第 9 章

物流信息管理平台

（ASP.NET 2.0+SQL Server 2000 实现）

随着国内信息化步伐的加快，加上物流企业对行业信息的需求越来越大，促使物流信息平台迅速发展，以保证物流信息平台信息的及时性、准确性，在最大程度上满足国内物流企业对行业信息的要求，以适应物流行业的市场变化，使之成为国内物流企业信息的主要来源。本章主要介绍如何利用 ASP.NET 2.0 +SQL Server 2000 快速开发一个对物流信息进行发布和管理的操作平台。通过学习本章，你将学到：

- 物流信息管理平台开发的基本过程
- 如何设计公共类
- 发布信息模块的实现方法
- 管理信息模块的实现方法
- SQL Server 2000 数据库在物流信息管理平台中的应用
- 面向对象的开发思想
- 分层开发模式
- 第三方控件在网站开发中的应用
- 电子邮件的发送

9.1 开发背景

随着经济全球化进程的加快，现代企业的专业分工和协作，对现代物流提出了越来越高的要求，信息化、自动化、网络化、智能化已成为现代物流的鲜明特征。随着物流行业的发展壮大，物流的信息化日益被从业者和信息系统提供商所重视。同时，现代企业的供应链也时刻提醒我们，若想在激烈的市场竞争中占据绝对优势，企业必须及时、准确地掌握客户的需求，同时对客户的需求作出快速的反应，在最短的时间内以最大限度挖掘和优化物流资源来满足客户的需求，从而建立高效的数字化物流经济。

9.2 需求分析

随着物流业在我国的蓬勃发展及物流市场的激烈竞争，现代物流信息逐步从定性转变为更精确的定量要求，这就需要物流信息管理平台提供大量准确、及时的信息数据，以帮助企业了解市场的变化以调整企业发展策略，所以物流信息管理平台最基本的功能就是保证浏览者查看到准确的信息、最新的信息。

9.3 系统设计

9.3.1 系统目标

物流信息管理平台是针对中小型物流企业设计的。主要实现如下目标：

- ☑ 操作简单方便、界面简洁美观。
- ☑ 网站整体结构和操作流程合理顺畅，实现人性化设计。
- ☑ 注册功能。提供两种注册途径：一种是个人用户注册，一种是企业用户注册。
- ☑ 货源信息的发布和浏览功能。
- ☑ 车源信息的发布和浏览功能。
- ☑ 专线信息的发布和浏览功能。
- ☑ 仓储信息的发布和浏览功能。
- ☑ 招聘信息的发布和浏览功能。
- ☑ 管理网站会员信息。
- ☑ 系统最大限度地实现易安装性、易维护性和易操作性。
- ☑ 系统运行稳定、安全可靠。

9.3.2　系统业务流程图

物流信息管理平台业务流程图如图 9.1 所示。

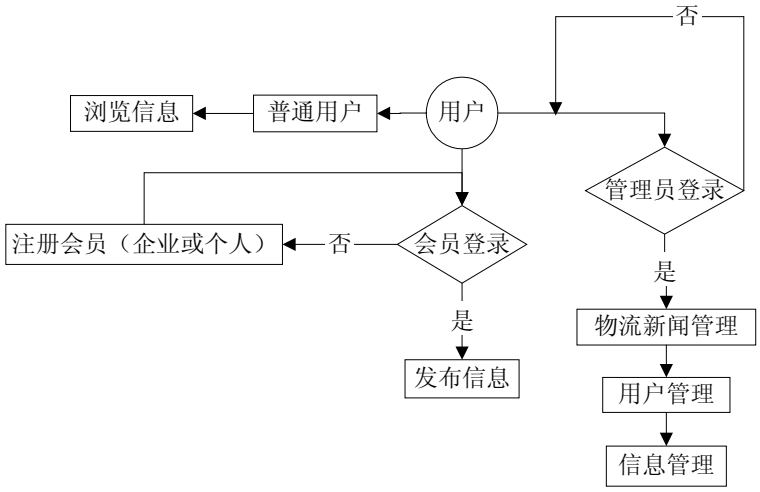

图 9.1　业务流程图

9.3.3　系统功能结构

根据物流信息管理平台的特点，可以将其分为前台和后台两个部分设计。前台主要实现功能为浏览信息（浏览信息包括货源信息、车源信息、招聘信息、企业信息、专线信息、仓储信息）、发布信息（个人用户发布信息、企业用户发布信息）、查询功能、注册功能（个人用户注册、企业用户注册）。后台主要实现功能为物流新闻管理（发布新闻、管理新闻）、信息管理（车源信息管理、货源信息管理、专线信息管理、招聘信息管理、仓储信息管理）、用户管理（个人用户管理、企业用户管理）。物流信息管理平台的前台功能结构如图 9.2 所示。

物流信息管理平台的后台功能结构如图 9.3 所示。

9.3.4　系统预览

物流信息管理平台由多个页面组成，下面仅列出几个典型页面，其他页面参见光盘中的源程序。

系统首页如图 9.4 所示，主要实现显示导航、最新物流信息、物流新闻、物流招聘和登录及搜索功能。发布信息页面如图 9.5 所示，主要实现企业或个人用户发布物流信息功能。

企业用户管理页面如图 9.6 所示，主要实现删除企业用户、锁定企业用户、查看企业用户详细信息。货源信息管理页面如图 9.7 所示，主要实现查看货源详细信息、删除货源信息、审核货源信息。

说明：由于路径太长，因此省略了部分路径，图 9.4 和图 9.5 省略的路径为“TM\09\WuLiu”，图 9.6 和图 9.7 省略的路径为“TM\09\WuLiu\manage”。

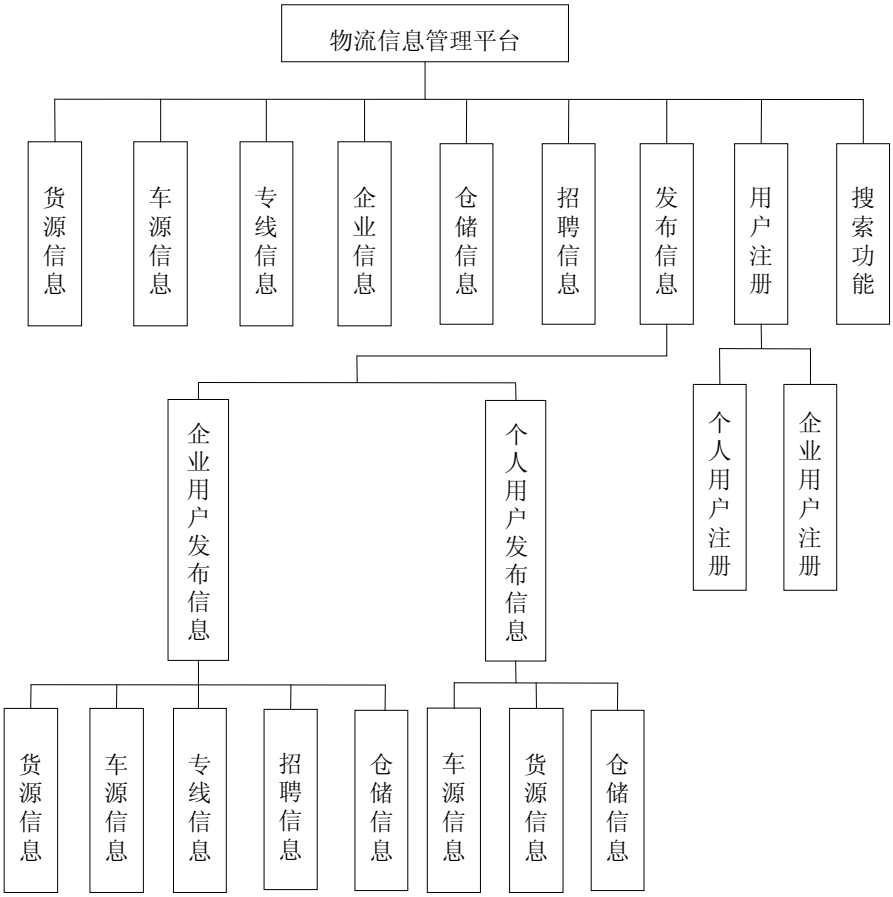

图 9.2　前台系统功能结构图

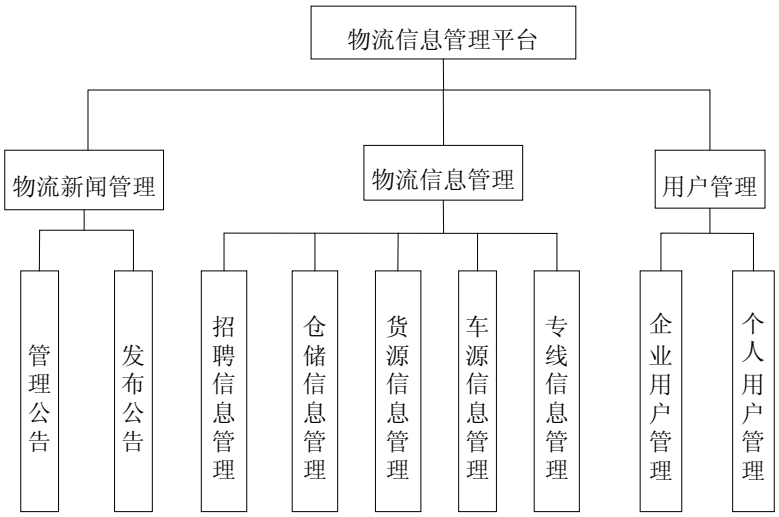

图 9.3　后台系统功能结构图

图 9.4　首页（光盘\…\index.aspx）

图 9.5　发布信息页（光盘\…\issuanceFreight.aspx）

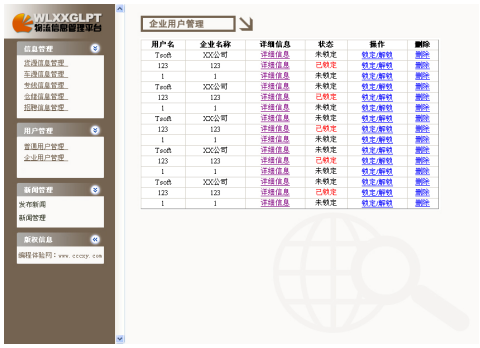

图 9.6　企业用户管理（光盘\…\manage_qyUser.aspx）

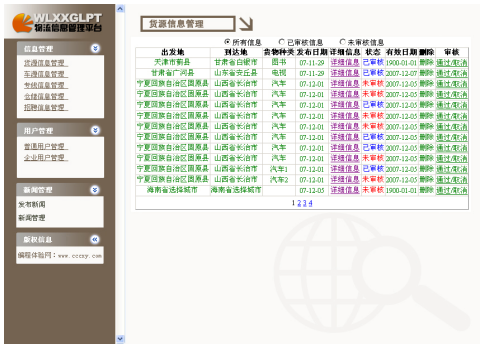

图 9.7　货源信息管理（光盘\…\manage_Freight.aspx）

说明：由于路径太长，因此省略了部分路径，图 9.4 和图 9.5 省略的路径为：TM\09\WuLiu。图 9.6 和图 9.7 省略的路径为：TM\06\WuLiu\Manage。

9.3.5　构建开发环境

1．网站开发环境

网站开发环境：Microsoft Visual Studio 2005 集成开发环境。
网站开发语言：ASP.NET+C#。
网站后台数据库：SQL Server 2000。
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 Server 2003（SP1）。

Web 服务器：Internet 信息服务（IIS）管理器。
数据库服务器：SQL Server 2000。
浏览器：IE 6.0。
网站服务器运行环境：Microsoft .NET Framework SDK v2.0。

3．客户端

浏览器：Internet Explorer 6.0。
分辨率：最佳效果 1024×768 像素。

9.3.6 数据库设计

本网站采用 SQL Server 2000 作为后台数据库，数据库名称为 db_WL，其中包含 10 个数据表，下面将分别介绍。

1．数据库概要说明

为了使读者对本程序系统后台数据库中的数据表有一个更清晰的认识，在此给出了数据库的结构图，该结构图包括系统所有的数据表，如图 9.8 所示。

图 9.8　数据库结构

2．数据库 E-R 图分析

物流信息化的一个重要步骤就是建立稳固的物流信息平台，通过物流信息平台了解到及时、有效的物流信息。因此，对物流信息平台的合理化设计尤为重要，而建立物流信息平台的一个关键问题是数据库的设计。

通过对网站进行的需求分析、网站流程设计以及系统功能结构的确定，规划出系统中使用的数据库实体对象分别为“物流新闻”、“货源信息”、“仓储信息”、“企业用户”、“招聘信息”、“搜索功能”，实体 E-R 图如下所示。

