以在该子目录下新建一个 Web.config 文件。它可以提供除从父目录继承的配置信息以外的配置信息，也可以重写或修改父目录中定义的设置。

在运行时对 Web.config 文件的修改不需要重启就可以生效（注：<processModel>节例外）。当然 Web.config 文件是可以扩展的。可以自定义新配置参数并编写配置节处理程序以对它们进行处理。

2. Web.config 配置文件（默认的配置设置）

以下所有的代码都应该位于：

```
<configuration>
<system.web>
```

和

```
</system.web>
</configuration>
```

之间。出于学习的目的，下面示例都省略了这段 XML 标记。

（1）<authentication>节。

作用：配置 ASP.NET 身份验证支持（Windows、Forms、PassPort 和 None）。该元素只能在计算机、站点或应用程序级别声明。<authentication>元素必须与</authorization>节配合使用。

示例：以下为基于窗体（Forms）的身份验证配置站点，用于当没有登录的用户访问该网页时，要求通过身份验证，同时网页自动跳转到登录网页。

```
<authentication mode="Forms" >
<forms loginUrl="logon.aspx" name=".FormsAuthCookie"/>
</authentication>
```

元素 loginUrl 表示登录网页的名称，name 表示 Cookie 的名称。

（2）<authorization>节。

作用：控制对 URL 资源的客户端访问（如允许匿名用户访问）。此元素可以在任何级别（计算机、站点、应用程序、子目录或页）上声明，必须与</authentication>首尾搭配使用。

示例，下面代码用于禁止匿名用户的访问。

```
<authorization>
<deny users="?"/>
</authorization>
```

注意

可使用 user.identity.name 来获取已经通过验证的当前用户名，可使用 web.Security.FormsAuthentication.RedirectFromLoginPage 方法将已验证的用户重定向到用户刚才请求的页面。

（3）＜compilation＞节。

作用：设置 ASP.NET 使用的所有编译。默认的 debug 属性为“True”。在程序编译完成交付使用之后应将其设为 True（Web.config 文件中有详细说明，此处省略示例）。

（4）＜customErrors＞节。

作用：为 ASP.NET 应用程序提供有关自定义错误信息的信息，它不适用于 XML Web Services 中发生的错误。

示例：当发生错误时，将网页跳转到自定义的错误页面。

```
<customErrors defaultRedirect="ErrorPage.aspx" mode="RemoteOnly">
</customErrors>
```

其中，元素 defaultRedirect 表示自定义的错误网页的名称。mode 元素表示对不在本地 Web 服务器上运行的用户显示自定义（友好的）信息。

（5）＜httpRuntime＞节。

作用：设置 ASP.NET HTTP 运行库。该节可以在计算机、站点、应用程序和子目录级别声明。

示例：控制用户上传文件最大为 4MB，最长时间为 60s，最多请求数为 100。

```
<httpRuntime maxRequestLength="4096" executionTimeout="60" appRequestQueueLimit="100"/>
```

（6）＜pages＞节。

作用：设置标识特定于页的配置（如是否启用会话状态、视图状态、是否检测用户的输入等）。＜pages＞可以在计算机、站点、应用程序和子目录级别声明。

示例：不检测用户在浏览器输入的内容中是否存在潜在的危险数据（注：该项默认是检测，如果你使用了不检测，一定要对用户的输入进行编码或验证），从客户端回发页时将检查加密的视图状态，以验证视图状态是否已在客户端被篡改（注：该项默认是不验证）。

```
<pages buffer="true" enableViewStateMac="true" validateRequest="false"/>
```

（7）＜sessionState＞节。

作用：设置为当前应用程序配置会话状态（如设置是否启用会话状态，会话状态保存位置）。

示例：

```
<sessionState mode="InProc" cookieless="true" timeout="20"/>
</sessionState>
```

说明

（1）mode="InProc"表示在本地储存会话状态（你也可以选择储存在远程服务器、SAL 服务器中或不启用会话状态）。

（2）cookieless="true"表示如果用户浏览器不支持 Cookie 时启用会话状态（默认为 False）。

（3）timeout="20"表示会话可以处于空闲状态的分钟数。

（8）＜trace＞节。

作用：配置 ASP.NET 跟踪服务，主要用来判断程序哪里出现了错误。

示例：以下为 Web.config 中的默认配置：

```
<trace enabled="false" requestLimit="10" pageOutput="false" traceMode="SortByTime" localOnly="true" />
```

说明

（1）enabled="false"表示不启用跟踪。

（2）requestLimit="10"表示指定在服务器上存储的跟踪请求的数目。

（3）pageOutput="false"表示只能通过跟踪实用工具访问跟踪输出。

（4）traceMode="SortByTime"表示以处理跟踪的顺序来显示跟踪信息。

（5）localOnly="true"表示跟踪查看器（trace.axd）只用于宿主 Web 服务器。

3. 自定义 Web.config 文件配置节

自定义 Web.config 文件配置节过程分为两步。

（1）在配置文件顶部＜configSections＞和＜/configSections＞标记之间声明配置节的名称和处理该节中配置数据的.NET Framework 类的名称。

（2）在＜configSections＞区域之后为声明的节做实际的设置。

示例：创建一个节存储数据库连接字符串。

```
<configuration>
<configSections>
<section name="appSettings" type="System.Configuration.NameValueFileSectionHandler, System, Version=1.0.3329.0,
Culture=neutral, PublicKeyToken=b77a5c561934e089"/>
</configSections>
  <appSettings>
    <add key="conStr"    value="Server=(local);database=db_news;Uid=sa;Pwd=" />
  </appSettings>
<system.web>
……//省略部分自动生成的代码
</system.web>
</configuration>
```

配置 Web.Config 文件可以在 IIS“站点属性”\“ASP.NET”中选择“编辑全局配置”按钮，即可配置 Web.Config 文件，如图 29.1 所示。

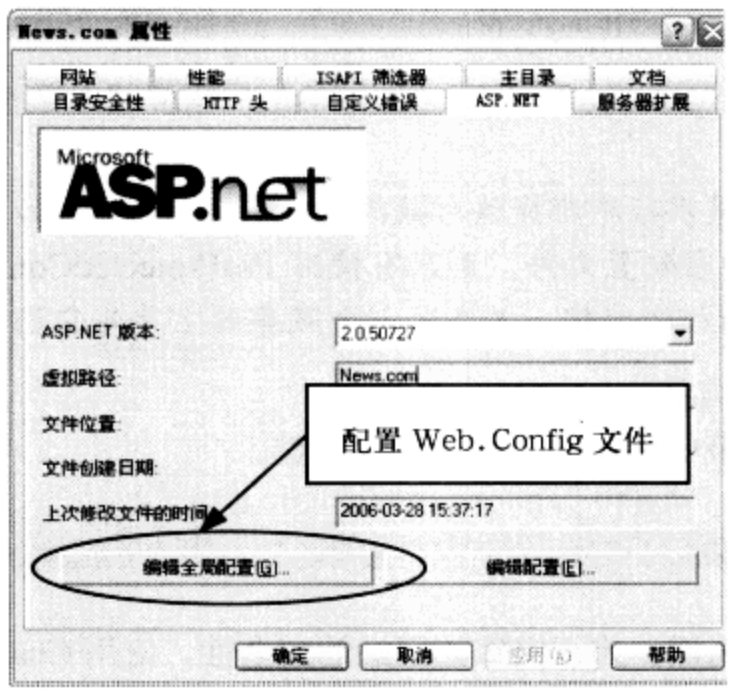


图 29.1　配置 Web.Config 文件

4. 访问 Web.config 文件

用户可以通过使用 ConfigurationSettings.AppSettings 静态字符串集合来访问 Web.config 文件。示例：获取上面例子中建立的连接字符串。

```
string str = ConfigurationManager.AppSettings["conStr"];
SqlConnection con = new SqlConnection(str);
```

29.1.2　Web.config 文件加密与解密的意义

上一节相信读者已经认识了 Web.config 文件，它是标准 XML 格式的文本文件，包含标准的 XML 文档元素。但由于 XML 语法结构简单易懂，出于对应用程序维护和修改考虑，Web 应用程序开发者往往会将应用程序的一些关键信息配置在 Web.config 文件中，例如，数据库连

接字符串。然而，Web.config 文件也存在安全性隐患，这种以明码方式存储的关键信息，可能会成为别有用心的人的突破口，造成不必要的损失。目前，解决这个问题最好的方式是将 Web.config 文件的字符串进行加密，当要修改信息时再对 Web.config 文件进行解密。

29.1.3 使用 SectionInformation 类实现加密与解密

System.Web.Configuration.SectionInformation 类对一个配置节的描述进行了抽象，使用它可以对 Web.config 文件进行加密或解密，下面详细介绍如何使用该类对 Web.config 文件进行加密与解密。

1. 语法讲解

如果要加密一个配置节，只需要简单地使用 SectionInformation 类的 ProtectSection（提供程序）方法，传递想要使用的提供程序的名字来执行加密。当需要解密文件配置节时只需调用 SectionInformation 类的 UnprotectSection 方法即可轻松完成解密。

（1）ProtectSection 方法。

此方法对 Web.config 文件中标记的节进行加密。

语法如下：

```
public void ProtectSection(string protectionProvider)
```

参数说明如下。

- protectionProvider：要使用的保护提供程序的名称。默认情况下将包含以下保护提供程序加密。
- DPAPIProtectedConfigurationProvider，使用 Windows 数据保护 API（DPAPI）数据进行加密和解密。
- RSAProtectedConfigurationProvider 使用 RSA 加密算法对数据进行加密和解密。

注意

这两个提供程序都提供对数据的强加密，但是，如果打算在多台服务器（如网络场）上使用同一个加密配置文件，则只有使用 PsaProtectedConfigurationProvider 才能导出用于对数据进行加密的加密密钥，并在另一台服务器上导入它们。

（2）UnprotectSection 方法。

此方法从关联的配置节中移除受保护的配置加密。

语法如下：

```
public void UnprotectSection()
```

2. 实现过程

以开发电子商务网站为例，为了保障其网站的安全性，使用 ConfigurationSection 类加密了网站中的数据连接字符串。这里主要实现的是加密和解密 Web.config 文件中的数据库连接字符串。在 ASP.NET 应用程序中，从工具箱中将两个 ImageButton 控件拖放到 Default.aspx 窗体中，运行效果如图 29.2 所示。

图 29.2 加密、解密

程序步骤如下。

（1）在“加密 Web.config”按钮的单击事件中编写加密代码，加密 Web.config 文件中“appSetting”节中的数据，主要代码如下。

例程 1 代码位置：光盘\mr\29\Webconfig\ Default.aspx.cs

```
protected void ImageButton2_Click(object sender, ImageClickEventArgs e)
{
    // 打开该应用程序中的配置文件
    Configuration config = WebConfigurationManager.OpenWebConfiguration(Request.ApplicationPath);
```

```
        //在打开的配置文件中找到指定的节
        ConfigurationSection section = config.GetSection("appSettings");
        if (section != null && !section.SectionInformation.IsProtected)
        {
            section.SectionInformation.ProtectSection("RsaProtectedConfigurationProvider");
            //保存加密后的文件
            config.Save();
            //加密成功给出成功提示
            Response.Write("<script>alert('加密成功!');</script>");
        }
    }
```

注 意

这里需要引入 using System.Web.Configuration 命名空间。

（2）在“解密 Web.config”按钮的单击事件中编写解密代码，解密 Web.config 文件中“appSettings”节中的数据，主要程序代码如下。

例程 2 代码位置：光盘\mr\29\Webconfig\ Default.aspx.cs

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{// 解密
    Configuration config = WebConfigurationManager.OpenWebConfiguration(Request.ApplicationPath);
        ConfigurationSection section = config.GetSection("appSettings");
        if (section != null && section.SectionInformation.IsProtected)
        {
            section.SectionInformation.UnprotectSection();
            config.Save();
            Response.Write("<script>alert('解密成功');</script>");
        }
    }
```

在实际运行效果中，单击“加密 Web.config”按钮加密 Web.config 文件，加密后 Web.config 文件代码如下。

例程 3 代码位置：光盘\mr\29\Webconfig\ web.config

```
<configSections>
  ... //省略前后没有变化的代码段
                    <KeyName>Rsa Key</KeyName>
                  </KeyInfo>
                    <CipherData> <CipherValue>BFpfxMOClGaSoqhksa7nIrfSRjIn9qJbOlUBD4nTF29rMG
GxCies5fEsTgsY9eLUBDEzJLwXsHyOUdPhnAUZvyKXevntCyy85RYvpL8AatbwaayPGgurJ4xsQi48m7MQb7At5kf0Mj
MYkBLjjV50uPrhXbwxCYyduV57txTEFr8=</CipherValue>
                  </CipherData>
                </EncryptedKey>
              </KeyInfo>
              <CipherData> <CipherValue>b6Qw+QFpF3YlGPjORxzraLLcqYd1Rg8gIFdfTz2t/6R2uhqFjgDXZ+xkm
Gpt5cz5GczuXksAM4LK6yc0iMui30p1bG4xbYgVSVoB1yLvlFYZV2nCFbjDbgXh3DZowDxzFEJHL579KvGK0ToI+tu7Gq
9eVpvSC2IRj96FRPIEsC4bZD0RRyN1POlP6ccLZIMxj4Dy9rP5q6IsrlnVSWKccxfpC8fJvAJ+loF1utUaygzlMPZZtiQ6G5LY
I1K3bOU+oB2n7mC7cnu+fz/EvMUBm+eavVTXunUTn47PmV2i56ldOY3F4O6ocpCAShtDR796FellDizcKE7bI2B1VtALf
KXpAaB/rxfipILXN0MlXDnMNMAfSuPW6lo6u0vd0X6x/tTdYidzbu0=</CipherValue>
              </CipherData>
            </EncryptedData>
        </appSettings>
    <connectionStrings/>
```

（3）单击“解密 Web.config”按钮，解密 Web.config 文件，解密后 Web.config 主要代码如下。

例程 4 代码位置：光盘\mr\29\Webconfig\ web.config

```
<configSections>
    ... //省略前后没有变化的代码段
</configSections>
```

```
<appSettings>
    <add key="Constring08" value="server=(loclhost);uid=sa;pwd=;database=db_08;" />
    <add key="Constring09" value="server=(loclhost);uid=sa;pwd=;database=db_09;" />
    <add key="Constring07" value="server=(loclhost);uid=sa;pwd=;database=db_07;" />
  </appSettings>
<connectionStrings/>
```

说 明 ASP.NET 在处理 Web.config 文件时会自动对该文件的内容进行解密。因此，不需要任何附加步骤即可对已加密的设置进行解密，供其他 ASP.NET 功能使用或用于访问代码中的值。

29.1.4 命令行工具 aspnet_regiis.exe 实现加密与解密

命令行工具 aspnet_regiis.exe，是一个类似于 DOS 的命令工具，称之为命令行解释器，该命令可以执行程序并在屏幕上显示输出。下面介绍如何使用 aspnet_regiis.exe 命令行工具加密和解密 Web.config 文件中的数据库连接字符串。

1. 语法讲解

使用 aspnet_regiis.exe 命令行工具加密与解密 Web.config 文件中的数据库连接字符串时，只需简单地按语法输入命令即可。

（1）加密语法

```
aspnet_regiis –pef   "connctionStrings"   "Path"
```

（2）解密语法

```
aspnet_regiis –pdf   "connctionStrings"   "Path"
```

注 意 上述加密和解密语法在输入时每个语句间要有空格。

参数说明如下。

- –pef：表示根据文件绝对路径执行加密配置节。
- –pdf：表示根据文件绝对路径执行解密配置节。
- ConnctionStrings：表示要加密或解密的配置点名称（注意：并不是连接字符串的名称）。
- Path：表示 Web.config 文件所在的绝对路径。

2. 实现过程

加密 Web.config 文件中的<connectionStrings>与<appSetting>节中的字符串，步骤如下。

（1）打开命令提示符，输入命令提示符 aspnet_regiis -pef" connecti onStirngs" "E:\ASP.NET 典型模块\Web.config 加密与解密"命令后，按下<回车>键，运行效果如 图 29.3 所示。

图 29.3 加密 connectionStings 节字符

（2）打开命令提示符，输入命令提示符 aspnet_regiis -pef" appSettings" "E:\ASP.NET 典型模块\Web.config 加密与解密"命令后，按下<回车>键，步骤如图 29.4 所示。

图 29.4　加密 appSettings 节字符串

加密后的 Web.config 文件主要代码如下。

例程 5　代码位置：光盘\mr\29\ Web.config 命令行\ web.config

```
<connectionStrings configProtectionProvider="RsaProtectedConfigurationProvider">
  ... //省略前后没有变化的代码段
          <KeyName>Rsa Key</KeyName>
        </KeyInfo>
        <CipherData> <CipherValue>cwcfUxO63DJlvQIxfyvBx5lqDLCukPtSynPWB5vEkzG0xwT1uRXYi3N690qO
KBm4odWbDt0/pxppirbmsBSHPEXZl26YGVnPaZ4U+u81S2uJNpoUvDuztEo/pl7MjYnRWWabmHIWmsmW4GqI62CvwK
LGY/C+9/SnQqiiJb6b308=</CipherValue>
        </CipherData>
      </EncryptedKey>
    </KeyInfo>
    <CipherData> <CipherValue>poXkFkttMRq/Jd pxH8679Sj4v/ViyzYwNfpMPXuiOdfVm5vJZMKONSMtL7S5Q
eXymdzL6UJdgf3/c5G7hPQDae6yqiaANLL6VjhAfZ0i9AQWs+qcRvVCvzTekjYd/UaPyZyWWXV74DemjqFNQj65dnItGW
zUgKpPbeEGh8yAvT3wwRFaHwr8lu0K4pySbu45HIRJUPkg3HEBfLDaOE1eN4gTPe0pkKgWoiaDVau1Xwi6CegRXg0jug=
=</CipherValue>
        </CipherData>
    </EncryptedData>
</connectionStrings>
```

解密 Web.config 文件中的<connectionStirngs>与<appSettings>节中的字符串，步骤如下。

（1）打开命令提示符，输入命令提示符 aspnet_regiis –pdf " connectionStirngs" "E:\ASP.NET 典型模块\Web.config 加密与解密"命令后，按下<回车>键，如图 29.5 所示。

图 29.5　解密 connectionStings 节字符

（2）打开命令提示符，输入命令提示符 aspnet_regiis –pdf " appSettings" "E:\ASP.NET 典型模块\Web.config 加密与解密"命令后，按下<回车>键，如图 29.6 所示。

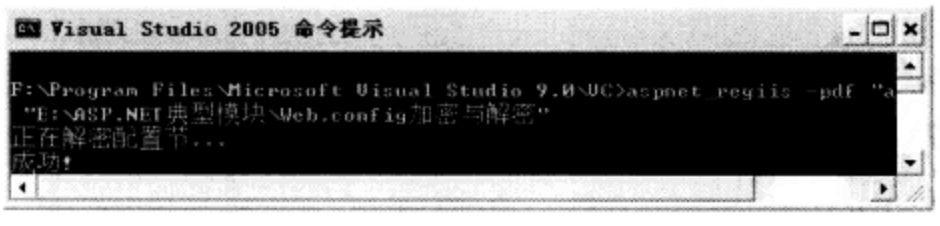


图 29.6　解密 appSettings 节字符串

解密后的 Web.config 文件代码如下。

例程 6　代码位置：光盘\mr\29\ Web.config 命令行\ web.config

```
<configuration>
    <configSections>
<appSettings>
    </appSettings>
    <connectionStrings>
```

```
    <remove name="LocalSqlServer" />
    <add name="LocalSqlServer" connectionString="server=zhy;uid=sa;pwd=;database=mrgwh;" />
</connectionStrings>
```

说明 由于 aspnet_regiis.exe 工具与特定版本的.NET Framework 相关联，因此必须使用适当版本的 aspnet_regiis.ext 工具为 ASP.NET 应用程序重新配置脚本映射。aspnet_regiis.exe 工具将 ASP.NET 应用程序的脚本重新配置为匹配该工具的 ASP.NET ISQPI 扩展版本。

使用此工具也可用来显示所有已安装的 ASP.NET 版本的状态、注册关联的 ASP.NET 版本、创建客户端脚本目录以及执行其他配置操作。

29.2 图文验证技术

29.2.1 图文验证技术概述

1. 验证作用

目前，网络上一些不法分子常常利用机器人程序自动注册、登录、灌水，从而对网站安全性构成了严重威胁。为了进一步保证安全性，越来越多的网站开始采用动态生成的图形码或附加码进行验证。验证码技术就是在服务器端生成一个随机数，并保存在内存中，然后将随机数写入设计好的图片中，发送给浏览器，并以图片形式显示给最终的用户。使用图文验证码后，自动注册程序无法再轻易地注册和破坏。

2. 实现原理

在服务器端，用编程语言生成一个随机数，并保存在内存中，然后将随机数写入设计好的图片中，发送给浏览器，并以图片形式显示给最终用户。

用户在浏览器端输入验证码，然后提交用户名、密码和验证码，服务器端获取用户的提交信息，判断验证码字符和服务器端保存的字符是否相同，如果相同，通过验证判断用户名和密码是否正确，否则提示错误信息，验证流程如图 29.7 所示。

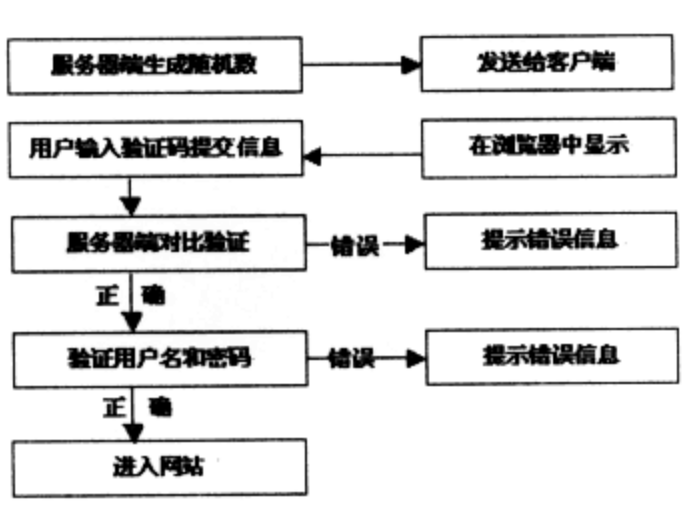


图 29.7 验证码流程图

29.2.2 纯数字验证码

纯数字验证码指的是生成的验证码全部由阿拉伯数字组成，其背景图片由 GDI+绘制出，页面运行效果如图 29.8 所示。

图 29.8 纯数字验证码

1. 创建生成随机数字

在应用程序的 App_Code 文件夹下，创建一个名为 CheckCode 的公共类，在该类中首先用 GDI+绘制出验证码图片，然后应用 Random 类即伪随机数生成器创建生成随机数字，生成随机数代码如下。

例程 7 代码位置：光盘\mr\29\ ValidateWeb \ App_Code\ CheckCode.cs

```
/// <summary>
/// 生成随机的字母
/// </summary>
/// <param name="VcodeNum">生成字母的个数</param>
/// <returns>string</returns>
private string RndNum(int VcodeNum)
{
    string Vchar = "0,1,2,3,4,5,6,7,8,9";
    string[] VcArray = Vchar.Split(',');
    string VNum = ""; //由于字符串很短,就不用StringBuilder了
    int temp = -1; //记录上次随机数值,尽量避免生产几个一样的随机数
    //采用一个简单的算法以保证生成随机数的不同
    Random rand = new Random();
    for (int i = 1; i < VcodeNum + 1; i++)
    {
        if (temp != -1)
        {
            rand = new Random(i * temp * unchecked((int)DateTime.Now.Ticks));
        }
        int t = rand.Next(VcArray.Length);
        if (temp != -1 && temp == t)
        {
            return RndNum(VcodeNum);
        }
        temp = t;
        VNum += VcArray[t];
    }
    return VNum;
}
```

生成验证码图片的代码由于篇幅有限这里不给出，说细代码请读者参见随书附带的光盘。

2. 验证功能设计

验证界面上除了要显示随机数外，还要求用户输入产生的随机数，并对用户的输入做检查，设计步骤如下。

首先在应用程序中创建 1 个 Web 窗体，将其命名为 MumberValidate.aspx。在该界面中拖曳 3 个 TextBox 控件、1 个 ImageButton 控件和 1 个 RequiredFiledValidator 到窗体中，其控件属性及说明如表 29.1 所示。

表 29.1 MumberValidate.aspx 页面用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	用 途
TextBox	txtUserID	无	用于显示用户名
	txtPwd	无	用于显示用户密码
	Validator	无	用于输入用户验证码
RequiredFieldValidator	Validator	无	用于判断验证码输入是否正确
ImageButton	imgBtnLogin	无	用于提交用户信息

绘制验证码的页面完成后，在需要验证码的页面 MumberVolidator.aspx 中利用 Image 控件将其显示出来，Imager 控件显示验证码的 HTML 源代码设置如下。

例程 8 代码位置：光盘\mr\29\ ValidateWeb \ MumberValidate.aspx

```
<img id="Img1" alt="看不清,请点击我!" onclick="this.src=this.src+'?'" src="UserValidator/checkCode.aspx"
style="width: 73px; height: 22px" align="left" />
```

3. 服务器端验证

用户输入验证码后，服务器端要与内存中的验证码进行比较，开发人员根据比较的结果，给予用户不同的提示信息。主要程序代码编写在页面“登录”按钮的 imgBtnLogin_Click 事件中，代码如下。

例程 9 代码位置：光盘\mr\29\ ValidateWeb \ MumberValidate.aspx.cs

```
protected void imgBtnLogin_Click(object sender, ImageClickEventArgs e)
{
    //判断用户输入的验证码是否正确
        if(this.Validator.Text= =Session["CheckCode"].ToString())
    {
        if (this.txtUserID.Text= ="mr"&&this.txtPwd.Text= ="mrsoft")
        {
            Response.Write("<script>alert('恭喜您!登录成功!');Location='MumberValidate.aspx'</script>");
            }
            else
            {
                Response.Write("<script>alert('用户名或密码错误,请重新输入!');Location='MumberValidate.aspx'</script>");
            }
        }
        else
        {
            Response.Write("<script>alert('验证码输入错误,请重新输入!');Location='MumberValidate.aspx'</script>");
            return;
        }
}
```

29.2.3 字母与数字混合验证码

既然是图文验证，当然包括图片和文字。上一节中简单介绍了验证的过程，其中使用的验证方式仅仅是数字，本节将介绍字母与数字混合验证码，来进一步讲解复杂的图文验证的实现过程。实际运行的效果如图 29.9 所示。

图 29.9 字母与数字混合验证码

1. 创建生成随机字符的方法

绘制数字与字母组合的验证码主要是在应用程序的 UserValidator 文件夹下 NumAndLetCheckCode.aspx 窗体页面中实现的。在绘制验证码之前，必须先随机生成字符串，这里虽然是创建随机字符串，但必须将所有的字符或数字罗列出来，这样系统才可以随机抽出一些字符，并根据这些随机的字符使用随机生成器“Random”生成需要的验证码，具体实现代

码如下。

例程 10 代码位置：光盘\mr\29\ ValidateWeb \ UserValidator\ NumAndLetCheckCode.aspx.cs

```
private string GenerateCheckCode()
{
    int number;
    char code;
    string checkCode = String.Empty;
    //生成随机生成器
    Random random = new Random();
    for (int i = 0; i < 4; i++)
    {
        number = random.Next();
        //罗列数字
        if (number % 2 == 0)
            code = (char)('0' + (char)(number % 10));
        else
        //罗列字母
            code = (char)('A' + (char)(number % 26));
        checkCode += code.ToString();
    }
    Response.Cookies.Add(new HttpCookie("CheckCode", checkCode));
    return checkCode;
}
```

2. 创建生成随机图片的方法

字符串生成后，下一步就要将该字符串绘制成图片显示出来，大多数网站都提供有图片背景的验证码，这样才更有效地保障不会被轻意攻破，将字符串绘制成图片并显示出来的主要程序代码如下。

例程 11 代码位置：光盘\mr\29\ ValidateWeb \ UserValidator\ NumAndLetCheckCode.aspx.cs

```
private void CreateCheckCodeImage(string checkCode)
{
    if (checkCode == null || checkCode.Trim() == String.Empty)
        return;
    System.Drawing.Bitmap image = new System.Drawing.Bitmap((int)Math.Ceiling((checkCode.Length * 12.5)), 22);
        Graphics g = Graphics.FromImage(image);
        try
        {
            //生成随机生成器
            Random random = new Random();
            //清空图片背景色
            g.Clear(Color.White);
            //画图片的背景噪音线
            for (int i = 0; i < 2; i++)
            {
                int x1 = random.Next(image.Width);
                int x2 = random.Next(image.Width);
                int y1 = random.Next(image.Height);
                int y2 = random.Next(image.Height);
                g.DrawLine(new Pen(Color.Black), x1, y1, x2, y2);
            }
            Font font = new System.Drawing.Font("Arial", 12, (System.Drawing.FontStyle.Bold));
            System.Drawing.Drawing2D.LinearGradientBrush brush = new System.Drawing.Drawing2D.Linear
GradientBrush(new Rectangle(0, 0, image.Width, image.Height), Color.Blue, Color.DarkRed, 1.2f, true);
            g.DrawString(checkCode, font, brush, 2, 2);
            //画图片的前景噪音点
            for (int i = 0; i < 100; i++)
            {
                int x = random.Next(image.Width);
                int y = random.Next(image.Height);
```

```
                image.SetPixel(x, y, Color.FromArgb(random.Next()));
            }
            //画图片的边框线
            g.DrawRectangle(new Pen(Color.Silver), 0, 0, image.Width - 1, image.Height - 1);
            System.IO.MemoryStream ms = new System.IO.MemoryStream();
            image.Save(ms, System.Drawing.Imaging.ImageFormat.Gif);
            Response.ClearContent();
            Response.ContentType = "image/Gif";
            Response.BinaryWrite(ms.ToArray());
        }
        finally
        {
            //释放系统所占资源
            g.Dispose();
            image.Dispose();
        }
    }
```

3. 实现验证功能

应用字母与数字混合验证码的页面为应用程序中的 NumberAndLetter.aspx 页面，在该页面中利用 Image 控件将其显示出来，Image 控件显示验证的 HTML 源码设置如下。

例程 12 代码位置：光盘\mr\29\ ValidateWeb \ NumberAndLetter.aspx

