第 3 章

小区物业内部管理网
（ASP.NET 2.0+Access 实现）

随着我国市场经济的快速发展和人们生活水平的不断提高，简单的社区服务已不能满足人们的需求。如何利用先进的管理手段提高物业的管理水平，是当今社会所面临的一个重要课题。要想提高物业的管理水平，必须全方位地提高物业管理意识。只有高标准、高质量的社区服务才能满足人们的需求。面对信息时代的挑战，利用计算机网络管理系统来提高物业管理无疑是一条行之有效的途径。物业管理方面的信息化与科学化，已成为现代化生活水平步入高台阶的重要标志。通过本章学习，你将学到：

- 小区物业内部管理网开发的基本过程
- 系统设计的方法
- 如何分析并设计数据库
- 如何设计公共类
- 主要功能模块的实现方法
- Web 用户控件在网站中的应用

3.1 开发背景

随着市场经济的发展和人们生活水平的提高，住宅小区越来越成为居住的主流，小区物业管理正是针对当代社会这一市场需要应运而生的。人们在对居住条件的要求越来越高的同时，对小区物业的服务和管理水平的要求也相应提高。伴随着小区的规模不断扩大和业主的不断增多，物业管理的工作量越来越大，也更复杂。为了适应这种形势，要求物业管理者在不断完善小区的各项硬件条件的同时，还要配备现代化的物业管理系统。

3.2 需求分析

物业管理在手工操作时代，工作程序非常繁琐，需要大量的人力、物力和财力，极大地浪费了小区物业的资源，不能及时了解业主的各项需要、提供细致周到的服务使物业管理水平大打折扣。小区物业管理系统是现代居住小区不可缺少的一部分。一个好的小区物业管理系统可以提升小区的管理水平，使小区的日常管理更加方便，同时确保取得最大的经济效益。将计算机的强大功能与现代的管理思想相结合，建立现代的智能小区是物业管理发展的方向。

3.3 系统设计

3.3.1 系统目标

小区物业内部管理网属于中小型网络管理系统，可以对中小型物业进行有效管理。通过本系统可以达到以下目标：

- ☑ 系统采用人机对话方式，界面美观友好，操作方便。
- ☑ 业主可以及时获得物业公告及小区新闻。
- ☑ 业主可以随时了解各项费用的欠费情况。
- ☑ 实现业主在线投诉及设备报修，并提供及时周到的服务。
- ☑ 智能化管理业主车位及车费的缴纳情况。
- ☑ 全方位的数据查询，提高工作效率。
- ☑ 完整的数据及服务档案管理提高管理水平。
- ☑ 完善的权限管理，提高系统的安全性。
- ☑ 系统最大限度地实现了易安装性、易维护性和易操作性。

3.3.2　系统功能结构

根据小区物业内部管理的特点，可以将小区物业内部管理信息网分为前台和后台两个部分进行设计。前台主要用于用户注册和登录网站、站内链接、查询用户的相关信息以及查看站内新闻和站内公告；后台主要用于管理员对业主投诉、业主报修、反馈信息受理、物业设备维修、业主停车位、业主费用、费用催缴和值班员工信息等进行管理。