物流新闻为浏览者提供物流行业的最新动态，物流新闻实体 E-R 图如图 9.9 所示。

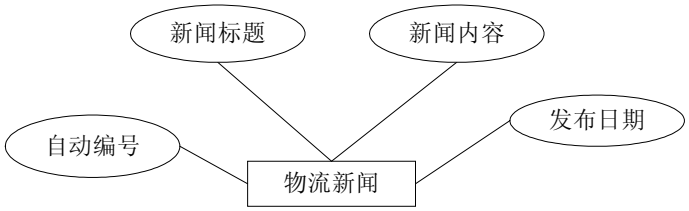

图 9.9　物流新闻实体 E-R 图

浏览者通过货源信息可以了解到用户需要运送货物的详细信息。货源信息实体 E-R 图如图 9.10 所示。

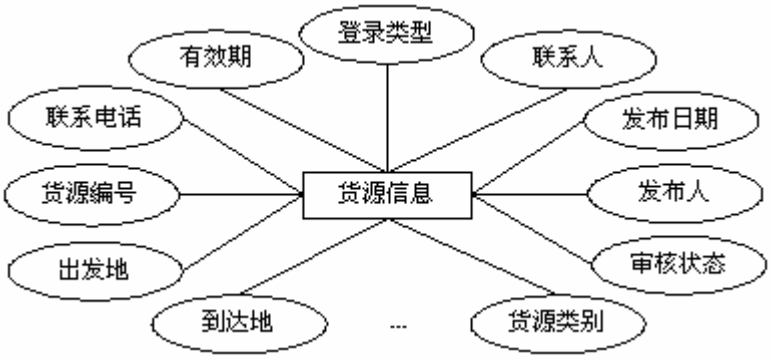

图 9.10　货源信息实体 E-R 图

浏览者可以通过仓储信息了解到某地出租的仓库信息，仓储信息实体 E-R 图如图 9.11 所示。

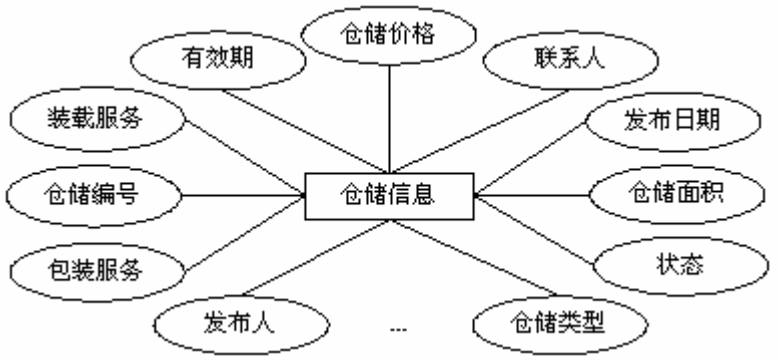

图 9.11　仓储信息实体 E-R 图

企业用户在注册时需要提供企业的详细信息，并提供给浏览者来增强企业的信誉度。企业用户实体 E-R 图如图 9.12 所示。

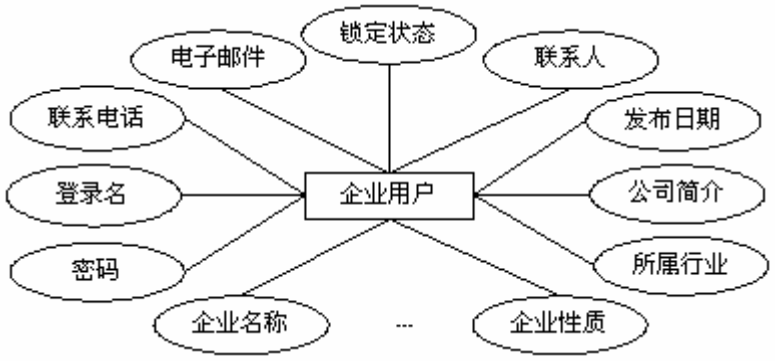

图 9.12　企业用户实体 E-R 图

企业用户可以通过发布招聘信息为本企业招贤纳士，浏览者可以通过招聘信息寻求到符合自身条

件的工作信息。招聘信息实体 E-R 图如图 9.13 所示。

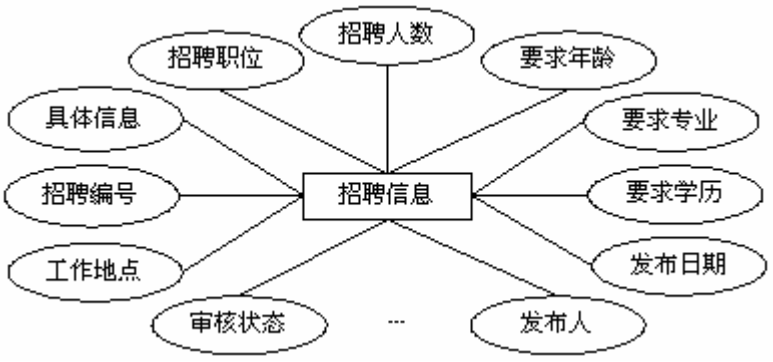

图 9.13　招聘信息实体 E-R 图

搜索功能可以使浏览者快速有效地查找到需要的信息，搜索功能实体 E-R 图如图 9.14 所示。

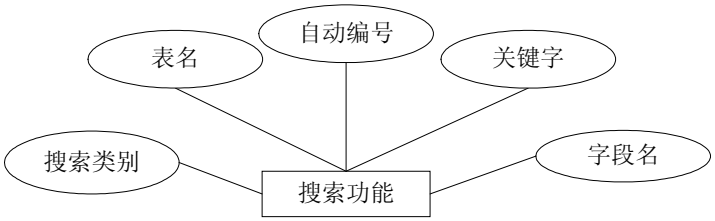

图 9.14　搜索功能实体 E-R 图

3．数据表结构

数据库实体 E-R 图设计完毕之后，就要根据实体 E-R 图设计数据表结构。下面将主要的数据表的数据结构和用途分别列出来，其他数据表参见本书附带光盘。

☑　tb_Depot（仓储信息表）

仓储信息表主要存储仓储详细信息，tb_Depot 表的结构如图 9.15 所示。

列名	数据类型	默认值	允许空	描述
ID	int			仓储编号
UserName	varchar			发布人
DepotType	varchar			仓储类型
DepotCity	varchar			仓储所在城市
DepotSite	varchar		✓	仓储所在地点
DepotAcreage	int			仓储面积
DepotSum	int		✓	仓储间数
DepotPrice	int		✓	仓储价格
Loading	int		✓	装载服务
Packing	int		✓	包装服务
Send	int		✓	陪送服务
Linkman	varchar			联系人
Phone	varchar			联系电话
Term	datetime			有效日期
Content	varchar		✓	详细信息
FBDate	datetime		✓	发布日期
userType	char			登录类型
Auditing	bit	(0)		审核状态

图 9.15　tb_Depot 表的结构

☑　tb_Freight（货源信息表）

货源信息表主要存储货源的详细信息，tb_Freight 表的结构如图 9.16 所示。

☑　tb_User（企业用户表）

企业用户表主要存储企业用户的详细信息，tb_User 表的结构如图 9.17 所示。

列名	数据类型	默认值	允许空	描述
ID	int			货源编号
UserName	varchar			发布人
Start	varchar			出发地
Terminal	varchar			到达地
FreightType	varchar			货源类别
FreightWeight	int		✓	货源重量
WeightUnit	char		✓	重量单位
Linkman	varchar			联系人
Phone	varchar			联系电话
Term	datetime			有效日期
Content	varchar		✓	备注
FBDate	datetime		✓	发布日期
userType	char			登录类型
Auditing	bit	(0)		审核状态

图 9.16　tb_Freight 表的结构

列名	数据类型	默认值	允许空	描述
ID	int			用户编号
UserName	varchar			登录名
UserPass	varchar			密码
PassQuestion	varchar			密码提示问题
PassSolution	varchar		✓	密码提示答案
Linkman	varchar		✓	联系人
CompanyName	varchar		✓	企业名称
Kind	varchar		✓	企业性质
Calling	varchar		✓	所属行业
LicenceNumber	varchar			营业执照号
Address	varchar		✓	地址
Phone	varchar		✓	联系电话
Fax	varchar		✓	传真
Email	varchar			电子邮件
NetworkIP	varchar		✓	网址
Content	varchar		✓	企业简介
Lock	bit	(0)		锁定状态
LockCause	varchar		✓	锁定原因

图 9.17　tb_User 表的结构

☑　tb_Job（招聘信息表）

招聘信息表主要存储招聘的详细信息，tb_Job 表的结构如图 9.18 所示。

☑　tb_News（物流新闻表）

物流新闻表主要存储物流新闻信息，tb_News 表的结构如图 9.19 所示。

列名	数据类型	默认值	允许空	描述
JobID	int			招聘编号
Job	varchar			招聘职位
Number	char		✓	招聘人数
Sex	char		✓	要求性别
Age	char		✓	要求年龄
Knowledge	varchar		✓	要求学历
Specialty	varchar		✓	要求专业
Experience	varchar		✓	工作经验
City	varchar		✓	工作地点
Pay	char		✓	月薪
ParticularInfo	varchar		✓	具体信息
FBDate	datetime			发布日期
UserName	varchar			发布人
Auditing	bit			审核状态

图 9.18　tb_Job 表的结构

列名	数据类型	默认值	允许空	描述
ID	int			新闻编号
NewsTitle	varchar			新闻标题
NewsContent	varchar			新闻内容
FBDate	datetime			发布日期

图 9.19　tb_News 表的结构

☑　tb_Search（搜索功能表）

搜索功能表主要存储各表名和字段名，tb_Search 表的结构如图 9.20 所示。

列名	数据类型	默认值	允许空	描述
ID	int			自动编号
searchType	varchar		✓	搜索类别
type	varchar		✓	表名
searchKey	varchar		✓	关键字
keyword	varchar		✓	字段名

图 9.20　tb_Search 表的结构

9.3.7 文件夹组织结构

为了便于读者对本网站的学习，在此笔者将网站文件的组织结构展示出来。前台文件组织结构如图 9.21 所示。

后台文件组织结构如图 9.22 所示。

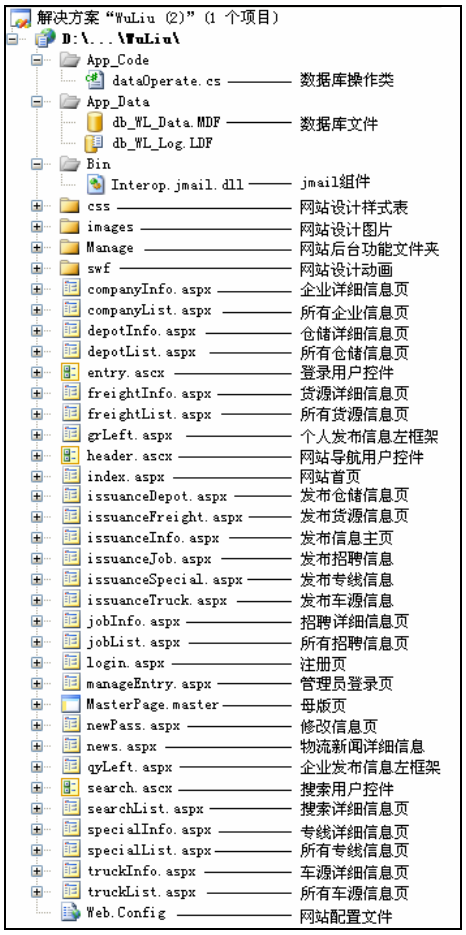

图 9.21 前台文件组织结构图

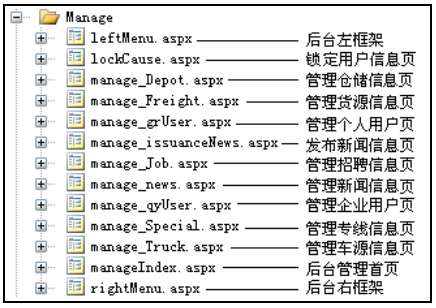

图 9.22 后台管理文件组织结构图

9.4 公共类设计

数据库操作类设计

数据库操作类用来完成数据库的连接操作以及数据库的查询、添加、删除和修改操作。将这几种操作编写到一个公共类里，可以减少重复代码的编写，有利于代码的维护。在 dataOperate 类中一共定

义了 5 个方法，下面分别对这几个方法进行讲解。

☑　createCon 方法

createCon 方法返回的类型为 SqlConnection，主要用来构造数据库的连接。代码如下：

例程01　代码位置：光盘\TM\09\WuLiu\App_Code\ dataOperate.cs

```
public static SqlConnection createCon()
    {
        //生成SqlConnection的一个对象用于连接数据库
        SqlConnection con = new SqlConnection("server=.;database=db_WL;uid=sa;pwd=;");
        return con;
    }
```

☑　execSQL 方法

execSQL(string sql)方法用来添加、插入和删除数据。此方法返回一个布尔值用来表示添加、插入和删除数据是否成功，执行成功返回 true，否则返回 false。调用此方法时应传入一个 string 类型的参数，此参数表示所要执行的 SQL 语句。代码如下：

例程02　代码位置：光盘\TM\09\ WuLiu \App_Code\ dataOperate.cs

```
    public static bool execSQL(string sql)
    {
        SqlConnection con = createCon();                    //创建连接对象
        con.Open();
        SqlCommand com = new SqlCommand(sql, con);
        try
        {
            com.ExecuteNonQuery();                          //执行SQL语句
            con.Close();                                    //关闭连接对象
        }
        catch (Exception e)
        {
            con.Close();
            return false;                                   //执行失败返回false
        }
        return true;                                        //执行成功返回true
    }
```

☑　seleSQL 方法

seleSQL(string sql)方法用来查找数据是否存在。此方法返回一个布尔型值用来表示是否查找到数据，如查找到数据则返回 true，否则返回 false。调用此方法时应传入一个 string 类型的参数，此参数表示所要执行的 SQL 语句。代码如下：