```
<img id="Img1" alt="看不清,请点击我!" onclick="this.src=this.src+'?'"
src="UserValidator/NumAndLetCheckCode.aspx"style="width: 73px; height: 22px" align="left" />
```


说明

该页面的前台页面设计与“纯数字验证码”页基本上一样，这里由于篇幅有限，不详细做介绍。

双击 NumberAndLetter.aspx 页面中的“登录”按钮，触发其 imgBtnLogin_Click 事件实现验证功能，具体实现代码如下。

例程 13 代码位置：光盘\mr\29\ ValidateWeb \ NumberAndLetter.aspx.cs

```
protected void imgBtnLogin_Click(object sender, ImageClickEventArgs e)
{
    HttpCookie cookie = Request.Cookies["CheckCode"];
    if (cookie.Value == this.Validator.Text.Trim())
    {
        if (this.txtUserID.Text == "mr" && this.txtPwd.Text = = "mrsoft")
        {
            Response.Write("<script>alert('恭喜您!登录成功!');Location='NumberAndLetter.aspx'</script>");
            }
            else
            {
                Response.Write("<script>alert('用户名或密码错误,请重新输入!');Location='NumberAndLetter.aspx'</script>");
            }
        }
        else
        {
            Response.Write("<script>alert('验证码输入错误,请重新输入!');Location='NumberAndLetter.aspx'</script>");
            return;
    }
```


29.2.4 纯汉字验证码

在申请 QQ 号码时，可以看到网页中的验证码是以汉字形式出现的。汉字验证码技术比字母和数字混合验证码技术更为先进。通过汉字验证码，可以更加有效地防止非法用户灌水，实际运行效果如图 29.10 所示。

图 29.10 纯汉字验证码

1. 绘制汉字验证码技术

绘制汉字验证码主要通过生成的汉字区位码将其转换成汉字。区位码是汉字一一对应的编码，用 4 位数字表示，前两位从 01 到 94 称为区码，后两位从 01 到 94 称为位码，一个汉字的前一半是 ASC Ⅱ码位“160 + 区码”的字符，后一半是 ASC Ⅱ码位“160 + 区码”的字符。例如，“房”的区位码是 2331，表示的是区码 23，位码 31，它是由 ASC Ⅱ码位“160 + 23 = 183”和“160 + 31 = 191”的两个字符组成。

在绘制汉字验证码中生成的汉字的汉字区位码是以十六进制进行组合的，因此，程序中定义由 16 个字符组成的数组，以便能够生成随机字符，组成一个十六进制数据，声明数姐如下：

```
//定义一个数组存储汉字编码的组成元素
string[] str = new string[16] { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "a", "b", "c", "d", "e", "f" };
```

2. 绘制汉字验证码功能实现

程序中绘制验证码的功能是在 ChineseCheckCode.aspx 页面中完成的，首先通过 CreateString(int strlength)方法来存储产生的随机汉字区位码，CreateString(int strlength)方法关键代码如下。

例程 14 代码位置：光盘\mr\29\ ValidateWeb \ UserValidator\ ChineseCheckCode.aspx.cs

```
private object[] CreateString(int strlength)
{
    //定义一个数组存储汉字编码的组成元素
    string[] str = new string[16] { "0", "1", "2", "3", "4", "5", "6", "7", "8", "9", "a", "b", "c", "d", "e", "f" };
    Random ran = new Random();  //定义一个随机数对象
    object[] bytes = new object[strlength];
    for (int i = 0; i < strlength; i++)
    {
        //获取区位码第1位
        int ran1 = ran.Next(11, 14);
        string str1 = str[ran1].Trim();
        //获取区位码第2位并防止数据重复
        ran = new Random(ran1 * unchecked((int)DateTime.Now.Ticks) + i);
        int ran2;
        if (ran1 == 13)
        {
        ran2 = ran.Next(0, 7);
    }
    else
    {
        ran2 = ran.Next(0, 16);
    }
    string str2 = str[ran2].Trim();
    //获取区位码第3位
    ran = new Random(ran2 * unchecked((int)DateTime.Now.Ticks) + i);
    int ran3 = ran.Next(10, 16);
    string str3 = str[ran3].Trim();
```

```
        //获取区位码第4位
        ran = new Random(ran3 * unchecked((int)DateTime.Now.Ticks) + i);
        int ran4;
        if (ran3 == 10)
        {
            ran4 = ran.Next(1, 16);
        }
        else if (ran3 == 15)
        {
            ran4 = ran.Next(0, 15);
        }
        else
        {
            ran4 = ran.Next(0, 16);
        }
        string str4 = str[ran4].Trim();
        //定义字节变量存储产生的随机汉字区位码
        byte byte1 = Convert.ToByte(str1 + str2, 16);
        byte byte2 = Convert.ToByte(str3 + str4, 16);
                byte[] stradd = new byte[] { byte1, byte2 };
                //将产生的汉字字节放入数组
                bytes.SetValue(stradd, i);
            }
            return bytes;
        }
```

程序中自定义了方法 GetString(int length)，用来将 CreateString(int strlength)方法产生的汉字区位码解码成中文汉字，GetString(int length)代码如下。

例程 15 代码位置：光盘\mr\29\ ValidateWeb \ UserValidator\ ChineseCheckCode.aspx.cs

```
private string GetString(int length)
{
    Encoding gb = Encoding.GetEncoding("gb2312");
    object[] bytes = CreateString(length);
    //根据汉字字节解码出中文汉字
    string str1 = gb.GetString((byte[])Convert.ChangeType(bytes[0], typeof(byte[])));
    string str2 = gb.GetString((byte[])Convert.ChangeType(bytes[1], typeof(byte[])));
    string str3 = gb.GetString((byte[])Convert.ChangeType(bytes[2], typeof(byte[])));
    string str4 = gb.GetString((byte[])Convert.ChangeType(bytes[3], typeof(byte[])));
    string str = str1 + str2 + str3 + str4;
    Response.Cookies.Add(new HttpCookie("CheckCode", str));
    return str;
}
```

29.3 防盗链技术

29.3.1 盗链对网站的危害

盗链是指服务提供商自己不提供服务的内容，通过技术手段绕过其他有利益的最终用户界面（如广告），直接在自己的网站上向最终用户提供其他服务提供商的服务内容，骗取最终用户的浏览和点击率，受益者不提供或提供很少的资源，而真正的服务提供商却得不到任何收益。

要了解盗链，首先来了解 HTTP 文件的两种主要下载方式。

- URL 方式直接下载，优点是占用服务器资源少、速度快；缺点是不能准确计量下载次数，无法防止盗链，保存在数据库中的文件无法下载，常见格式的文件如.html 直接在浏览器中打开，不能直接下载。
- 二进制数据流输出方式，优点是准确计量下载次数、能防盗链、所有文件格式都能直

接下载而不是打开、保存在数据库中等非文件数据能以文件方式下载等；缺点是占用服务器资源多。

盗链导致被盗链站的直接经济损失、服务器外来压力以及盗链网站非正当手段获得其他利益（包括网友信任度以及广告投资等），网站的空间和流量有大量的损失。为了保障自己站点的安全，为网站创建合理的防盗链模块是一种保障措施。

29.3.2 防盗链的解决措施

发生防盗链的问题，是因为 IIS 在处理浏览器请求的时候只是将动态页面交由 HTTP 处理模块处理，而没有将一些图片和下载文件进行处理。在 ASP.NET 中，主要是利用“HttpHandler”处理来解决盗链。

默认情况下，IIS 只安装了处理“ASPX”、“ASP”等 ASP.NET 框架自带文件格式的处理模块。防盗链接的解决措施就是要在 IIS 中安装处理图片和下载文件的处理模块（HttpHandler）。

注意

防盗链的应对措施并不是不允许其他网站再链接到本站点，而是要求用户必须登录本站点才可以使用站内的图片或文件，这样就解决了数据访问量大而网站访问量不大的问题。

29.3.3 图片资源防盗链下载

1. HttpHandler 处理程序处理图片

首先创建一个“Handler.ashx”处理文件，此文件就是一个继承自“IHttpHander”的处理程序。具体步骤如下。

（1）右键单击网站根目录，在弹出的快捷菜单中选择“添加新项”菜单命令，打开“添加新项”对话框。

（2）选择“一般处理程序“模板，单击”添加“按钮，此时在网站根目录中就会出现一个“Handler.ashx”处理文件，默认生成的文件代码如下。

```
<%@ WebHandler Language="C#" Class="MyHandler" %>
using System;
using System.Web;
public class MyHandler : IHttpHandler {
    public void ProcessRequest (HttpContext context) {
        context.Response.ContentType = "text/plain";
        context.Response.Write("Hello World");
    }
    public bool IsReusable {
        get {
            return false;
        }
    }
}
```

说明

“Handler “类包含两个成员：属性“IsReusable”和“ProcessRequest”方法 。其中“IsReusable”表示 HTTP 请求可以使用当前处理，“ProcessRequest”表示用户自定义的对特殊 HTTP 请求的处理。

（3）在“ProcessRequest”方法中，添加“.jpg”文件的处理程序，同时修改属性“IsReusable”，返回“True”，具体代码如下。

例程 16 代码位置：光盘\mr\29\HandlerPicture\ Handler.ashx

```
public class Handler : IHttpHandler {
    public void ProcessRequest(HttpContext context)
    {
        //判断是否是本地引用，如果是则返回给客户端正确的图片
        //这里的判断就是用到了http请求中所记录的页信息
        //如果是网站，可将“localhost”修改为网站地址
        if (context.Request.UrlReferrer.Host == "localhost")
        {
            //设置客户端缓冲中文件过期时间为0,即立即过期。
            context.Response.Expires = 0;
            //清空服务器端为此会话开辟的输出缓存
            context.Response.Clear();
            //获得文件类型
            context.Response.ContentType = "image/jpg";
            //将请求文件写入到输出缓存中
            context.Response.WriteFile(context.Request.PhysicalPath);
            //将输出缓存中的信息传送到客户端
            context.Response.End();
        }
        //如果不是本地引用，则属于盗链引用，返回给客户端错误的图片
        else
        {
            //设置客户端缓冲中文件过期时间为0，即立即过期
            context.Response.Expires = 0;
            //清空服务器端为此会话开辟的输出缓存
            context.Response.Clear();
            //获得文件类型
            context.Response.ContentType = "image/jpg";
            //将特殊的报告错误的图片文件写入到输出缓存中
            context.Response.WriteFile(context.Request.PhysicalApplicationPath + "error.jpg");
            //将输出缓存中的信息传送到客户端
            context.Response.End();
        }
    }
    public bool IsReusable
    {
        get
        {
            return true;
        }
    }
```

（4）编译“.ashx”文件。“.ashx”文件在 Visual Studio 2008 中并不能自动编译，而不编译应用程序就无法自动识别。解决办法是在网站根目录下添加一个类“Handler”，系统自动将此类添加到“App_Code”目录下，然后将“Handler.ashx”中的所有内容复制到“Handler.cs”中，最后保存编译的代码，系统将自动编译“Handler.cs”中的代码。

2. Web.config 配置文件中注册 HttpHandler

自定义处理程序以后，要在配置文件中进行注册，打开“Web.config”文件，在“system.web”节点下，添加处理程序的注册信息，代码如下。

例程 17 代码位置：光盘\mr\29\HandlerPicture\ Web.Config

```
<httpHandlers>
    <add verb="*" path="*.jpg" type="Handler"/>
</httpHandlers>
```

3. IIS 中配置图片的特殊处理程序

在上述两个小节中在应用程序中已经配置好了处理“.jpg”文件的程序，但浏览器的请求是直接发送给 IIS 的，而不是给应用程序，所以还需要在 IIS 中进行特殊的配置才能使得当浏览器请求“.jpg”文件类型时，IIS 能交由 ASP.NET 处理，具体配置步骤如下。

（1）依次选择“开始”→“设置”→“控制面板”→“管理工具”→“Internet 信息服务(IIS)管理器”选项，弹出“Internet 信息服务(IIS)管理器”窗口，如图 29.11 所示。

（2）在左侧目录树中，右键单击“网站”节点。在弹出的快捷菜单中，选中“属性”菜单命令，打开网站的属性对话框，切换到“主目录”选项卡界面，如图 29.12 所示。

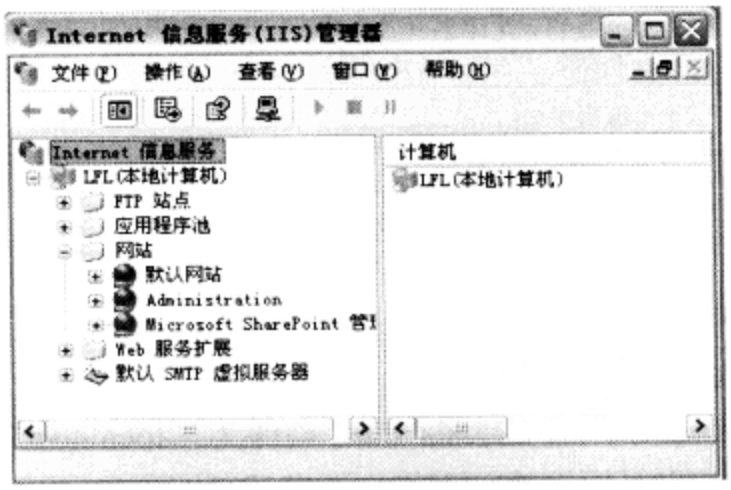


图 29.11　Internet 信息服务管理器

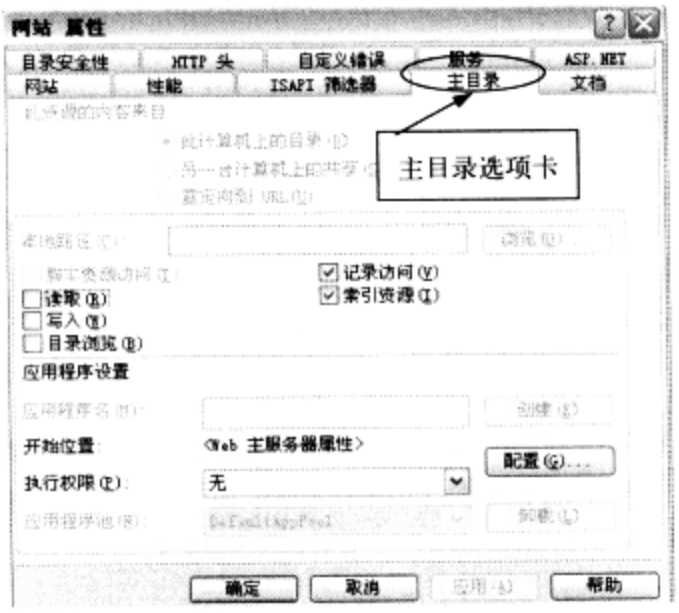


图 29.12　网站属性对话框

（3）单击“配置”按钮，打开“应用程序配置”对话框，如图 29.13 所示。在“应用程序映射”列表框中，罗列出了 IIS 所有能够处理的文件格式，以及这些格式对应的处理程序。在此列表框中，选中“.apsx”条目，单击“编辑”按钮，将打开“添加/编辑应用程序扩展名映射”对话框，如图 29.14 所示。

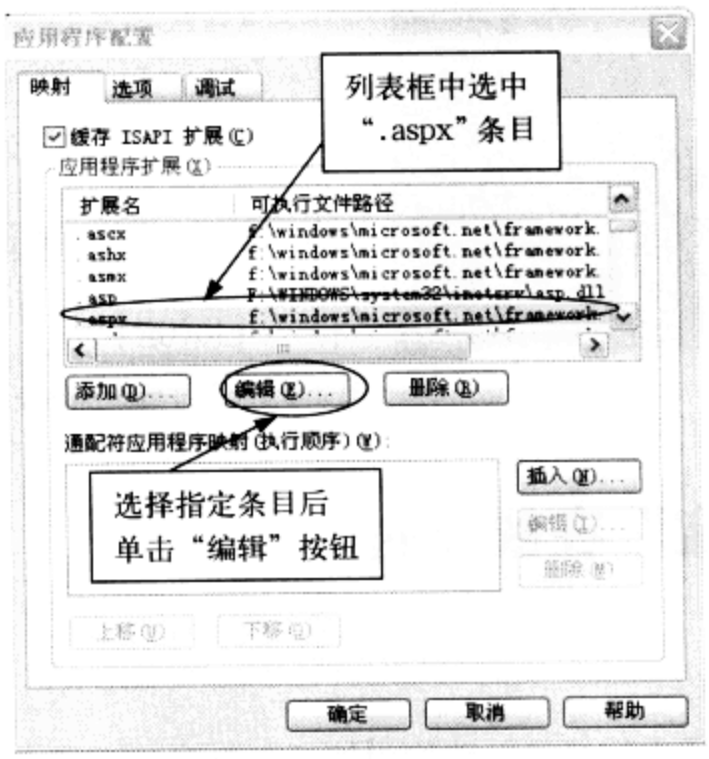


图 29.13　应用程序配置对话框

图 29.14　添加/编辑应用程序扩展名映射

（4）在“可执行文件”文本框中的内容为：“F:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll”，此路径主要是 Visual Studio 2008 软件的安装路径（读者根据本机配置可不同）。其中 aspnet_isapi.dll 是.Net Framework 3.5 的一部分，主要是用来指导 IIS 映射到 Visual Studio 2008 的相应处理程序。在此处，只把其路径复制下来，其他不做任何操作。单击“取消”按钮回到“应用程序配置”对话框。

（5）接下来这一步就是为了让 IIS 能将“JPG”格式的文件正确映射到前面编写的“HttpHandler”处理中，需要为其加载“aspnet_isapi.dll”文件，单击图 29.13（应用程序配置对话框）中的“添加”按钮，打开空白的“添加/编辑应用程序扩展名映射”对话框，如图 29.15 所示。

（6）在“扩展名”文本框中输入“.jpg”。在“可执行文本”文本框中粘贴上步中复制的“F:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\aspnet_isapi.dll”。接下来的步骤只需连续单击“确定”按钮即可。

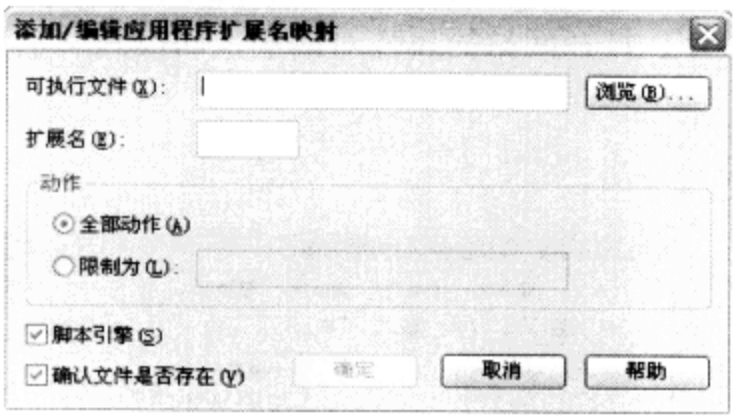


图 29.15　空白的对话框

4. 图片防盗链程序测试

（1）在网站根目录下，添加 2 个“.jpg”文件，一个是合法调用的文件“mrsoft.jpg”，一个是“error.jpg”，错误图像文件。

（2）在网站根目录下创建一个 Web 窗体，命名为 Default.aspx，在该页面中添加 1 个 Image 控件，并将其属性“ImageUrl”设置为“~/mrsoft.jpg”。

（3）运行该 Default.aspx 页，测试“.jpg”文件格式的处理程序。

在“Handler.cs”类中，如果方法“ProcessRequest”中的判断 Host 条件为“localhost”，则 Image 控件正常显示图片“mrsoft.jpg”。合法调用的图片文件如图 29.16 所示。

如果将“localhost”修改为其他网站地址，如 192.168.1.118，则 Image 控件显示的是“error.jpg”图片，表示服务器并不是从本地站点获取图片，如图 29.17 所示。

图 29.16　合法下载的图片资源

图 29.17　图片资源非法下载

29.4　“支付宝”在线支付

29.4.1　支付宝接口概述

支付宝对外接口分为两种：一种是接受外部请求的接口，统称为外部服务接口；另一种是主动通知外部系统的接口，统称为外部通知接口。

外部服务接口的主要目的是让外部合作伙伴主动使用支付宝的服务（如创建交易）。外部通知接口的主要目的是为外部合作伙伴提供数据同步服务（如，交易状态同步）以及异步处理结果返回服务。

- 支付宝的接入 URL：http://www.alipay.com/cooperate/gateway.do。
- 对付宝的接入方式：POST/GET 方式。

29.4.2　应用支付宝实现在线支付

1. 添加支付宝类文件 AliPay

支付宝类 AliPay 是由第三方支付软件提供，在应用程序中需要将这个支付类添加到自己的

网站中才能实现正确的调用关系。添加支付宝类文件 AliPay 步骤如下。

（1）右键单击网站根目录，在弹出的快捷菜单中选择“添加新项”菜单命令，打开“添加新项”对话框。

（2）选择“类”文件图标，单击“添加”按钮，系统会自动提示是否将类放在“App_Code”目录下，选择“是”，待系统生“App_Code”目录后，再删除刚添加类，将第三方提供的支付类 AliPay 放在该目录下即可。

注意

一定要将此类文件放在“App_Code”目录下，因为 Visual Studio 2005 可以自动编辑此“App_Code”目录下的类文件。

2. 设计提交定单功能

添加完主要类文件后，要设计用户选择完商品后的提交界面，具体步骤如下。

（1）在应用程序下添加一个 Web 窗体，命名为 SubmitPayPage.aspx。

（2）支付宝提供的虚拟支付程序的收款方式是支付宝网站，在连接互联网状态下测试该程序时，会真正收取用户存储在支付宝中的金额，所以在设计支付金额时要改尽量小的金额（该设计在开发实际的大型网站时应用）。下面简单设计一个商品订单提交页，界面设计效果如图 29.18 所示。

图 29.18　用户的提交界面

界面中几个属性的说明如表 29.2 所示，其中详细说明了提交界面中的特殊属性设置。

表 29.2　　提交界面的特殊属性说明

属　性	详 细 注 释
卖家账号	默认为支付宝网站管理员的邮箱，如果商品所有人已经是支付宝的注册用户，则此处直接填写其注册邮箱
支付网关	检查付款内容，默认值是支付宝网站的网关
服务参数	一般指接口名称，虚拟程序中指定默认值
合作商	用户注册支付宝几账号时默认生成的 ID 号
支付类型	指明商品是购买、捐赠或者是奖品等

续表

属　　性	详 细 注 释
加密码协议	信息加密类型
安全校验码	用户注册支付宝账号时默认生成的检验信息
重定向地址	用户支付成功后页面跳转的页面
服务器编码	页面的编码格式

注意　如果读者测试此实例，必须先注册为支付宝用户，并修改界面中的“合作商”和“安全校验码”。

（3）双击页面 SubmitPayPage.aspx 中的“支付宝”按钮，触发其 Click 事件，实现添加支付功能的代码如下。

例程 18　代码位置：光盘\mr\29\ BankPay \ SubmitPayPage.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
{
    //按时构造订单号
    System.DateTime currentTime = new System.DateTime();
    currentTime = System.DateTime.Now;
    //获取适合使用习惯的日期,例如1981-10-17
    string out_trade_no = currentTime.ToString("g");
    //替换日期中的特殊字符
    out_trade_no = out_trade_no.Replace("-", "");
    out_trade_no = out_trade_no.Replace(":", "");
    out_trade_no = out_trade_no.Replace(" ", "");
    //业务参数赋值
    //支付接口
    string gateway = T_gateway.Text;
    //服务接口名称,此处采用测试默认值
    string service = T_service.Text;
    //合作伙伴ID，注册为支付宝用户后获取
    string partner = T_partner.Text;
    //加密类型
    string sign_type = T_sign_type.Text;
    //商品名称
    string subject = T_subject.Text;
    //商品描述
    string body = T_body.Text;
    //支付类型此处默认为商品购买，具体类型可参考下载的文档
    string payment_type = T_payment_type.Text;
    //总金额    0.01～50000.00
    string total_fee = T_total_fee.Text;
    //商品的展示地址
    string show_url = T_show_url.Text;
    //卖家账号
    string seller_email = T_seller_email.Text;
    //partner账户的支付宝安全校验码
    string key = T_key.Text;
    //服务器返回接口
    string return_url = T_return_url.Text;
    //服务器通知返回接口
    string notify_url = T_notify_url.Text;
    //编码格式
    string _input_charset = T_inputchatset.Text;
    //生成一个支付对象
```

```
        AliPay ap = new AliPay();
        //根据网关校验，并返回完成地址
        string aliay_url = ap.CreatUrl(
              gateway,          service,          partner,
              sign_type,        out_trade_no, subject,
              body,             payment_type, total_fee,
              show_url,         seller_email, key,
              return_url,    _input_charset,notify_url );
        //导航到支付宝交付页面
        Response.Redirect(aliay_url);
    }
```

注 意

在该页面中首先需要引用第三方控件的命名空间，引用的命名空间如下：

using Gateway。

3. 获取支付成功后的返回信息

用户提交完订单并使用支付宝付款后，将跳转到支付宝的验证页面或者银行验证页面，当付款成功后，才返回网站指定的页面，实现步骤如下。

（1）在应用程序中创建一个 Web 窗体命名为 Default.aspx，用来获取支付成功后的返回信息。

（2）该页面中不需要设计任何控件，但用户可根据自己的要求自行设计界面来显示成功信息。

（3）在 Default.aspx 页面的 Page_Load 事件中，编写获取返回数据的代码，具体实现的代码如下。

例程 19 代码位置：光盘\mr\29\ BankPay \ Default.aspx.cs

```
    protected void Page_Load(object sender, EventArgs e)
    {
        //支付宝的网关
        string alipayNotifyURL = "https://www.alipay.com/cooperate/gateway.do?";
        //用户注册支付宝时生成的校验码（必须填写自己的）
        string key = "xxxxxxx";
        //页面编码格式
        string _input_charset = "utf-8";
        //用户注册支付宝时生成的合作伙伴id（必须填写自己的）
        string partner = "xxxxxx";
         alipayNotifyURL = alipayNotifyURL + "service=create_digital_goods_trade_p" + "&partner=" + partner +
"&notify_id=" + Request.QueryString["notify_id"];
        //获取支付宝ATN返回结果，True是正确的订单信息，False 是无效的
        string responseTxt = Get_Http(alipayNotifyURL, 120000);
        int i;
        NameValueCollection coll;
        //在集合中装载返回信息
        coll = Request.QueryString;
        // 将所有的键值保存在数组中
        String[] requestarr = coll.AllKeys;
        //进行排序
        string[] Sortedstr = BubbleSort(requestarr);
        for (i = 0; i < Sortedstr.Length; i++)
        {
            Response.Write("Form: " + Sortedstr[i] + "=" + Request.QueryString[Sortedstr[i]] + "<br>");
        }
        //构造待md5摘要字符串
        StringBuilder prestr = new StringBuilder();
        for (i = 0; i < Sortedstr.Length; i++)
        {
            if (Request.Form[Sortedstr[i]] != "" && Sortedstr[i] != "sign" && Sortedstr[i] != "sign_type")
```

```
            {
                if (i == Sortedstr.Length - 1)
                {
                    prestr.Append(Sortedstr[i] + "=" + Request.QueryString[Sortedstr[i]]);
                }
                else
                {
                    prestr.Append(Sortedstr[i] + "=" + Request.QueryString[Sortedstr[i]] + "&");
                }
            }
        }
        prestr.Append(key);
        //生成Md5摘要
        string mysign = GetMD5(prestr.ToString(), _input_charset);
        string sign = Request.QueryString["sign"];
        //测试返回的结果
    //    Response.Write(prestr.ToString());
        //验证支付发过来的消息，签名是否正确
         if (mysign == sign && responseTxt == "true")
        {
            //此时可以更新网站的数据，比如商品的减少等
            Response.Write("success");        //返回给支付宝消息,成功
        }
        else
        {
            // Response.Write("-----------------------------------------");
            // Response.Write("<br>Result:responseTxt=" + responseTxt);
            // Response.Write("<br>Result:mysign=" + mysign);
            // Response.Write("<br>Result:sign=" + sign);
            Response.Write("fail");
        }
    }
```

（4）上述 Page_Load 事件中，调用了 3 个方法 GetMD5、BubbleSort 和 Get_Http，分别介绍如下。

GetMD5 主要应用的是 MD5 加密算法，用于生成 MD5 摘要，代码如下。

例程 20 代码位置：光盘\mr\29\ BankPay \ Default.aspx.cs

```
public static string GetMD5(string s, string _input_charset)
{
    /// <summary>
    /// 与ASP兼容的MD5加密算法
    /// </summary>
    MD5 md5 = new MD5CryptoServiceProvider();
    byte[] t = md5.ComputeHash(Encoding.GetEncoding(_input_charset).GetBytes(s));
    StringBuilder sb = new StringBuilder(32);
    for (int i = 0; i < t.Length; i++)
    {
        sb.Append(t[i].ToString("x").PadLeft(2, '0'));
    }
    return sb.ToString();
}
```

BubbleSort 方法主要应用了典型的冒泡排序法，来对获取的相关信息执行排序操作，代码如下。

例程 21 代码位置：光盘\mr\29\ BankPay \ Default.aspx.cs

```
public static string[] BubbleSort(string[] r)
{
    /// <summary>
    /// 冒泡排序法
    /// </summary>
    //交换标志
```

```
        int i, j;
        string temp;
        bool exchange;
        //最多做R.Length-1次排序
        for (i = 0; i < r.Length; i++)
        {
            //本次排序开始前，交换标志应为假
            exchange = false;
            for (j = r.Length - 2; j >= i; j--)
            {
                //交换条件
                if (System.String.CompareOrdinal(r[j + 1], r[j]) < 0)
                {
                    temp = r[j + 1];
                    r[j + 1] = r[j];
                    r[j] = temp;
                    //发生了交换，故将交换标志置为真
                    exchange = true;
                }
            }
            //本次排序未发生交换，提前终止算法
            if (!exchange)
            {
                break;
            }
        }
        return r;
    }
```