详细规划网站的功能，给出系统功能结构图，前台功能结构图如图 3.1 所示。

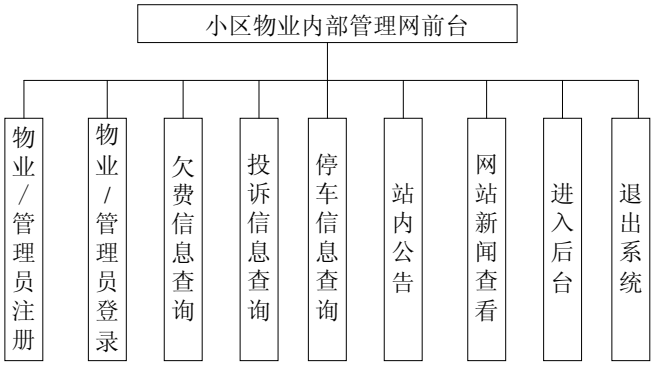

图 3.1　小区物业内部管理网前台功能结构图

小区物业内部管理网的后台功能结构，如图 3.2 所示。

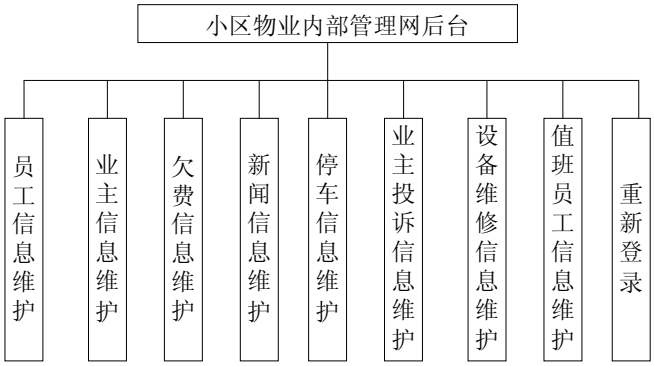

图 3.2　小区物业内部管理网后台功能结构图

3.3.3　业务流程图

为了读者能够更好地学习，下面给出小区物业内部管理网的网站流程概述及网站流程图。

业主在小区物业内部管理网上通过注册成为网站用户，登录网站进行相关操作。业主登录后，可以进行在线查询欠费信息、业主投诉信息查询、停车信息查询、设备维护信息查询、新闻信息查询和站内公告信息查看的操作。网站的管理员通过登录模块可以登录到网站的后台系统管理员主页面，对业主信息、欠费信息、停车信息、新闻信息、投诉信息和值班员工信息等进行管理。网站业务流程图

如图 3.3 所示。

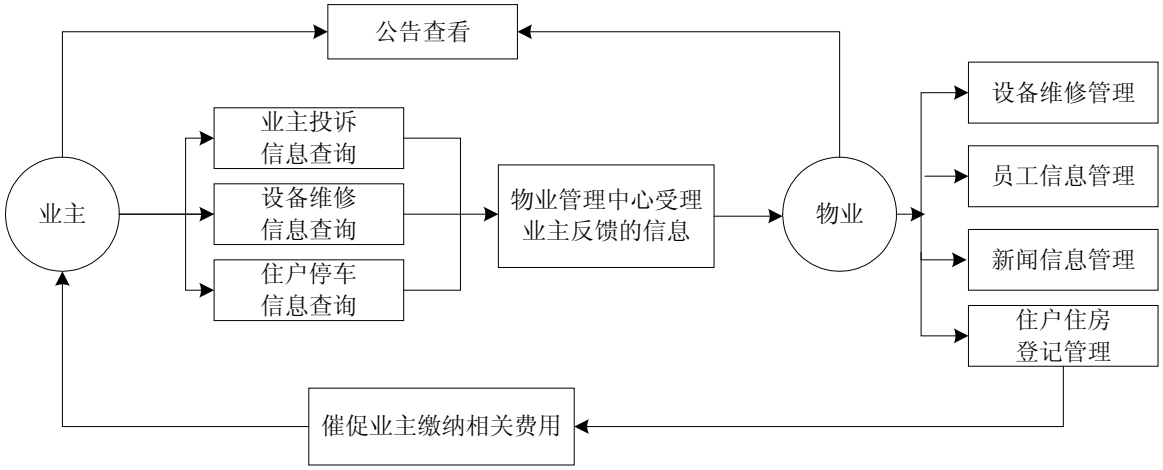

图 3.3　业务流程图

3.3.4　系统预览

小区物业内部管理信息网由多个页面组成，下面仅列出几个典型页面，其他页面参见光盘中的源程序。

网站主页如图 3.4 所示，主要实现用户注册和登录网站、站内链接、查询用户的相关信息以及查看站内新闻和站内公告等功能。小区投诉信息查询页如图 3.5 所示，主要按照编号、状态、投诉状态来查询相应的信息。