例程03　代码位置：光盘\TM\09\ WuLiu \App_Code\ dataOperate.cs

```
public static bool seleSQL(string sql)
    {
        int i;
        SqlConnection con = createCon();                    //创建连接对象
        con.Open();
        SqlCommand com = new SqlCommand(sql, con);
```

```
            try
            {
                i =Convert.ToInt32(com.ExecuteScalar());          //执行SQL语句返回第一行第一列值
                con.Close();                                      //关闭连接
            }
            catch (Exception e)
            {
                con.Close();                                      //关闭连接
                return false;
            }
            if (i > 0)                                            //判断是否大于0，大于返回true，否则返回false
            {
                return true;
            }
            else
            {
                return false;
            }
        }
```

☑ getDataset 方法

getDataset(string sql string table)方法用来查找并返回多行数据。此方法返回一个 DataSet 数据集。在调用此方法时应传入两个 string 类型的参数，第一个参数表示要执行的 SQL 语句，第二个参数表示表名。代码如下：

例程04 代码位置：光盘\TM\09\ WuLiu \App_Code\ dataOperate.cs

```
public static DataSet getDataset(string sql,string table)
    {
        SqlConnection con = createCon();                          //创建数据库连接对象
        con.Open();                                               //打开连接
        SqlDataAdapter sda = new SqlDataAdapter(sql, con);        //执行SQL语句
        DataSet ds = new DataSet();                               //创建数据集
        sda.Fill(ds, table);                                      //填充数据集
        return ds;                                                //返回数据集
    }
```

☑ getRow 方法

getRow (string sql)方法用来查找并返回一行数据。此方法返回一个 SqlCommand 对象。在调用此方法时应传入一个 string 类型的参数，此参数表示所要执行的 SQL 语句。代码如下：

例程05 代码位置：光盘\TM\09\ WuLiu \App_Code\ dataOperate.cs

```
public static SqlDataReader getRow(string sql)
{
    SqlConnection con = createCon();
    con.Open();                                   //打开数据库连接
    SqlCommand com = new SqlCommand(sql, con);
    return    com.ExecuteReader();
}
```

9.5　网站首页设计

9.5.1　网站首页概述

在网站的首页中把网站的主要功能都显示出来以方便访问者使用，使浏览者通过首页对本网站有一个全面的了解，并在第一时间浏览到本站的最新信息。首页中主要包括以下模块：

- ☑ 网站导航
- ☑ 搜索功能
- ☑ 企业推荐
- ☑ 用户登录
- ☑ 物流新闻
- ☑ 招聘信息
- ☑ 最新货源信息
- ☑ 最新车源信息
- ☑ 最新专线信息
- ☑ 最新仓储信息

网站首页的运行效果如图 9.23 所示。

图 9.23　物流信息管理平台首页

9.5.2 网站首页技术分析

在货源信息模块中有效日期列的显示格式，可以通过 GridView 控件中的 DataFormatString 属性来实现。

DataFormatString 属性语法如下：

```
{A:B}
```

冒号前的值 A 指定在从零开始的参数列表中的参数索引。此值只能设置为 0。

冒号后的值 B 指定值 A 所显示的格式。常用的数值格式如表 9.1 所示。

表 9.1 常用的数值格式

格式字符	说明	输入数值	输出格式
C	以货币格式显示数值	12345.6789	$12,345.68
D	以十进制格式显示数值	12345	12345
E	以科学记数法（指数）格式显示数值	12345.6789	1234568E+004
F	以固定格式显示数值	12345.6789	12345.68
G	以常规格式显示数值	12345.6789	12345.6789
N	以数字格式显示数值	12345.6789	12,345.68

常用的日期时间格式如表 9.2 所示。

表 9.2 常用的日期时间格式

格式字符	说明	输出格式
d	精简日期格式	yyyy-MM-dd
D	详细日期格式	yyyy年MM月dd日
f	完整格式	yyyy年MM月dd日 HH:mm
F	完整日期时间格式	yyyy年MM月dd日 HH:mm:ss
g	一般格式	MM/dd/yyyy HH:mm
G	一般格式	MM/dd/yyyy HH:mm:ss
s	适中日期时间格式	yyyy-MM-dd HH:mm:ss
t	精简时间格式	HH:mm
T	详细时间格式	HH:mm:ss

还可以直接输入时间格式，如{0:yyyy-MM-dd}将显示与{0:d}相同的格式。要注意的是 MM 必须是大写，因为 MM 大写表示的是月份，而 mm 小写表示的是时间里的分钟。

注意：把需要设置的字段的 HtmlEncode 属性设置为 False，才能显示出所设置的格式。

9.5.3　网站首页实现过程

本模块使用的数据表：tb_Freight、tb_Truck、tb_Special、tb_Depot。

1．设计步骤

（1）在该网站中新建一个 Web 窗体，将其命名为 index.aspx，用于显示网站首页。

（2）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（3）在 Tabel 表格中添加相关的服务器控件，控件的属性设置及用途如表 9.3 所示。

表 9.3　各控件名称、属性设置及用途

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
母版页	MasterPage.master	均为默认值	显示导航、登录信息等
GridView	gvNews	均为默认值	显示物流新闻
	gvJob	均为默认值	显示招聘信息
	gvFreight	均为默认值	显示货源信息
	gvTruck	均为默认值	显示车源信息
	gvSpecial	均为默认值	显示专线信息
	gvDepot	均为默认值	显示仓储信息

由于篇幅有限，这里只给出显示货源信息的 GridView 控件的前台绑定代码。实现代码如下：

例程06　代码位置：光盘\TM\09\ WuLiu \index.aspx

```
<asp:GridView ID="gvFreight" runat="server" Width="585px" AutoGenerateColumns="False">
  <Columns>
      <asp:BoundField DataField="Start" HeaderText="出发地" />
      <asp:BoundField DataField="Terminal" HeaderText="到达地" />
      <asp:BoundField DataField="FreightType" HeaderText="货物种类" />
      <asp:TemplateField HeaderText="重量">
          <ItemTemplate>
              <%#Eval("FreightWeight") %>
              <%#Eval("WeightUnit")%>
          </ItemTemplate>
       </asp:TemplateField>
       <asp:BoundField DataField="FBDate" DataFormatString="{0:yy-MM-dd}" HeaderText="发布日期"
         HtmlEncode="False" />
       <asp:TemplateField HeaderText="详细信息">
          <ItemTemplate>
             <a  href="#"  onclick='openPWD("freightInfo.aspx?ID=<%#Eval("ID")%>",340,370)'>详细信息
</a>
          </ItemTemplate>
       </asp:TemplateField>
  </Columns>
          </asp:GridView>
```

2．实现代码

在主页 Web 窗体的加载事件中调用各个功能绑定到 DataList 控件上的方法。实现代码如下：

例程07　代码位置：光盘\TM\09\ WuLiu \index.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        bindFreight();                                          //自定义方法绑定货源信息
        bindTruck();                                            //自定义方法绑定车源信息
        bindSpecial();                                          //自定义方法绑定专线信息
        bindCompany();                                          //自定义方法绑定公司信息
        bindDepot();                                            //自定义方法绑定仓储信息
       if(!IsPostBack)                                          //判断是否是首次加载
       {
            if (Session["UserName"] !=null)                     //在首次加载中判断用户是否登录
            {
                pelEntry.Visible = false;                       //不显示登录窗口
                pelWelcom.Visible = true;                       //显示欢迎词
               this.Label1.Text = Session["UserName"].ToString();//显示用户名
            }
            else
            {
                pelEntry.Visible = true;                        //显示登录窗口
                pelWelcom.Visible = false;                      //不显示欢迎词
            }
        }
    }
```

物流信息、招聘信息、最新车源信息、最新货源信息、最新专线信息、最新仓储信息和企业推荐这几个信息的显示都是通过 GridView 控件实现的。由于以上几个信息绑定的方法类似，这里主要介绍最新货源信息的绑定。最新货源信息通过自定义方法 bindFreight 将数据源绑定到 GridView 控件上。代码如下：

例程08　代码位置：光盘\TM\09\ WuLiu \index.aspx.cs

```
protected void bindFreight()
    {
        string sql = "select * from tb_Freight where FBDate >='" + day + "'";
        //调用数据库操作类getDataset方法将其返回值绑定到GridView控件上
❶       gvFreight.DataSource = dataOperate.getDataset(sql, "tb_Freight");
❷       gvFreight.DataBind();
    }
```

代码贴士

❶ GridView.DataSource 属性：表示数据源的对象，数据绑定控件从该对象中检索其数据。

❷ GridView.DataBind：将数据源绑定到 GridView 控件。

由于最新货源信息量比较大，这里使用了 GridView 控件的分页功能。在 GridView 控件的 PageIndexChanging 事件中设置当前索引并重新绑定数据源。主要代码如下：

例程09　代码位置：光盘\TM\09\ WuLiu \index.aspx.cs

```
protected void gvFreight_PageIndexChanging(object sender, GridViewPageEventArgs e)
 {
      gvFreight.PageIndex = e.NewPageIndex;                    //设置当前索引
      gvTruck.DataBind();                                      //重新绑定GridView控件
 }
```

9.5.4　单元测试

在编写程序的过程中，难免会遇到一些错误。这些错误可能是非常隐蔽而且难以发现的，因此，开发人员需要进行大量故障排查才能发现错误的根源。错误可以分为语法错误、语义错误和逻辑错误 3 种类型，下面将分别介绍。

☑　语法错误

语法错误是一种程序错误，它会影响编译器完成工作。它也是最简单的错误，几乎所有的语法错误都能被编译器或解释器发现，并将错误信息显示出来提醒程序开发人员。在 Visual Studio 2005 中遇到语法错误时，错误信息将显示在错误列表窗口中。这些信息将会告诉程序开发人员语法错误的位置（行、列和文件），并给出错误的简要说明。

☑　语义错误

程序源代码的语法正确而语义或意思与程序开发人员本意不同时，就是语义错误。此类错误比较难以察觉，它通常在程序运行过程中出现。语义错误会导致程序非正常终止。例如，在将数据信息绑定到表格控件时，经常会出现“未将对象引用设置到对象的实例中”的错误提示，此类语义错误在程序运行时，将会被调试器以异常的形式告诉程序开发人员。

☑　逻辑错误

不是所有的语义错误都容易发现，它们可能隐藏得很深。在某些语义错误下，程序仍可以继续执行，但执行结果却不是程序开发人员想要的，此类错误就是逻辑错误。例如，在程序中，需要计算表达式 c=a+b 的值，但在编程的过程中，将表达式中的“+”，写成了“-”，像这样的错误，调试器不能以异常的形式告诉程序开发人员，这种错误就是逻辑错误。程序开发人员可以通过调试解决此类错误。

9.6　用户注册页面设计

9.6.1　用户注册页面概述

浏览者可以通过用户注册功能注册成为本网站的会员。用户注册有两种注册方式，一种为个人用户注册，另一种为企业用户注册。分为两种注册方式主要考虑到用户的发布信息不一样，企业用户可以发布专线信息和招聘信息，而个人用户不可以发布这些信息。用户注册页面如图 9.24 所示。

图 9.24　用户注册页

9.6.2　用户注册技术分析

在用户注册模块中用户的注册，主要使用 Insert 语句将用户的注册信息添加到数据库中。Insert 语句用于向现有数据库中添加新的数据。

语法：

```
INSERT[INTO]
   {table_name WITH(<table_hint_limited>[…n])
|view_name
|rowset_function_limited
}
{[(column_list)]
    {VALUES
     ({DEFAULT|NULL|expression}[,..n])
     |derived_table
     |execute_statement
    }
 }
    |DEFAULT VALUES
```

Insert 语句的参数说明如表 9.4 所示。

表 9.4　Insert 语句的参数说明

参　数	参 数 说 明
[INTO]	一个可选的关键字，可以将它用在INSERT和目标表之前
table_name	将要接收数据的表或table变量的名称

续表

参　数	参 数 说 明
view_name	视图的名称及可选的别名。通过view_name来引用的视图必须是可更新的
(column_list)	要在其中插入数据的一列或多列的列表。必须用圆括号将clumn_list括起来，并且用逗号进行分隔
VALUES	引入要插入的数据值的列表。对于column_list（如果已指定）中或者表中的每个列，都必须有一个数据值。必须用圆括号将值列表括起来。如果VALUES列表中的值与表中列的顺序不相同，或者未包含表中所有列的值，那么必须使用column_list明确地指定存储每个传入值的列
DEFAULT	强制SQL Server装载为列定义的默认值。如果对于某列并不存在默认值，并且该列允许NULL，那么就插入NULL
expression	一个常量、变量或表达式。表达式不能包含SELECT或EXECUTE语句
derived_table	任何有效的SELECT语句，它返回将装载到表中的数据行

用户在使用 INSERT 语句插入数据时，必须注意以下几点。

☑ 插入项的顺序和数据类型必须与表或视图中列的顺序和数据类型相对应。

☑ 如果表中某列定义为不允许 NULL，插入数据时，该列必须存在合法值。

☑ 如果某列是字符型或日期型数据类型，插入的数据应该加上单引号。

例如使用 SQL 语句向用户信息表中插入一条记录，代码如下：