Get_Http 方法主要用来获取远程服务器 ATN 结果，代码如下。

例程 22 代码位置：光盘\mr\29\ BankPay \ Default.aspx.cs

```
    public String Get_Http(String a_strUrl, int timeout)
    {
        string strResult;
        try
        {
            //创建访问页面
            HttpWebRequest myReq = (HttpWebRequest)HttpWebRequest.Create(a_strUrl);
            myReq.Timeout = timeout;
            HttpWebResponse HttpWResp = (HttpWebResponse)myReq.GetResponse();
            //获取页面返回数据流
            Stream myStream = HttpWResp.GetResponseStream();
            StreamReader sr = new StreamReader(myStream, Encoding.Default);
            StringBuilder strBuilder = new StringBuilder();
            //获取内容
            while (-1 != sr.Peek())
            {
                strBuilder.Append(sr.ReadLine());
            }
            strResult = strBuilder.ToString();
        }
        catch (Exception exp)
        {
            //给出错误提示
            strResult = "错误:" + exp.Message;
        }
        return strResult;
    }
```

（5）在编写上述代码前需要引入如下命名空间，以便使用这些命名空间中的类，代码声明如下。

例程 23 代码位置：光盘\mr\29\ BankPay \ Default.aspx.cs

```
using System.Text;
using System.Security.Cryptography;
using System.Collections.Specialized;
using System.IO;
using System.Data.SqlClient;
using System.Data.OleDb;
using System.Net;
```

4. 设置支付过程中的通知

在支付宝中设置支付过程中的通知由 NoticeReturn.aspx 页实现，获取返回的通知与获取返回信息的步骤基本上类似，但方法略有区别，具体获取步骤如下。

（1）在 Page_Load 事件中，添加获取的代码，具体代码如下。

例程 24 代码位置：光盘\mr\29\ BankPay \ NoticeReturn.aspx .cs

```
protected void Page_Load(object sender, EventArgs e)
{
    string alipayNotifyURL = "https://www.alipay.com/cooperate/gateway.do?";
    //partner合作伙伴id（必须填写）
    string partner = "xxxxxxxxxx";
    //partner 的对应交易安全校验码（必须填写）
    string key = "xxxxxxxx";
    alipayNotifyURL = alipayNotifyURL + "service=create_digital_goods_trade_p" + "&partner=" + partner +
"&notify_id=" + Request.Form["notify_id"];
        //获取支付宝ATN返回结果，True是正确的订单信息，False 是无效的
        string responseTxt = Get_Http(alipayNotifyURL, 120000);
        int i;
        NameValueCollection coll;
        //在集合中装载返回信息
        coll = Request.Form;
        // 将所有的键值保存在数组中
        String[] requestarr = coll.AllKeys;
        //进行排序
string[] Sortedstr = BubbleSort(requestarr);
//构造待md5摘要字符串
string prestr = "";
for (i = 0; i < Sortedstr.Length; i++)
{
    if (Request.Form[Sortedstr[i]] != "" && Sortedstr[i] != "sign" && Sortedstr[i] != "sign_type")
    {
        if (i == Sortedstr.Length - 1)
        {
            prestr = prestr + Sortedstr[i] + "=" + Request.Form[Sortedstr[i]];
        }
        else
        {
            prestr = prestr + Sortedstr[i] + "=" + Request.Form[Sortedstr[i]] + "&";
        }
    }
}
prestr = prestr + key;
string mysign = GetMD5(prestr);
string sign = Request.Form["sign"];
//验证支付发过来的消息，签名是否正确
if (mysign == sign && responseTxt == "true")
{
    //    判断支付状态TRADE_FINISHED（文档中有枚举表可以参考）
    if (Request.Form["trade_status"] == "WAIT_SELLER_SEND_GOODS")
        {
            //更新自己数据库的订单语句
            //返回给支付宝消息，成功
            Response.Write("success");
```

```
                }
                else
                {
                    Response.Write("fail");
                }
            }
        }
```

（2）上述 Page_Load 事件代码中分加用到了 3 个自定义方法：GetMD5、BubbleSort 和 Get_Http，分别介绍如下。

例程 25 代码位置：光盘\mr\29\ BankPay \ NoticeReturn.aspx .cs

```
    public static string GetMD5(string s)
    {
        /// <summary>
        /// 与ASP兼容的MD5加密算法
        /// </summary>
        MD5 md5 = new MD5CryptoServiceProvider();
        byte[] t = md5.ComputeHash(Encoding.GetEncoding("utf-8").GetBytes(s));
        StringBuilder sb = new StringBuilder(32);
        for (int i = 0; i < t.Length; i++)
        {
            sb.Append(t[i].ToString("x").PadLeft(2, '0'));
        }
        return sb.ToString();
    }
    public static string[] BubbleSort(string[] R)
    {
        // <summary>
        //冒泡排序法
        // </summary>
        int i, j; //交换标志
        string temp;
        bool exchange;
        //最多做R.Length-1趟排序
        for (i = 0; i < R.Length; i++)
        {
            //本趟排序开始前，交换标志应为假
            exchange = false;
            for (j = R.Length - 2; j >= i; j--)
            {
                //交换条件
                if (System.String.CompareOrdinal(R[j + 1], R[j]) < 0)
                {
                    temp = R[j + 1];
                    R[j + 1] = R[j];
                    R[j] = temp;
                    //发生了交换，故将交换标志置为真
                    exchange = true;
                }
            }
            //本趟排序未发生交换，提前终止算法
            if (!exchange)
            {
                break;
            }
        }
        return R;
    }
    //获取远程服务器ATN结果
    public String Get_Http(String a_strUrl, int timeout)
    {
        string strResult;
```

```
        try
        {
            //创建访问页面
            HttpWebRequest myReq = (HttpWebRequest)HttpWebRequest.Create(a_strUrl);
            myReq.Timeout = timeout;
            HttpWebResponse HttpWResp = (HttpWebResponse)myReq.GetResponse();
            //获取页面返回数据流
            Stream myStream = HttpWResp.GetResponseStream();
            StreamReader sr = new StreamReader(myStream, Encoding.Default);
            StringBuilder strBuilder = new StringBuilder();
            //获取内容
            while (-1 != sr.Peek())
            {
                strBuilder.Append(sr.ReadLine());
            }
            strResult = strBuilder.ToString();
        }
        catch (Exception exp)
        {
            //给出错提示信息
            strResult = "错误:" + exp.Message;
        }
        return strResult;
    }
```

服务类技术

第 30 章

实例位置：光盘\mr\30\

30.1 Web Service 实现天气预报

30.1.1 Web Service 天气预报功能概述

实现天气预报的查询有两种方法：一种是比较简单的，是在 ASP 时代通用的方法，就是直接转到天气预报的网站；另一种方法就是使用最新技术 Web Service，又叫 Web 服务。本节将介绍如何在 ASP.NET 中使用 Web Service 实现天气预报的查询。实际运行效果如图 30.1 所示，主要通过选择省份及相应的城市来获取该城市的天气预报信息。

图 30.1 Web Service 实现天气预报运行效果图

30.1.2 介绍 Web Service

1. Web Service 概述

Web Service 即 Web 服务。所谓服务就是系统提供一组接口，并通过接口使用系统提供的功能。同在 Windows 系统中，应用程序通过 API 接口函数使用系统提供的服务一样，在 Web 站点之间，如果想要使用其他站点的资源，就需要其他站点提供服务，这个服务就是 Web 服务。

Web 服务是建立可互操作的分布式应用程序的新平台，它是一套标准，定义了应用程序如何在 Web 上实现互操作性。在这个新的平台上，开发人员可以使用任何语言，以及在任何操作系统平台上进行编程，只要保证遵循 Web 服务标准，就能够实现对服务进行查询和访问。Web 服务的服务器端和客户端都要支持行业标准协议 HTTP、SOAP 和 XML。

Web 服务缩小了 Web 应用程序之间的“通信隔阂”，增强了交互性。因此，Web 站点可以组合起来，形成能够为用户提供丰富体验的合成体。

Web 服务中表示数据和交换数据的基本格式是可扩展标记语言(XML)。Web 服务使用 XML 作为基本的数据通信方式，来消除使用不同组件模型、操作系统和编程语言的系统之间存在的差异。开发人员可以使用同使用组件创建分布式应用程序一样的方法，创建不同来源的 Web 服务所组合在一起的应用程序。

网络是多样性的，要在 Web 的多样性世界里取得成功，Web 服务在涉及到操作系统、对象模型和编程语言的选择时不能有任何倾向性。并且，要使 Web 服务像其他基于 Web 的技术一样被广泛采用，还必须满足以下特性。

- 服务器端和客户端的系统都是松散耦合的。也就是说，Web 服务与服务器端和客户端

所使用的操作系统、编程语言都无关。

- Web 服务的服务器端和客户端应用程序具有连接到 Internet 的能力。
- 用于进行通信的数据格式必须是开放式标准，而不是封闭通信方式。在采用自我描述的文本消息时，Web 服务及其客户端无须知道每个基础系统的构成即可共享消息，这使得自行开发系统和不同的系统之间能够进行通信。Web 服务使用 XML 实现此功能。

2. 了解 XML Web 服务

Web Service 技术其实就是利用 SOAP 在 HTTP 上实现远程调用的一种方法。因为 Web 服务中表示数据和交换数据的基本格式就是可扩展标记语言（XML），所以读者需要首先了解 XML Web 服务，简单介绍如下。

使用 ASP.NET 构造一个简单的 XML Web 服务是相对容易的，然而，XML Web 服务真正强大的功能只有在研究了其基础结构以后才能领悟。

XML Web 服务是建立在.NET 框架和公共语言运行时间基础上的。XML Web 服务的基础结构是构建符合像 SOAP、XML 和 WSDL 这样的行业标准，并且它允许其他的平台的客户端与 XML Web 服务互操作。只要一个客户端可以发送符合标准的 SOAP 消息、依据格式化的服务描述，那么这个客户端可以调用一个使用 ASP.NET 创建的 XML Web 服务，不管客户端存在于何种平台。

当使用 ASP.NET 构造一个 XML Web 服务时，它自动支持客户端使用 SOAP、HTTP-GET 和 HTTP-POST 协议通信。即使 HTTP-GET 和 HTTP-POST 支持使用 URL 编码的变量名/变量值对来传送消息，支持这两个协议的数据类型也没有支持 SOAP 协议的数据类型丰富。在 SOAP 中，使用 XML 把数据传送到 XML Web 服务或从 XML Web 服务取回消息，你可以使用支持丰富的数据类型集的 XSD 模式定义复杂的数据类型。使用 ASP.NET 构造一个 XML Web 服务的开发者不必明确地定义复杂的数据类型。它们可以只构造一个管理类。ASP.NET 处理定义到一个 XSD 模式的映射类和到 XML 数据的映射对象实例，以便通过网络传输。

注意

XML Web 服务并不能取代 DCOM（分布式 COM），应该说 XML Web 服务是跨越使用行业标准的平台通信的一种消息传递基础结构。

3. Web Service 优点

Web Service 的主要应用一般是数据库访问流量大的时候，将数据通过 Web 服务存在本地，实现数据的快速供应。使用 Web Service 其优势如下。

- 提供 Web 服务端，用户可以访问公用的 Web 接口。
- 独立于在应用程序中使用的用户接口类型（如 Web 客户端和 Windows 客户端）表示。
- 是对应用程序服务器进行远程处理的方式之一。
- 屏蔽服务端其他层，更安全地提供服务。

30.1.3 创建一个简单 Web Service

下面介绍如何创建一个简单的 Web 服务，主要调用 Web 服务返回 Hello Word 字符串。实现步骤如下。

（1）打开 Visual Studio 2008 开发环境，依次选择“文件”/“新建”/“网站”命令，弹出“新建网站”对话框，在该对话框中选择“ASP.NET Web 服务”模板，并命名为“MyWebService”，如图 30.2 所示。

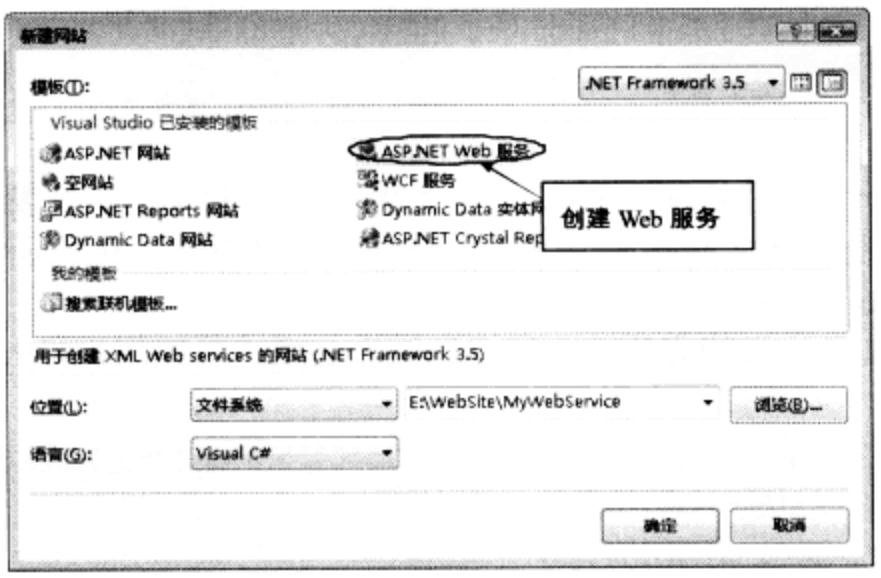


图 30.2　新建 ASP.NET Web 服务

（2）单击 “确定”按钮，将显示如图 30.3 所示页面，在该页面中主要显示的是一个 Web Service 默认生成的服务代码。

图 30.3　Web 服务的代码隐藏文件

下面介绍 Web 服务的代码隐藏文件。它包含了自动生成的一个类，并生成一个名为 HelloWorld 的模板方法，返回一个字符串。默认生成的代码如下：

```
using System;
using System.Web;
using System.Web.Services;
using System.Web.Services.Protocols;
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
public class Service : System.Web.Services.WebService
{
    public Service () {
        //如果使用设计的组件，请取消注释以下行
        //InitializeComponent();
    }
```

```
    [WebMethod]
    public string HelloWorld() {
        return "Hello World";
    }
}
```

（3）通过 Web Service 特性应用到实现一个 Web 服务的类上，开发者可以使用一个描述 Web 服务的字符串来设置这个 Web 服务的默认 XML 命名空间。代码如下：

```
[WebService(Namespace = "http://contoso.org/")]
```

（4）在代码中添加自定义的方法 Login ()，该方法用于验证用户输入的用户名及密码。代码如下：

```
    [WebMethod(Description = "该方法用来验证用户名及密码是否正确,正确返回true,否则返回false")]
    public bool Login(string UserName,string PassWord)
    {
        SqlConnection conn = new SqlConnection("server=MRLFL\\MRLFL;uid=sa;pwd=;database=db_26");
        conn.Open();
        SqlCommand cmd = new SqlCommand("select * from tb_User where UserName='" + UserName + "' and
PassWord='" + PassWord + "'", conn);
        SqlDataReader dr = cmd.ExecuteReader();
        bool Message =false;
        if (dr.Read())
        {
            Message = true;
        }
        else
        {
            Message = false;
        }
        cmd.Dispose();
        dr.Dispose();
        conn.Dispose();
        return Message;  //返回用户名及密码是否输入正确
    }
```

说 明 在通过 Web 服务添加可访问的方法时，需要把方法定义为 public，并使方法具有 WebMethod 属性。

（5）在“生成”菜单中，选择“生成解决方案”选项，生成 Web 服务。

（6）为了测试生成的 Web 服务，直接单击“▶”按钮，将显示 Web 服务帮助页面，如图 30.4 所示。

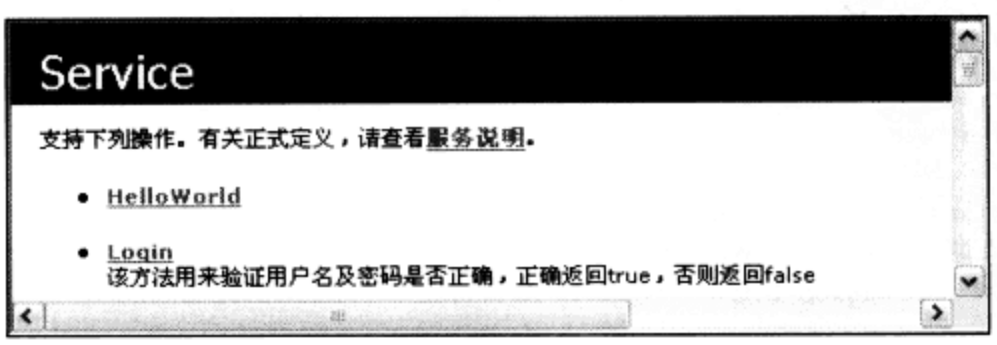


图 30.4　Web 服务帮助页面

（7）在图 30.4 中看到 Web 服务包含两个方法，一个是 HelloWorld 模板方法，另外一

个为自定义的 Login 查询方法。单击 Login 方法的链接将显示它的测试页面，如图 30.5 所示。

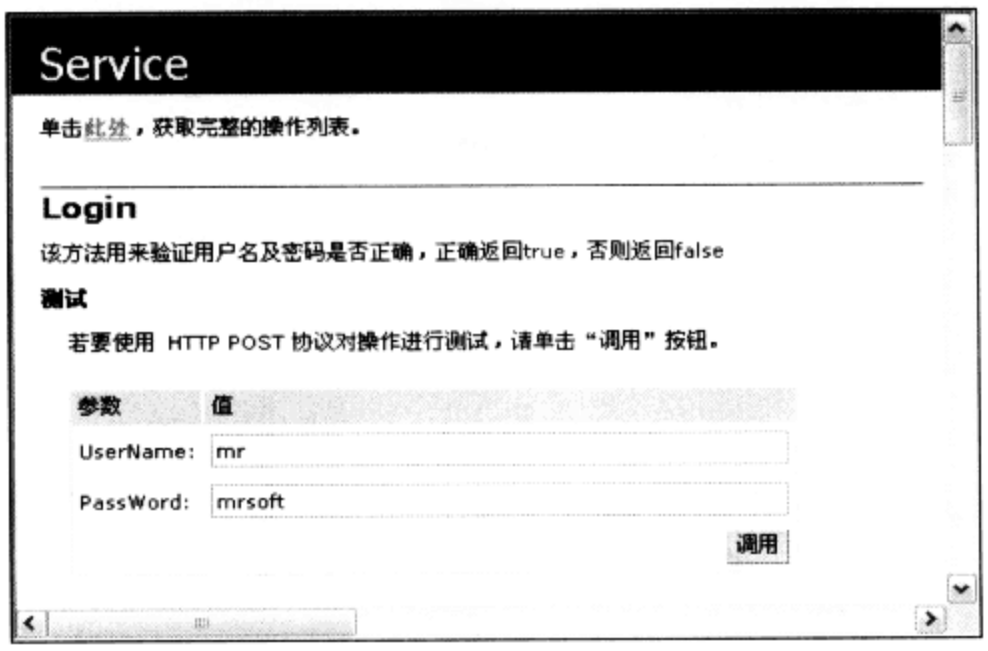


图 30.5　Login 方法的测试页面

（8）在测试页中输入要进行验证的用户名及密码，单击【调用】按钮即可调用 Web 服务的相应方法并显示方法的返回结果，如图 30.6 所示。

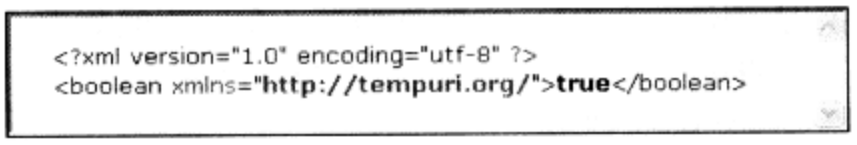


图 30.6　Login 方法返回的结果页面

从上面的测试结果可以看出，Web 服务方法的返回结果是使用 XML 进行编码的。

利用 Web Service 技术，不仅仅可以实现简单的数据操作，还可以实现在 Web 页面上能够完成的操作。其中.NET 平台内建了对 Web service 的支持，包括 Web service 的构建和使用。与其他开发平台不同，使用.NET 平台，用户不需要其他的工具或者 SDK 就可以完成 Web service 的开发。.NET Framework 本身就全面支持 Web Service，包括服务器端的请求处理器和对客户端发送和接收 SOAP 消息的支持。

30.1.4　使用 Web Service 获取天气预报

1. 设计步骤

（1）新建 1 个网站，将其命名为 WeatherService，将默认主页命名为 Default.aspx。

（2）在 Default.aspx 页面中添加 1 个 Table 表格、2 个 DropDowList 控件和 1 个 Button 控件，分别用来布局页面、选择省份及获取相应的城市的天气预报信息。

（3）右键单击创建的应用程序的根目录，在弹出的对话框中选择“添加 Web 引用”选项，如图 30.7 所示。

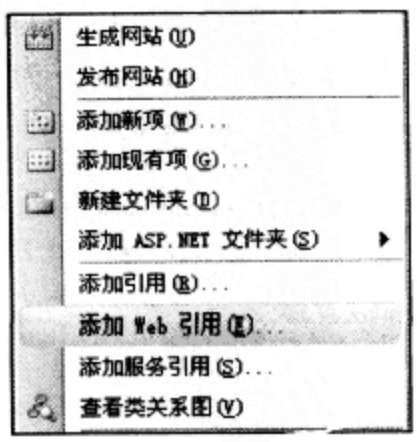


图 30.7　“添加 Web 引用”选项

此时将会弹出如图 30.8 所示的对话框，在该对话框的 URL 地址中输入 Web Service 的服务地址：http://www.ayandy.com/Service.asmx，然后单击“前往”按钮，最后在“Web 引用名”文本框中输入“App_WebReferences“Web 引用名。

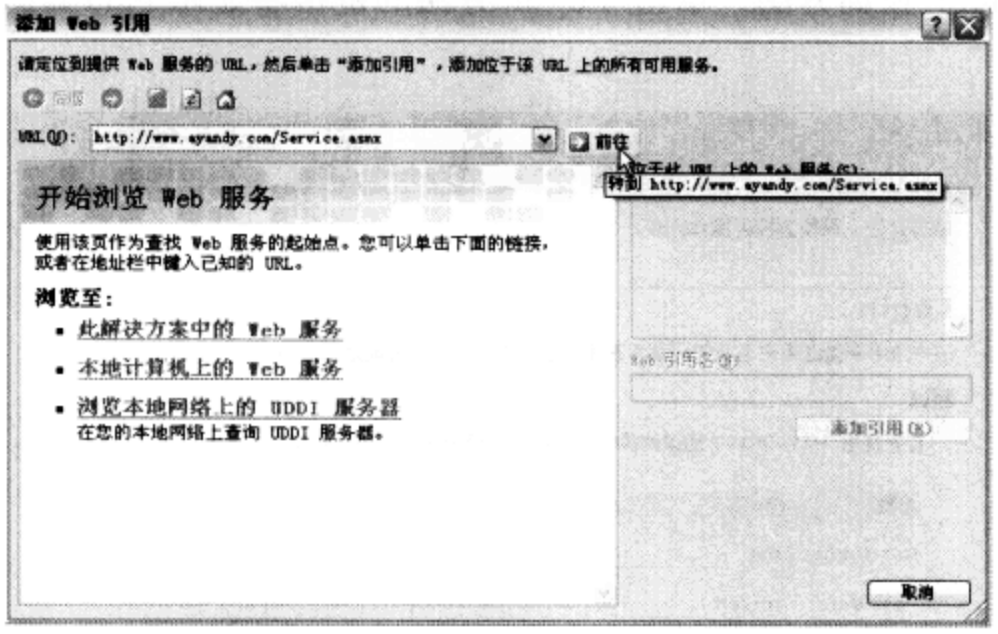


图 30.8　添加 Web Service 服务地址

说　明

这里添加的 Web Service 服务地址：http://www.ayandy.com/Service.asmx 为 Internet 互联网上免费提供的天气预报服务地址。

2. 代码实现

首先，在页面代码中实例化一个 Web Service 服务对象，声明如下：

例程 1　代码位置：光盘\mr\30\ MyWebService \ Default.aspx.cs

```
obj.Service myobj = new obj.Service(); //实例化
```

在页面 Page_Load 事件中调用 3 个自定义方法，分别用来绑定 WebService 服务中的省份、城市及相应的天气预报状况，编写的代码如下。

例程 2　代码位置：光盘\mr\30\ MyWebService \ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        //绑定省份信息
        BindPro();
        //绑定相应的城市信息
        BindCity();
        //绑定查找到的相应的天气预报
        BindWeather();
        Label4.Text = DateTime.Now.ToShortDateString();
    }
}
```

自定义 1 个 BindPro()方法来绑定所要查询的省份信息，代码如下。

例程 3　代码位置：光盘\mr\30\ MyWebService \ Default.aspx.cs

```
protected void BindPro()
{
    string [] pro=myobj.getSupportProvince();
    for (int i = 1; i <= Int32 .Parse (pro[0]); i++)
    {
        DropDownList1.Items.Add(new ListItem (pro[i].ToString (),pro[i].ToString()));
    }
 }
```

自定义一个 BindCity()方法来绑定所要查询的相应省份的城市信息，代码如下。

例程 4　代码位置：光盘\mr\30\ MyWebService \ Default.aspx.cs

```
protected void BindCity()
{
        DropDownList2.Items.Clear();
        string[] city = myobj.getSupportCity (DropDownList1 .SelectedValue );
        for (int i = 1; i <= Int32.Parse(city[0]); i++)
        {
            DropDownList2.Items.Add(new ListItem(city[i].ToString(), city[i].ToString()));
        }
 }
```

自定义 1 个 BindCity ()方法来绑定所查询的省份及城市相应的天气预报信息，代码如下。

例程 5　代码位置：光盘\mr\30\ MyWebService \ Default.aspx.cs

```
protected void BindWeather()
{
      string[] mystr = myobj.getWeatherbyCityName(DropDownList2 .SelectedValue , theDayFlagEnum.Today);
      Label1.Text = mystr[1].ToString();
      Label2.Text = mystr[2].ToString();
      Label3.Text = mystr[3].ToString();
      Label5.Text = mystr[5].ToString();
      Image1.ImageUrl = mystr[6].ToString();
 }
```

在命名空间代码区域内需要引入如下命名空间。

例程 6　代码位置：光盘\mr\30\ MyWebService \ Default.aspx.cs

```
using System .Web .Services ;
using obj;
```

30.2　社会标签（Tags）技术

Tags，即社会标签（Bookmark）。它是一种更为灵活、有趣的文章或图片等信息的分类方式。用户可以为每篇文章、每张图片或每条信息添加一个或多个标签，从而根据这些文章、图片或信息进行分类。

30.2.1　社会标签简介

1. 社会标签概述

用户可以把社会标签理解为一个文章和图片的分类。标签，又被称为软分类，即根据文章、图片或者信息的意义，由信息的组织者为信息指定一人或者多个“标签”。传统分的分类又称为硬分类，即就是发布文章、图片或信息时所选择的系统现有的固定的分类。标签和分类相比具有以下特点。

- 分类一般是事先预定好的，即文章或图片属于哪一个分类，事先就已经规定好了，而标签不同，它是在文章或图片完成之后再由用户添加的。
- 一般情况下，一篇文章或一张图片，只能有一个分类。而标签不同，一般文章或一张图片可以同时拥有一个或多个标签。
- 文章或图片的分类往往是由一个人指定（通常为文章或图片的作者），而标签则是由多人指定。

2. 社会标签应用

随着博客的发展，标签也渐渐地应用于博客中。同时，它也是体现博客社会性的最好方式。标签在博客中具有以下优点。

- 标签是分类（Category）最好的补充和扩展。
- 标签是一个开发、简易、有效的分类方法，有着广泛的应用前景。

- 标签能够帮助博客的访问迅速获取其所需要的信息。
- 搜索引擎和基于标签的分类软件能够更好地通过标签搜索和检索分类信息。

说 明

社会标签应用当然不仅仅只是用于博客，如论坛、电子商务网等都可以，其应用范围非常广。

简单的实现管理社会标签，程序的运行效果如图 30.9 所示。

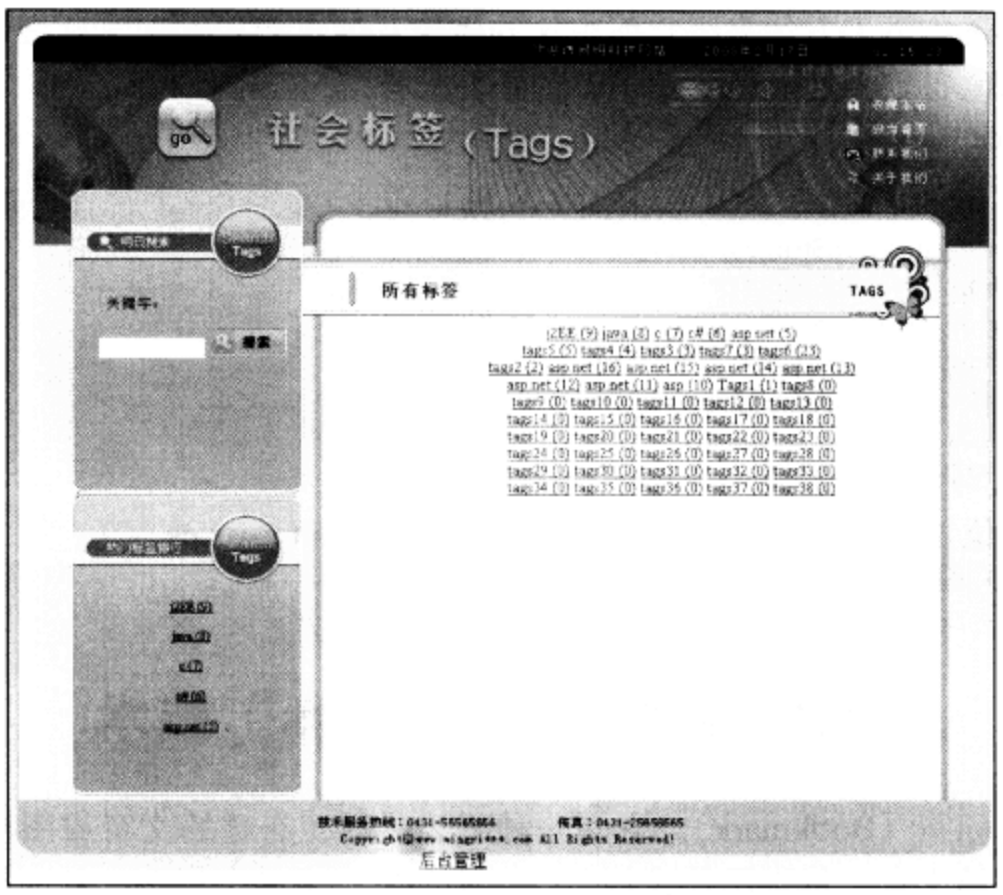


图 30.9　社会标签（Tags）运行效果图

30.2.2　热门标签排行

1. 页面设计

在标签管理程序首页中设置了热门标签，用来显示标签的分类名称和访问量。热门标签排行由 Default.aspx 页面实现，热门标签排行按照帖子的访问次数，以倒序方式显示系统中热门的标签，该页面运行效果如图 30.10 所示。

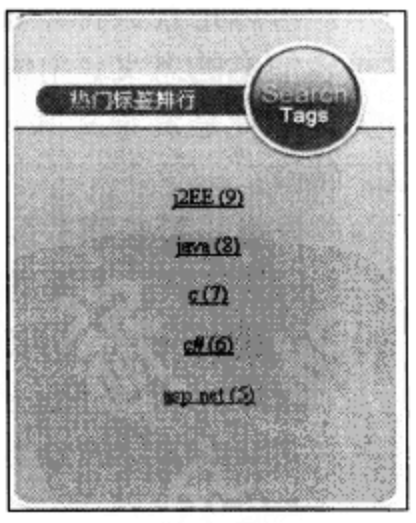


图 30.10　热门标签排行运行效果

程序开发的步骤如下。

（1）在应用程序 Tags 中创建 1 个 Web 窗体，将其命名为 Default.aspx。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 DataList 控件到窗体中，设置控件的属性。

页面前台主要设计代码如下。

例程 7　代码位置：光盘\mr\30\ Tags \ Default.aspx