图 3.4　网站主页（光盘\…\Default.aspx）

图 3.5　投诉信息查询页（光盘\…\ tousuxinxi\ tousucx.aspx）

后台管理页面如图 3.6 所示，主要实现业主投诉管理、业主报修管理、反馈信息受理、物业设备维修、业主停车位管理、业主费用管理、费用催缴管理和值班员工信息管理等功能。网站后台管理员登录页面如图 3.7 所示，主要实现管理员登录功能。

图 3.6　后台管理页面（光盘\…\ tousuxinxi\ tousushenhe.aspx）　图 3.7　管理员登录页（光盘\…\ guanliyuan\denglu.aspx）

注意：由于路径太长，因此省略了部分路径，省略的路径是“TM\03\物业\WebSite3”。

3.3.5　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：Access。
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：Access。
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。

3.3.6　数据库设计

本系统采用 Access 数据库，名称为 db_wygl，其中包含 9 张表。下面分别给出数据表概要说明、

数据表 E-R 图及主要数据表结构。

1．数据库概要说明

从读者角度出发，为了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形结构图，如图 3.8 所示，其中包含了对系统中所有数据表的相关描述。

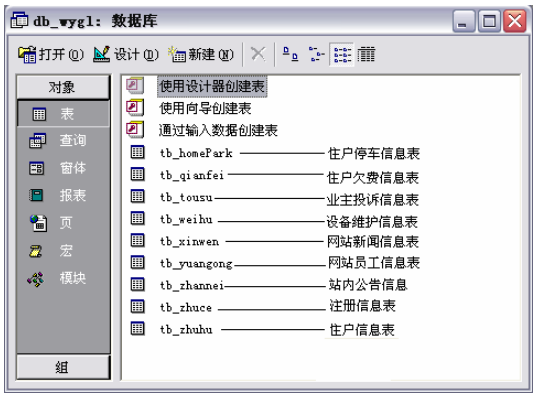

图 3.8　数据表树形结构图

2．数据库概念设计

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业主欠费信息实体、设备维修信息实体、新闻信息实体、住户信息实体、业主投诉信息实体。

业主欠费信息实体 E-R 图如图 3.9 所示。

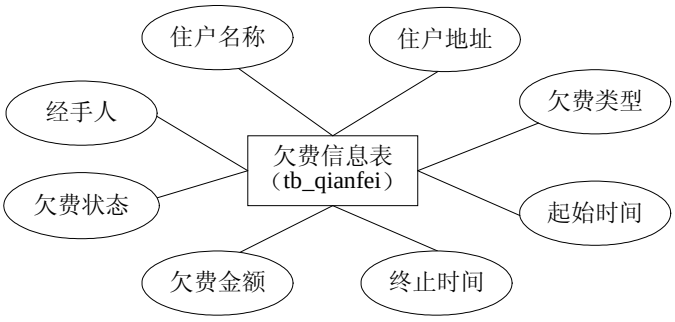

图 3.9　业主欠费信息实体 E-R 图

设备维护信息实体 E-R 图如图 3.10 所示。

新闻信息实体 E-R 图如图 3.11 所示。

住户信息实体 E-R 图如图 3.12 所示。

业主投诉信息实体 E-R 图如图 3.13 所示。

3．数据库逻辑结构设计

在设计完数据库实体 E-R 图之后，下面将根据实体 E-R 图设计数据表结构。下面以创建 tb_homePark（业主停车信息表）为例演示如何根据 E-R 图创建数据表。具体操作如下：