```
insert into 用户信息表 values(需要添加的字段)
```

INSERT 语句还可以一次给数据表添加多条记录，即将某一查询结果插入到指定的表中，这也是 INSERT 语句的第二种用法——批量插入。VALUES 子句指定的是一个 SELECT 子查询的结果集。INSERT 语句第二种用法的语法如下：

```
Insert Into table_name Select {* | fieldname1 [,fieldname2…]]} From table_source [Where search_condition ]
```

参数说明：

☑ Insert Into：关键字。

☑ table_name：存储数据的数据表，该数据表必须已经存在。

☑ Select：表示其后是一个查询语句。

例如将货源信息表中的数据添加到车源信息表中，代码如下：

```
insert into 车源信息表 select * from 货源信息表
```

注意：在批量插入时如果不给出列名，则必须保证两个表中的列数相同。

9.6.3　用户注册实现过程

本模块使用的数据表：tb_User、tb_GrUser。

1．设计步骤

（1）在该网站中新建一个 Web 窗体，将其命名为 login.aspx，用于实现用户注册。

（2）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（3）在 Tabel 表格中添加相关的服务器控件，控件的属性设置及用途如表 9.5 所示。

表 9.5　各控件名称、属性设置及用途

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
TextBox	txtName	均为默认值	用户注册输入用户名
	txtPass	均为默认值	用户注册输入密码
	txtQrPass	均为默认值	用户注册输入确认密码
	txtPassQuestion	均为默认值	用户注册输入密码提示问题
	txtPassSolution	均为默认值	用户注册输入密码提示答案
	txtGrLinkman	均为默认值	用户注册输入个人用户联系人
	txtGrPhone	均为默认值	用户注册输入个人用户联系电话
	txtGrAddress	均为默认值	用户注册输入个人用户所在地
	txtLinkman	均为默认值	用户注册输入企业用户联系人
	txtCompanyName	均为默认值	用户注册输入企业名称
	txtCalling	均为默认值	用户注册输入企业所属行业
	txtLicenceNumber	均为默认值	用户注册输入企业营业执照号
	txtAddress	均为默认值	用户注册输入企业地址
	txtPhone	均为默认值	用户注册输入企业用户联系电话
	txtFax	均为默认值	用户注册输入企业传真
	txtEmail	均为默认值	用户注册输入企业电子邮件
	txtNetworkIP	均为默认值	用户注册输入企业网址
	txtContent	将 TextMode 属性设置为“MultiLine”（设置文本框模式）	用户注册输入企业简介

续表

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
Web 用户控件	dh.ascx	均为默认值	用于导航
Panel	pelDaohan	均为默认值	导航会员注册方式
	pelBase	均为默认值	显示基本信息
	pelGrInfo	均为默认值	显示个人详细信息
	pelQyInfo	均为默认值	显示企业详细信息
Button	btnNext	均为默认值	“下一步”按钮
	btnQyLogin	均为默认值	企业用户注册按钮
	btnGrLogin	均为默认值	个人用户注册按钮
RequiredFieldValidator	RequiredFieldValidatorName	将ControlToValidate属性设置为txtName（要验证控件的ID）	验证注册用户是否输入用户名
	RequiredFieldValidatorPass	将ControlToValidate属性设置为txtPass（要验证控件的ID）	验证注册用户是否输入密码
	RequiredFieldValidatorEmail	将ControlToValidate属性设置为txtEmail（要验证控件的ID）	验证注册用户是否输入电子邮件
CompareValidator	CompareValidatorQpass	将ControlToCompare属性设置为txtPass（用于比较控件的ID），将ControlToValidate属性设置为txtQpass（要验证控件的ID）	验证注册用户输入的两次密码是否一致
RegularExpressionValidator	RegularExpressionValidatorEamil	将ValidationExpression属性设置为“\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*”（用来设置正则表达式）	验证用户输入的电子邮件地址是否正确

2．实现代码

在用户注册页面中，选择不同的注册方式进入相应的注册详细信息页面，来进行不同级别的会员注册。在页面加载事件中，使用 Panel 控件显示导航选择会员注册方式。实现代码如下：

例程10　代码位置：光盘\TM\09\ WuLiu \login.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
❶       pelDaohan.Visible = true;                    //导航用户注册方式
        pelBase.Visible = false;                     //用户基本注册信息
        pelQyInfo.Visible = false;                   //企业用户注册详细信息
        pelGrInfo.Visible = false;                   //个人用户注册详细信息
}
```

代码贴士

❶ Visible 属性：获取或设置一个值，该值指示服务器控件是否呈现在页面上。

当用户单击“企业会员注册”按钮时，在此按钮的 Click 事件中将注册方式记录下来，并通过 Panel 控件显示输入基本信息的页面。实现代码如下：

例程11　代码位置：光盘\TM\09\ WuLiu \login.aspx.cs

```
protected void LinkButton2_Click(object sender, EventArgs e)
    {
        loginType = 1;                  //记录注册方式是企业用户注册
        pelDaohan.Visible = false;      //不显示导航注册方式
        pelBase.Visible = true;         //显示基本注册信息
        pelQyInfo.Visible = false;      //不显示个人注册详细信息
        pelGrInfo.Visible = false;      //不显示企业注册详细信息
    }
```

用户输入完基本信息后，单击“下一步”按钮，在此按钮的 Click 事件中将用户注册的基本信息存储下来，再根据记录的用户注册方式检测用户名是否存在，如果存在将显示企业会员注册的详细信息页面，如果不存在将给出相应的提示信息。实现代码如下：

例程12　代码位置：光盘\TM\09\ WuLiu \login.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
    {
        name = this.txtName.Text;                                   //存储用户名
        pass = this.txtPass.Text;                                   //存储密码
        passQuestion = this.txtPassQuestion.Text;                   //存储密码提示问题
        passSolution = this.txtPassSolution.Text;                   //存储密码提示答案
        string QySql = "select * from tb_User where UserName='" + name + "'"; //查询企业用户名是否存在SQL语句
        string GrSql = "select * from tb_GrUser where Name='" + name + "'";  //查询个人用户名是否存在SQL语句
        if (loginType == 0)                                         //判断会员注册方式
        {
            if (!dataOperate.seleSQL(GrSql))                        //判断个人用户名是否存在
            {
                pelDaohan.Visible = false;                          //不显示导航会员注册方式
                pelBase.Visible = false;                            //不显示基本注册信息
                pelQyInfo.Visible = false;                          //不显示企业注册详细信息
                pelGrInfo.Visible = true;                           //显示个人注册详细信息
            }
            else
                RegisterStartupScript("false", "<script>alert('用户名已经此在')</script>");
        }
        else
        {
            if (!dataOperate.seleSQL(QySql))                        //判断个人用户名是否存在
            {
                pelDaohan.Visible = false;                          //不显示导航会员注册方式
                pelBase.Visible = false;                            //不显示基本注册信息
                pelQyInfo.Visible = true;                           //显示企业注册详细信息
```

```
                    pelGrInfo.Visible = false;                    //不显示个人注册详细信息
                }
                else
                    RegisterStartupScript("false", "<script>alert('用户名已经此在')</script>");
            }
        }
```

用户输入完企业注册的详细信息后，单击“注册”按钮，通过 SQL 语句利用数据库操作类中的 execSQL 方法将企业注册信息添加到数据库中。实现代码如下：

例程13　代码位置：光盘\TM\09\ WuLiu \login.aspx.cs

```
    protected void Button2_Click(object sender, EventArgs e)
        {
            string linkman = this.txtLinkman.Text;                    //存储联系人
            string companyName = this.txtCompanyName.Text;            //存储企业名称
            string ddlKind = this.ddlKind.SelectedValue;              //存储企业性质
            string calling = this.txtCalling.Text;                    //存储所属行业
            string licenceNumber = this.txtLicenceNumber.Text;        //存储营业执照号
            string address = this.txtAddress.Text;                    //存储公司地址
            string phone = this.txtPhone.Text;                        //存储联系电话
            string fax = this.txtFax.Text;                            //存储传真
            string email = this.txtEmail.Text;                        //存储电子邮件
            string networkIP = this.txtNetworkIP.Text;                //存储公司网址
            string content = this.txtContent.Text;                    //存储内容简介
            string adSql = "insert into tb_User values('" + name + "','" + pass + "','" + passQuestion + "','" + passSolution +
"','" + linkman + "','" +
                companyName + "','" + ddlKind + "','" + calling + "','" + licenceNumber + "','" + address + "','" + phone +
"','" + fax + "','" +
                email + "','" + networkIP + "','" + content + "') ";
            if (dataOperate.execSQL(adSql))                           //判断是否添加成功
            {
                bindEmail();                                  //自定义方法将用户的登录名和密码发送到E-mail中
                Response.Write("<script>alert('添加成功！')</script>");
            }
            else
            {
                RegisterStartupScript("false", "<script>alert('添加失败！')</script>");
            }
        }
```

9.7　搜索信息功能

9.7.1　搜索信息功能概述

搜索功能可以使浏览者快速有效地查找到需要的信息。搜索功能可以按不同的信息类型进行搜索，如图 9.25 所示。

图 9.25　搜索信息功能

9.7.2　搜索信息功能技术分析

搜索信息功能主要使用了 SQL 语句中的 LIKE 模糊查询。在对要查询的数据表中的数据了解得不全面的情况下，可以使用 LIKE 模糊查询。例如不能确定所要查询人的姓名，只知道姓李；查询某个人的联系方式只知道是以“3451”结尾等，这时都可以使用 LIKE 进行模糊查询。LIKE 关键字需要使用通配符在字符串内查找指定的模式，所以读者需要了解通配符及其含义。通配符的含义如表 9.6 所示。

表 9.6　LIKE 关键字中的通配符及其含义

通　配　符	说　　明
%	由零个或更多字符组成的任意字符串
_	任意单个字符
[ ]	用于指定范围，例如[A~F]，表示A~F范围内的任何单个字符
[^]	表示指定范围之外的，例如[^ A~F]，表示A~F范围以外的任何单个字符

☑　“%”通配符

“%”通配符能匹配零个或更多个字符的任意长度的字符串。

例如，在货源信息表出发地字段中，查询第一个字为“天”的记录。SQL 语句如下：

```
select * from    tb_Freight    where start    like '天%'
```

☑　“_”通配符

“_”号表示任意单个字符，该符号只能匹配一个字符，利用“_”号可以作为通配符组成匹配模式进行查询。

例如，在货源信息表货源类型字段中，查询只有两个字但第一个字必须是“汽”的记录。SQL 语句如下：

```
select * from    tb_Freight where FreightType    like '汽_'
```

☑　“[ ]”通配符

在模糊查询中可以使用“[ ]”符号来查询一定范围内的数据。“[ ]”符号用于表示一定范围内的任意单个字符，它包括两端数据。

例如，在货源信息表联系电话字段中，查询电话号码以“23”结尾并且开头数字位于 1~5 之间的记录。SQL 语句如下：

```
select * from    tb_Freight where Phone    like '[1-5]23'
```

☑　“[^]”通配符

在模糊查询中可以使用“[^]”符号来查询不在指定范围内的数据。“[^]”符号用于表示不在某范围内的任意单个字符，它包括两端数据。

例如，在货源信息表联系电话字段中，查询电话号码以“23”结尾，但不以 1 开头的记录。SQL 语句如下：

```
select * from    tb_Freight where Phone    like '[^1]23'
```

9.7.3　搜索信息功能实现过程

本模块使用的数据表：tb_Search。

1．设计步骤

（1）在该网站中新建一个 Web 用户控件，将其命名为 search.ascx，用于实现搜索功能。

（2）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（3）在 Tabel 表格中添加相关的服务器控件，控件的属性设置及用途如表 9.7 所示。

表 9.7　各控件名称、属性设置及用途

控 件 类 型	控 件 名 称	主要属性设置及用途	控 件 用 途
DropDownList	ddlSearchType	均为默认值	选择信息类型
	ddlKeyType	均为默认值	关键字类型
TextBox	txtKey	均为默认值	输入关键字
	txtTerminal	将Visible属性设置为“false”	输入到达地
Button	btnSearch	均为默认值	“搜索”按钮
Label	labTerminal	均为默认值	显示到达地关键字

2．实现代码

在搜索功能的加载事件中调用自定义方法 bindSearchType，将 DropDownList 控件的数据源进行绑定。实现代码如下：

例程14　代码位置：光盘\TM\09\ WuLiu \search.ascx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)                          //判断是否是首次加载页面
        {
            bindSearchType();
        }
    }
```

自定义方法 bindSearchType 将显示信息类别的 DropDownList 控件进行绑定，并初始化显示关键字类型的 DropDownList 控件。实现代码如下：

例程15　代码位置：光盘\TM\09\ WuLiu \search.ascx.cs

```
  public void bindSearchType()
    {
        string sql = "select distinct searchType,type from tb_Search";      //调用数据库操作类中的getDataset方法
        //绑定显示信息类别的DropDownList控件的数据源
        DataSet ds = dataOperate.getDataset(sql, "tb_Search");
❶       ddlSearchType.DataSource = ds.Tables["tb_Search"].DefaultView;
```

```
❷              ddlSearchType.DataTextField = "searchType";
❸              ddlSearchType.DataValueField = "type";
               ddlSearchType.DataBind();
               bindKey();                                          //自定义方法绑定关键字类型
          }
```