```
<td valign="top">
<asp:DataList ID="DataList1" runat="server" Width="145px" Font-Size="9pt" Height="92px">
<ItemTemplate>
   <a href='Info.aspx?id=<%# Eval("id") %>'>
```

```
        <%# Eval("name") %>
        (<%# Eval("visit") %>)</a>
    </ItemTemplate>
    <AlternatingItemTemplate>
        <a href='Info.aspx?id=<%# Eval("id") %>'>
          <%# Eval("name") %>
          (<%# Eval("visit") %>)</a>
    </AlternatingItemTemplate>
        </asp:DataList></td>
      </tr>
    </table>
    </td>
```

2. 代码实现

首先，声明并实例化创建的一个公共类对象，以便调用其中编写的代码，声明的代码如下。

例程 8 代码位置：光盘\mr\30\ Tags \ Default.aspx.cs

```
Data data = new Data();
```

在 Page_Load 事件中，调用用户自定义的 bind 方法，以实现社会标签的绑定，代码如下。

例程 9 代码位置：光盘\mr\30\ Tags \ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
  {
    //调用自定义方法绑定列表中数据
    this.bind();
  }
```

定义一个用户自定义的 bind 方法，在该方法中调用公共类中的 EXECDataList1 将查找出访问量在前 5 名的热门标签的信息绑定到 DataList 控件当中，实现的代码如下。

例程 10 代码位置：光盘\mr\30\ Tags \ Default.aspx.cs

```
public void bind()
{
    //调用公共类的EXECDataList1执行查询的SQL语句
    this.data.EXECDataList1(this.DataList1, "select top 5 * from tags order by visit desc");
}
```

30.2.3 标签的检索

1. 页面设计

在标签管理程序首页中设置了查询标签的功能，主要用于查询社会标签（Tags），该检索标签页面运行效果如图 30.11 所示。

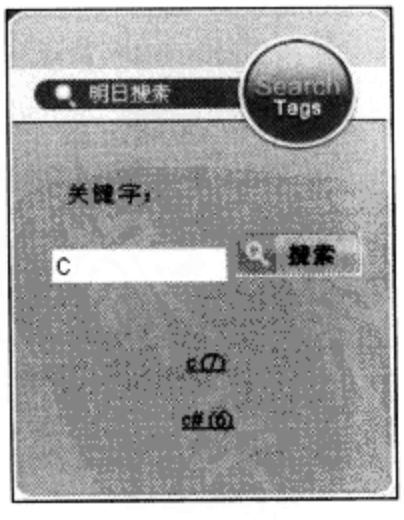


图 30.11 检索标签运行效果

程序开发的步骤如下。

（1）在应用程序 Tags 中创建 1 个 Web 窗体，其主页默认为 Default.aspx，其中设置的搜索标签功能是网站首页的一部分。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 TextBox 控件、1 个 ImageButton 控件和 1 个 DataList 控件到窗体中，分别用来填写要搜索的标签名称、执行搜索操作和绑定搜索的标签列表。前台页面代码设计如下。

例程 11 代码位置：光盘\mr\30\ Tags \ Default.aspx

```
<td style="height: 115px" valign="top">
        <asp:DataList ID="dlTag" runat="server" RepeatColumns="1" Width="100%" Height="71px" Font-Size="10pt">
         <ItemTemplate>
         <a href='Info.aspx?id=<%# Eval("id") %>'>
```

```
            <%# Eval("name") %>
            (<%# Eval("visit") %>)</a>
              </ItemTemplate>
            <AlternatingItemTemplate>
            <a href='Info.aspx?id=<%# Eval("id") %>'>
            <%# Eval("name") %>
           (<%# Eval("visit") %>)</a>
           </AlternatingItemTemplate>
           </asp:DataList>
      <asp:Label ID="Label2" runat="server" Width="290px"></asp:Label>
                                          </td>
```

2. 代码实现

首先，声明并且实例化创建的公共类对象，以便调用其中编写的方法，声明的代码如下。

例程 12 代码位置：光盘\mr\30\ Tags \ Default.aspx.cs

```
Data data = new Data();
```

在 Page_Load 事件中，调用用户自定义的 bind 方法，以实现社会标签内容的绑定，实现的代码如下。

例程 13 代码位置：光盘\mr\30\ Tags \ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
    //调用自定义方法绑定列表中数据
  this.bind();
}
```

定义一个用户自定义的 bind 方法，将查找出访问量在前 5 名的热门标签的信息绑定到 DataList 控件当中，实现的代码如下。

例程 14 代码位置：光盘\mr\30\ Tags \ Default.aspx.cs

```
public void bind()
{
    //调用公共类中的EXECDataList1执行指定的查询SQL语句
    this.data.EXECDataList1(this.DataList1, "select top 5 * from tags order by visit desc");
}
```

双击页面中的“检索”按钮，触发其“ImageButton1_Click”事件，在该事件中实现的是将检索到相应的标签显示到 DataList 控件中，实现的代码如下。

例程 15 代码位置：光盘\mr\30\ Tags \ Default.aspx.cs

```
protected void ImageButton1_Click(object sender, ImageClickEventArgs e)
{
        if (this.tbKey.Text == "")
        {
            string sqlstr = "select * from tags";
            data.EXECDataList1(this.DataList1, sqlstr);
        }
        else
        {
            string sqlstr = "select * from tags where name like '%" + this.tbKey.Text.Trim() + "%' order by visit desc";
            data.EXECDataList1(this.dlTag, sqlstr);
        }
    }
```


30.3 在 线 客 服

在线客服是网站的一种交流方式，类似于 MSN 和 QQ，但无须将客服方添加到自己的好友里，即可以实现对话。

30.3.1 在线客服概述

QQ 是网民上网时重要的交流方式，可实现在线和非在线进的信息发送，本节介绍的客服系统是基于 QQ 软件的一个模块，所以要求测试者机器上必须安装 QQ 软件。

在线客服功能使用户可以轻松地找到咨询人员，而无须加对方为好友，而且可以通过客服系统的在线功能，了解有哪些技术人员在提供支持，此功能通常用于大型技术论坛和技术型网站。如果用户在论坛中遇到技术难题的时候，可通过实时在线客服系统，查看在线的技术人员，以获得技术上的帮助。在线客服模块通常在页面上处于悬浮状态，这样用户需要服务时，直接就可以通过单击服务按钮，获得帮助。

本节将通过 QQ 上提供的客户服务的代码接口，实现在线客服的功能，在线客服的运行效果如图 30.12 所示。

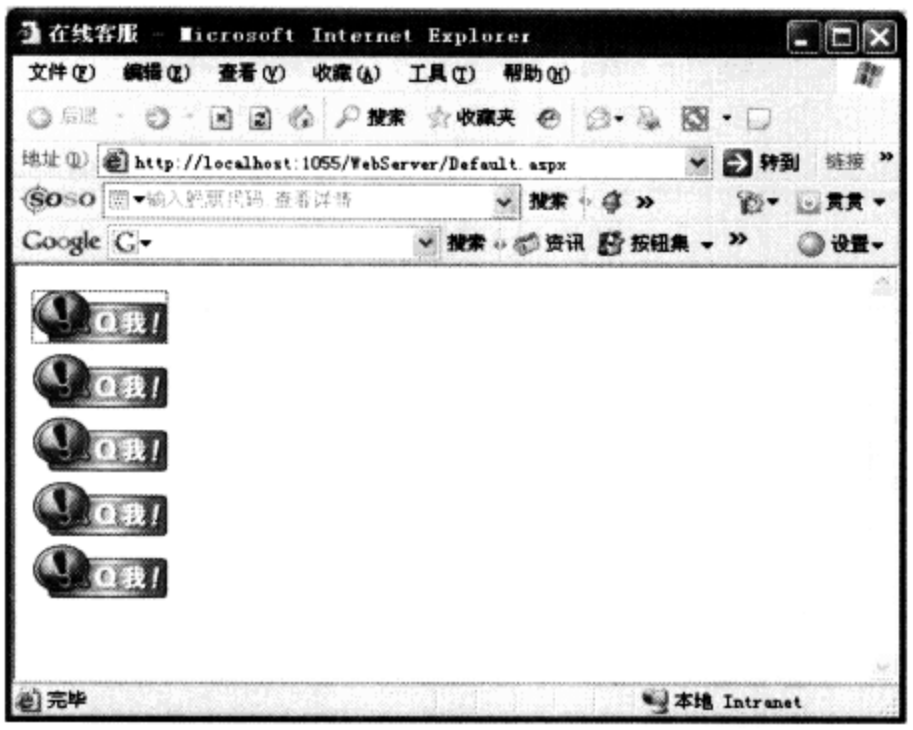


图 30.12 在线客服运行效果图

30.3.2 QQ 网站上自动生成代码

由于 QQ 是目前比较流行的一款在线聊天的工具，所以很多在线客服都采用 QQ 提供的在线客服服务。在 QQ 网站中的代码生成页中，当用户按照步骤填写完相关信息后，单击“生成网页代码”按钮后，便可生成你所需要的代码。然后将代码粘贴到自己的网站中，并且适当修改代码以符合实际需要，然后读者便可以到网上测试您的程序是否可用，测试的机器必须安装 QQ。

在线客服中使用 QQ 网站自动生成代码的具体步骤如下。

（1）打开浏览器，并且在打开的浏览器中地址栏中输入网址为“http:/is.qq.com/webpresence/code.shtml”。

（2）打开该网页后，根据腾讯公司提供的生成代码的步骤选择生成代码的第一步，从单选按钮中选择图片风格，在这里官方一共提供 13 种风格的图片供大家选择，如图 30.13 所示。

（3）在第一步中选择要显示的风格后，便开始第二步填写您的相关信息中，依次填写“QQ/TM 号码（必填）”、“该在线状态的使用范围（最多 5 个域名，域名间请用分号隔开）”和“图片旁的留言”，并且在下拉列表框中选择“有效时间”和从复选框当中选择会话权限，如图 30.14 所示。

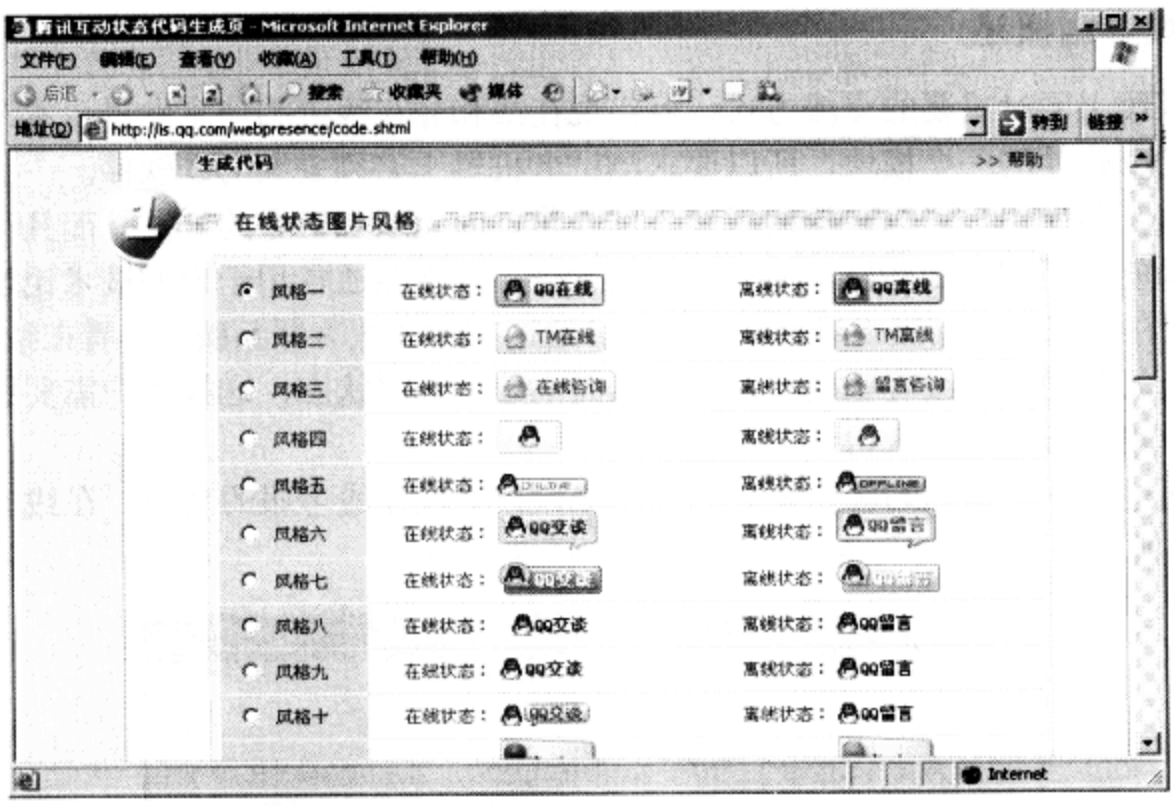


图 30.13　选择显示风格

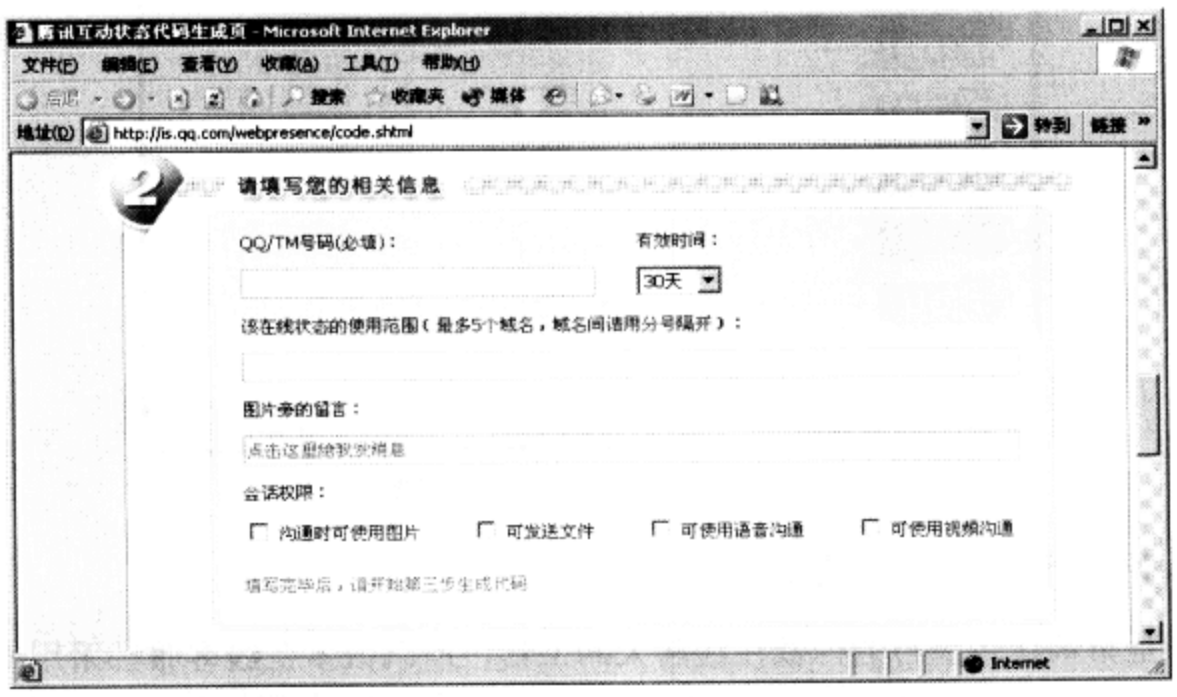


图 30.14　填写相关信息

(4) 完成第二步操作，接下来进入第三步当中，如图 30.15 所示。

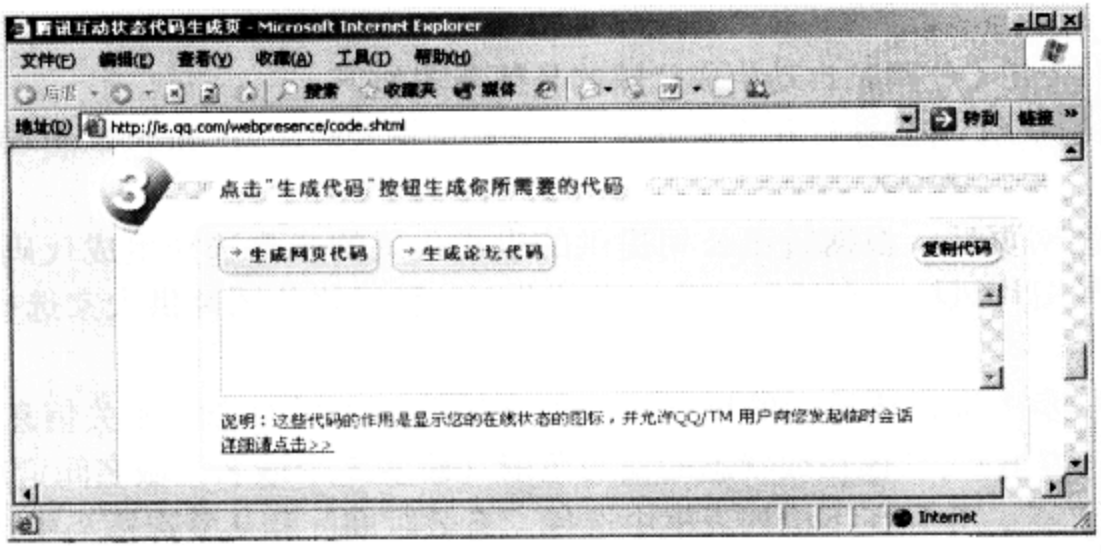


图 30.15　自动生成代码界面

（5）在第三步中单击“生成网页代码”按钮便可在文本框中生成您所需要的代码。在网站中，把你需要添加互动状态的地方直接粘贴到所编写的网站当中。

说 明

QQ 目前不支持自己和自己交谈，如果自己和自己交谈则给出提示信息，如图 30.16 所示。

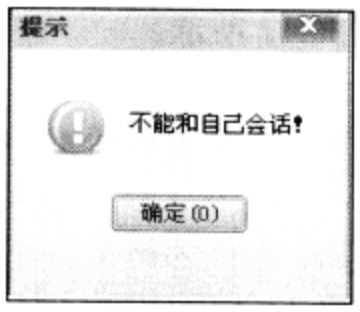


图 30.16　自己和自己会话弹出提示对话框

30.3.3　在线客服实现

在 30.3.2 节中能够在文本框中获得一段 QQ 网站上自动生成的一段代码，下面我们将对这段代码进行详细的讲解，生成的代码如下：

```
<a href="http://sighttp.qq.com/cgi-bin/check?sigkey=72d35736d0c176d20392cacea5c713a50cdc8f0ab29a97609
8af0fc7763444a92471998949b2121fbd3ed993de4b026f7444aac5b2034942"; target=_blank; onclick="var tempSrc='http:
//sighttp.qq.com/wpa.js?rantime='+Math.random()+'&sigkey=72d35736d0c176d20392cacea5c713a50cdc8f0ab29a97609
8af0fc7763444a92471998949b2121fbd3ed993de4b026f7444aac5b2034942';varoldscript=document.getEleme ntById('te
stJs');varnewscript=document.createElement('script');newscript.setAttribute ('type',' text/javascript'); newscript.setAttribute('id', 'testJ
s');newscript.setAttribute('src',tempSrc);if(oldscript == null){document.body.appendChild(newscript);}else {olds
cript.parentNode.replaceChild(newscript, oldscript);}return false;"><imgborder="0"SRC='http://wpa.qq.com/pa? p=
1:100310063:1' alt="点击这里给我发消息">
</a>
```

表示取状态图片的接口为：'http://wpa.qq.com/pa?p=1:100310063:1' 。

说 明

Web 网页中引用此接口，获取指定 QQ 用户的在线状态；该接口返回一张图片用来表示用户当前的状态。QQ 公司只支持“离线”和“在线”两种状态。

对 P=1：100310063：1 的参数说明。

第 1 个参数为“P=”后面的“1”。

第 2 个参数为“100310063”。

第 3 个参数为最后的数字“1”。

30.3.4　将代码应用于网站中

将生成的代码应用到网站中，实现的具体步骤如下。

（1）打开在线客服程序，打开网站的首页“Default.aspx”页面，将代码粘贴到源代码视图中。

（2）将生成的代码粘贴到指定的标签中，代码如下：

```
<a href="http://sighttp.qq.com/cgi-bin/check?sigkey=72d35736d0cb176d20392cacea5c713a50cdc8f0ab29a976098af0fc776344
4a92471998949b2121fbd3ed993de4b026f7444aac5b2034942";target=_blank;onclick="vartempSrc='http://sigh ttp .qq.com/
wpa.js?rantime='+Math.random()+'&sigkey=72d35736d0c176d20392cacea5c713a50cdc8f0ab29a976098af0fc7763444a92471
998949b2121fbd3ed993de4b026f7444aac5b2034942';var oldscript=document.getElementById('testJs');varnewscript=docum
ent.createElement('script');newscript.setAttribute('type','text/javascript'); newscript.setAttribute('id','testJs');newscript.setAtt
ribute('src',tempSrc);if(oldscript == null){document.body.appendChild(newscript);}else{oldscript.parentNode.replace
Child(newscript, oldscript);}return false;"><img border="0" SRC='http://wpa.qq.com/pa?p=1:100310063:1' alt="点击这里给
我发消息">
</a>
```

（3）代码粘贴完毕后，还需要修改代码以适应程序的需要，在该代码中将“target =_blank”修改为“target=_self”。

（4）在自动生成代码时，填写的名称需要修改成 http://locathost/Default.aspx。

30.3.5 客服后台管理

客服后台管理客服人员的信息，可以添加客服人员，同时也可以对客服人员进行编辑和删除操作，客服后台管理运行效果如图 30.17 所示。

图 30.17 运行效果图

1. 页面设计

（1）在应用程序中创建 1 个 Web 窗体，其主页默认为 Manage.aspx。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 TextBox 控件、1 个 ButtonList 控件和 2 个 GridView 控件到窗体中，设置控件的属性。

页面中各个控件的属性设置及其用途如表 30.1 所示。

表 30.1 客服后台管理页控件列表

控件类型	控件名称	主要属性设置	用途
TextBox	控件的名称为 txtname	无	用于输入客服人员的名称
Button	控件的名称为 BtnAdd	将控件的 OnClick 属性设置为 BtnAdd_Click，将 Text 属性设置为添加	用于执行添加操作
GridView	控件的名称为 GvWebServer	无	用于显示客服人员

2. 代码实现

首先，在命名空间代码区域中引入如下命名空间。

例程 16 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
using System.Data.SqlClient;
```

然后声明 3 个自定义变量，实现后台的调用，声明如下。

例程 17 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
SqlConnection sqlcon;
SqlCommand sqlcom;
string strCon = "Data Source=(local);Database=db_WebServer;Uid=sa;Pwd=";
```

在事件加载 Page_Load 事件中，调用用户自定义的 bind 方法实现将客服信息绑定到 GridView 控件当中，实现的代码如下。

例程 18 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)//判断页面是否是首次加载
    {
      //调用自定义方法绑定列表中数据
      bind();
    }
}
```

在 GridView1_RowEditing 事件中，主要用来实现表格控件的分页，实现的代码如下。

例程 19 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
protected void GridView1_RowEditing(object sender, GridViewEditEventArgs e)
{
```

```
        //实现表格分页
        GvWebServer.EditIndex = e.NewEditIndex;
        //重新绑定下列表中数据
        bind();
    }
```

在 GridView1_RowDeleting 事件中，单击 GridView 控件中的删除按钮，实现按照客服编号删除指定的客服信息，实现的代码如下。

例程 20 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
{
  //定义一个删除的SQL语句
string sqlstr = "delete from tb_QQLine where ListID='" + GvWebServer.DataKeys[e.RowIndex].Value.ToString() + "'";
  //创建一个数据连接对象
  sqlcon = new SqlConnection(strCon);
 //创建命令对象
  sqlcom = new SqlCommand(sqlstr,sqlcon);
 //打开数据库连接
  sqlcon.Open();
 //执行定义的删除SQL语句，并返回受影响的行数
  sqlcom.ExecuteNonQuery();
 //关闭数据库连接
  sqlcon.Close();
 //重新绑定下列表中的数据
  bind();
 }
```

在“GridView1_RowUpdating”事件中，实现使用 Update 语句执行修改客服服务信息，实现的代码如下。

例程 21 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
protected void GridView1_RowUpdating(object sender, GridViewUpdateEventArgs e)
{
    sqlcon = new SqlConnection(strCon);
    string sqlstr = "update tb_QQLine set ListQQ='"
          + ((TextBox)(GvWebServer.Rows[e.RowIndex].Cells[1].Controls[0])).Text.ToString().Trim() + "' where ListID='"
          + GvWebServer.DataKeys[e.RowIndex].Value.ToString() + "'";
    sqlcom=new SqlCommand(sqlstr,sqlcon);
    sqlcon.Open();
    sqlcom.ExecuteNonQuery();
    sqlcon.Close();
    //取消编辑状态
    GvWebServer.EditIndex = -1;
    //重新绑定下列表中数据
    bind();
}
```

在“GridView1_RowCancelingEdit”事件中，实现的是撤销操作的功能，实现的代码如下。

例程 22 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
protected void GridView1_RowCancelingEdit(object sender, GridViewCancelEditEventArgs e)
{
    //取消编辑状态
   GvWebServer.EditIndex = -1;
   //重新绑下列表中数据
    bind();
}
```

在用户自定义的 bind 方法中，使用 Select 语句将数据绑定到 GridView 控件中，实现的代码如下。

例程 23 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
public void bind()
{
    //定义一个查询的SQL语句
    string sqlstr = "select * from tb_QQLine ";
    sqlcon = new SqlConnection(strCon);
    SqlDataAdapter myda = new SqlDataAdapter(sqlstr, sqlcon);
    //创建一个数据集类型的变量myds
    DataSet myds = new DataSet();
    sqlcon.Open();
    myda.Fill(myds,"tb_QQLine ");
    //设置列表控件的数据源及主键字段
    GvWebServer.DataSource = myds;
    GvWebServer.DataKeyNames = new string[] { "ListID" };
    //从数据库中绑定数据
    GvWebServer.DataBind();
    sqlcon.Close();
}
```

双击页面中的“添加”按钮，执行添加客服人员操作。实现的代码如下。

例程 24 代码位置：光盘\mr\30\ WebServer \ Manage.aspx.cs

```
protected void BtnAdd_Click(object sender, EventArgs e)
{
    string sqlstr = "insert into tb_QQLine (ListQQ) values('"+this.txtname.Text+"')";
    sqlcon = new SqlConnection(strCon);
    sqlcom = new SqlCommand(sqlstr, sqlcon);
    sqlcon.Open();
    sqlcom.ExecuteNonQuery();
    sqlcon.Close();
    bind();
}
```

30.4 循环播放广告

30.4.1 循环播放广告功能概述

在一些大型网站中，有时会看到一些网页中有播放图片的效果，增加了网站动态感觉。通过添加图片的效果使整个网站内的信息更加丰富。在网站中使用循环播放广告，不但可以使网站更美观，还可以提高网站的访问量。

本节首先通过循环播放广告浏览页来浏览循环播放图片的效果，并且可以通过该页上的“广告位轮换管理”超级链接进入到广告管理页面中。在该页中用户可以通过添加播放图片，并且可以选择广告的播放位置。当添加完成广告后，返回到循环播放广告的浏览页面中（即首页中）时，用户将看见新添加的广告正在循环的播放着。

30.4.2 循环播放广告关键技术

网页中实现循环播放广告时，主要可以通过 JavaScript 脚本和 Flash 结合的方法实现，通过 JavaScript 脚本和 Flash 结合方法实现不间断显示循环广告。当公司美工人手不足时，开发人员可以使用 JavaScript 脚本和 Flash 结合方法控制循环播放图片。与 HTML 语言提供的 marquee 元素相比，通过 JavaScript 脚本和 Flash 结合方法可以使图片循环不间断显示，这样使网页看起来更加美观。

为了使读者更好地掌握循环播放图片，下面将详细介绍如何实现应用 JavaScript 脚本控制循环播放图片的技术。

在 Default.aspx 网站首页中通过 JavaScript 脚本和 Flash 结合方法控制播放图片的主要代

码如下，在该代码中主要实现的是设置播放广告的长和宽，以及播放广告的方式，实现的代码如下。

例程 25　代码位置：光盘\mr\30\ LoopAD \ Default.aspx