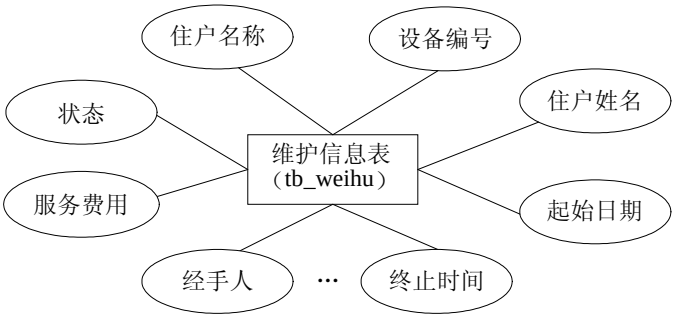

图 3.10　设备维护息实体 E-R 图

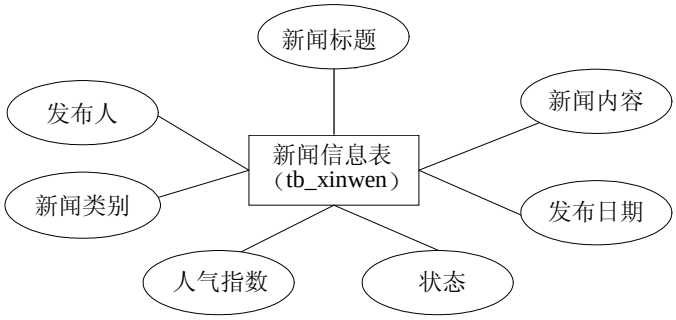

图 3.11　新闻信息实体 E-R 图

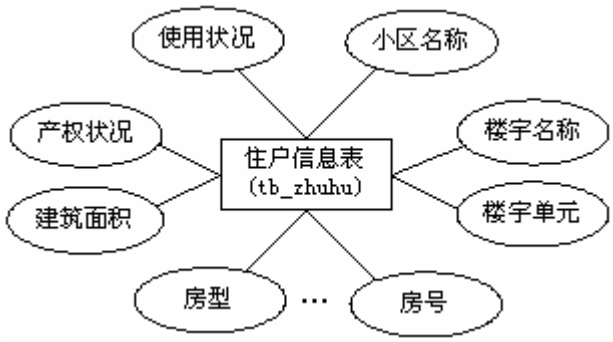

图 3.12　住户信息实体 E-R 图

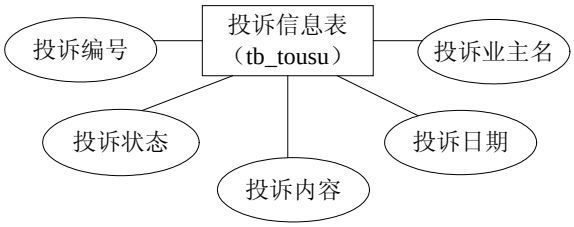

图 3.13　业主投诉信息实体 E-R 图

（1）运行 Microsoft Access 2000，打开如图 3.14 所示的对话框，选中“空 Access 数据库”单选按钮。

（2）单击“确定”按钮，打开“文件新建数据库”对话框，从中选择数据库存放的位置，并将数

据库命名为 db_wygl，如图 3.15 所示。

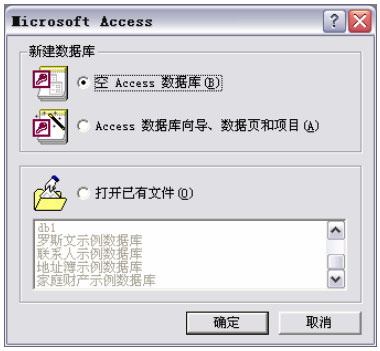

图 3.14　创建一个空 Access 数据库

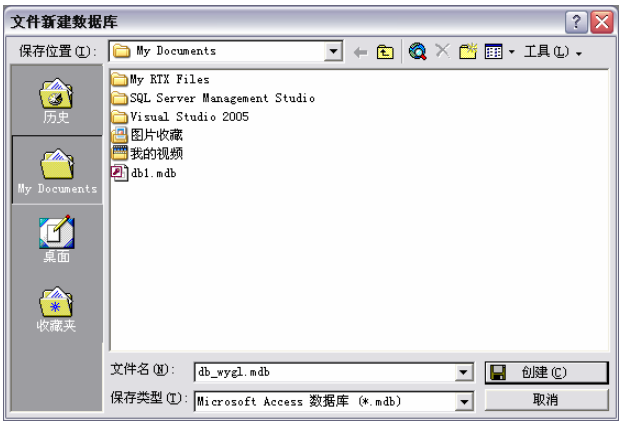

图 3.15　“文件新建数据库”对话框

（3）单击“创建”按钮，打开“db_wygl：数据库”对话框，右击“使用设计器创建表”，在弹出的快捷菜单中选择“设计视图”命令，如图 3.16 所示。

（4）在打开的“tb_homePark：表”窗口中根据业主停车信息实体 E-R 图，设置“字段名称”、“数据类型”及“说明”，在窗口下方的“常规”选项卡中，可以对字段进行详细设置，如图 3.17 所示。

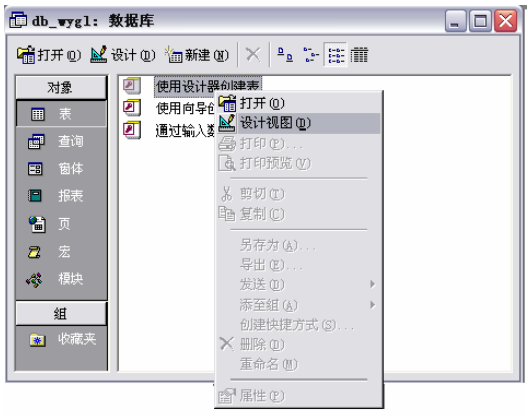

图 3.16　选择“设计视图”命令

图 3.17　“tb_homePark：表”窗口

（5）单击窗口右上角的“关闭”按钮，弹出提示是否保存数据表的对话框，单击“是”按钮，弹出“另存为”对话框，在“表名称”文本框中输入表的名称“tb_homePark”，如图 3.18 所示。单击“确定”按钮完成数据表的创建。