代码贴士

❶ DataSource 属性：数据源的对象，数据绑定控件从该对象中检索其数据。

❷ DataTextField：设置显示文本的字段名。

❸ DataValueField：设置值的字段名。

自定义方法 bindKey 将显示关键字类型的 DropDownList 控件进行绑定。实现代码如下：

例程16　代码位置：光盘\TM\09\ WuLiu \search.Ascx.cs

```
    public void bindKey()
        {
            //获取当前选择的信息类型的表名
❶           string type = ddlSearchType.SelectedValue.ToString();
            string sql = "select searchKey,keyword from tb_Search where type='" + type + "'";
            //调用数据库操作类中getDataset方法并获取返回的数据集
            DataSet ds = dataOperate.getDataset(sql, "tb_Search");
            //绑定关键字类别的DropDownList控件的数据源
            ddlKeyType.DataSource = ds.Tables["tb_Search"].DefaultView;
            //绑定关键字类别的DropDownList控件文本的字段名
            ddlKeyType.DataTextField = "searchKey";
            //绑定关键字类别的DropDownList控件值的字段名
            ddlKeyType.DataValueField = "keyword";
            ddlKeyType.DataBind();
            //调用自定义方法是否显示到达地文本框
            bindTerminal();
        }
```

代码贴士

❶ SelectedValue 属性：获取当前选择项的值。

自定义方法 bindTerminal 主要用来判断关键字类型是否选择出发地类型，如果选择出发地类型将显示到达地文本框，否则不显示。实现代码如下：

例程17　代码位置：光盘\TM\09\ WuLiu \search.Ascx.cs

```
public void bindTerminal()
 {
      //判断关键字类型是否选择了出发地
      if (ddlKeyType.SelectedValue.ToString() == "Start")
      {
          txtTerminal.Text = "";   //清空到达地文本框
          Label1.Visible = true;
          txtTerminal.Visible = true;
      }
      else
```

```
            {
                Label1.Visible = false;
                txtTerminal.Visible = false;
            }
        }
```

当浏览者添加完需要搜索的信息后单击“搜索”按钮，将浏览者添加的搜索信息转换成 SQL 语句存储到 Session 中，并跳转到搜索的详细信息页面。实现代码如下：

例程18　代码位置：光盘\TM\09\ WuLiu \search.ascx.cs

```
    protected void Button1_Click(object sender, EventArgs e)
        {
            string table = ddlSearchType.SelectedValue.ToString();                    //获取表名
            string keyType = ddlKeyType.SelectedValue.ToString();                     //获取字断名
            string keys = txtKey.Text;                                                //获取关键字
            string sql;
            if (txtTerminal.Text != "")
            {
                sql = "select * from " + table + " where " + keyType + " like '%" + keys + "%' and   terminal like '%" +
txtTerminal.Text + "%'";
            }
            else
            {
                sql = "select * from " + table + " where " + keyType + " like '%" + keys + "%'";
            }
            Session["searchSql"] = sql;                                               //j将SQL语句保存到Session中
            Session["searchType"] = ddlSearchType.SelectedValue.ToString();           //将表名保存到Session中
            Response.Redirect("searchList.aspx");
        }
```

9.8　发布信息页设计

9.8.1　发布信息页概述

会员通过发布信息模块发布信息。根据用户的登录方式不同发布的信息内容也不同，以个人方式登录的用户能发布货源信息、车源信息、仓储信息，如图 9.26 所示。

以企业方式登录的用户能发布货源信息、车源信息、仓储信息、专线信息、招聘信息，如图 9.27 所示。

9.8.2　发布信息页技术分析

在添加货源出发地时使用到了省与城市之间的联动功能。省市联动功能主要使用两个 DropDownList 控件绑定数据库中的省和市的详细信息。创建自定义方法将显示出发地的省和到达地的省的 DropDownList 控件进行绑定。实现代码如下：

图 9.26　个人用户发布信息页

图 9.27　企业用户发布信息页

例程19　代码位置：光盘\TM\09\ WuLiu \issuanceFreight.aspx.cs

```
public void bindSf()
    {
        string sql="select distinct sf from tb_City";
            //调用数据库操作类中的getDataset方法并接受返回的数据集
        DataSet ds=dataOperate.getDataset(sql, "tb_City");
            //绑定出发省的数据源
        this.ddlcSf.DataSource = ds.Tables["tb_City"].DefaultView;
            //绑定到达省的数据源
        this.ddldSf.DataSource = ds.Tables["tb_City"].DefaultView;
            //绑定出发省DropDownList控件的文本值
        ddlcSf.DataTextField = "sf";
            //绑定出发省DropDownList控件的值
        ddlcSf.DataValueField = "sf";
        ddldSf.DataTextField = "sf";
        ddldSf.DataValueField = "sf";
        this.ddlcSf.DataBind();
        this.ddldSf.DataBind();
    }
```

当用户改变出发省的选项时，显示城市的 DropDownList 控件的列表也随之改变，这个功能主要通过显示省的 DropDownList 控件中的 SelectedIndexChanged 事件来实现。实现代码如下：

例程20　代码位置：光盘\TM\09\ WuLiu \issuanceFreight.aspx.cs

```
protected void ddlcSf_SelectedIndexChanged(object sender, EventArgs e)
    {
            //获取所选择省的值
        string sf = ddlcSf.SelectedValue.ToString();
        string sql = "select cs from tb_City where sf='" + sf+"'";
            //调用数据库操作类中的getDataset方法并接受返回的数据集
        DataSet ds = dataOperate.getDataset(sql, "tb_City");
        this.ddlcCs.DataSource = ds.Tables["tb_City"].DefaultView;
            //绑定出发市DropDownList控件的文本值
        ddlcCs.DataTextField = "cs";
            //绑定出发市DropDownList控件的值
        ddlcCs.DataValueField = "cs";
        this.ddlcCs.DataBind();
    }
```

9.8.3　发布信息页实现过程

本模块使用的数据表：tb_Freight。

1．设计步骤

（1）在该网站中新建一个 Web 窗体，将其命名为 issuanceFreight.aspx，用于实现发布货源信息功能。

（2）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（3）在 Tabel 表格中添加相关的服务器控件，控件的属性设置及用途如表 9.8 所示。

表 9.8　各控件名称、属性设置及用途

控 件 类 型	控 件 名 称	主要属性设置	用　　途
TextBox	txtFreightType	均为默认值	输入货物种类
	txtFreightWeight	均为默认值	输入货物重量
	txtLinkman	均为默认值	输入发布货物联系人
	txtPhone	均为默认值	输入发布货物人联系电话
	txtTerm	均为默认值	输入货物信息有效日期
	txtContent	将TextMode属性设置为MultiLine（设置文本框模式）	输入货物信息备注
RadioButton	rdibtnDun	Checked设置为true（控件已选中状态），GrupName设置为weighte（单选按钮所属的组）	选择货物重量单位吨
	rdibtnFan	GrupName设置为weighte（单选按钮所属的组）	选择货物重量单位方
	rdibtnJian	GrupName设置为weighte（单选按钮所属的组）	选择货物重量单位件
DropDownList	ddlcSf	将AutoPostBack属性设置为True（自动回传到服务器）	选择货物出发地的省
	ddlcCs	将AutoPostBack属性设置为True（自动回传到服务器）	选择货物出发地的市
	ddldSf	将AutoPostBack属性设置为True（自动回传到服务器）	选择货物到达地的省
	ddldCs	将AutoPostBack属性设置为True（自动回传到服务器）	选择货物到达地的市
Button	btnIssuance	均为默认值	发布货物信息按钮

2．实现代码

发布各种信息的实现过程类似，这里主要讲解如何发布货源信息。当用户单击“发布”按钮时，在此按钮的 Click 事件中将用户添加的货源信息通过 SQL 语句使用数据库操作类中的 execSQL 方法将货源详细信息存储到数据库中。实现代码如下：

例程21　代码位置：光盘\TM\09\ WuLiu \issuanceFreight.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
    {
        string UserName = Session["UserName"].ToString();                                    //存储用户登录名
        string Start = ddlcSf.SelectedValue.ToString() + ddlcCs.SelectedValue.ToString();    //存储出发地
        tring Terminal = ddldSf.SelectedValue.ToString() + ddldCs.SelectedValue.ToString();  //存储到达地
        string FreightType = this.txtFreightType.Text;                                       //存储货物类型
        string FreightWeight = this.txtFreightWeight.Text;                                   //存储货物重量
        string WeightUnit;                                                                   /用于存储重量单位
        if (rdibtnDun.Checked)                                                               //判断重量单位类型
        {
            WeightUnit = "吨";
        }
        else
            if (rdibtnFan.Checked)
            {
                WeightUnit = "方";
            }
        else
            {
```