```
<SCRIPT type=text/javascript>
 <!--
 var focus_width=291
 var focus_height=265
 var text_height=0
 var swf_height = focus_height+text_height
 var pics='<%=str %>'
 var links='<%=loopId %>'
 var texts=''
 var banner='<param name="allowScriptAccess" value="sameDomain"><param name="movie"value="imgFile/
banner.swf"><param name="quality" value="high"><param name="bgcolor" value="#DADADA">'
 document.write('<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"  width="'+ focus_width +'" height="'+
swf_height +'">');
 document.write(banner);
 document.write('<param name="menu" value="false"><param name=wmode value="opaque">');
 document.write('<param name="FlashVars"value="pics='+pics+'&links='+links+'&texts='+texts+'&borderwidth
='+focus_width+'&borderheight='+focus_height+'&textheight='+text_height+'">');
 document.write('</object>');
 //-->
</SCRIPT>
```

30.4.3　浏览循环播放广告页面设计

并且如果该循环广告在后台循环广告管理页面中为它添加了链接的网站地址，当单击该广告便可跳转到链接的页，浏览循环播放广告页面运行效果如图 30.18 所示。

图 30.18　循环广告运行效果图

1. 页面设计

程序开发的步骤如下。

（1）在应用程序中创建 1 个 Web 窗体，其主页默认为 Default.aspx。作为浏览播放广告功能页。页面 Default.aspx 的设计界面如图 4.11 所示。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 LinkButton 控件到控件中，设置控件的属性。

前台页面设计的关键代码如下。

例程 26　代码位置：光盘\mr\30\ LoopAD \ Default.aspx

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs" Inherits="_Default" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server">
 <title>循环播放广告</title>
</head>
<body>
    <form id="form1" runat="server">
    <div><asp:LinkButton ID="likbtnAD" runat="server" OnClick="likbtnAD_Click" Font-Size="10pt" >广告位轮换
管理</asp:LinkButton>
        <SCRIPT type=text/javascript>
```

```
            <!--
            var focus_width=291
            var focus_height=265
            var text_height=0
            var swf_height = focus_height+text_height
            var pics='<%=str %>'
            var links='<%=loopId %>'
            var texts=''
            var banner='<param name="allowScriptAccess" value="sameDomain"><param name="movie"value="imgFile
/banner.swf"><param name="quality" value="high"><param name="bgcolor" value="#DADADA">'
              document.write('<object classid="clsid:d27cdb6e-ae6d-11cf-96b8-444553540000"   width="'+ focus_width
+'" height="'+ swf_height +'">');
            document.write(banner);
            document.write('<param name="menu" value="false"><param name=wmode value="opaque">');
            document.write('<param name="FlashVars" value="pics='+pics+'&links='+links+'&texts='+texts+'&borde
rwidth='+focus_width+'&borderheight='+focus_height+'&textheight='+text_height+'">');
            document.write('</object>');
              //-->
        </SCRIPT>
                 
                </div>
            </form>
        </body>
        </html>
```

2. 代码实现

首先声明两个字符串类型的变量“loopId”和变量“str”，实现的代码如下。

例程 27 代码位置：光盘\mr\30\ LoopAD \ Default.aspx.cs

```
public string loopId;
public string str;
```

在 Page_Load 页面加载事件中，实现设置循环播放广告图片的路径，并且实现单击循环广告图片时页面跳转到链接页面中，实现的代码如下。

例程 28 代码位置：光盘\mr\30\ LoopAD \ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
    string rd = DateTime.Now.Ticks.ToString();
    str = "imgFile/1.jpg?rd=" + rd + "limgFile/2.jpg?rd=" + rd + "limgFile/3.jpg?rd=" + rd + "limgFile/4.jpg?rd=" + rd + "";
    string path = Server.MapPath("./") + "imgFile/LoopId.txt";
    loopId = File.ReadAllText(path);
    loopId = loopId.Replace(',', '|');
}
```

双击页面中的“广告位轮换管理”超级链接按钮，触发其“likbtnAD_Click”事件。在该事件中使用 Response 的 Redirect 方法实现将页面跳转到“manage_AD.aspx”页面中，实现的代码如下。

例程 29 代码位置：光盘\mr\30\ LoopAD \ Default.aspx.cs

```
protected void likbtnAD_Click(object sender, EventArgs e)
{
    Response.Redirect("manage_AD.aspx");
}
```

30.4.4 广告位轮换管理页面设计

广告位轮换管理页面实现在“manage_AD.aspx”页，在广告位轮换管理页面中，能够添加循环广告，并且能够指定广告位轮换的位置和链接的网站的网址，并且该网址写在“imgFile/LoopId.txt”文本文件中，打开文本文件可以看到链接的网址已经写到文件中运行效果如图 30.19 所示，广告轮换管理页面运行效果如图 30.20 所示。

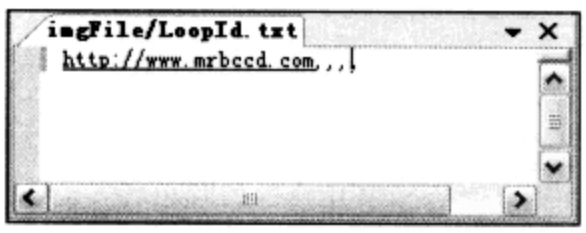


图 30.19　打开 imgFile/LoopId.txt 文本文件运行效果图

图 30.20　广告位轮换管理页面运行效果图

1. 页面设计

程序开发的步骤如下。

（1）在应用程序中创建 1 个 Web 窗体，其主页默认为 manage_AD.aspx。作为广告位轮换管理页。

（2）在页面中添加 1 个 Table（表格）控件为整个页面布局。从“工具箱”选项卡中拖放 1 个 FileUpload 控件、1 个 RadioButtonList 控件和 2 个 Button 按钮到窗体中，设置控件的属性。

页面中各个控件的属性设置及其用途如表 30.2 所示。

表 30.2　　广告位轮换管理页控件列表

控件类型	控件名称	主要属性设置	用　途
Table	无	无	页面布局
FileUpload	将控件的名称设置为 fileupAD	无	选择要上传的图片
RadioButtonList	将控件的名称设置为 radListPlace	向 Items 属性中添加“位置 1”、“位置 2”、“位置 3”、“位置 4”这 4 个成员。并且将“位置 1”的 Seleced 属性设置为 True	用于选择广告位的位置
TextBox	控件的名称设置为 txtADurl	无	用于输入链接的路径

2. 代码实现

在页面中双击“上传”按钮，触发其 btnUp_Click 事件，实现的是添加循环播放的广告，实现的代码如下。

例程 30　代码位置：光盘\mr\30\ LoopAD \ manage_AD.aspx.aspx.cs

```
protected void btnUp_Click(object sender, EventArgs e)
{
    string strExtension = fileupAD.FileName.Substring(fileupAD.FileName.LastIndexOf(".") + 1);
    if (fileupAD.HasFile)
    {
        string imgPath =Server.MapPath("./imgFile")+"/"+radListPlace.SelectedValue+".jpg";
        fileupAD.SaveAs(imgPath);
        addUrl();
     }
     else
     {
         RegisterStartupScript("", "<script>alert('上传文件不能为空')</script>");
      }
    }
```

用户自定义 addUrl 方法主要是用来获取视频 id 的 txt 文件，编写的代码如下，并在每行代码中有详细的注释。

例程 31　代码位置：光盘\mr\30\ LoopAD \ manage_AD.aspx.aspx.cs

```
protected void addUrl()
{
   int intPlace =Convert.ToInt32(radListPlace.SelectedValue);
   //获取保存视频id的txt文件
   string path = Server.MapPath("./") + "imgFile/LoopId.txt";
   //获取txt文件中的内容
   string loopId = File.ReadAllText(path);
   //将txt内容以“,”分隔的字符串保存到数组中
    string[] loopids = loopId.Split(',');
    //修改指定位置的视频id
    loopids[intPlace - 1] = txtADurl.Text;
    //将数组保存到字符串中
    loopId = loopids[0];
    for (int i = 1; i < loopids.Length; i++)
    {
          loopId += "," + loopids[i];
    }
    //将字符串中的内容保存到txt文件中
    File.WriteAllText(path, loopId);
}
```

双击页面上的“返回首页”按钮，触发其 BtnSy_Click 事件，在该事件中使用 Response 的 Redirect 方法将页面跳转到网站首页中，实现的代码如下。

例程 32 代码位置：光盘\mr\30\ LoopAD \ manage_AD.aspx.aspx.cs

```
protected void BtnSy_Click(object sender, EventArgs e)
{
  Response.Redirect("Default.aspx");//跳转到网站首页当中
}
```

Ajax

第 31 章

实例位置：光盘\mr\31\

31.1 Ajax 概述

31.1.1 Ajax 定义

Ajax 和 Web2.0 在软件开发界是目前比较常用的词汇。Web 2.0 是划分网络时代的一个概念，而 Ajax 则是属于这个时代的一种技术。Ajax 不是一种语言，而是用来描述一组技术的集合，提高互联网性能的一种关键的技术。Ajax 全称是 Asynchronous JavaScript and XML（异步 JavaScript 和 XML 技术），囊括 JavaScript 脚本语言、CSS 样式表、XMLHttpRequest 数据交换对象和 DOM 文档对象或 XMLDOM 文档对象。

Ajax、异步 JavaScript 与 XML，是使用客户端脚本与 Web 服务器交换数据的 Web 应用开发方法。通过 Ajax，Web 页面不用打断交互流程就可以进行重新加载，从而实现动态更新。使用 Ajax，可以创建类似本地桌面 Windows 应用程序。

Ajax 它更像是一个模式、一个标志，或是描述设计技巧的一种方法。对于刚开始了解 Ajax 的程序开发人员来说，会有一种全新的感觉。其实 Ajax 的所有组件在很多年前就已经开发出来了，在 2004 至 2005 年出现了一些基于 Ajax 的非常动态的 Web 用户接口（即 UI），尤其是 Google 的 GMail 与 Maps 应用系统和照片共享网站 Flickr，这些 UI 充分地使用了后台通道，被一些开发者称为“Web 2.0”，从而导致了大家对 Ajax 应用兴趣的猛涨。

31.1.2 Ajax 运行原理

Ajax 的核心是 JavaScript XmlHttpRequest 对象，该对象在 Internet Explorer 5 中首次引入，它是一种支持异步请求的技术。简而言之，XmlHttpRequest 对象可以使用 JavaScript 向服务器提出请求并处理响应，而不阻塞用户。

在创建 Web 站点时，在客户端执行屏幕更新为用户提供了很大的灵活性。下面是使用 Ajax 可以完成的功能。

- 动态检测网页中的数据，无须页面重复加载。例如，动态检测网络考试系统倒计时时间，无须页面重复加载，并等待服务器重新发送整个页面。
- 提升站点的性能，这是通过减少从服务器下载的数据量而实现的。例如，在电子商城网的购物车页面，当更新购物车中的一项物品的数量时，通常会重新载入整个页面。如果使用 Ajax 计算新物品的总数量，服务器只会返回新物品总数量，而无须重新载入整个页面。

31.2 搭建 Ajax 开发环境

31.2.1 Ajax 开发环境下载与安装

具体步骤如下。

（1）访问 http：//ajax.asp.net 网站，单击首页下方的“Download ASP.NET Ajax V1.0”超级连接，如图 31.1 所示。

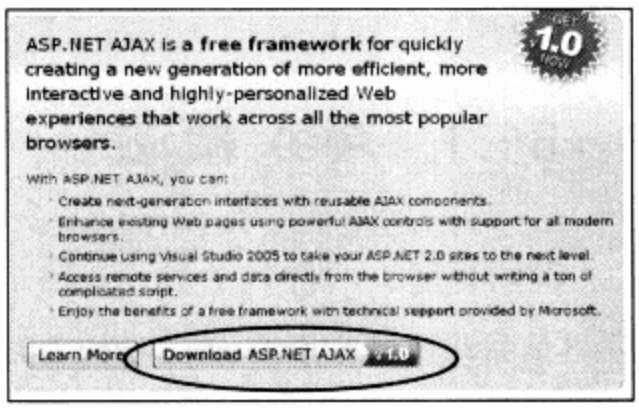


图 31.1 选择“Download ASP.NET Ajax V1.0”超级连接

（2）进入下一个页面，如图 31.2 所示，在页面中单击“Download ASP.NET Extensions v1.0”超级连接，进入微软公司网站页面。

图 31.2 选择“Download ASP.NET Extensions v1.0”超级连接

（3）如图 31.3 所示，在页面中单击“Download”按钮，弹出“文件下载”对话框，如图 31.4 所示，单击“保存”按钮将文件保存到相应的位置，并开始下载。

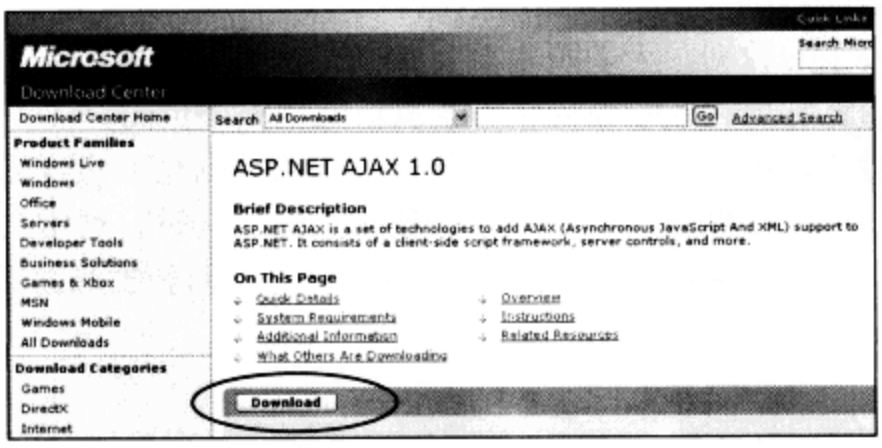


图 31.3 微软公司网站页面

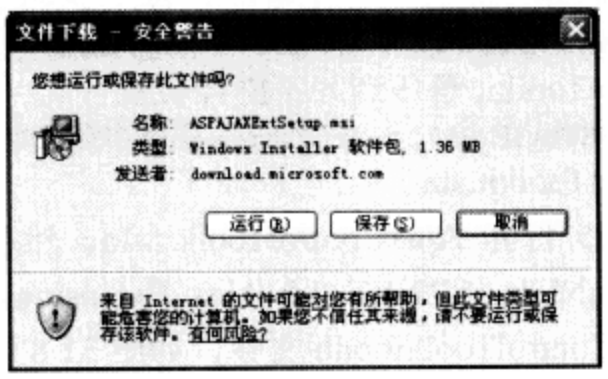


图 31.4　下载并保存到本地计算机

(4) 下载完成后，执行 ASPAjaxExtSetup.msi 安装文件，弹出如图 31.5 所示的界面。

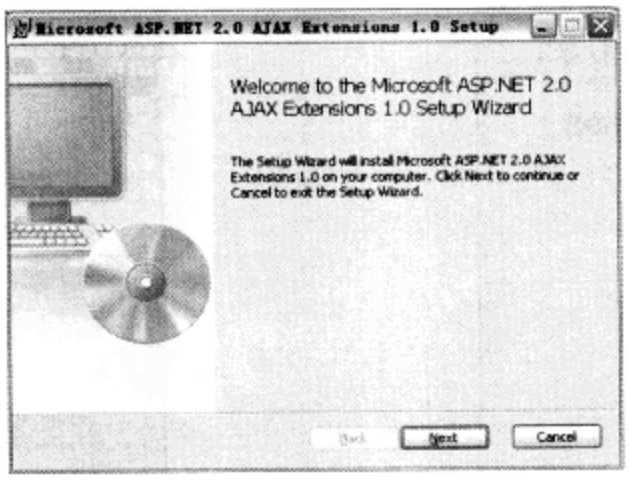


图 31.5　安装第一步界面

(5) 单击“Next”按钮，进入 Atlas 安装协议界面，如图 31.6 所示。

(6) 在如图 31.6 所示的页面中，将“I accept the terms in the License Agreement”复选框选中，意为接受安装协议，然后单击“Next”按钮，开始安装。安装完毕后将会出现如图 31.7 所示的界面，单击“Finish”按钮完成操作。

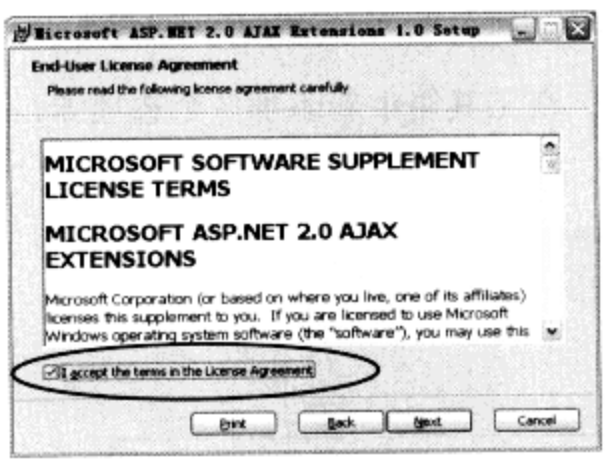


图 31.6　接受安装协议

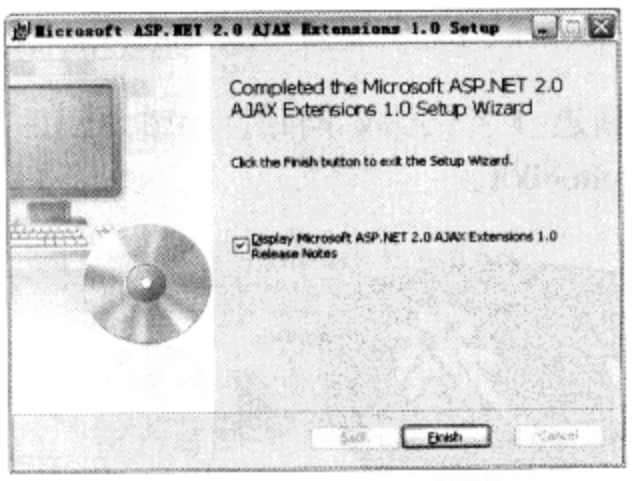


图 31.7　安装完毕

31.2.2　AjaxControlToolkit 下载与安装

ASP.NET AjaxControlToolkit 是在 ASP.NET Ajax 基础之上构建的，它是一个免费的资源，任何程序开发人员都可以使用该资源。下载 AjaxControlToolkit 地址为 http://ajax.asp.net/downloads/default.aspx?tabid=47。

下载完成后，需要对环境设置如下。

（1）在安装 ASPAjaxExtSetup.msi 后，会在系统 C 盘生成如下文件夹路径：

```
C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 ajax Extensions\v1.0.61025
```

（2）将下载的 AjaxControlToolkit 解压到如下路径位置：

```
C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 ajax Extensions\v1.0.61025\AjaxContronlToolkit\
```

（3）双击运行 AjaxControlToolkit.sln。

（4）用 Visual Studio 2005 打开 AjaxControlToolkit.sln，然后编译 TemplateVSI 项目，在 C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 Ajax Extensions\SampleWebSite\Bin 下面生成 AjaxControlToolkit.dll 和 AjaxControlToolkit.pdb 文件，如图 31.8 所示。

（5）将这个两个文件复制到 C:\Program Files\Microsoft asp.NET\ASP.NET 2.0 Ajax Extensions\Binaries 文件夹下面，这样就可以在 Visual Studio 2005 中使用 AJAX.NET 控件了，如图 31.9 所示。

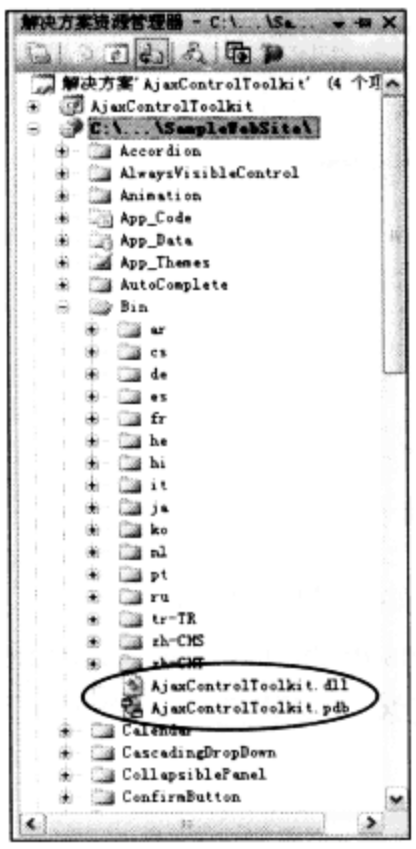


图 31.8　AjaxControlToolkit.dll 和 AjaxControlToolkit.pdb 文件

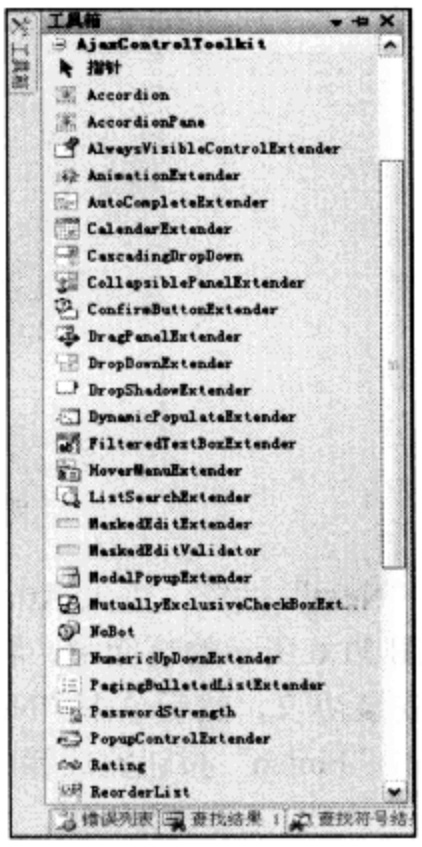


图 31.9　AjaxControlToolkit 控件

（6）新建 1 个 Ajax 网站，如图 31.10 所示，在工具箱中新添加一个选项卡，并命名为 AjaxControltoolkit。

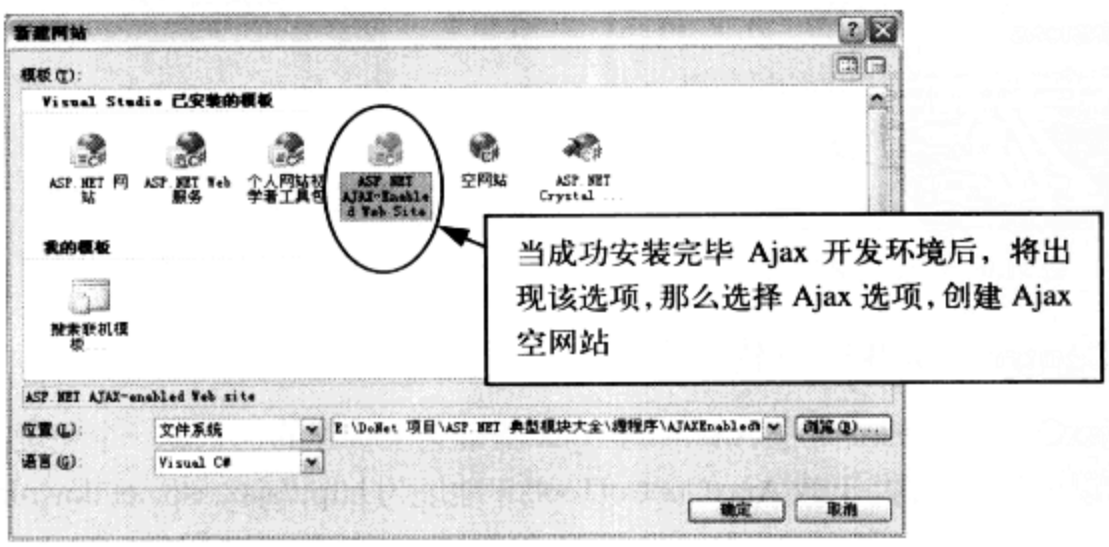


图 31.10　创建 Ajax 网站

（7）在 AjaxControltoolkit 选项卡上单击鼠标右键，在弹出的快捷菜单中选择“选择项”，浏览并找到 AjaxControlToolkit.dll 文件，然后添加到选项卡中，即可将 Ajax 控件引用到 Visual Studio 2005 集成开发环境中。

31.2.3 创建 Ajax 空网站

当成功安装 Ajax 开发环境和 AjaxControlToolkit 工具包后，读者就可以创建 Ajax 网站了，创建的具体步骤在 31.2.2 节中讲解安装 AjaxControlToolkit 工具包的时已经介绍过了，在此不再做详细介绍。

另外，Ajax 开发环境中的最基本的 ScriptManager 控件、UpdatePanel 控件、Timer 控件的详细讲解，笔者在 19.2.2 节中已经详细介绍。

31.3 Ajax 开发典型应用

31.3.1 Ajax 多样式验证

1. 概述

在开发 Web 应用程序过程中，数据的验证是不可缺少的一个功能，ASP.NET 中的验证控件可以实现验证功能，但是验证显示结果并不能友好地、明显地提示给用户，ASP.NET Ajax Control Toolkit 中的 ValidatorCallout 控件可以解决这一问题，如图 31.11 所示。

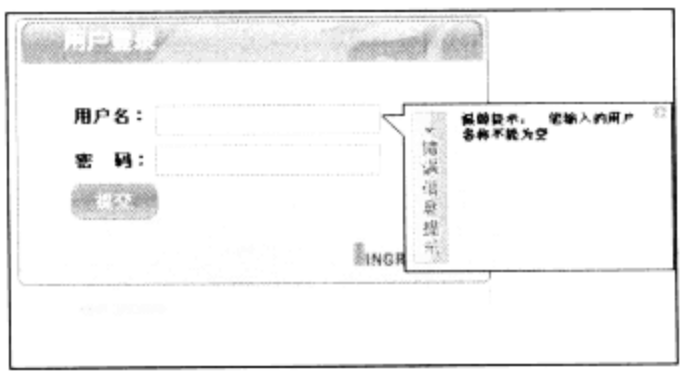


图 31.11 Ajax 多样式验证

2. 关键技术

本实例主要通过 ASP.NET Ajax Control Toolkit 中的 ValidatorCallout 控件实现，ValidatorCallout 控件主要实现多样式验证。通常情况下程序人员只需要设置相关属性就可以实现 ValidatorCallout 控件的多样式验证。ValidatorCallout 控件主要属性及说明如表 31.1 所示。

表 31.1 ValidatorCallout 控件主要属性及说明

属 性	说 明
TargetControllID	设置 ASP.NET 服务器端验证控件的 ID
Width	提示信息框的显示宽度
CloseImageUrl	指定 Close 图像
WarningIconImageUrl	指定警告图像
HighlightCssClass	设置验证结果的样式
Animations	设置验证结果的动画
OnShow	显示验证结果时的动画
OnHide	隐藏验证结果时的动画

3. 实现过程

（1）新建 1 个 Ajax 网站，将其命名为 ValidatorCallout，默认主页为 Default.aspx。

（2）在 Default.aspx 页中添加 1 个 ScriptManager 控件、1 个 RequiredFieldValidator 控件、1 个 ValidatorCallout 控件和 1 个 TextBox 控件，其中 ScriptManager 控件主要用于管理 Web 页面中的 Ajax 控件，RequiredFieldValidator 控件完成验证功能，ValidatorCallout 控件实现多样式验

证显示，TextBox 控件输入要验证的文本。

（3）RequiredFieldValidator 控件的主要属性设置，下代码如。

例程 1　代码位置：光盘\mr\31\ValidatorCallout\Default.aspx

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server" ControlToValidate="TextBox1"
                                        ErrorMessage="温馨提示：　您输入的用户名称不能为空"
Display="None"></asp:RequiredFieldValidator>
```

（4）ValidatorCallout 控件的主要属性设置，代码如下。

例程 2　代码位置：光盘\mr\31\ValidatorCallout\Default.aspx

```
                <cc1:ValidatorCalloutExtender ID="ValidatorCalloutExtender1" runat="server" TargetControlID="Required
FieldValidator1"   HighlightCssClass="Validator" CloseImageUrl="icon.gif" WarningIconImageUrl="error.gif">
                </cc1:ValidatorCalloutExtender>
```

31.3.2　Ajax 密码强度提示

1. 概述

智能密码强度检测提供的功能非常实用，而且是非常重要的一项功能，它能够提示用户所输入密码安全性的强弱，如图 31.12 所示。

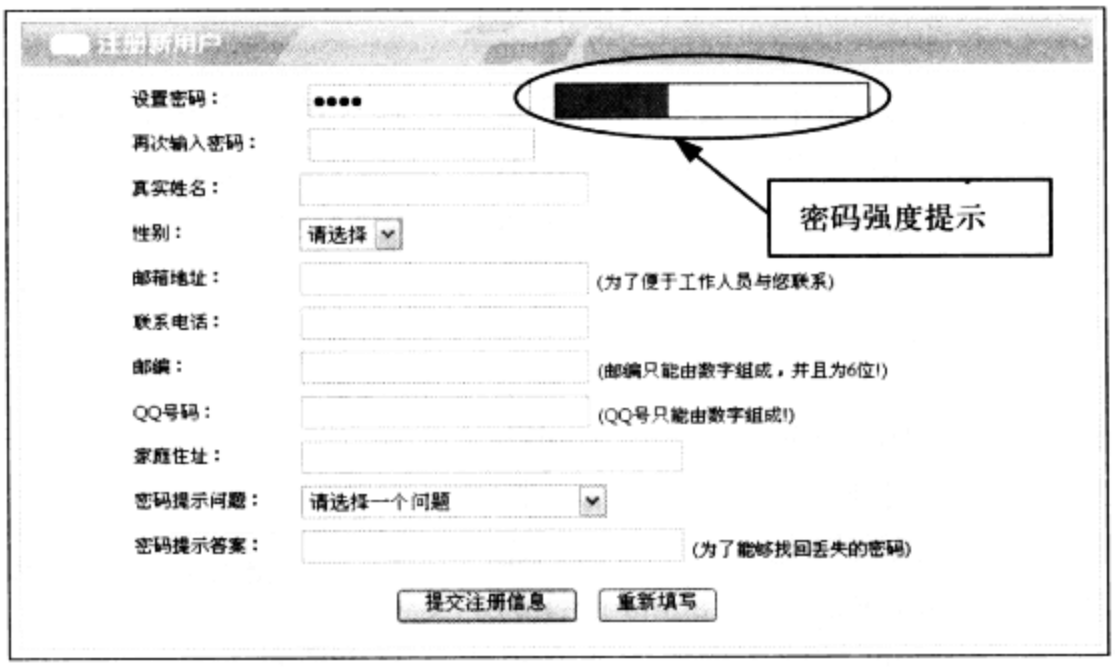


图 31.12　密码强度检测

2. 关键技术

PasswordStrength 控件是 ASP.NET Ajax Control Toolkit 版本里面提供的一个检测密码强度控件，当用户在密码框中输入密码时，文本框的后面会有一个密码强度提示，这种提示有两种方式：文本和进度条。提示信息的位置也可以由程序人员自己设置。另外，当密码框失去焦点时提示信息会自动消失。表 31.2 所示为 PasswordStrength 控件的主要属性及说明。