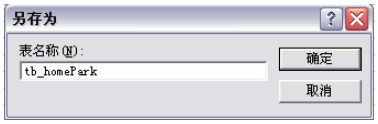

图 3.18　“另存为”对话框

重复以上操作，创建其他数据表。由于篇幅所限，笔者在此只给出较重要的数据表结构，其他数

据表可参见本书附带的光盘。

☑　tb_qianfei（业主欠费信息表）

业主欠费信息表主要存储用户欠费的相关信息，如表 3.1 所示。

表 3.1　欠费信息表结构

字　段　名	数 据 类 型	默　认　值	必 填 字 段	说　　明
name	文本	无	是	欠费业主名称
address	文本	无	是	欠费业主地址
leibie	文本	无	是	欠费类型
stime	日期/时间	无	是	欠费的起始日期
etime	日期/时间	无	是	欠费的终止日期
qianfei	文本	无	是	欠费金额
state	文本	无	是	欠费状态（已使用、未使用）
person	文本	无	是	经手人

☑　tb_tousu（业主投诉信息表）

业主投诉信息表主要存储用户投诉的相关信息，如表 3.2 所示。

表 3.2　业主投诉信息表结构

字　段　名	数 据 类 型	默　认　值	必 填 字 段	说　　明
id	文本	无	是	投诉编号
person	文本	无	是	投诉业主名
time	日期/时间	无	是	投诉日期
neirong	文本	无	是	投诉内容
zhuangtai	文本	无	是	投诉状态（未处理、已处理）

☑　tb_weihu（设备维护信息表）

设备维护信息表主要存储设备维修的相关信息，如表 3.3 所示。

表 3.3　设备维护信息表结构

字　段　名	数 据 类 型	默　认　值	必 填 字 段	说　　明
id	文本	无	是	设备维修编号
name	文本	无	是	业主名称
stime	日期/时间	无	是	起始日期
etime	日期/时间	无	是	终止日期
person	文本	无	是	经手人
feiyong	文本	无	是	服务费用
state	文本	无	是	备注
neirong	备注	无	是	维护内容