```
                    WeightUnit = "件";
                }
            string Linkman = this.txtLinkman.Text;                    //存储联系人
            string Phone = this.txtPhone.Text;                        //存储联系电话
            string Term = this.txtTerm.Text;                          //存储有效日期
            string Content = this.txtContent.Text;                    //存储详细信息
            string FBDate=DateTime.Now.ToString();                    //存储发布日期
            string UserType = Session["UserType"].ToString();
            string sql = "insert into tb_Freight values('" + UserName + "','" + Start + "','" + Terminal + "','" + FreightType + "','" +
                FreightWeight + "','" + WeightUnit + "','" + Linkman + "','" + Phone + "','" + Term + "','" + Content + "','" + FBDate + "','" + UserType + "','')";
                    //判断货物信息是否插入
            if (dataOperate.execSQL(sql))
            {
                RegisterStartupScript("true", "<script>alert('发布成功！')</script>");
            }
            else
            {
                RegisterStartupScript("false", "<script>alert('发布失败！')</script>");
            }
        }
```

9.8.4　单元测试

在编写完程序后需要对程序进行调试，而断点是调试的核心，它是.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。断点模式可以看作是一种超时，所有元素（如函数、变量和对象）都保留在内存中，但它们的移动和活动被挂起了。在中断模式下，可以检查它们的位置和状态，以查看是否存在冲突或 bug。可以在中断模式下对程序进行调整，如果没有这个功能，调试大的程序几乎是不可能的。

（1）设置断点

添加一个断点，当遇到该断点所在的代码时，就中断执行。单击该代码左边的灰色区域，或者右击该代码行，选择“断点”/“插入断点”命令，断点在该行的旁边显示为一个红色的圆，该行代码也突出显示，如图 9.28 所示。

（2）断点窗口

使用断点窗口可以查看文件中的断点信息。根据以下任何一种方法均可显示断点窗口：

☑　按下 Ctrl+Alt+B 组合键。

☑　选择菜单栏中的“调试”/“窗口”/“断点”命令。

☑　在调试工具栏中单击即时下拉图标，选择“断点”。

断点窗口如图 9.29 所示。通过选择断点窗口中对应的复选框，能够启用或禁用所有断点。

在该窗口中，可以禁用断点（删除描述信息左边的记号；禁用的断点用填充为红色的圆圈来表示）、删除断点、编辑断点的属性。该窗口中还显示了条件和命中次数两个可用属性，它们是非常有用的。右击“断点”，在弹出的快捷菜单中选择相应的条件，可对其进行编辑。

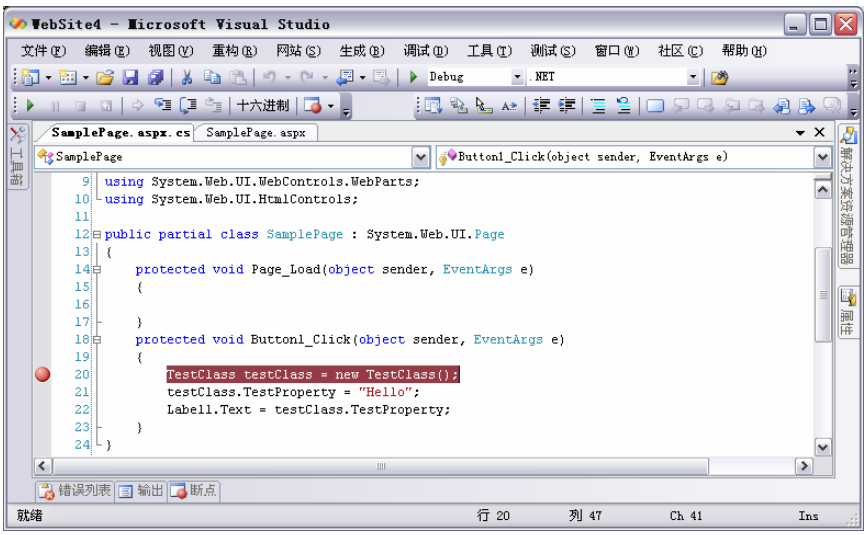

图 9.28　设置断点

图 9.29　断点窗口

（3）断点属性

为提供更大的灵活性，通过 Visual Studio 调试器能够设置属性以修改断点的行为。通过属性菜单可对它们进行设置。右击断点轮廓左边缘，弹出断点属性菜单，如图 9.30 所示。

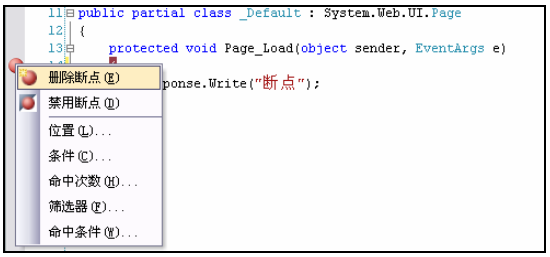

图 9.30　断点属性

在属性菜单中，前两个命令可删除或者禁用所选中断点。当选择“禁用断点”命令时，该命令会发生切换，断点图标将显示为一个空心圆。

①位置

选择“位置”命令，将会弹出如图 9.31 所示“文件断点”对话框，其中包括清晰的断点说明信息。

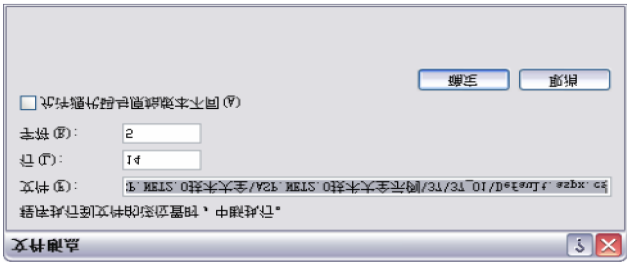

图 9.31　“文件断点”对话框

②条件

选择“条件”命令，将会弹出如图 9.32 所示“断点条件”对话框。在此对话框中的文本框内可以输入任何有效表达式。当程序运行到断点时，将计算该表达式值。依据所选中的单选按钮和条件表达式所计算的值，所运行应用程序或者停止，或者继续。两个单选按钮说明如下。

☑ 为 True(T)：如果输入条件计算所得是一个 Boolean 值 True，那么应用程序将会暂停。

☑ 已更改：如所输入条件发生了变化，那么应用程序将会暂停。第一次运行这段正在调试的代码时，断点并不会使程序暂停，因为没有内容可比较。第二次运行这段代码时，会初始化表达式，同时将对二者进行比较。

③命中次数

选择“命中次数”命令，将会弹出如图 9.33 所示“断点命中次数”对话框。在默认情况下，每次命中断点，调试器就中断执行。通过设置命中次数，可以让调试器每命中两次断点中断执行一次，也可以每 10 次、每 512 次或任意选择的次数中断一次。

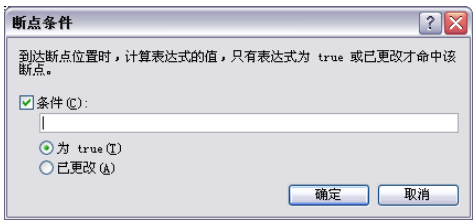

图 9.32　“断点条件”对话框

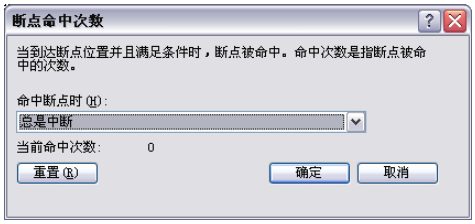

图 9.33　“断点命中次数”对话框

④筛选器

选择“筛选器”命令，将会弹出如图 9.34 所示“断点筛选器”对话框。在此对话框中可指定机器、进程、线程或者它们的任意组合设置断点，前提是该断点必须有效。

⑤命中条件

选择“命中条件”命令，将会弹出如图 9.35 所示“命中断点时”对话框。在此对话框中指定在命中断点时应执行的操作。默认情况下，调试器中断执行，但是可以选择打印一条消息或者运行一个 Visual Studio 宏。如果选择打印消息而不是中断，则断点的效果与 Trace 语句非常类似。这种使用断点的方法称为跟踪点。当“打印消息”复选框被选中时，窗口左边红色圆形的断点图标会变成菱形。

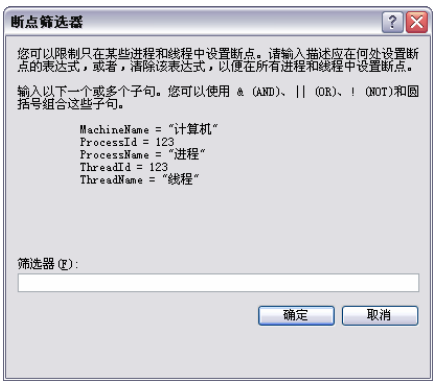

图 9.34 “断点筛选器”对话框

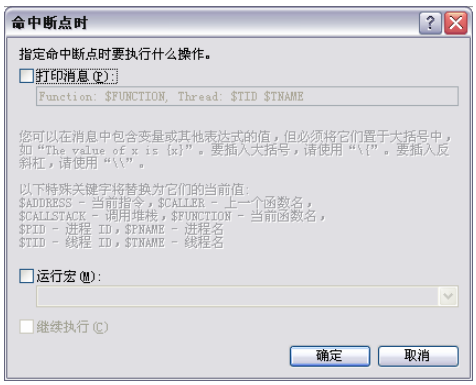

图 9.35 “命中断点时”对话框

9.9 货源信息页设计

9.9.1 货源信息页概述

货源信息页面主要显示所有用户发布的货源信息，浏览者可以通过此页浏览到自己需要的货源信息。货源信息页运行效果如图 9.36 所示。

图 9.36 货源信息页

9.9.2 货源信息页技术分析

在货源信息页面中主要通过 DataSet 对象将其绑定到 GridView 控件上将所有的货源信息显示出来，

下面主要介绍 DataSet 对象。

DataSet（数据集）对象相当于内存中的数据库，在命名空间 System.Data 中定义。DataSet 是一个完整的数据集。在 DataSet 内部，主要可以存储 5 种对象，如表 9.9 所示。

表 9.9　DataSet 的对象

属　性	说　明
DataTable	使用行、列形式来组织的一个矩形数据集
DataColumn	一个规则的集合，描述决定将什么数据存储到一个DataRow中
DataRow	由单行数据库数据构成的一个数据集合，该对象是实际的数据存储
Constraint	决定能进入DataTable的数据
DataRelation	描述了不同的DataTable之间如何关联

在 DataSet 内部是一个或多个 DataTable 的集合。在每个 DataTable 的集合中都包括 DataRow 对象、DataColumn 对象和 Constraint 集合以及 DataRelation 集合。DataTable 和其内部的 DataRelation 集合对应于父关系和子关系，二者建立了 DataTable 之间的连接。DataSet 内部的 DataRelation 集合是所有 DataTable 集合中的一个聚合视图。

显示货源信息的 GridView 控件将数据库操作类中 getDataSet 方法返回的 DataSet 对象绑定到数据源上，将所有货源信息显示出来。代码如下：

```
gvFreight.DataSource = dataOperate.getDataset(sql, "tb_Freight");
```

9.9.3　货源信息页实现过程

本模块使用的数据表：tb_Freight。

1．设计步骤

（1）在该网站中新建一个 Web 窗体，将其命名为 freightList.aspx，用于实现显示所有货源信息。

（2）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（3）在 Table 中添加一个 GridView 控件，此控件用于显示所有货源信息。

这里给出显示货源信息的 GridView 控件前台绑定代码。实现代码如下：

例程22　代码位置：光盘\TM\09\ WuLiu \freightList.aspx

```
    <asp:GridView ID="gvFreight" runat="server" AutoGenerateColumns="False" Width="585px" AllowPaging="True"
OnPageIndexChanging="gvFreight_PageIndexChanging">
        <Columns>
          <asp:BoundField DataField="Start" HeaderText="出发地" />
          <asp:BoundField DataField="Terminal" HeaderText="到达地" />
          <asp:BoundField DataField="FreightType" HeaderText="货物种类" />
          <asp:TemplateField HeaderText="重量">
              <ItemTemplate>
                  <%#Eval("FreightWeight") %>
                  <%#Eval("WeightUnit")%>
              </ItemTemplate>
          </asp:TemplateField>
```

```
            <asp:BoundField DataField="FBDate" DataFormatString="{0:yy-MM-dd}" HeaderText="发布日期"
                    HtmlEncode="False" />
                <asp:TemplateField HeaderText="详细信息">
                    <ItemTemplate>
        <a href="#" onclick='openPWD("freightInfo.aspx?ID=<%#Eval("ID")%>",340,370)'>详细信息</a>
                    </ItemTemplate>
                    </asp:TemplateField>
        </Columns>
        </asp:GridView>
```

2．实现代码

在 Web 窗体的加载事件中调用自定义 bindFreight 方法将货源信息绑定到 GridView 控件上。实现代码如下：

例程23　代码位置：光盘\TM\09\ WuLiu \freightList.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        bindFreight();                    //自定义方法
    }
```

自定义 bindFreight 方法通过 SQL 语句利用数据库操作类中的 getDataset 方法，将数据源绑定到 GridView 控件上将所有货源信息显示出来。实现代码如下：

例程24　代码位置：光盘\TM\09\ WuLiu \freightList.aspx.cs

```
protected void bindFreight()
    {
        string sql = "select * from tb_Freight order by ID DESC";
        //调用getDataset方法将返回值绑定到GridView上
        gvFreight.DataSource = dataOperate.getDataset(sql, "tb_Freight");
        gvFreight.DataBind();
    }
```

由于货源信息量很大，为了页面的美观和浏览方便，使用了 GridView 控件自带的分页功能。在 GridView 控件中的 PageIndexChanging 事件中设置当前页的索引并重新绑定 GridView 控件。实现代码如下：

例程25　代码位置：光盘\TM\09\ WuLiu \freightList.aspx.cs

```
protected void gvFreight_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {
❶       gvFreight.PageIndex = e.NewPageIndex;        //设置当前页索引
        gvFreight.DataBind();                        //重新绑定GridView控件
    }
```

代码贴士

❶ NewPageIndex 属性：获取或设置新页的索引。

9.10　货源详细信息页设计

9.10.1　货源详细信息页概述

当浏览者在货源信息页面单击某条信息的“详细信息”链接按钮时，将进入一个新的页面，在此页面中显示此条货源的详细信息，如图 9.37 所示。

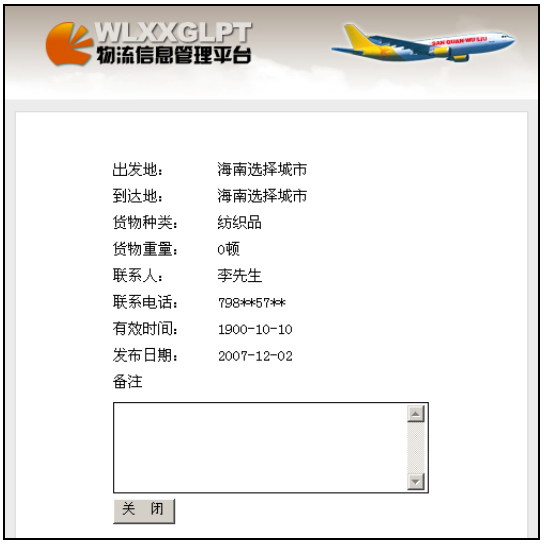

图 9.37　货源详细信息页

9.10.2　货源详细信息页技术分析

在货源详细信息页面中，主要使用 SqlDataReader 对象将货源信息表中的各个字段显示在页面上。以下介绍一下 SqlDataReader 对象。

DataReader 对象主要用来读取数据结果，使用它读取记录时通常比使用 DataSet 更快。DataReader 类有 3 种：SqlDataReader、OleDbDataReader 和 OdbcDataReader。DataReader 对象用 Commmand 对象从数据库中读取记录，每次只能返回一条记录保存到内存中，从而避免了使用大量内存，大大提高了性能。DataReader 离不开与数据库的连接，它是与底层数据库紧密联系在一起的。需要注意的是，DataReader 对象返回的结果是一个只读的且仅向前的数据流。DataReader 对象的常用属性及说明如表 9.10 所示。

表 9.10　DataReader 常用属性及说明

属　　性	说　　明
Depth	设置阅读器的深度
FieldCount	返回当前指定行的列数

续表

属　　性	说　　明
IsClosed	返回当前对象是否关闭
Item	指定字段的值
RecordsAffected	返回影响的记录数

注意：IsClosed 和 RecordsAffected 是在一个已经关闭的 DataReader 对象上可以调用的唯一属性。

DataReader 对象的常用方法及说明如表 9.11 所示。

表 9.11　DataReader 常用方法及说明

方　　法	说　　明
Close	关闭DataReader对象
GetBoolean	返回所获取的布尔型值
GetByte	返回所获取的Byte类型值
GetChar	返回所获取的Char类型值
GetDataTypeName	返回指定列的数据类型
GetDateTime	返回所获取的DateTime对象
GetInt32	返回所获取的Int类型值
GetName	返回指定列数的列名
GetString	返回所获取的String型值
GetType	返回当前对象的Type对象
GetValue	以本机格式返回指定字段的值
GetValues	返回包含指定列的数据的对象，该对象的类型为指定列的原始类型和格式
NextResult	读取批处理SQL语句的结果时，移动到下一个结果
Read	从数据源中读取一个或多个记录集，返回值为True，表示仍有记录未读取；否则表示已经读取到最后一条记录

例如，使用 SqlDataReader 对象获取货源信息表中编号等于 1 的记录，并将此记录中出发地信息显示在文本框中。代码如下：

```
SqlConnection con = new SqlConnection("server=.;database=db_WL;uid=sa;pwd=;")
con.Open();
string sql = "select * from tb_Freight where ID=1;
SqlCommand com = new SqlCommand(sql, con);
SqlDataReader sdr    com.ExecuteReader();
  sdr.Read();                                                    //读取下条记录
txtStart.Text = sdr["start"].ToString();
```