表 31.2　PasswordStrength 控件的主要属性及说明

属　性	说　明
TargetControlID	要检测密码的 TextBox 控件 ID
DisplayPosition	密码强度提示的信息的位置，如，DisplayPosition="RightSidelLeftSidel BelowLeft"
StrengthIndicatorType	强度信息提示方式，包括文本和进度条 StrengthIndicatorType="Textl BarIndicator"
PreferredPasswordLength	密码的长度
PrefixText	用文本方式时开头的文字 PrefixText="强度："
TextCssClass	用文本方时文字的 CSS 样式

续表

属　性	说　明
MinimumNumericCharacters	密码中最少要包含的数字数量
MinimumSymbolCharacters	密码中最好要包含的符号数量（*，#）
RequiresUpperAndLowerCaseCharacters	是否需要区分大小写
TextStrengthDescriptions	文本方式时的文字提示信息 TextStrengthDescriptions="极弱;弱;中等;强;超强"
BarIndicatorCssClass	进度条的 CSS 样式
BarBorderCssClass	进度条边框的 CSS 样式
HelpStatusLabelID	帮助提示信息的 Lable 控件 ID
CalculationWeightings	密码组成部门所占的比重，其值的格式为“A；B；C；D”。其中 A 表示长度比重，B 表示数字的比重，C 表示大写的比重，D 表示特殊符号的比重。A、B、C、D 四个值的和必须为 100，默认值为“50；15；15；20”

3. 实现过程

（1）新建 1 个 Ajax 网站，将其命名为 PasswordStrength，默认主页为 Default.aspx。

（2）在 Default.aspx 页中主要添加 1 个 ScriptManager 控件、1 个 PasswordStrength 控件和 1 个 TextBox 控件，其中 ScriptManager 控件主要用于管理 Web 页面中的 Ajax 控件，PasswordStrength 控件实现密码强度提示功能，TextBox 控件输入要验证的密码文本。

（3）在 Head 标记中添加验证进度条样式。代码如下。

例程 3　代码位置：光盘\mr\31\PasswordStrength\Default.aspx

```
<style type="text/css">
 .bartype
{
   color:blue;
   background-color:green;
}
 .barborder
{
 border-style:solid;
 border-width:1px;
 width:200px;
 vertical-align:middle;
}
        .aaa
        {
        background-color:#047AFD;
            color:#ffffff;
            font-family:Arial;
            font-size:9pt;
            padding: 2px 3px 2px 3px;
        }
  </style>
```

（4）设置 PasswordStrength 控件的属性，其代码如下。

例程 4　代码位置：光盘\mr\31\PasswordStrength\Default.aspx

```
<cc1:PasswordStrength ID="PasswordStrength1" runat="server" TargetControlID="TextBox1"
    DisplayPosition="RightSide" TextCssClass="aaa" HelpHandlePosition="BelowLeft"
    MinimumNumericCharacters="2" MinimumSymbolCharacters="2" StrengthIndicatorType="BarIndicator"
    PrefixText="密码强度：" PreferredPasswordLength="10" RequiresUpperAndLowerCaseCharacters="true"
    TextStrengthDescriptions="很差;差;一般;好;很好" CalculationWeightings="40;20;20;20" BarIndicator
CssClass="bartype"
    BarBorderCssClass="barborder">
</cc1:PasswordStrength>
```

31.3.3 Ajax 智能匹配检索

1. 概述

使用谷歌搜索引擎的用户都知道，只要在文本框中输入部分关键字，就能够显示相关搜索提示信息列表。本例中通过 ASP.NET 来实现这样的功能，程序运行结果如图 31.13 所示。

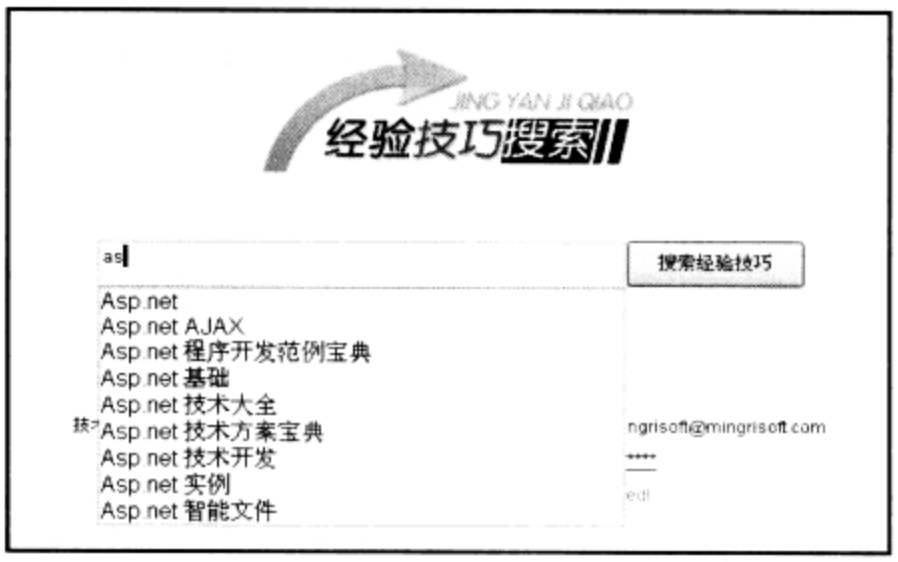


图 31.13 职能匹配检索

2. 关键技术

本实例的核心技术是通过 ASP.NET Ajax Control Toolkit 中的 AutoCompleteExtender 控件实现。

AutoCompleteExtender 控件实现自动输入建议的功能，通过调用 WebService 或本页面对应的方法名来获取提示数据，供用户能达到自动选择的功能。AutoCompleteExtender 控件的主要属性及说明如表 31.3 所示。

表 31.3 AutoCompleteExtender 控件的主要属性及说明

属　性	说　明
TargetControlID	指定将被辅助完成自动输入的控件 ID，这里的控件只能是 TextBox
ServicePath	指出提供服务的 Web 服务路径，若不指出则 ServiceMethod 表示本页面对应的方法名
ServiceMethod	指出提供服务的方法名，例如，public string[] Method(string prefixText, int count)，其中参数 prefixText 是用户输入的关键字；参数 count 是所需要获取提示数据的数量；2 个参数都会自动传给 WebService 的 ServiceMethod 方法），返回值是用户所获得提示数据的来源数组
MinimumPrefixLength	指出开始提供提示服务时，TextBox 控件应有的最小字符数，默认值为 3
CompletionInterval	从服务器读取数据的时间间隔，默认为 1000，单位：毫秒
EnableCaching	是否在客户端缓存数据，默认为 True
CompletionSetCount	显示的条数，默认值为 10

3. 实现过程

（1）新建 1 个 Ajax 网站，将其命名为 AutoCompleteExtender，默认主页为 Default.aspx。

（2）在 Default.aspx 页中主要添加 1 个 ScriptManager 控件、1 个 AutoCompleteExtender 控件和 1 个 TextBox 控件，其中 ScriptManager 控件主要用于管理 Web 页面中的 Ajax 控件，AutoCompleteExtender 控件实现自动完成功能，TextBox 控件接收输入检索关键字。

（3）创建 1 个 Web 服务，将其命名为 KeyFind.asmx，该服务主要完成智能检索功能。

（4）在 KeyFind.asmx Web 服务的 KeyFind.cs 文件下实现代码如下。

例程 5　代码位置：光盘\mr\31\AutoCompleteExtender\App_Code\KeyFind.cs

```
using System;
using System.Web;
using System.Collections;
using System.Web.Services;
using System.Web.Services.Protocols;
//引入空间
using System.Data;
using System.Data.OleDb;
using System.Configuration;
/// <summary>
/// KeyFind 的摘要说明
/// </summary>
[WebService(Namespace = "http://tempuri.org/")]
[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
//添加服务脚本（必须添加，否则程序不能正常运行）
[System.Web.Script.Services.ScriptService]
public class KeyFind : System.Web.Services.WebService
{

    public KeyFind()
    {
        //如果使用设计的组件，请取消注释以下行
        //InitializeComponent();
    }
    //定义数组保存获取的内容
    private string[] autoCompleteWordList = null;
    //2个参数“prefixText”表示用户输入的前缀，count表示返回的个数
    [WebMethod]
    public String[] GetCompleteDepart(string prefixText, int count)
    {
        ///检测参数是否为空
        if (string.IsNullOrEmpty(prefixText) == true || count <= 0) return null;
        // 如果数组为空
        if (autoCompleteWordList == null)
        {
            //读取数据库的内容
            OleDbConnection conn = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" +
Server.MapPath("Ex18_02.mdb"));
            conn.Open();
            OleDbDataAdapter da = new OleDbDataAdapter("select keyName from keyInfo   where keyName like'" +
prefixText + "%' order by keyName", conn);
            DataSet ds = new DataSet();
            da.Fill(ds);
            //读取内容文件的数据到临时数组
            string[] temp = new string[ds.Tables[0].Rows.Count];
            int i = 0;
            foreach (DataRow dr in ds.Tables[0].Rows)
            {
                temp[i] = dr["keyName"].ToString();
                i++;
            }
            Array.Sort(temp, new CaseInsensitiveComparer());
            //将临时数组的内容赋给返回数组
            autoCompleteWordList = temp;
            if (conn.State == ConnectionState.Open)
                conn.Close();
        }
        //定位二叉树搜索的起点
        int index = Array.BinarySearch(autoCompleteWordList, prefixText, new CaseInsensitiveComparer());
        if (index < 0)
```

```
            {    //修正起点
                index = ~index;
            }
            //搜索符合条件的数据
            int matchCount = 0;
            for (matchCount = 0; matchCount < count && matchCount + index < autoCompleteWordList.Length; matchCount++)
            {    ///查看开头字符串相同的项
                if (autoCompleteWordList[index + matchCount].StartsWith(prefixText, StringComparison.CurrentCulture
IgnoreCase) == false)
                {
                    break;
                }
            }
            //处理搜索结果
            string[] matchResultList = new string[matchCount];
            if (matchCount > 0)
            {    //复制搜索结果
                Array.Copy(autoCompleteWordList, index, matchResultList, 0, matchCount);
            }
            return matchResultList;
        }
    }
```

(5) 回到 Default.aspx 页的源视图，设置 AutoCompleteExtender 控件的属性，代码如下。

例程 6　代码位置：光盘\mr\31\AutoCompleteExtender\Default.aspx

```
    <cc1:AutoCompleteExtender ID="AutoCompleteExtender1" runat="server" TargetControlID="TextBox1"
                    ServicePath="KeyFind.asmx" CompletionSetCount="10" MinimumPrefixLength="1" ServiceMethod=
"GetCompleteDepart">
                </cc1:AutoCompleteExtender>
```

31.3.4　Ajax 实现许愿墙

1. 概述

许愿墙是目前网络比较流行的功能，用来发送祝福，将自己的愿望放在网页上通过网络传播，如图 31.14 所示。

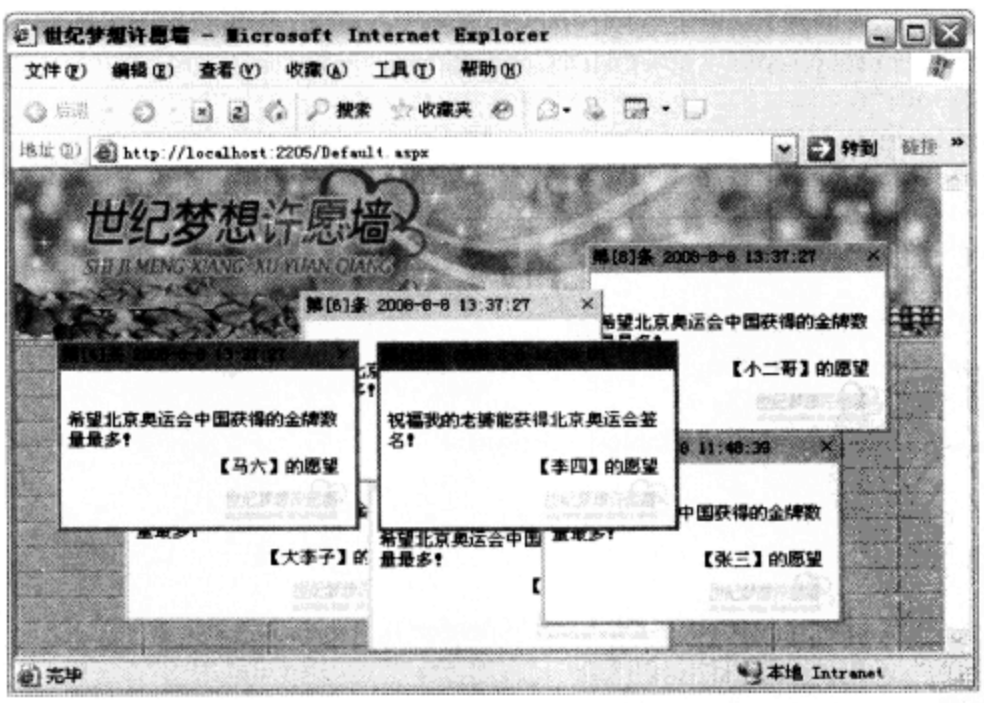


图 31.14　许愿墙

2. 关键技术

许愿墙的每条愿望是从数据库中获取的，然后通过程序生成 Html 语言输出到 Web 页中。主要代码如下。

```
//创建许愿墙，并显示许愿信息
foreach (DataRow row in ds.Tables[0].Rows)
{    ///产生位置的随机起始位置
     lefIndex = indexRandom.Next(30, 750);
     topIndex = indexRandom.Next(30, 420);
     wall = new StringBuilder();
     //创建一个<div></div>，用来作为许愿墙
     wall.Append("<div id=\"divBless" + row["ID"].ToString() + "\" class=\"BlessPanel\" ");
     //添加样式
     wall.Append("style=\"position:absolute;");
     wall.Append("left:" + lefIndex + "px;");
     wall.Append("top:" + topIndex + "px;");
     wall.Append("background-color:" + row["BackColor"].ToString() + ";");
     wall.Append("z-index:" + row["ID"].ToString() + ";\" ");
     //添加鼠标事件
     wall.Append("onmousedown=\"getPanelFocus(this)\">");
     //添加表格
     wall.Append("<table border=\"0\">");
     wall.Append("<td style=\"cursor:move;\" width=\"98%\" ");
     //添加鼠标事件
     wall.Append("onmousedown=Down(divBless" + row["ID"].ToString() + ")>");
     wall.Append("第[" + row["ID"].ToString() + "]条 ");
     wall.Append(row["dreamDate"].ToString() + " " + "</td><td style=\"cursor:hand;\" ");
     wall.Append("onclick=\"ssdel()\" width=\"2%\">×</td></tr>");
     wall.Append("<tr><td style=\"height:100px;padding:5px;\" colspan=\"2\">");
     wall.Append(row["dream"].ToString().Trim());
     //添加许愿人姓名
     wall.Append("<div style=\"padding:5px;float:right;\">【" + row["dreamName"].ToString() + "】的愿望
</div></td></tr></table>");
     wall.Append("</div>");
     //追加到输出字符串中
     allWall.Append(wall.ToString());
}
```

每条愿望的移动、删除、获得焦点功能是通过 JavaScript 来控制，关键代码如下：

```
//创建许愿墙，并显示许愿信息
foreach (DataRow row in ds.Tables[0].Rows)
{    ///产生位置的随机起始位置
     lefIndex = indexRandom.Next(30, 750);
     topIndex = indexRandom.Next(30, 420);
     wall = new StringBuilder();
     //创建一个<div></div>，用来作为许愿墙
     wall.Append("<div id=\"divBless" + row["ID"].ToString() + "\" class=\"BlessPanel\" ");
     //添加样式
     wall.Append("style=\"position:absolute;");
     wall.Append("left:" + lefIndex + "px;");
     wall.Append("top:" + topIndex + "px;");
     wall.Append("background-color:" + row["BackColor"].ToString() + ";");
     wall.Append("z-index:" + row["ID"].ToString() + ";\" ");
     //添加鼠标事件
     wall.Append("onmousedown=\"getPanelFocus(this)\">");
     //添加表格
     wall.Append("<table border=\"0\">");
     wall.Append("<td style=\"cursor:move;\" width=\"98%\" ");
     //添加鼠标事件
     wall.Append("onmousedown=Down(divBless" + row["ID"].ToString() + ")>");
     wall.Append("第[" + row["ID"].ToString() + "]条 ");
     wall.Append(row["dreamDate"].ToString() + " " + "</td><td style=\"cursor:hand;\" ");
     wall.Append("onclick=\"ssdel()\" width=\"2%\">×</td></tr>");
     wall.Append("<tr><td style=\"height:100px;padding:5px;\" colspan=\"2\">");
     wall.Append(row["dream"].ToString().Trim());
     //添加许愿人姓名
```

```
                wall.Append("<div style=\"padding:5px;float:right;\">【" + row["dreamName"].ToString() + "】的愿望
</div></td></tr></table>");
                wall.Append("</div>");
                //追加到输出字符串中
                allWall.Append(wall.ToString());
            }
```

3. 实现过程

（1）新建 1 个 Ajax 网站，将其命名为 Wall，默认主页为 Default.aspx。

（2）创建 Jscript 文件，并且命名为 js_wallControl.js。该文件用于控制 DIV 的移动、删除和获得焦点。实现代码如下：

```
// JScript 文件
    //-- 控制层删除(删除许愿墙) -->
    function ssdel()
    {
        if(event)
        {
            lObj = event.srcElement;
            while (lObj && lObj.tagName != "DIV") lObj = lObj.parentElement ;
        }
        var id = lObj.id
        document.getElementById(id).removeNode(true);
    }
    //-- 控制层删除 -->

    //-- 控制层移动 （移动许愿墙）-->
    var Obj=''
    var index=10000;    //z-index的值;
    document.onmouseup=Up;
    document.onmousemove=Move;
    function Down(Object)
    {
        Obj = Object.id;
        document.all(Obj).setCapture();
        pX = event.x - document.all(Obj).style.pixelLeft;
        pY = event.y - document.all(Obj).style.pixelTop;
    }
    function Move()
    {
        if(Obj != '')
        {
            document.all(Obj).style.left = event.x - pX;
            document.all(Obj).style.top = event.y - pY;
        }
    }
    //-- 控制层移动 -->
    function Up()
    {
        if(Obj != '')
        {
            document.all(Obj).releaseCapture();
            Obj='';
        }
    }
    ///获取控制层焦点（获取许愿墙焦点）;
    function getPanelFocus(obj)
    {
        if(obj.style.zIndex!=index)
        {
            index = index + 2;
            var idx = index;
```

```
                obj.style.zIndex=idx;
            }
        }
```

（3）创建样式表 StyleSheet.css，主要用来设置 DIV 样式（每条愿望的样式）。代码如下：

```
body
{
    font-family: 宋体;
    font-size: 12px;
}
.BlessPanel                    /*设置DIV样式。
{
    position:absolute;
    width:200px;
    height:auto;
    border-top-style:Ridge;
    border-right-style:Ridge;
    border-left-style:Ridge;
    border-bottom-style:Ridge;
    border-width:1pt;
}
```

（4）在 Default.aspx.cs 编写代码来实现许愿墙的核心功能。代码如下：

```
using System;
using System.Data;
using System.Configuration;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.OleDb;
using System.Text;
public partial class _Default : System.Web.UI.Page
{
    // 许愿墙坐标的随机生成器
    private Random indexRandom = new Random();
    // 保存页面输出的内容
    protected string AllBlessString = string.Empty;
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!Page.IsPostBack)
        {
            BindPageData();
        }
    }
    //获取许愿墙信息
    private void BindPageData()
    {
        OleDbConnection con = new OleDbConnection(@"Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + Server.
MapPath("db_wall.mdb"));
        con.Open();
        OleDbDataAdapter dap = new OleDbDataAdapter("select * from tb_wall", con);
        DataSet ds = new DataSet();
        dap.Fill(ds);
        if (ds == null || ds.Tables.Count <= 0 || ds.Tables[0].Rows.Count <= 0) return;
        StringBuilder wall;
        StringBuilder allWall = new StringBuilder();
        int lefIndex;
        int topIndex;
        //创建许愿墙，并显示许愿信息
        foreach (DataRow row in ds.Tables[0].Rows)
```

```
        {    ///产生位置的随机起始位置
             lefIndex = indexRandom.Next(30, 750);
             topIndex = indexRandom.Next(30, 420);
             wall = new StringBuilder();
             //创建一个<div></div>，用来作为许愿墙
             wall.Append("<div id=\"divBless" + row["ID"].ToString() + "\" class=\"BlessPanel\" ");
             //添加样式
             wall.Append("style=\"position:absolute;");
             wall.Append("left:" + lefIndex + "px;");
             wall.Append("top:" + topIndex + "px;");
             wall.Append("background-color:" + row["BackColor"].ToString() + ";");
             wall.Append("z-index:" + row["ID"].ToString() + ";\" ");
             //添加鼠标事件
             wall.Append("onmousedown=\"getPanelFocus(this)\">");
             //添加表格
             wall.Append("<table border=\"0\">");
             wall.Append("<td style=\"cursor:move;\" width=\"98%\" ");
             //添加鼠标事件
             wall.Append("onmousedown=Down(divBless" + row["ID"].ToString() + ")>");
             wall.Append("第[" + row["ID"].ToString() + "]条 ");
             wall.Append(row["dreamDate"].ToString() + " " + "</td><td style=\"cursor:hand;\" ");
             wall.Append("onclick=\"ssdel()\" width=\"2%\">×</td></tr>");
             wall.Append("<tr><td style=\"height:100px;padding:5px;\" colspan=\"2\">");
             wall.Append(row["dream"].ToString().Trim());
             //添加许愿人姓名
             wall.Append("<div style=\"padding:5px;float:right;\">【" + row["dreamName"].ToString() + "】的愿望
</div></td></tr></table>");
             wall.Append("</div>");
             //追加到输出字符串中
             allWall.Append(wall.ToString());
        }
        //将当前DIV许愿墙的内容添加到输出字符串中
        AllBlessString += allWall.ToString();
    }
}
```

（5）在 Default.aspx 页中调用 JavaScript 文件和 CSS 文件，并且将 DIV 许愿墙内容输出到 Web 页面中，实现代码如下：

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>温馨许愿墙</title>
    <script language="JavaScript" src="js_wallControl.js"></script>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" />
</head>
<body style="background-image: url(墙.jpg)">
    <form id="form1" runat="server">
        <%= AllBlessString %>
    </form>
</body>
</html>
```

高级应用技术

第 32 章

实例位置：光盘\mr\32\

32.1 在线文本编辑器

32.1.1 在线文本编辑器的概述

在网站开发过程中，经常会开发网站评论、留言或发送信息等功能。为了使单一的文字更加突出和美观，就需要使用到在线文本编辑器。在线文本编辑器可以改变文字的大小、颜色、字体以及添加图片等效果。在线文本编辑器的主要特点是可以达到对文本所见即所得的修改方式。例如，修改发表评论文本的字体及颜色后就会马上呈现出文本修改后的样式。

目前比较流行的在线文本编辑器有很多种，包括 FCKEditor 文本编辑器、FreeTextBox 文本编辑器等，但任何在线文本编辑器都大致由菜单栏和编辑栏两部分组成。下面介绍一下这两部分的作用。

- 菜单栏：菜单栏是用来布置功能按钮的，包括粗体按钮、斜体按钮、字体选择框、字号选择框等，使用这些功能来编辑文本。
- 编辑栏：编辑栏是用户的文本输入区域，用户可以在编辑栏中输入文本。使用菜单栏编辑后的文本效果都会在该区域呈现出来。

32.1.2 制作简单的文本编辑器

在本节中将制作简单的文本编辑器，该文本编辑器的主要编辑功能包括粗体、斜体、下划线、字号设置等，如图 32.1 所示。

用户可以通过单击“提交”按钮将编辑后的文本显示在 showTxt.aspx 页面中，如图 32.2 所示。

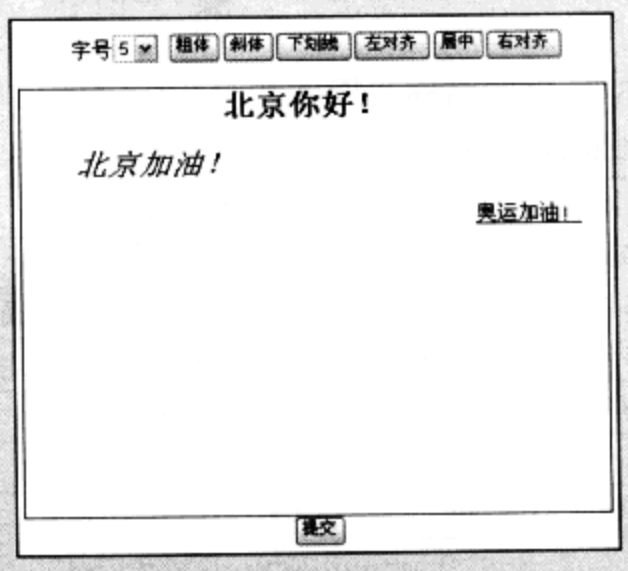


图 32.1 简单的文本编辑器

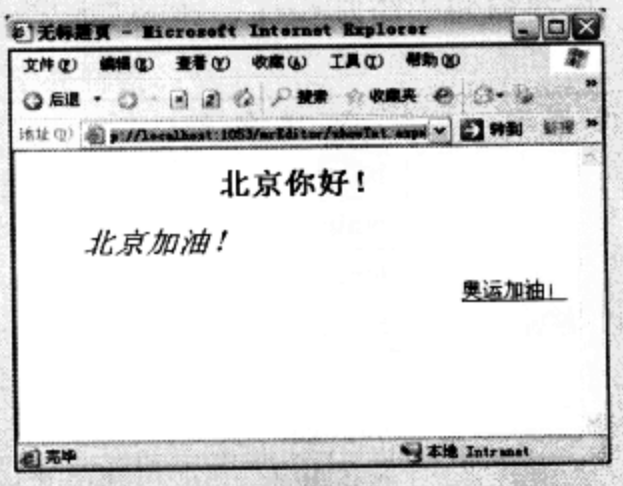


图 32.2 在 showText.aspx 页显示文本

1. 关键技术

大多数的在线文本编辑器都是使用 JavaScript 语言来实现编辑文本的，在本程序中也是利

用 JavaScript 实现的。文本的编辑区域是通过使用 iframe 框架来创建的一个区域，并使用 document 对象中的 designMode 属性来设置该区域为可编辑区域。下面介绍 designMode 属性。

designMode 属性：该属性只有在 IE5 以上的浏览器中可以应用。该属性可以控制浏览器中的某块为 HTML 编辑区域。当设置该属性的参数为“On”时表示可以编辑该区域，当为“Off”时表示不可以编辑，默认为“Off”。

文本的编辑功能主要是使用 document 对象的 execCommand 方法来实现的。下面介绍一下 execCommand 方法。

ExecCommand 方法：该方法用来执行一个命令。该命令对当前文档或所选择的区域有效。该方法常用的命令如表 32.1 所示。

表 32.1　　常用命令及说明

命　令	说　明
BackColor	设置或获取当前选中区的背景颜色
Bold	设置当前选中区为粗体
Copy	将当前选中区复制到剪贴板
Cut	将当前选中区复制到剪贴板并删除
CreateLink	将当前选中区上插入超级链接，或显示一个对话框允许用户输入超级链接的 URL
Delete	删除当前选中区
FontName	设置或获取当前选中区的字体
Italic	设置当前选中区为斜体
FontSize	设置或获取当前选中区的字体大小
ForeColor	设置或获取当前选中区的前景（文本）颜色
Indent	增加选中区的缩进
JustifyLeft	设置当前选中区左对齐
JustifyRight	设置当前选中区右对齐
justifyCenter	设置当前选中区居中对齐
Underline	设当前选中区的下划线
Outdent	减少选中区的缩进

2. 前台页面设计

（1）创建 1 个 Web 窗体，默认名为 Default.aspx。

（2）在该窗体中添加 1 个表格，该表格设置为两行，第 1 行用来布置功能按钮，在第 2 行中添加 1 个 iframe 框架用来设置为文本编辑区域。

（3）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 32.2 所示。

表 32.2　　控件类型、控件名称及说明

控件类型	控件名称	主要属性	说　明
标准/Button 控件	btnBold	设置 onclick 事件为 txtbold	实现文字设置为粗体操作
	btnItalic	设置 onclick 事件为 txtItalic	实现文字设置为斜体操作
	btnUnder	设置 onclick 事件为 txtUnderline	实现设置文字的下划线
	btnLeft	设置 onclick 事件为 txtLeft	实现文字居左操作
	btnCenter	设置 onclick 事件为 txtCenter	实现文字居中操作
	btnRight	设置 onclick 事件为 txtRight	实现文字居右操作
	btnSubmit	均为默认值	实现提交操作
HTML/ Select 控件	Select1	设置 onchange 事件为 txtFontName	实现选择字号操作
HTML/input 控件	hiddenText	设置 type 属性为 hidden 设置 runat 属性为 server	用来保存文本

在 Default.aspx 页面中编写 JavaScript 代码来实现文本的编辑功能，包括粗体设置、斜体设置、下划线设置等。JavaScript 代码如下。

例程 1 代码位置：光盘\mr\32\mrEditor\Default.aspx