☑ tb_xinwen（网站新闻信息表）

网站新闻信息表主要存储网站新闻的相关信息，如表 3.4 所示。

表 3.4 网站新闻信息表结构

字 段 名	数 据 类 型	默 认 值	必 填 字 段	说 明
title	文本	无	是	新闻标题
neirong	文本	无	是	新闻内容
fabushijian	日期/时间	无	是	发布日期
faburen	文本	无	是	发布人
xinwenleibie	文本	无	是	新闻类别
renqi	文本	无	是	人气指数
state	文本	无	是	状态（已处理、未处理）

3.3.7 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，如图 3.19 所示。

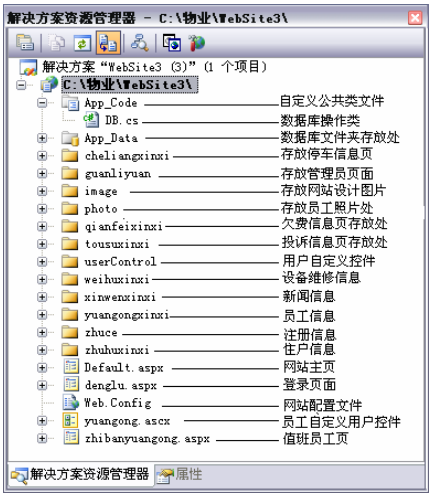

图 3.19 网站文件组织结构图

3.4 公共类设计

在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。本系统中创建了一个公共类——DB 类，主要用来访问 Microsoft Access 数据库，数据库操作类主要的功能是连接数据库。

首先在命名空间区域引用 using System.Data.OleDb 命名空间。代码如下：

```
using System.Data.OleDb;
```

小区物业内部管理网中采用 Access 数据库，下面给出连接 Access 数据库的字符串。具体代码如下：

```
/// <summary>
/// DB 的摘要说明
/// </summary>
public class DB
{
    public DB()
    {
        //
        // TODO: 在此处添加构造函数逻辑
        //
    }
    public static OleDbConnection createDB()      //声明一个用户自定义的方法
    {
        OleDbConnection odbc = new OleDbConnection("Provider=Microsoft.Jet.OLEDB.4.0;Data
source=|DataDirectory|db_物业\WebSite3gl.mdb;");      //连接Access数据库
        return odbc;                                  //返回一个OleDbConnection对象类型的对象
    }
}
```

3.5　网站首页设计

3.5.1　网站首页概述

网站首页的设计效果关系到网站的建设及形象宣传，它对网站的生存和发展起着非常重要的作用。网站首页应该是一个信息含量较高、内容较丰富的宣传平台，主要包含以下内容：

- ☑ 网站菜单导航。
- ☑ 信息快速查询。
- ☑ 用户注册及登录。
- ☑ 站内公告显示。
- ☑ 网站新闻显示，按时间先后顺序显示新闻信息。
- ☑ 后台登录入口：为管理员进入后台提供一个入口。

小区物业内部管理信息网主页如图 3.20 所示。

3.5.2　网站首页技术分析

小区物业内部管理网的主页主要使用了用户控件技术。用户控件基本的应用就是把网页中经常用到的且使用频率较高的程序封装到一个模块中，以便在其他页面中重复使用，以此提高代码的重用性和程序开发的效率。用户控件的应用始终融会着一个高层的设计思想，即“模块化设计，模块化应用”的原则。

图 3.20　小区物业内部管理网主页

用户控件的好处是更好地解决 UI（用户界面）的复杂问题，创建完成用户控件后，不需要预编译。当用户在.aspx 页面中引用用户控件时，页面解析器将从.aspx 文件中动态地生成一个页面并将其编译到一个配件中（托管的库中），这种页面式的创建方式和不用编译的部署方式使得用户控件的开发变得极为容易。用户控件在小区物业内部管理网内页面中频繁地使用，从而提升了整个程序开发的效率。本节将从以下几个方面来介绍用户控件。