9.10.3　货源详细信息页实现过程

本模块使用的数据表：tb_Freight。

1．设计步骤

（1）在该网站中新建一个 Web 窗体，将其命名为 freightInfo.aspx，用于实现显示货源详细信息。

（2）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（3）在 Tabel 表格中添加相关的服务器控件，控件的属性设置及用途如表 9.12 所示。

表 9.12　各控件名称、属性设置及用途

控 件 类 型	控 件 名 称	主要属性设置	用　　途
TextBox	txtFreightType	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示货物种类
	txtFreightWeight	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示货物重量
	txtLinkman	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示联系人
	txtPhone	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示联系电话
	txtTerm	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示有效日期
	txtContent	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示公司
	txtFBDate	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示发布日期
	txtStart	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示出发地
	txtTerminal	将ReadOnly属性设置为True（设置文本为只读，不可以更改文本）	显示到达地
Input (Button)	btnClose	均为默认值	关闭页面按钮

2．实现代码

在 Web 窗体页面的加载事件中调用自定义方法 bindFreightInfo()将货源的详细信息显示出来。实现代码如下：

例程26　代码位置：光盘\TM\09\ WuLiu \freightInfo.aspx.cs

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

在 bindFreightInfo 自定义方法中通过货源信息页面传入的编号，在数据库中查找符合该编号的货源详细信息并显示出来。实现代码如下：

例程27　代码位置：光盘\TM\09\ WuLiu \freightInfo.aspx.cs

```
    protected void bindFreightInfo()
{
    string ID = Request.QueryString["ID"].ToString();
```

```
        string sql = "select * from tb_Freight where ID=" + ID;
        //通过数据库操作类中的row方法找到此条货源详细信息
        SqlDataReader sdr = dataOperate.getRow(sql);
        sdr.Read();                                                    //读取下条记录
        //获取货源详细信息
        txtStart.Text = sdr["start"].ToString();
        txtTerminal.Text = sdr["Terminal"].ToString();
        txtFreightType.Text = sdr["FreightType"].ToString();
        txtFreightWeight.Text = sdr["FreightWeight"].ToString() + sdr["WeightUnit"].ToString();
        txtLinkman.Text = sdr["Linkman"].ToString();
        txtPhone.Text = sdr["Phone"].ToString();
        txtTerm.Text = Convert.ToDateTime(sdr["Term"]).ToLongDateString();
        txtContent.Text = sdr["Content"].ToString();
        txtFBDate.Text = Convert.ToDateTime(sdr["FBDate"]).ToLongDateString();
    }
```

9.11 货源信息管理页

9.11.1 货源信息管理页概述

管理员通过货源信息管理模块对货源信息进行审核、查看详细信息和将过期信息删除。货源信息管理页如图 9.38 所示。

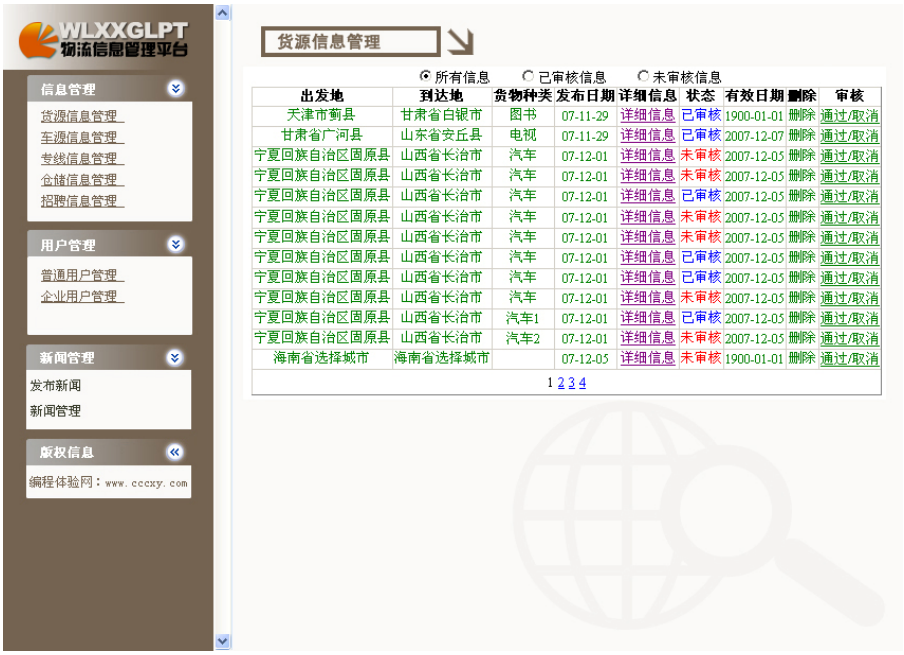

图 9.38 货源信息管理页

9.11.2　货源信息管理页技术分析

信息的有效性尤为重要，如果信息已经过期就需要管理员将其删除。为了方便管理员查阅，将过期的信息以特殊颜色显示出来。实现此功能可以在 GridView 控件中的 RowDataBound 事件中实现，在此事件中先将当天的日期获取，再将此条信息的有效日期获取，如果有效日期小于当前日期说明此条信息已经过期，并通过 ForeColor 属性改变此条信息的颜色。改变颜色通过 Color 类实现，Color 类需要引用命名空间 System.Drawing。实现代码如下：

例程28　代码位置：光盘\TM\09\ WuLiu \manage_Freight.aspx.cs

```
protected void gvFreight_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
❶           DateTime term = Convert.ToDateTime(e.Row.Cells[6].Text);          //获取有效日期
❷           DateTime nowDate =DateTime.Now.Date;                              //获取当前日期
            if (term < nowDate)                                               //判断是否过期
            {
                e.Row.ForeColor = Color.Green;
            }
        }
    }
```

代码贴士

❶ Convert：将一种基本数据类型转换成另一种基本数据类型。

❷ DateTime.Now.Date：获取本地时间的日期部分。

9.11.3　货源信息管理页实现过程

本模块使用的数据表：tb_Freight。

1．设计步骤

（1）在该网站中创建 Manage 文件夹，用于存放网站后台管理窗体。

（2）在该 Manage 文件夹下新建一个 Web 窗体，将其命名为 manage_Freight.aspx，用于货源信息管理。

（3）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（4）在 Table 中添加一个 GridView 控件，用于货源信息管理。

这里给出显示货源信息的 GridView 控件前台绑定代码。实现代码如下：

例程29　代码位置：光盘\TM\09\ WuLiu \manage_Freight.aspx

```
    <asp:GridView ID="gvFreight" runat="server" AutoGenerateColumns="False"
OnRowDataBound="gvFreight_RowDataBound" OnRowDeleting="gvFreight_RowDeleting"
OnSelectedIndexChanging="gvFreight_SelectedIndexChanging" Font-Size="10pt">
        <Columns>
```

```
            <asp:BoundField DataField="Start" HeaderText="出发地" />
            <asp:BoundField DataField="Terminal" HeaderText="到达地" />
            <asp:BoundField DataField="FreightType" HeaderText="货物种类" />
            <asp:BoundField DataField="FBDate" HeaderText="发布日期" DataFormatString="{0:yy-MM-dd}"
HtmlEncode="False" />
             <asp:TemplateField HeaderText="详细信息">
                 <ItemTemplate>
                     <a   href="#" onclick='openPWD("../freightInfo.aspx?ID=<%#Eval("ID")%>",340,370)' >详细信
息</a>
                 </ItemTemplate>
             </asp:TemplateField>
             <asp:BoundField AccessibleHeaderText="sh" DataField="Auditing" HeaderText="状态" />
             <asp:CommandField HeaderText="删除" ShowDeleteButton="True">
                 <ControlStyle Font-Underline="False" />
             </asp:CommandField>
             <asp:CommandField HeaderText="审核" SelectText="通过/取消" ShowSelectButton="True" />
         </Columns>
     </asp:GridView>
```

2．实现代码

在 Web 窗体页面的加载事件中调用自定义方法 bindFreight()将所有的货源信息显示出来。实现代码如下：

例程30　代码位置：光盘\TM\09\ WuLiu \manage_Freight.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        bindFreight();                                  //调用自定义方法显示货源信息
    }
```

bindFreight 自定义方法将所有货源信息绑定到 GridView 控件上，在此方法中先判断管理员选择的是哪种显示方式，根据管理员选择的显示方式来绑定 GridView 控件。实现代码如下：

例程31　代码位置：光盘\TM\09\ WuLiu \manage_Freight.aspx.cs

```
protected void bindFreight()
    {
        string sql = "";
        if (rdibtnW.Checked)                            //判断是否选择未审核显示方式
        {
            sql = "select * from tb_Freight where Auditing=0";
        }
        else
            if (rdibtnY.Checked)                        //判断是否选择已审核显示方式
            {
                sql = "select * from tb_Freight where Auditing=1";
            }
            else
                if (rdibtnS.Checked)                    //判断是否选择所有信息显示方式
                {
                    sql = "select * from tb_Freight";
```

```
                }
            //调用数据库操作类中getDataset方法将数据源绑定到GridView控件上
            gvFreight.DataSource = dataOperate.getDataset(sql, "tb_Freight");
❶           gvFreight.DataKeyNames = new string[] { "ID" };
            gvFreight.DataBind();
        }
```

代码贴士

❶ GridView. DataKeyNames 属性：设置主键字段。

如果货源信息已被审核，此条信息“状态”列将显示为“已审核”且字体颜色为红色，如货源信息未被审核将显示为“未审核”且字体颜色为蓝色。此功能通过 GridView 控件的 RowDataBound 事件实现，此事件在数据绑定后引发。在此事件中判断每条信息的审核状态，根据信息当前的锁定状态来改变“状态”列的显示文本和颜色。实现代码如下：

例程32　代码位置：光盘\TM\09\ WuLiu \manage_Freight.aspx.cs

```
protected void gvFreight_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            //获取货源信息的有效日期
            DateTime term = Convert.ToDateTime(e.Row.Cells[6].Text);
            //获取当前日期
            DateTime nowDate =DateTime.Now.Date;
            if (term < nowDate)                                    //判断此条显示是否过期
            {
                e.Row.ForeColor = Color.Green;                     //如果过期改变此行的颜色
            }
            if (e.Row.Cells[5].Text == "False")                    //判断当前信息的审核状态
            {
                e.Row.Cells[5].Text = "未审核";                    //改变文本值
                e.Row.Cells[5].ForeColor =Color.Red;               //改变显示颜色
            }
            else
            {
                e.Row.Cells[5].Text = "已审核";
                e.Row.Cells[5].ForeColor = Color.Blue;
            }
        }
    }
```

审核信息通过“通过/取消”按钮来实现，当管理员单击某条信息“通过/取消”按钮时，改变审核状态一列的文本和文本颜色。此功能在 GridView 控件中的 SelectedIndexChanging 事件中实现，此事件当选择某行时引发。实现代码如下

例程33　代码位置：光盘\TM\09\ WuLiu \manage_Freight.aspx.cs

```
protected void gvFreight_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
{
```

```
            //获取当前货源信息的编号
            string    ID = this.gvFreight.DataKeys[e.NewSelectedIndex].Value.ToString();
            string selSql="select Auditing from tb_Freight where ID= "+ID;
            //调用数据库操作类中getRow方法并接受此方法返回的SqlDataReader对象
            SqlDataReader sdr = dataOperate.getRow(selSql);
            sdr.Read();                                          //读取下条记录
            int Auditing =Convert.ToInt32(sdr["Auditing"]);      //获取当前信息的审核状态
            if (Auditing == 0)                                   //判断货源信息的审核状态
            {
                Auditing = 1;                                    //改变当前信息的审核状态
            }
            else {
                Auditing = 0;
            }
            string updSql = "update tb_Freight set Auditing='" + Auditing + "' where ID=" + ID;
            dataOperate.execSQL(updSql);                         //将改变后的审核状态存储到数据库中
            bindFreight();                                       //调用自定义方法重新绑定货源信息
        }
```

9.12　企业用户管理页

9.12.1　企业用户管理页概述

管理员可以通过企业用户管理模块对企业用户的相关信息进行删除和锁定操作。企业用户管理页面如图 9.39 所示。

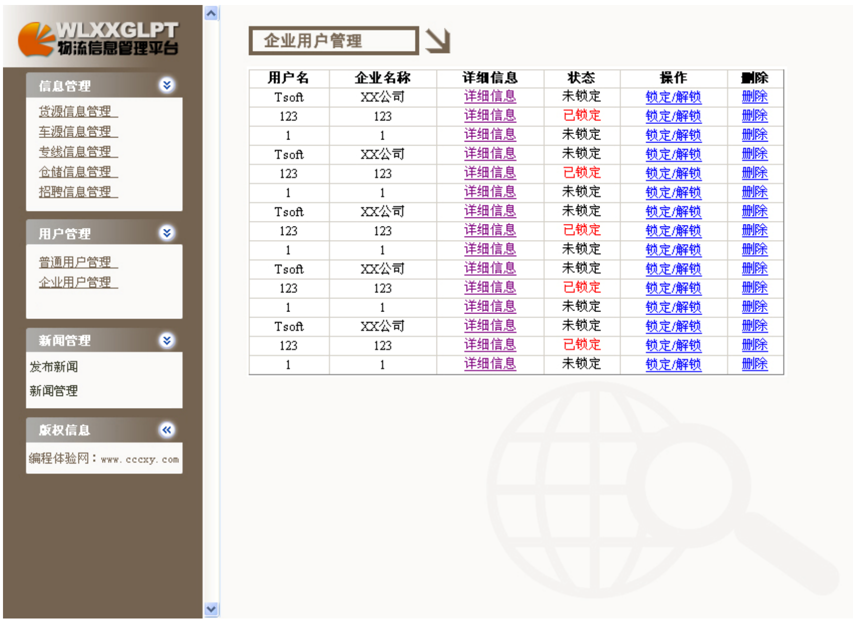

图 9.39　企业用户管理页

9.12.2　企业用户管理页技术分析

当企业用户发布了非法的信息，可以通过锁定此用户进行警告。管理员单击需要锁定的用户后面的“锁定/解锁”链接按钮时会弹出一个新的窗口，在文本框中输入锁定此用户的原因后，单击“关闭”按钮，此用户就会被锁定。当被锁定的用户登录时将会弹出提示对话框，提示此用户被锁定的原因。实现此功能需要通过 GridView 控件中的 SelectedIndexChanging 事件实现，在此事件中先判断用户是否被锁定，如此用户已经被锁定不用实现此操作；如果用户未锁定，通过 JavaScript 中的 Window 对象的 open()方法，打开一个指定大小的新窗口并向此窗口传入表名和当前用户的编号。代码实现如下：

例程34　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx

```
RegisterStartupScript("false",
"<script>window.open('lockCause.aspx?table=tb_User&ID="+ID+"',",'width=310,height=190')</script>");
```

在新窗口中输入锁定此用户的原因，当单击“关闭”按钮时此用户的锁定原因将保存到数据库中。实现代码如下：

例程35　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx.cs

```
protected void Button1_Click(object sender, EventArgs e)
    {
        string table = Request.QueryString["table"].ToString();                    //获取表名
        string id = Request.QueryString["ID"].ToString();                          //获取编号
        string sql = "update " + table + " set LockCause='"+txtCause.Text+"' where ID="+id;
        dataOperate.execSQL(sql);
        Response.Write("<script>window.close();</script>");
    }
```

9.12.3　实现过程

本模块使用的数据表：tb_User。

1．设计步骤

（1）在该网站中创建 Manage 文件夹，用于存放网站后台管理窗体。

（2）在该 Manage 文件夹下新建一个 Web 窗体，将其命名为 manage_qyUser.aspx，用于企业用户管理。

（3）在 Web 窗体中添加 Tabel 表格，用于页面的布局。

（4）在 Table 中添加一个 GridView 控件，用于企业用户管理。

这里给出显示企业用户的 GridView 控件前台绑定代码。实现代码如下：

例程36　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx

```
<asp:GridView ID="gvGrUser" runat="server" AutoGenerateColumns="False"
OnRowDataBound="gvGrUser_RowDataBound" OnRowDeleting="gvGrUser_RowDeleting"
OnSelectedIndexChanging="gvGrUser_SelectedIndexChanging">
    <Columns>
```

```
            <asp:BoundField HeaderText="用户名" DataField="Name" />
            <asp:BoundField DataField="Linkman" HeaderText="联系人" />
            <asp:BoundField HeaderText="状态" DataField="Lock" />
            <asp:BoundField DataField="Phone" HeaderText="联系电话" />
            <asp:BoundField DataField="Address" HeaderText="所在地址" />
            <asp:CommandField HeaderText="操作" SelectText="锁定/解锁" ShowSelectButton="True" />
            <asp:CommandField HeaderText="删除" ShowDeleteButton="True" />
        </Columns>
    </asp:GridView>
```

2．实现代码

在 Web 窗体页面的加载事件中调用自定义方法 bindQyUser()将所有企业用户显示出来。实现代码如下：

例程37　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx.cs

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

bindQyUser 自定义方法将所有企业用户绑定到 GridView 控件上。实现代码如下：

例程38　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx.cs

```
protected void bindQyUser()
 {
     string sql = "select * from tb_User";
     //调用数据库操作类中的getDataset方法将返回值绑定到GridView控件上
     gvQyUser.DataSource = dataOperate.getDataset(sql, "tb_User");
     //设置主键字段
     gvQyUser.DataKeyNames = new string[] { "ID" };
     gvQyUser.DataBind();
 }
```

当管理员想要删除企业用户时，通过使用 GridView 控件的 RowDeleting 事件来实现，此事件在删除前引发。在此事件中通过获取此用户的编号，使用 SQL 语句利用数据库操作类中的 execSQL 方法将此用户删除，如果删除成功将给出相应的提示。实现代码如下：

例程39　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx.cs

```
protected void gvQyUser_RowDeleting(object sender, GridViewDeleteEventArgs e)
    {
        string ID = gvQyUser.DataKeys[e.RowIndex].Value.ToString();                    //获取当前用户编号
        string sql = "delete from tb_User where ID=" + ID;
        if (dataOperate.execSQL(sql))
        {
            Page.RegisterStartupScript("true", "<script>alert('删除成功！')</script>");
        }
        else
        {
            Page.RegisterStartupScript("false", "<script>alert('删除失败！')</script>");
```

```
            }
            bindQyUser();                                                    //重新绑定GridView控件
        }
```

如果用户被锁定，此用户的“状态”列将显示为“已锁定”并且字体颜色为红色。此功能通过 GridView 控件的 RowDataBound 事件实现，此事件在数据绑定后引发。在此事件中首先判断用户的当前锁定状态，根据锁定状态，改变“状态”列中的显示文本和颜色。实现代码如下：

例程40　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx.cs

```
protected void gvQyUser_RowDataBound(object sender, GridViewRowEventArgs e)
    {
        if (e.Row.RowType == DataControlRowType.DataRow)
        {
            if (e.Row.Cells[3].Text == "False")                              //判断用户是否被锁定
            {
               e.Row.Cells[3].Text = "未锁定";                               //设置状态文本
            }
            else
            {
                e.Row.Cells[3].Text = "已锁定";                              //设置状态文本
                e.Row.Cells[3].ForeColor = System.Drawing.Color.Red;         //设置文本显示颜色
            }
        }
    }
```

锁定用户通过“锁定/取消”按钮来实现。当管理员单击某用户“锁定/取消”按钮时，改变当前用户的锁定状态并添加到数据库中，此功能在 GridView 控件中的 SelectedIndexChanging 事件中实现。此事件当选择某行时引发。实现代码如下：

例程41　代码位置：光盘\TM\09\ WuLiu \manage_qyUser.aspx.cs

```
    protected void gvQyUser_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
    {
        string ID = gvQyUser.DataKeys[e.NewSelectedIndex].Value.ToString();      //获取当前用户的编号
        string selSql = "select Lock from tb_User where ID=" + ID;
        //调用数据库操作类中的getRow方法并接受返回的SqlDataReader对象
        SqlDataReader sdr = dataOperate.getRow(selSql);
        sdr.Read();                                                              //前进一条记录
        int Lk = Convert.ToInt32(sdr["Lock"]);
        if (Lk == 0)                                                             //判断此用户是否锁定
        {
            RegisterStartupScript("false",
"<script>window.open('lockCause.aspx?table=tb_User&ID="+ID+"','','width=310,height=190')</script>");
            Lk = 1;
        }
        else
        {
            Lk = 0;
        }
        string updSql = "update tb_User set Lock='" + Lk + "' where ID=" + ID;
```

```
            dataOperate.execSQL(updSql);                                    //更新此用户
            bindQyUser();                                                   //重新绑定GridView控件
        }
```

9.13 网站编译与发布

网站编译与发布与 1.11 节实现过程完全相同，可参见 1.11 节，这里不再详细介绍。

9.14 网站文件清单

物流信息管理平台的网站文件清单如表 9.13 所示。

表 9.13 网站文件清单

文件位置及名称	说　明
WuLiu\App_Code\dataOperate.cs	数据库操作类
WuLiu\ App_Data\ db_WL_Data.MDF	数据库文件
WuLiu\ Bin \ Interop.jmail.dll	引用的Jmail组件
WuLiu\ Manage\ leftMenu.aspx	后台管理左框架
WuLiu\ Manage\ lockCause.aspx	后台管理锁定用户详细信息
WuLiu\ Manage\ manage_Depot.aspx	后台仓储信息管理页
WuLiu\ Manage\ manage_Freight.aspx	后台货源信息管理页
WuLiu\ Manage\ manage_grUser.aspx	后台个人用户管理页
WuLiu\ Manage\ manage_issuanceNews.aspx	后台发布物流信息页
WuLiu\ Manage\ manage_Job.aspx	后台招聘信息管理页
WuLiu\ Manage\ manage_news.aspx	后台物流新闻管理页
WuLiu\ Manage\ manage_qyUser.aspx	后台企业用户管理页
WuLiu\ Manage\ manage_Special.aspx	后台专线信息管理页
WuLiu\ Manage\ manage_Truck.aspx	后台车源信息管理页
WuLiu\ Manage\ manageIndex.aspx	后台管理首页
WuLiu\ Manage\ rightMenu.aspx	后台管理右框架
WuLiu\ companyInfo.aspx	企业详细信息页
WuLiu\ companyList.aspx	所有企业信息页
WuLiu\ depotInfo.aspx	仓储详细信息页
WuLiu\ depotList.aspx	所有仓储信息页
WuLiu\ entry.ascx	登录用户控件
WuLiu\ freightInfo.aspx	货源详细信息页

续表

文件位置及名称	说　　明
WuLiu\ freightList.aspx	所有货源信息页
WuLiu\ grLeft.aspx	个人用户发布信息左框架
WuLiu\ header.ascx	网站导航用户控件
WuLiu\ index.aspx	网站首页
WuLiu\ issuanceDepot.aspx	发布仓储信息页
WuLiu\ issuanceFreight.aspx	发布货源信息页
WuLiu\ issuanceInfo.aspx	发布信息首页
WuLiu\ issuanceJob.aspx	发布招聘信息页
WuLiu\ issuanceSpecial.aspx	发布专线信息页
WuLiu\ issuanceTruck.aspx	发布车源信息页
WuLiu\ jobInfo.aspx	招聘详细信息页
WuLiu\ jobList.aspx	所有招聘信息页
WuLiu\ login.aspx	用户注册页
WuLiu\ manageEntry.aspx	后台登录页
WuLiu\ MasterPage.master	母版页
WuLiu\ newPass.aspx	修改个人信息页
WuLiu\news.aspx	新闻详细信息页
WuLiu\ qyLeft.aspx	企业用户发布信息左框架
WuLiu\ search.ascx	搜索功能用户控件
WuLiu\ searchList.aspx	搜索详细信息页
WuLiu\ specialInfo.aspx	专线详细信息页
WuLiu\ specialList.aspx	所有专线信息页
WuLiu\ truckInfo.aspx	车源详细信息页
WuLiu\ truckList.aspx	所有车源信息页

注意：上面的文件列表清单中，凡是与 ASPX 文件对应的都有一个 CS 文件，在此没有一一列出。

9.15　邮件发送

在本程序用户注册中使用了邮件发送功能。在开发电子邮件发送功能时，主要使用 Jmail 组件发送电子邮件。因为使用 Jmail 组件不需要书写大量的代码，却能实现非常完美的功能。在使用 Jmail 组件的同时，需要特别强调的是在使用过程中要将此组件引用到项目当中，而且还要在本地计算机上注册该组件。

9.15.1 Jmail 组件介绍

Jmail 组件是由 Dimac 公司开发，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，如图 9.40 所示。

（2）在打开的“添加引用”对话框（如图 9.41 所示）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了。

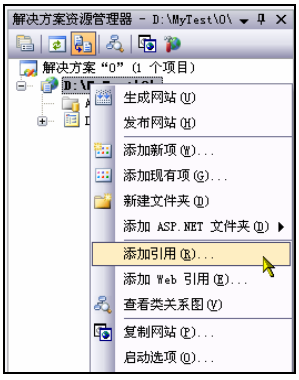

图 9.40 在项目中添加引用

图 9.41 “添加引用”对话框

注意：Jmail 组件不是 ASP.NET 2.0 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。

9.15.2 邮件发送的实现

发送功能主要通过 Jmail.. MessageClass 类中几个主要的属性和方法来实现。Jmail.. MessageClass 类中的主要属性如表 9.14 所示。

表 9.14 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，默认为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

续表

属　　性	说　　明
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

☑ Prority 属性

此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 9.15 所示。

表 9.15　Jmail.. MessageClass 类中的方法及说明

方　　法	说　　明
AddRecipient(emailAddress,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

☑ Send 方法

此方法用来发送邮件，一般情况下只使用服务器参数即可。用法如下：

```
Message.Send(server)
```

9.16　本章小结

本章主要的内容是根据物流信息的实际情况设计一个物流信息管理平台。在开发过程中，首要考虑的问题就是系统的需求分析以及如何设计数据库，因为数据库设计直接影响到管理系统的好坏；然后考虑公共类的编写，一个好的公共类不但可以提高开发速度，还有利于系统的维护。本章通过详细的讲解以及简洁的代码使读者能够更快、更好地掌握物流信息管理平台开发技术。