```
<script language="javascript">
    //设置编辑区域
        function document.onreadystatechange()
        {
         txtArea.document.designMode="On";
        }
         //设置文字居左
         function txtLeft()
         {
              txtArea.document.execCommand("justifyLeft");
         }
         //设置文字居中
         function txtCenter()
         {
              txtArea.document.execCommand("justifyCenter");
         }
         //设置文字居右
         function txtRight()
         {
              txtArea.document.execCommand("justifyRight");
         }
         //设置文字为粗体
         function txtbold()
         {
              txtArea.document.execCommand("Bold");
         }
         //设置文字为斜体
         function txtItalic()
         {
              txtArea.document.execCommand("Italic");
         }
         //设置文字字号
         function txtFontName()
         {
              var txtSize=document.getElementById("Select1").value;
            txtArea.document.execCommand("fontsize","",txtSize)
         }
         //设置文字下划线
         function txtUnderline()
         {
              txtArea.document.execCommand("Underline");
         }
         //将文本保存到隐藏域中
         function saveTxt()
         {
              var s =txtArea.document.body.innerHTML;
              document.getElementById("hiddenText").value=s
         }
</script>
```

3. 后台代码编写

在页面的加载事件中，设置“提交”按钮的单击事件。实现代码如下。

例程 2 代码位置：光盘\mr\32\mrEditor\Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
     //设置提交按钮的单击事件
     btnSubmit.Attributes.Add("onclick", "saveTxt();");
 }
```

在“提交”按钮的单击事件中，获取隐藏域中的文本，并将文本保存到 Session 中，最后将跳转到 showTxt.aspx 页面中显示所编辑后的文本。实现代码如下。

例程 3　代码位置：光盘\mr\32\mrEditor\Default.aspx.cs

```
protected void btnSubmit_Click1(object sender, EventArgs e)
{
    //获取隐藏域中的文本
    string s = hiddenText.Value;
    //将文本保存到session中
    Session["txt"] = s;
    //跳转到showTxt页面显示文本
    Response.Redirect("showTxt.aspx");
}
```

32.1.3　应用 FCKEditor 在线文本编辑器

FCKEditor 是开源的 HTML 在线文本编辑器，该文本编辑器可以让 Web 程序拥有如 Microsoft Word 强大的编辑功能。FCKEditor 在线文本编辑器支持目前比较流行的大多数浏览器，包括 IE、FirFox 等。目前 FCKEditor 在线文本编辑器最新版本是 2.6.3。FCKEditor 在线文本编辑器界面如图 32.3 所示。

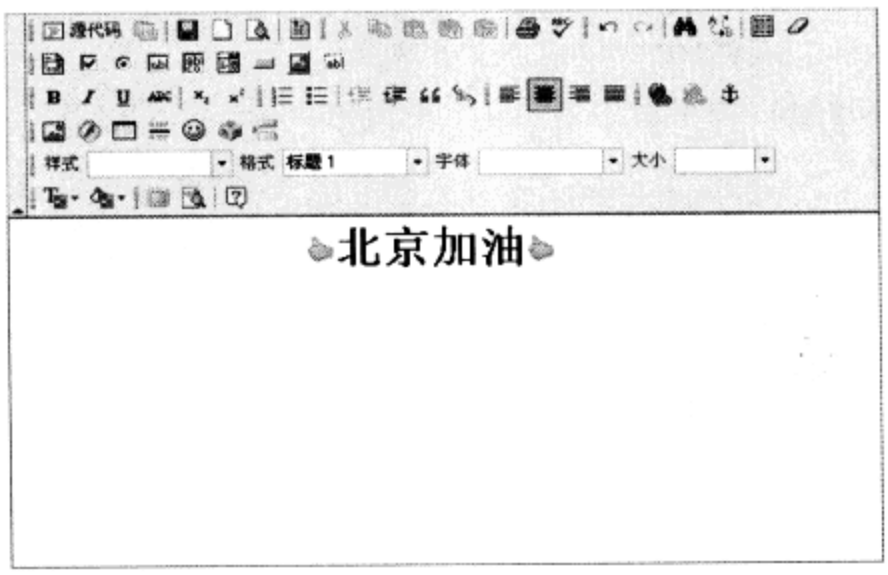


图 32.3　FCKEditor 在线文本编辑器

1. 功能介绍

FCKEditor 在线文本编辑器是目前功能比较完善、兼容性良好、使用方便的在线编辑器，该编辑器的主要功能如下。

（1）CSS 支持，更好地结合网站风格。
（2）字体格式，包括类型，大小、颜色、风格、粗体、斜体等。
（3）文本格式，包括对齐、缩进等。
（4）操作功能，包括粘贴和粘贴为纯文本、撤销和重做。
（5）图像插入、上传和服务器浏览支持。
（6）表的建立和编辑，包括添加、删除行等，表元素的编辑包括大小、颜色等。
（7）右键菜单支持。
（8）皮肤支持、插件支持、拼写检查。
（9）多国语言支持，自动用户语言检测。

2. 简化及修改 FCKEditor 文件

首先需要下载 FCKEditor 2.6.3 和 FCKEditor.NET 2.5 版的 2 个压缩包。FCKEditor 2.6.3 是目前最新的版本，该版本支持 Javascript 文件和图片等，FCKEditor.NET 是 1 个 ASP.NET 控件

DLL 文件。下面介绍一下简化 FCKEditor 文件的步骤。

（1）首先将所下载的 FCKEditor 2.6.3 压缩包解压缩，解压后的文件如图 32.4 所示。由于有很多文件是范例或源文件用不到的，因此需要将该文件简化。

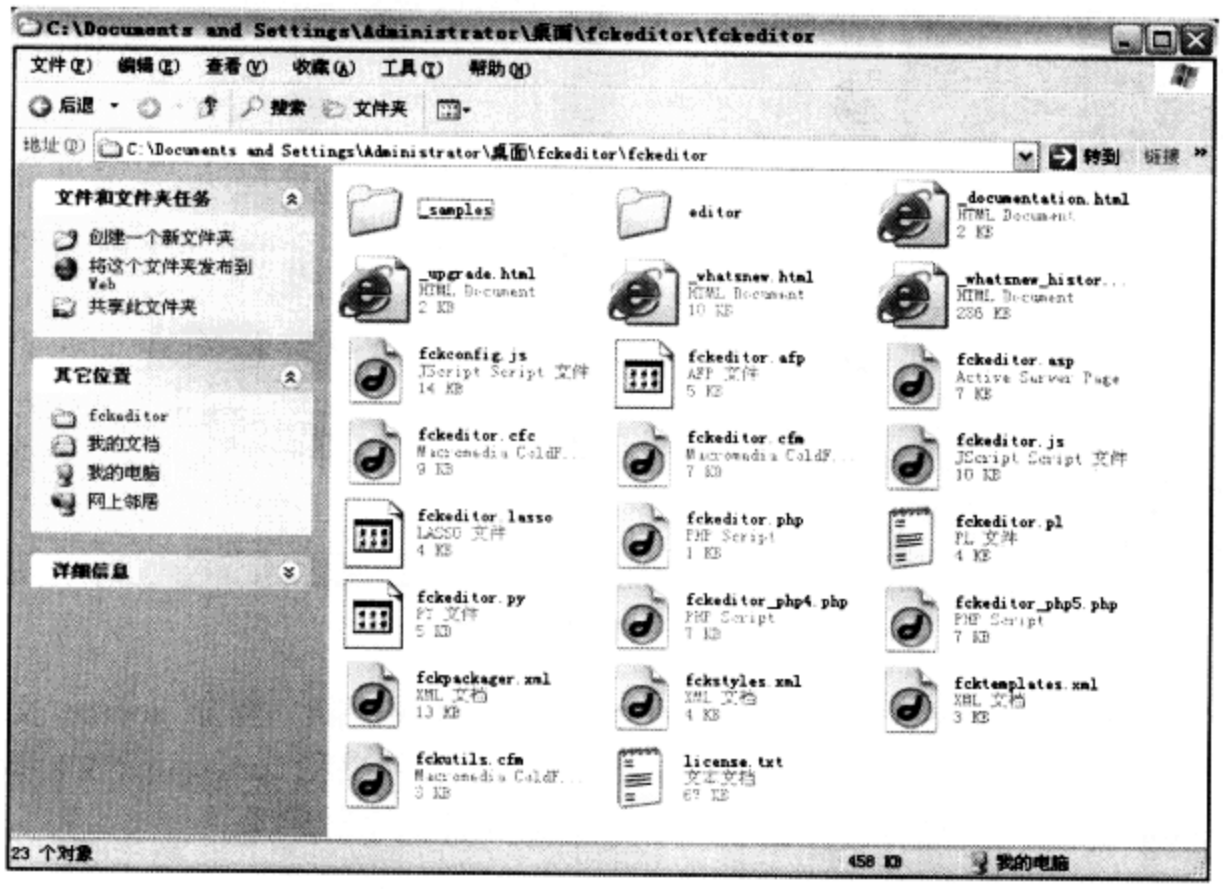


图 32.4　解压后的文件夹

（2）选中所有以“_”开头的文件和文件夹，如图 32.5 所示，将这些文件都删除掉，因为这些文件都是范例，在使用 FCKEditor 文本编辑器时不会用到。

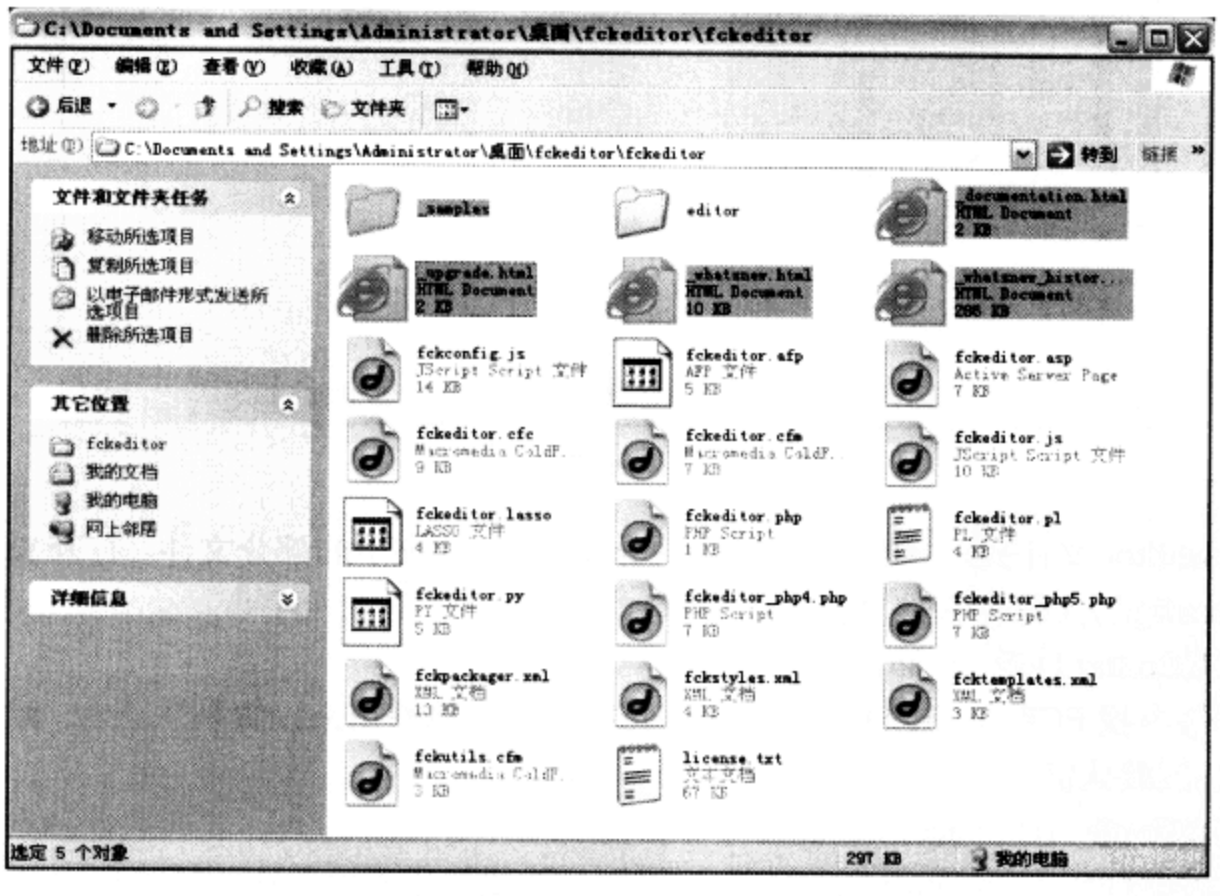


图 32.5　选中所有“_”开头的文件

（3）保留 editor 文件夹、fckconfig.js 文件、fckeditor.js 文件、fckstyles.xml 文件和 fcktemplates.xml 文件，如图 32.6 所示，将其余的文件全部删除。

图 32.6　保留指定文件

（4）进入 editor 文件夹，将该文件中的_source 文件夹删除，删除_source 文件夹后进入 filemanager 文件夹，该文件夹下有两个文件夹为 browser 文件夹和 connectors 文件夹，如图 32.7 所示。connectors 文件夹中只保留.aspx 文件夹，其余文件夹全部删除。

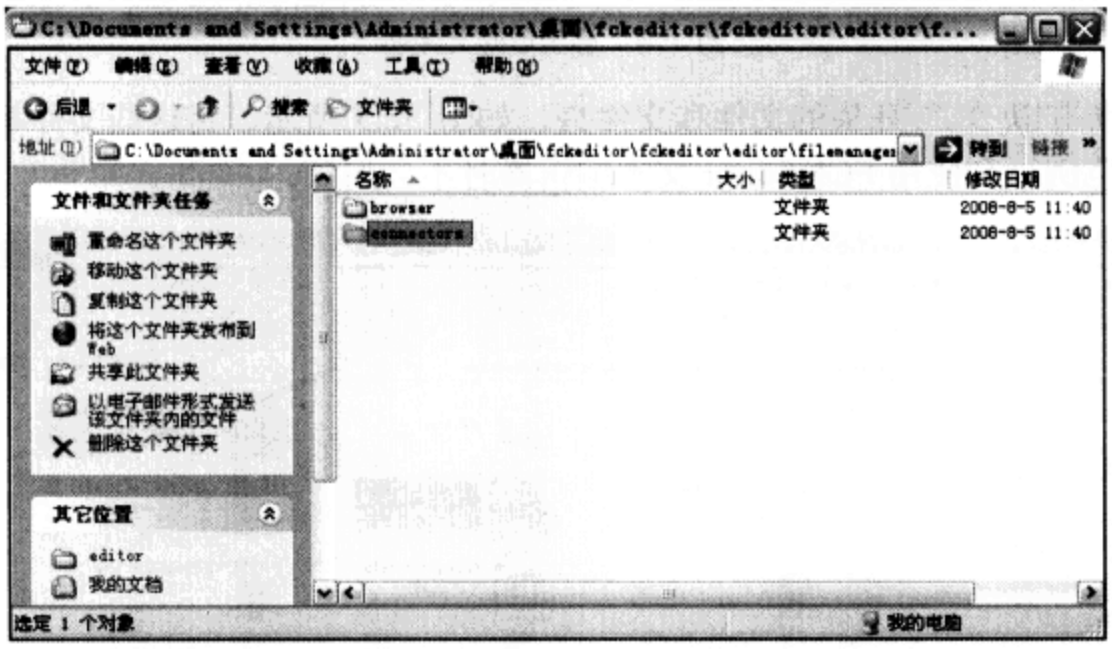


图 32.7　filemanager 文件夹下

简化 fckeditor 文件夹后，下面需要修改 fckeditor 文件夹下的部分文件。打开 fckeditor 文件夹下的 fckconfig.js 文件，在该文件中的设置如下。

- FCKConfig.DefaultLanguage 属性

在该文件中找 FCKConfig.DefaultLanguage 属性，该属性用来设置默认语言，默认的值为英文（en），把该默认值修改为中文（zh-cn）。

- FCKConfig.TabSpaces 属性

在该文件中找到 FCKConfig.TabSpaces 属性，该属性用来设置在编辑过程中是否使用 TAB 键。默认值 0 为不使用，将默认值修改为 1 使用。

- FCKConfig.FontNames 属性

在该文件中找到 FCKConfig.FontNames 属性，该属性用来添加字体。在该属性中添加中文

常用字体，包括宋体、黑体、隶书、楷体_GB2312。各字体之间使用“;”间隔。

- _FileBrowserLanguage 变量

在该文件中找到_FileBrowserLanguage 变量，该变量用来设置文件浏览器默认语言，默认值为 PHP，将默认值修改为 aspx。

- _QuickUploadLanguage 变量

在该文件中找到_FileBrowserLanguage 变量，该变量用来设置上传默认语言，默认值为 PHP，将默认值修改为 aspx。

下面将设置允许上传文件，打开 fckeditor\editor\filemanager\connectors\asp\config.aspx 文件，找到该文件中的 CheckAuthentication 方法，将该方法的返回值设置为 True。

3. 在 ASP.NET 中配置 FCKEditor 在线文本编辑器

本节将介绍在 ASP.NET 中配置 FCKEditor 在线文本编辑器，在配置前先把 FCKEditor.NET2.5 压缩包解压缩，找到解压后的文件夹下的 bin\Release 目录下的 FredCK.FCKeditorV2.dll 文件备用。具体步骤如下。

（1）打开在制作简单文本编辑器一节中所创建的项目，并将所找到的 FredCK.FCKeditorV2.dll 文件和简化修改后的 fckeditro 文件夹复制到该项目中，并在该项目中创建一个名为 userfiles 的文件夹，该文件夹用于保存所上传的文件。

（2）在项目名称上单击鼠标右键，在弹出的快捷菜单中选择“添加新项”选项。在添加新项窗口中，添加 1 个 Web 窗体，并将该窗体命名为 useFck.aspx。

（3）在工具箱中，右键单击“常规”选项卡，在弹出的快捷菜单中单击“选择项”菜单项，如图 32.8 所示。

（4）在打开的“选择工具箱项”窗口中，选择“浏览”按钮，如图 32.9 所示。在弹出的窗口中查找到项目中的 FredCK.FCKeditorV2.dll 文件后单击“确定”按钮。

图 32.8 选择“选择项”菜单项

图 32.9 “选择工具箱项”窗口

（5）完成上一步的操作后，在“常规”选项卡中会自动生成 1 个名为 FCKeditor 的控件。读者可以将该控件拖曳到设计页面中。

（6）在 Web.Config 文件中还需要配置源文件路径和上传文件路径。配置代码如下。

例程 4 代码位置：光盘\mr\32\mrEditor\Web.Config

```
<appSettings>
    <add key="FCKeditor:BasePath" value="~/fckeditor/" />
    <add key="FCKeditor:UserFilesPath" value="~/userfiles/" />
  </appSettings>
```

通过以上步骤的操作在 ASP.NET 中配置 FCKEditor 在线文本编辑器已经完成，读者可以通过 FCKeditor 控件的 Value 属性获取到 FCKeditor 控件中所编辑的文本值。

32.2 在线获取客户端网卡（MAC）地址

32.2.1 网卡（MAC）地址简介

MAC（Media Access Control，介质访问控制）地址是在媒体接入层上使用的地址，直白地说就是网卡的物理地址。网卡的物理地址通常是由网卡生产商写入至网卡的 EPROM（一种闪存芯片，通常可以通过程序擦写），它存储的是传输数据时，标识发出数据的电脑和接收数据主机的地址。这个地址与网络无关，也即无论将带有这个地址的硬件（如网卡、集线器、路由器等）接入到网络的何处，它都有相同的 MAC 地址，MAC 地址一般是不可改变的。更形象地说 MAC 地址就如同我们现实中的身份证号码一样，具有全球惟一性。

32.2.2 为什么使用网卡（MAC）地址

在开发程序时有时会使用到投票功能或限制某台机器只能领取一个账号等功能。这些功能在开发时常常使用 IP 地址来作为限制的标准。但是 IP 地址有些时候也不是很标准的。例如动态 IP 会在每次连接网络时分配不同的 IP，所以就需要考虑到 MAC 地址的全球惟一性。利用 MAC 地址来控制某机器只能领取一个账号是可取的。本节中的程序就是利用 MAC 地址来控制一台机器只能获取一个账号。

32.2.3 获取网卡（MAC）地址关键技术

在 ASP.NET 中获取 MAC 地址，首先需要获取到客户端的 IP 地址，然后使用进程执行 nbtstat –A 客户端 IP 命令获取到 NetBIOS 名称表，再使用正则表达式截取到 MAC 的地址。下面首先来介绍一下 nbtstat 命令。

nbtstat 命令：该命令用来显示基于 TCP/IP 的 NetBIOS（NetBT）协议统计资料、本地计算机和远程计算机的 NetBIOS 名称表和 NetBIOS 名称缓存。该命令的语法如下：

```
nbtstat[-a RemoteName] [-A IPAddress] [-c] [-n] [-r] [-R] [-RR] [-s] [-S] [Interval]
```

在本程序中主要使用到了-A IPAddress 参数，该参数用来显示远程计算机的 NetBIOS 名称表，其名称由远程计算机的 IP 地址指定。读者可以在“开始”菜单中选择运行选项，在弹出的“运行”窗口中输入 cmd 命令，就会打开 DOS 窗口。在该窗口中输入 nbtstat –A 读者机器的 IP，会显示 NetBIOS 名称表。在该表中可以查看 MAC 地址信息，如图 32.10 所示。

图 32.10 MAC 地址信息

在 ASP.NET 中启动进程使用的是 ProcessStartInfo 对象和 Process 对象实现的。这两个对象的说明如下。

- ProcessStartInfo 对象

该对象用来指定启动进程时使用的一组值。在该对象中常用的属性及说明如表 32.3 所示。

表 32.3 常用的属性及说明

属　性	说　明
FileName	获取或设置要启动的应用程序或文档
RedirectStandardInput	获取或设置一个 bool 值，该值指示应用程序的输入是否从 Process.StandardInput 流中读取
RedirectStandardOutput	获取或设置一个 bool 值，该值指示是否将应用程序的输出写入 Process.StandardOutput 流中
Arguments	获取或设置启动应用程序时要使用的一组命令行参数
UseShellExecute	获取或设置一个 bool 值，该值指示是否使用操作系统外壳程序启动进程

- Process 对象

该对象提供对本地和远程进程的访问并使您能够启动和停止本地系统进程。在本程序中使用了该对象中的两个方法来启动和停止系统进程。这两个方法说明如下。

- Start 方法

该方法用来启动此对象的 StartInfo 属性指定的进程资源，并将其与该组件关联。

- WaitForExit 方法

该方法用来指示 Process 组件无限期地等待关联进程退出。

32.2.4 限制每台机器只能领取一个账号

在本程序中每台计算机只能领取一个试用账号，当用户没有领取过账号时单击“领取账号密码”按钮，将会显示一个试用的账号和密码信息，如图 32.11 所示。当用户领取过账号时单击“领取账号密码”按钮，将会弹出一个提示框提示用户已领取过账号不可以领取的信息，如图 32.12 所示。

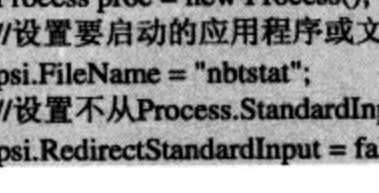


图 32.11 显示账号密码

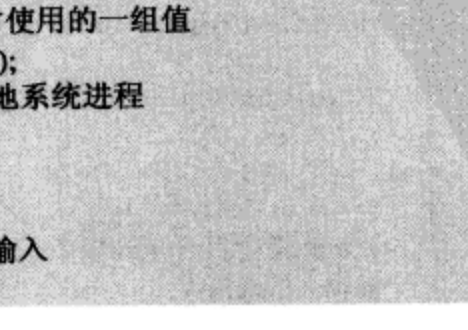


图 32.12 提示不可以领取账号

1. 前台页面设计

(1) 创建 1 个 Web 窗体，默认名为 Default.aspx。

(2) 在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 32.4 所示。

表 32.4 控件类型、控件名称及说明

控件类型	控件名称	主要属性	说明
标准/Button 控件	getNamePass	均为默认值	实现获取账号和密码操作
标准/Label 控件	labName	均为默认值	显示获取到的账号
	labPass	均为默认值	显示获取到的密码

2. 后台代码编写

在“领取账号密码”按钮的单击事件中用来实现获取账号和密码操作。在该事件中先获取到客户端的 MAC 地址，再判断 Cookie 中是否已存储过该地址，如果存储过将提示不可以再领取账号，如果未领取将显示账号和密码信息。实现代码如下。

例程 5 代码位置：光盘\mr\32\getMAC\Default.aspx.cs

```
protected void getNamePass_Click(object sender, EventArgs e)
    {
        //获取客户端的IP地址
         string IP = Request.UserHostAddress;
        //创建字符串变量
        string dirResults = "";
        //创建ProcessStartInfo对象表示启动进程时使用的一组值
        ProcessStartInfo psi = new ProcessStartInfo();
        //创建Process对象使您能够启动和停止本地系统进程
        Process proc = new Process();
        //设置要启动的应用程序或文档
        psi.FileName = "nbtstat";
        //设置不从Process.StandardInput流中读取输入
        psi.RedirectStandardInput = false;
```

```
                    //设置要输出写入 Process.StandardOutput流
                    psi.RedirectStandardOutput = true;
                    //设置启动的应用程序中的一组命令参数
                    psi.Arguments = "-A " + IP;
                    //设置从可执行文件创建进程
                    psi.UseShellExecute = false;
                    //设置启动进程
                    proc = Process.Start(psi);
                    //获取StandardOutput输出流
                    dirResults = proc.StandardOutput.ReadToEnd();
                    //设置Process 组件无限期地等待关联进程退出
                    proc.WaitForExit();
                    //替换掉StandardOutput输出流中的"/r,/n,/t"
                    dirResults = dirResults.Replace("\r", "").Replace("\n", "").Replace("\t", "");
                    //设置正则表达式
                    Regex reg = new Regex("MAC[ ]{0,}Address[ ]{0,}=[ ]{0,}(?<key>((.)*?))MAC", RegexOptions.Ignore
Case | RegexOptions.Compiled);
                    //向获取的StandardOutput输出流添加"MAC"字符串
                    dirResults = dirResults + "MAC";
                    //获取Cookie
                    HttpCookie oldCookie = Request.Cookies["netCard"];
                    //获取正则表达式中的匹配项
                    Match mc = reg.Match(dirResults);
                    //获取网卡号去除掉“-”符合
                    string networkCard = mc.Groups["key"].Value.Replace("-", "");
                    //判断Cookie是否为空
                    if (oldCookie == null)
                    {
                        //判断是否符合正则表达式的要求
                        if (mc.Success)
                        {
                            //显示账号
                            labName.Text = "您的账号为： " + networkCard;
                            //显示密码
                            labPass.Text = "您的密码为： 1234";
                            //创建Cookie对象
                            HttpCookie newCookie = new HttpCookie("netCard");
                            //设置Cookie的有效时间
                            newCookie.Expires = DateTime.MaxValue;
                            //添加Cookie中的值
                            newCookie.Values.Add("numberCard", networkCard);
                            //将Cookie添加到Cookie集合中
                            Response.Cookies.Add(newCookie);
                        }
                        else
                        {
                            RegisterStartupScript("", "<script>alert( '您没有联网！ ');</script>");
                        }
                    }
                    else
                    {
                        //获取Cookie中的网卡号
                        string numberCard = oldCookie.Values["numberCard"];
                        //判断Cookie中的网卡号是否和获取到的网卡号一致
                        if (numberCard.Trim() == networkCard.Trim())
                        {
                            RegisterStartupScript("", "<script>alert('很抱歉！ 您的计算机已领取过账号。')</script>");
                        }
                        else
                        {
                            //判断是否符合正则表达式的要求
                            if (mc.Success)
```

```
                {
                    //显示账号
                    labName.Text = "您的账号为：" + networkCard;
                    //显示密码
                    labPass.Text = "您的密码为：1234";
                    //修改Cookie中的值
                    oldCookie.Values.Set("numberCard", networkCard);
                    //将Cookie添加到Cookie集合中
                    Response.Cookies.Add(oldCookie);
                }
                else
                {
                    RegisterStartupScript("", "<script>alert( '您没有联网！ ');</script>");
                }
            }
        }
    }
```

32.3 处理 PDF 文档

32.3.1 PDF 文档简介

PDF（Portable Document Format）可翻译为可移植文件格式。PDF 文档是由 Adobe System 创立的。PDF 文档可以将文字、字型、格式、颜色及独立于设备和分辨率的图形图像等封装在一个文件中。该格式文件还可以包含超文本链接、声音和动态影像等电子信息，支持比较长的文件，集成度和安全可靠性都比较高。该文档常用于制作电子图书、产品说明、公司文告、网络资料等文件。

PDF 文档另一个特点就是 PDF 文档与操作系统平台无关，可以实现跨平台浏览，也就是说 PDF 文档不管是在 Windows、UNIX 还是在苹果公司的 Mac OS 操作系统中都可以使用。这一特点使它成为在 Internet 上进行电子文档发行和数字化信息传播的理想文档格式，很受大家欢迎。

32.3.2 配置 iTextSharp 组件

了解了 PDF 文档后，下面将讲解如何制作一个 PDF 格式化工具。制作 PDF 格式化工具需要借助第三方组件 iTextSharp 的帮助来完成。iTextSharp 组件是一个很强大的动态创建 PDF 的工具，是使用比较广泛的组件。要使用 iTextSharp 组件首先需要将其添加引用到项目中。添加引用 iTextSharp 组件的步骤如下。

（1）打开 VS2008 开发工具，新建一个网站，将其命名为 managePDF。

（2）在菜单栏中选择“网站”，在弹出的快捷菜单中选择“添加引用”菜单项，如图 32.13 所示。

（3）在弹出的“添加引用”窗口中，选择“浏览”选项卡并查找到所下载的 itextsharp.dll 文件，如图 32.14 所示。

（4）选择 itextsharp.dll 文件后单击确定按钮将返回到 VS2008 开发工具中。此时将会自动添加一个 Bin 文件夹，在该文件夹中存放着 itextsharp.dll 文件，如图 32.15 所示。