1．用户控件的概念

用户控件是一种服务器控件，它与 ASP.NET 页面有着同样的“所见即所得”的特点和声明性样式，并以.ascx 为扩展名存储为文本文件。Web 用户控件与完整的 ASP.NET 网页（即.aspx 文件）非常相似，同时具有自己的用户界面页和代码。开发人员可以采取与创建 ASP.NET 页相似的方式创建 Web 用户控件，然后向其中添加所需的标记和子控件。Web 用户控件可以像页面一样包含对其内容进行操作（包括执行数据绑定等任务）的代码。

说明：ASP.NET 用户控件的创建方式很灵活，可以通过创建以.ascx 为扩展名的文本文件进行创建，也可以以编程方式动态创建，还可以通过修改 Web 窗体页（.aspx 文件），将其转换为 ASP.NET 用户控件。

2．创建 Web 用户控件

创建普通用户控件的基本步骤如下：

（1）创建一个新文件并为其指定扩展名为.ascx。

（2）在该页面的顶部创建一个@Control 指令，并指定要为控件（如果有）使用的编程语言。

（3）添加希望用户控件显示的控件。

（4）添加用户控件要执行的任务（如处理控件事件或从数据源读取数据）的代码。

（5）如果希望在用户控件和宿主页之间共享信息，需要在控件中创建相应的属性。根据需要可以创建任何类的属性。

注意：不能将用户控件放入网站的 App_Code 文件夹中，如果某个用户控件在 App_Code 文件夹中，则运行包含该控件的页面时将发生分析错误。另外，用户控件属于 System.Web.UI.UserContol 类型，它直接继承于 System.Web.UI.Control。

下面介绍在 Visual Studio 2005 中手动创建一个用户控件的过程。主要操作步骤如下：

（1）打开解决方案资源管理器，右击项目名称，在弹出的快捷菜单中选择“添加新项”命令，将会弹出如图 3.21 所示的“添加新项”对话框。在该对话框中，选择“Web 用户控件”项，并为其命名，然后单击“添加”按钮即可将 Web 用户控件添加到项目中。

图 3.21　“添加新项”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 文件中可以直接向页面中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多次，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，将用户控件拖曳到网页上，然后释放鼠标左键即可，如图 3.22 所示。

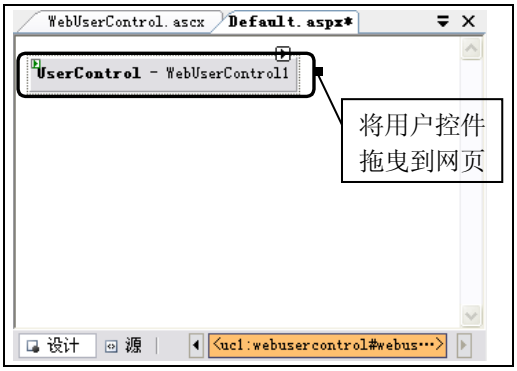

图 3.22　将 Web 用户控件添加至网页

（3）在已添加的用户控件上，单击鼠标右键，在弹出的快捷菜单中选择“属性”命令，打开“属性”窗口，如图 3.23 所示，用户可以在“属性”窗口中修改用户控件的属性。

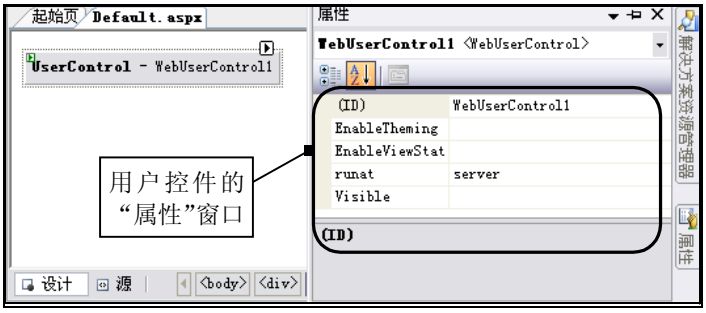

图 3.23　用户控件的“属性”窗口

4．Web 用户控件的优点

通过 Web 用户控件，可以很好地重用公共用户界面（UI）功能，从而提升程序开发效率。

用户控件提供了一个面向对象的编程模型，在一定程度上取代了服务器端文件包含（<!--#include-->）指令，并且提供的功能比服务器端包含文件提供的功能更多。使用用户控件的主要优点如下：