（5）添加完成后，想要使用该组件还需要添加该组件的命名空间。引用命名空间的代码如下：

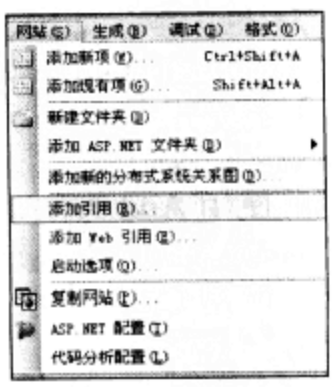


图 32.13 选择“添加引用”菜单项

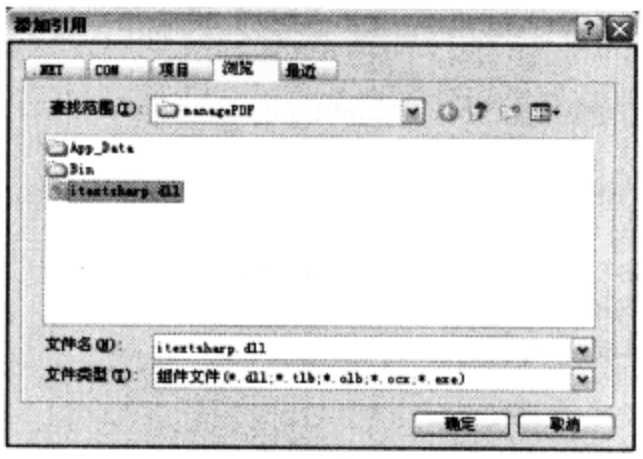


图 32.14 “浏览”选项卡

图 32.15 Bin 文件夹

```
using iTextSharp.text;
using iTextSharp.text.pdf;
```

32.3.3 制作简单的 PDF 格式化工具

本程序中制作的 PDF 格式化工具可以实现 PDF 文档页面大小设置功能、页边距设置功能、页眉和页脚设置等功能。本实例运行效果如图 32.16 所示。

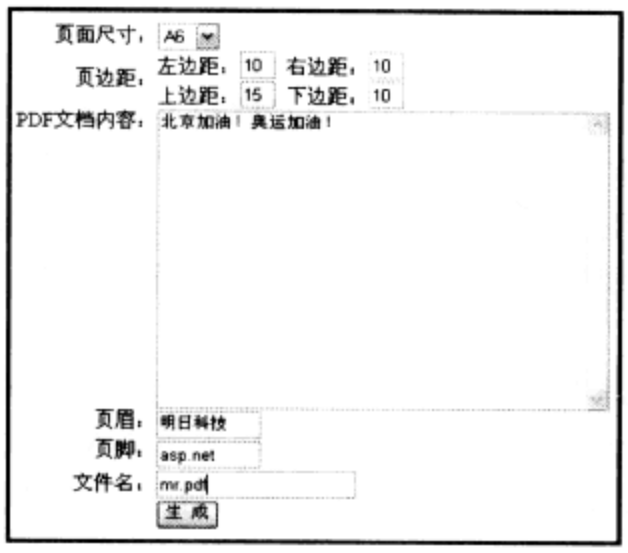


图 32.16 简单的 PDF 格式化工具

1. 关键技术

PDF 文档的创建是利用 iTextSharp 组件来完成的，创建一个简单的 PDF 文档可以使用以下 5 步来完成。

（1）创建 Document 对象的实例，Document 对象在 iTextSharp.text 对象下。代码如下：

```
Document document = new Document();
```

（2）创建 Document 对象的 Writer 实例，该实例的创建是使用 PdfWriter 对象中的 GetInstance 方法来实现的，使用该方法需要传入 2 个参数，第 1 个参数为 Documneg 对象，第 2 个参数为 Stream 对象。代码如下：

```
PdfWriter.GetInstance(document, new FileStream("C:\mr.pdf", FileMode.Create));
```

（3）打开 Document 对象，打开该对象需要使用该对象中的 Open 方法。代码如下：

```
document.Open();
```

（4）使用 Add 方法，向 Document 对象中添加内容。代码如下：

```
document.Add(new Paragraph("One World One Dream"));
```

（5）使用 Close 方法，关闭 Document 对象。代码如下：

```
document.Close();
```

通过以上 5 步的操作可以实现一个简单的 PDF 文档，该文档可以使用 Adobe Acrobat 软件或其他支持 PDF 文档的软件来打开。PDF 文档打开后的效果如图 32.17 所示。

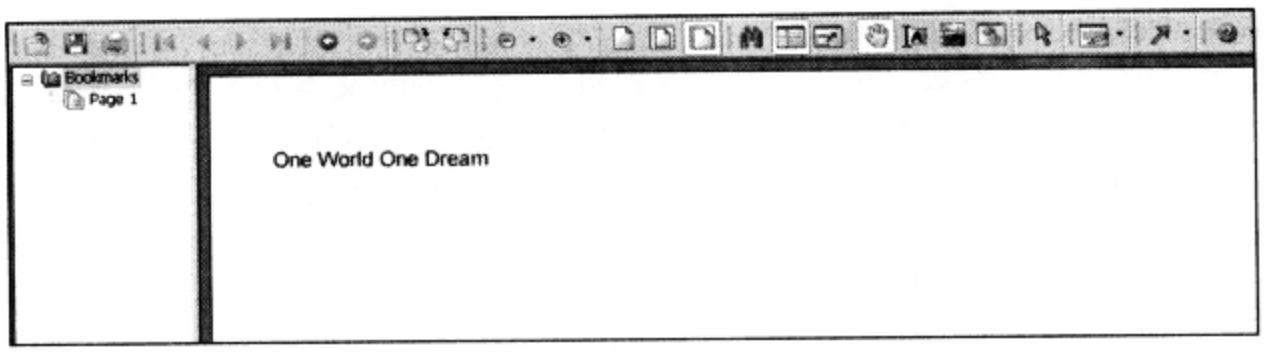


图 32.17　PDF 文档

通过上面的方法只能创建一个简单的文档，还不能设置页边距、页眉、页脚等操作。如果实现这些效果还需要使用 iTextSharp 组件提供的更多方法。首先介绍一下如何设置页面大小及页边距。设置页面的大小及页边距使用的是 Document 对象的构造函数。Document 对象有 3 个构造函数分别如下。

- Document()构造函数

该构造函数是以默认的页面大小及页边距来创建一个 Document 对象。

- Document(Rectangle pageSize)构造函数

该构造函数需要传入一个 Rectangle 对象，该对象可以创建纸张的大小、颜色等。该对象也提供了比较常用的页面大小。如从 A0 到 A10，从 B0 到 B10 等，用户可以在其中选择需要的页面大小。

- Document（Rectangle pageSize,int marginLeft,int marginRight,int marginTop,int marginBottom）构造函数

该构造函数需要 5 个参数，这 5 个参数的说明如下。

（1）Rectangle pageSize 对象：该对象可以选择页面的大小，如 A4 大小的页面。

（2）Int marginLeft 变量：该变量用来设置页面的左边距。

（3）int marginRight 变量：该变量用来设置页面的右边距。

（4）int marginTop 变量：该变量用来设置页面的上边距。

（5）int marginBottom 变量：该变量用来设置页面的下边距。

在使用 Document 对象创建 PDF 文档时，默认情况下是不支持中文的。如果想要使 Document 对象支持中文必须要设置一下文字的字体、大小及样式等。下面来介绍一下 BaseFont 对象及 Font 对象。

- BaseFont 对象

该对象用来存储文字的字体，也可以使用该对象中的 CreateFont 方法来指定某一个字体、文字的显示方向及是否可以嵌入字体。在系统中字体文件都存储在 C:\Windows\fonts 目录下。字体文件是以.ttf 为扩展名的。比较常用的字体有 simkai.ttf 楷体_GB2312、simhei.ttf 黑体等。如果使用 CreateFont 方法来设置某一个字体必须将字体的完整路径及名称全部给出，该方法还可以设置文字的显示方向，如 BaseFont.IDENTITY_V 属性用来设置文字纵向显示。设置字体的代码如下：

```
BaseFont bfChinese = BaseFont.CreateFont(@"c:\WINDOWS\fonts\simhei.ttf", BaseFont.IDENTITY_H,
BaseFont.NOT_EMBEDDED);
```

- Font 对象

在对象中用来设置文字所使用的字体及文字大小等参数。该对象有 10 种构造函数，这里就不逐个介绍了，只介绍一下带有两个参数的 Font 构造函数。第 1 个参数是 BaseFont 对象，用来设置文字的字体，第 2 个参数是 float 类型的变量，该变量表示字体的大小。下面使用刚刚创建的 BaseFont 对象构造 Font 对象并设置文字的大小为 12。实现代码如下：

```
Font fontChinese = new Font(bfChinese, 12);
```

下面介绍如何设置页眉和页脚，文档中的页眉和页脚常用来显示文档的页数或文档的名称等一些信息。在 PDF 文档中页眉和页脚的文本是使用 HeaderFooter 对象的构造函数来设置的。设置完文本后通过 document.Header 属性和 document.Footer 属性将文本显示出来。HeaderFooter 对象的构造函数说明如下。

```
HeaderFooter(Phrase before, bool numbered)
```

该构造函数中的第一参数 Phrase 用来设置页眉或页脚的文本及文本的字体，参数 bool numbered 用来设置是否显示页数，True 为显示页数，False 为不显示。下面使用 HeaderFooter 构造函数来设置页眉和页脚的文本并显示在 PDF 文档中。代码如下：

```
//设置页脚文本
HeaderFooter footer = new HeaderFooter(new Phrase("asp.net",fontChinese), false);
//显示页脚
document.Footer = footer;
//设置页眉文本
HeaderFooter header = new HeaderFooter(new Phrase("明日科技",fontChinese), false);
//显示页眉
document.Header = header;;
```

2. 前台页面设计

（1）创建 1 个 Web 窗体，将默认名改为 createPDF.aspx。

（2）在该窗体中添加控件，所添加的控件类型、控件名称及说明如表 32.5 所示。

表 32.5　　控件类型、控件名称及说明

控 件 类 型	控件名称	主 要 属 性	说　　明
标准/ TextBox 控件	txtLeft	均为默认值	输入页面的左边距
	txtRight	均为默认值	输入页面的右边距
	txtTop	均为默认值	输入页面的上边距
	txtBottom	均为默认值	输入页面的下边距
	txtContent	设置 TextMode 属性为 MultiLine	输入文档内容
	txtHeader	均为默认值	输入页眉内容
	txtFooter	均为默认值	输入页脚内容
	txtName	均为默认值	输入要保存的文件名包括扩展名 pdf
标准/ DropDownList 控件	ddlPs	均为默认值	用来选择页面大小
标准/ Button 控件	btnCreate	均为默认值	实现生成 PDF 文档操作
	btnOpen	均为默认值	实现打开 PDF 文档操作

3. 后台代码编写

在“生成”按钮的单击事件中，通过获取用户输入的页面大小、页边距等信息来设置 PDF 文档的相应参数，并将 PDF 文档保存到当前项目下。实现代码如下。

例程 6　代码位置：光盘\mr\32\managePDF\createPDF.aspx.cs

```
protected void btnCreate_Click(object sender, EventArgs e)
{
    //获取左边距
    int psLeft= Convert.ToInt32(txtLeft.Text);
    //获取右边距
    int psRight=Convert.ToInt32(txtRight.Text);
    //获取上边距
    int psTop=Convert.ToInt32(txtTop.Text);
    //获取下边距
    int psBottom=Convert.ToInt32(txtBottom.Text);
    //保存文档的路径
    pdfPath = txtName.Text;
    //创建documnet对象，并设置文档的页面大小，页边距等。
    Document document = new Document(getPs(ddlPs.SelectedValue),psLeft,psRight,psTop,psBottom);
    //创建BaseFont对象，用来设置字体
    BaseFont bfChinese = BaseFont.CreateFont(@"c:\WINDOWS\fonts\simhei.ttf", BaseFont.IDENTITY_H,
BaseFont.NOT_EMBEDDED);
    //设置文字的字体，大小，样式
```

```
        Font fontChinese = new Font(bfChinese, 12);
        //创建documnet对象的实例
        PdfWriter.GetInstance(document, new FileStream( Server.MapPath(".") +"\\"+ txtName.Text, FileMode.Create));
        //设置页眉
        HeaderFooter footer = new HeaderFooter(new Phrase(txtFooter.Text,fontChinese), false);
        document.Footer = footer;
        //设置页脚
        HeaderFooter header = new HeaderFooter(new Phrase(txtHeader.Text,fontChinese), false);
        document.Header = header;
        //打开文档
        document.Open();
        //在文档中添加文本内容
        document.Add(new Paragraph(txtContent.Text, fontChinese));
        //关闭文档对象
        document.Close();
    }
```

自定义 getPs 方法根据传入的参数，查询出指定的 PageSize 对象中的属性，并将该属性返回。实现代码如下。

例程 7 代码位置：光盘\mr\32\managePDF\createPDF.aspx.cs

```
protected Rectangle getPs(string str)
    {
        //创建Rectangle对象
        Rectangle ps=null;
        //获取使用的纸张
        switch (str)
        {
            case "A4": { ps = PageSize.A4; break; }
            case "A5": { ps = PageSize.A5; break; }
            case "A6": { ps = PageSize.A6; break; }
            case "B5": { ps = PageSize.B5; break; }
        }
        return ps;
    }
```

32.4 OWC 生成图表

32.4.1 OWC 简介

在开发 Web 程序时，常常会使用图形来表示数据，例如使用柱型图或饼型图来表示公司的季度营业额等。使用图形来表示数据会使数据更加鲜明。在 ASP.NET 中可以使用 GDI+来绘制图形，但使用 GDI+来绘制图形会比较麻烦。为了节省开发时间常常会使用绘图组件来绘制图形，如 OWC 组件。

OWC（Office Web Components）是微软公司随 Office 提供的绘图组件，使用它能够绘制绝大部分的图形。OWC 是随着 Microsoft Office 2003 软件一起安装的一组 ActiveX 组件。使用 OWC 组件就必须要保证本机上已安装了 Microsoft Office 2003 软件。OWC 组件可以很方便地在浏览器中或在传统的编程环境中进行数据分析，例如电子表格、图表、数据透视表等。

32.4.2 添加 OWC 组件

本节将介绍如何在 ASP.NET 中添加该 OWC 组件的引用。添加该组件的步骤如下。

（1）打开 VS2008 开发工具，新建一个网站将其命名为 createShart，将页面默认名 Default.aspx 修改为 createColumn.aspx。

（2）在菜单栏中选择“网站”选项，在弹出的快捷菜单中选择“添加引用”选择项，如

图 32.18 所示。

(3) 在弹出的“添加引用”窗口中选择“COM”选项卡，在该选项卡中选中“Microsoft Office Web Components 11.0”选项，如图 32.19 所示。

图 32.18 选择“添加引用”选择项

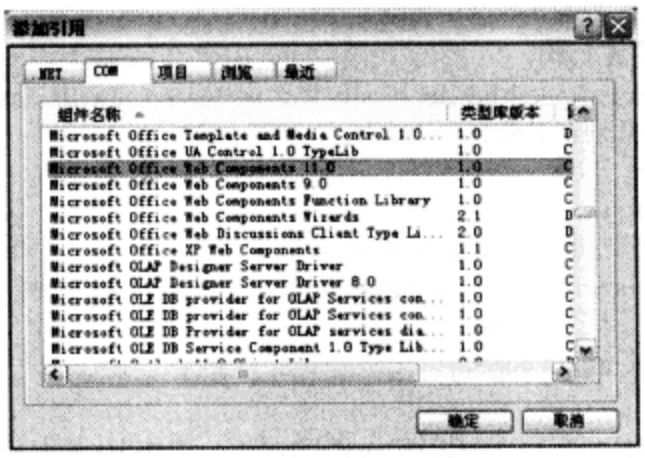


图 32.19 “添加引用”窗口

(4) 单击“确定”按钮，将会返回到 createColumn.aspx 页面中完成 OWC 组件的添加。要使用 OWC 组件还需要在该页面中添加引用 Microsoft.Office.Interop.Owc11 命名空间，代码如下：

```
using Microsoft.Office.Interop.Owc11;
```

32.4.3 OWC 绘制图形的关键技术

OWC 组件可以绘制出大部分常用的图形，包括柱型图、饼型图、折线图等。使用 OWC 绘制图形首先应该创建两个对象，这两个对象为 ChartSpace 对象和 ChChart 对象。下面介绍一下这两个对象。

- ChartSpace 对象

该对象表示图表工作区。图表工作区为图表容器的顶层，可以包含多个图表，每个图表以 ChChart 对象表示。在新建图表工作区时，工作区为空（不包含任何图表）。

- ChChart 对象

该对象表示图表工作区中的单个图表。图表工作区最多可以包含 16 个图表。ChChart 对象为 ChCharts 集合的成员

创建完 ChartSpace 对象和 ChChart 对象后需要设置图表对象类型，也就是说设置图表的显示样式，设置图表对象类型是使用 ChChart 对象中的 Type 属性来完成的。该对象的说明如下。

Type 属性：该属性用来设置或获取图表对象的类型。在该属性中可以设置常见的图表，包括柱型图、饼型图等。该属性的常用参数及说明如表 32.6 所示。

表 32.6 常用参数及说明

参　数	说　明
chChartTypeArea	该参数用来设置区域图形
chChartTypeArea3D	该参数用来设置 3D 效果的区域图形
chChartTypeBarClustered	该参数用来设置条形图
chChartTypeBarClustered3D	该参数用来设置 3D 效果的条形图
chChartTypeColumnClustered	该参数用来设置柱型图
chChartTypeColumnClustered3D	该参数用来设置 3D 效果的柱型图
chChartTypePie	该参数用来设置饼型图
chChartTypePie3D	该参数用来设置 3D 效果的饼型图
chChartTypeRadarLine	该参数用来设置雷达线型图

绘制图形需要使用 ChChart 对象中 ChSeriesCollection 集合的 SetData 方法。该方法用来设置指定图表对象的数据。使用该方法需要传入 3 个参数。参数说明如下。

- Dimension 参数

该参数为 ChartDimensionsEnum 常量，是必需参数。用来指定设置的数据维。ChartDimensions Enum 常量如表 32.7 所示。

表 32.7　　ChartDimensionsEnum 常量

常　量	说　明
chDimBubbleValues	设置气泡图上的标志值
chDimCategories	设置用作分类的值
chDimCharts	在 HasMultipleCharts 属性设置为 True 时设置新图表的来源字段
chDimCloseValues	设置股票图的收盘值
chDimFilter	设置放置于筛选坐标轴的字段
chDimFormatValues	设置格式映射中的值
chDimHighValues	设置股票图的盘高值
chDimLowValues	设置股票图的盘低值
chDimOpenValues	设置股票图的开盘值
chDimRValues	设置极坐标图的 R 值
chDimSeriesNames	设置作为系列名称的值
chDimThetaValues	设置极坐标图的极角值
chDimValues	设置绘制图表的值
chDimXValues	设置 XY 散点图或气泡图的 X 值
chDimYValues	设置 XY 散点图或气泡图的 Y 值

- DataSourceIndex 参数

该参数值为 Long 类型的 ChartSpecialDataSourcesEnum 常量，该参数是必需的。ChartSpecial DataSourcesEnum 常量如表 32.8 所示。

表 32.8　　ChartSpecialDataSourcesEnum 常量

参　数	说　明
chDataBound	将指定对象绑定到 DataReference 参数指定的外部数据源
chDataLinked	将指定对象绑定到其他维。在 Dimension 参数中指定 chDimFormatValues 创建格式映射时使用该值
chDataLiteral	将指定对象绑定到 DataReference 参数指定的文字类型数据
chDataNone	清除指定对象

- DataReference 参数

该参数是 Microsoft Excel 样式的区域引用（例如，“A1:D4”）或行集的列名指定数据引用。如果 DataSourceIndex 参数设置为 chDataLiteral，可以将 DataReference 设置为一维数组或逗号分隔列表。对于 ChErrorBars 对象，该参数指定可用作误差线数值的 Double 或 String 数组。

创建完图形后还需要将图形保存起来，例如将图形保存为 GIF 格式的图片。保存图形是使用 ChartSpace 对象中的 ExportPicture 方法来实现的，调用该方法需要传入 4 个参数。这 4 个参数的说明如下。

- FileName 参数

该参数为 String 类型，可选。指定保存的文件名。如果不指定该参数，对于图表工作区默认文件名为 Chart.gif，对于数据透视表列表为 Pivot.gif。

- FilterName 参数

该参数为 String 类型，可选。指定所用的图形过滤器名称。支持的文件名为 GIF、JPG 和 PNG。默认值为 GIF。

Width 参数

该参数为 Long 类型，可选。以像素为单位指定图像的宽度。必须为服务器中的图表指定该参数。

- Height 参数

该参数为 Long 类型，可选。以像素为单位指定图像的高度。必须为服务器中的图表指定该参数。

32.4.4 绘制 3D 柱型图

在本程序中绘制的 3D 柱型图用来表示某公司的全年季度营业额。在图形中分别以 4 个 3D 柱型图来表示各季度的营业额，如图 32.20 所示。

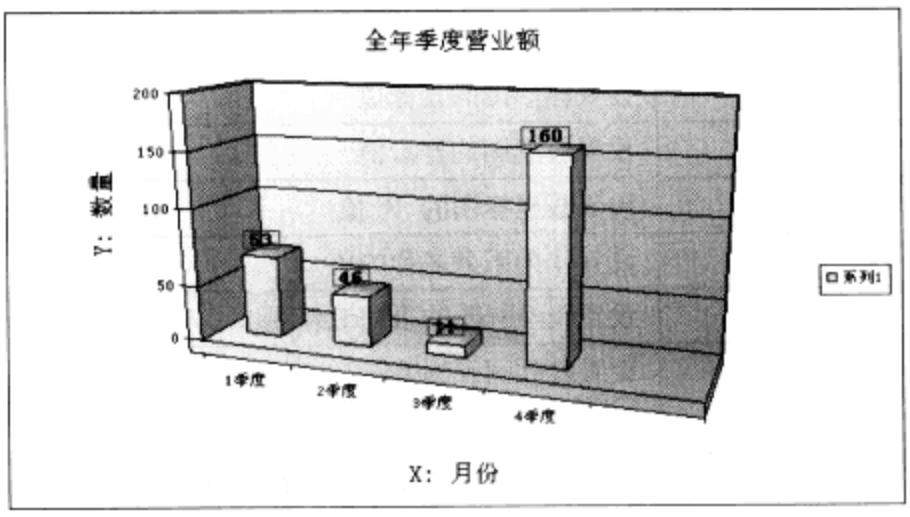


图 32.20　3D 柱型图

1. 前台页面设计

（1）打开在前面所创建的 createColumn.aspx 页面。

（2）在该页面中添加 1 个 PlaceHolder 控件，用来显示 3D 柱型图。

2. 后台代码编写

在页面的加载事件中实现 3D 柱型图的绘制，在该事件中首先将创建一个字符串数组，在该数组中保存 4 个季度的营业额。再创建图表工作区，并向图表工作去中添加一个图表，设置图表的类型为 3D 柱型图。接下来要设置图表中的标题，*x* 轴标题和 *y* 轴标题，并向图表中的柱型图添加数据。最后将图表保存到指定目录下并显示在页面中。实现代码如下。

例程 8　代码位置：光盘\mr\32\createShart\createColumn.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        //创建一个字符串数组，保存各季度名称
        string[] monNum = new string[4];
        //创建一个字符串数组，保存各季度营业额
        string[] monCount = new string[4];
        //创建生成随机数对象
        Random rd = new Random();
        //使用for循环赋值
        for (int i = 0; i < 4; i++)
        {
            monNum[i] = Convert.ToString(i + 1);
            monCount[i] = (rd.Next(200)).ToString();
        }
```

```
//创建一个字符串变量，保存各季度名称
string strXdata = string.Empty;
foreach (string strData in monNum)
{
    strXdata += strData + "季度\t";
}
//创建一个字符串变量，保存各季度营业额
string strYdata = string.Empty;
foreach (string strValue in monCount)
{
    strYdata += strValue + "\t";
}
//创建图表工作区
ChartSpace laySpace = new ChartSpaceClass();
//在图表工作区中添加一个图表
ChChart InsertChart = laySpace.Charts.Add(0);
//设置图表类型为3d柱型图
InsertChart.Type = ChartChartTypeEnum.chChartTypeColumnClustered3D;
//设置图表是否具有图例
InsertChart.HasLegend = true;
//设置图表是否具有标题
InsertChart.HasTitle = true;
InsertChart.Title.Caption = "全年季度营业额";
//设置x坐标是否具有标题
InsertChart.Axes[0].HasTitle = true;
//设置x坐标标题内容
InsertChart.Axes[0].Title.Caption = "X: 月份";
//设置y坐标是否具有标题
InsertChart.Axes[1].HasTitle = true;
//设置最小的数轴值
InsertChart.Axes[1].Scaling.SplitMinimum = 50;
//设置y坐标标题内容
InsertChart.Axes[1].Title.Caption = "Y: 数量";
//指定新图表的位置
InsertChart.SeriesCollection.Add(0);
//设置分类的值
InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories, +(int)ChartSpecialData
SourcesEnum.chDataLiteral, strXdata);
//设置图表的值
InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues, +(int)ChartSpecialDataSources
Enum.chDataLiteral,strYdata) ;
//设置图表颜色
InsertChart.SeriesCollection[0].Interior.Color = "#99E6FC";
//显示图表中所有标志
ChDataLabels dls = InsertChart.SeriesCollection[0].DataLabelsCollection.Add();
//设置标志的大小
dls.Font.Size = 10;
//设置标志边框颜色
dls.Border.Color = "red";
//设置标志为粗体
dls.Font.Bold = true;
//字符串变量用来保存生成图片的路径
string strAbsolutePath = Server.MapPath(".") + "\\sq.gif";
//将图表保存为图片文件
laySpace.ExportPicture(strAbsolutePath, "GIF", 600, 350);
//创建图像标记
string strImageTag = "<IMG Src='" + strAbsolutePath + "'/>";
//将图像显示在页面中
this.PlaceHolder1.Controls.Add(new LiteralControl(strImageTag));
}
```

32.4.5 绘制 3D 饼型图

在本程序中绘制的 3D 饼型图用来表示某公司的全年季度营业额。在图形中分别以 4 个扇形区域来表示各季度的营业额，如图 32.21 所示。

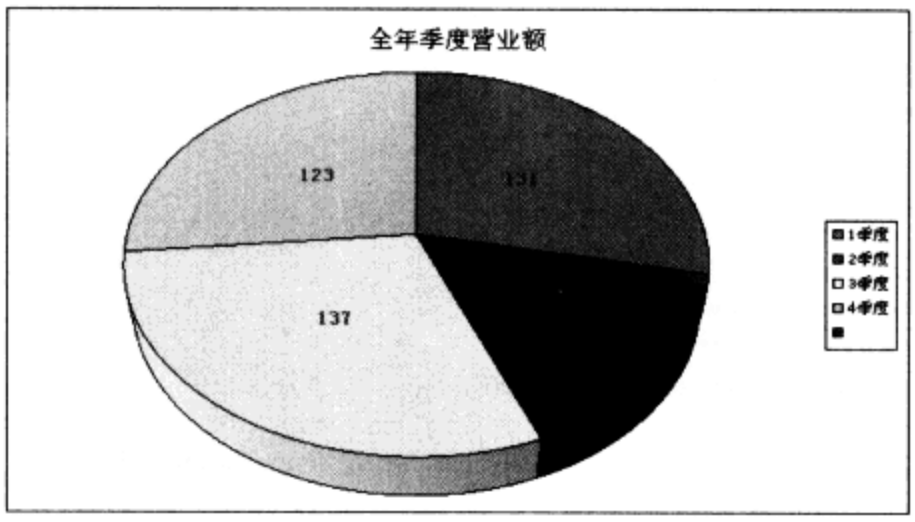


图 32.21 3D 饼型图

1. 前台页面设计

(1) 创建一个 Web 窗体，将其命名为 createPieChart.aspx。

(2) 在该页面中添加 1 个 PlaceHolder 控件，用来显示 3D 饼型图。

2. 后台代码编写

在页面的加载事件中用来实现 3D 饼型图的绘制，在该事件中首先将创建一个字符串数组，在该数组中保存 4 个季度的营业额。再创建图表工作区，并向图表工作去中添加一个图表，设置图表的类型为 3D 饼型图。接下来设置图表中的标题等信息，并向图表中的饼型图添加数据。最后将图表保存到指定目录下并显示在页面中。实现代码如下。

例程 9 代码位置：光盘\mr\32\createShart\createPieChart.aspx。

```
protected void Page_Load(object sender, EventArgs e)
    {
        //创建一个字符串数组，保存各季度名称
        string[] monNum = new string[4];
        //创建一个字符串数组，保存各季度营业额
        string[] monCount = new string[4];
        //创建生成随机数对象
        Random rd = new Random();
        //使用for循环赋值
        for (int i = 0; i < 4; i++)
        {
            monNum[i] = Convert.ToString(i + 1);
            monCount[i] = (rd.Next(200)).ToString();
        }
        //创建一个字符串变量，保存各季度名称
        string strXdata = string.Empty;
        foreach (string strData in monNum)
        {
            strXdata += strData + "季度\t";
        }
        //创建一个字符串变量，保存各季度营业额
        string strYdata = string.Empty;
        foreach (string strValue in monCount)
        {
            strYdata += strValue + "\t";
        }
        //创建图表工作区
```

```
        ChartSpace laySpace = new ChartSpaceClass();
        //在图表工作区中添加一个图表
        ChChart InsertChart = laySpace.Charts.Add(0);
        //设置图表类型为3d饼型图
        InsertChart.Type = ChartChartTypeEnum.chChartTypePie3D;
        //设置图表是否具有图例
        InsertChart.HasLegend = true;
        //设置图表是否具有标题
        InsertChart.HasTitle = true;
        //设置标题内容
        InsertChart.Title.Caption = "全年季度营业额";
        //指定新图表的位置
        InsertChart.SeriesCollection.Add(0);
        //设置分类的值
        InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimCategories, (int)ChartSpecial
DataSourcesEnum.chDataLiteral, strXdata);
        //设置图表的值
        InsertChart.SeriesCollection[0].SetData(ChartDimensionsEnum.chDimValues, (int)ChartSpecialDataSources
Enum.chDataLiteral, strYdata);
        //显示图表中所有标志
        ChDataLabels dls = InsertChart.SeriesCollection[0].DataLabelsCollection.Add();
        //设置标志大小
        dls.Font.Size = 10;
        //设置标志颜色
        dls.Font.Color = "red";
        //设置标志为粗体
        dls.Font.Bold = true;
        //字符串变量用来保存生成图片的路径
        string strAbsolutePath = Server.MapPath(".") + "\\sq.gif";
        //将图表保存为图片文件
        laySpace.ExportPicture(strAbsolutePath, "GIF", 600, 350);
        //创建图像标记
        string strImageTag = "<IMG Src='" + strAbsolutePath + "'/>";
        //将图像显示在页面中
        this.PlaceHolder1.Controls.Add(new LiteralControl(strImageTag));
    }
```

[General Information]
书名=ASP.NET开发典型模块大全　　修订版_12536795
SS号=11665398