- ☑ 可以将常用的内容或者控件以及控件的运行程序逻辑，设计为用户控件，然后便可以在多个网页中重复使用该用户控件，从而省却许多重复性的工作。例如网页上的导航栏，几乎每个页都需要相同的导航栏，这时便可以将其设计为一个用户控件，在多个页中使用。
- ☑ 如果网页内容需要改变，只需修改用户控件中的内容，其他添加使用该用户控件的网页会自动随之改变，因此网页的设计以及维护变得简单易行。

3.5.3　网站首页实现过程

本模块使用的数据表：tb_xinwen、tb_zhannei、tb_qianfei、tb_zhuce。

1．设计步骤

下面对网站首页的制作过程进行详细的讲解。具体步骤如下：

（1）在该网站中创建一个 Web 窗体，将其命名为 Default.aspx，用于显示网站首页的相关信息。

（2）在 Web 窗体中添加一个 Table 表格，用于页面的布局。

（3）在 Table1 表格中添加相关服务器控件，属性设置及其用途如表 3.5 所示。

表 3.5　网站首页中用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
TextBox	txtyhname	无	输入登录用户名
	txtyhmm	PasswordChar属性设置为“*”	输入登录用户密码
Button	btnDl	Text属性设置为“登录”	登录
	btnZc	Text属性设置为“注册”	注册
GridView	gvXw	无	显示网站新闻信息
	gvQf	无	显示业主欠费信息
Calendar	CalRq	无	显示当前的年、月、日
LinkButton	lbgly	Text属性设置为“进入后台”	进入后台

2．实现代码

首先导入必要的命名空间 using System.Data.OleDb。引入命名空间的代码如下：

```
using System.Data.OleDb;
```

在 page_load 单击事件中，分别调用 xinwenfillgv 自定义方法、fillguogao 自定义方法、qiangfeifillgv 自定义方法，用于显示网站新闻、站内公告、欠费用户通知。实现代码如下：

例程01　代码位置：光盘\TM\03\物业\ WebSite3\ Default.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
 {
      this.xinwenfillgv();                    //调用用户自定义的xinwenfillgv()方法
      this.fillgougao();                      //调用用户自定义的fillgougao()方法
      this.qianfeifillgv();                   //调用用户自定义的qianfeifillgv()方法
 }
```

在 xinwenfillgv 自定义方法中，主要用来实现网站新闻的显示，在此方法内主要通过 OleDbDataAdapter 对象获取网站新闻信息并将信息填充到数据集中。实现代码如下：

例程02　代码位置：光盘\TM\03\物业\ WebSite3\ Default.aspx.cs

```
    public void xinwenfillgv()                //用户自定义的方法
    {
        //调用用户自定义的DB类的createDB()的方法
        OleDbConnection con = DB.createDB();
```

```
            //声明一个OleDbCommand的对象，并将该对象cmd实例化
            OleDbCommand cmd = new OleDbCommand();
        cmd.Connection = con;
        con.Open();//打开数据库连接
        //从新闻表tb_xinwen按发布时间fabushijian的降序排序
        OleDbDataAdapter sda = new OleDbDataAdapter("select top 9 * from tb_xinwen order by fabushijian desc", con);
         DataSet ds = new DataSet();                    //声明一个DataSet的对象，并将该对象ds实例化
         sda.Fill(ds, "tb_xinwen");                     //将新闻表tb_xinwen填充到数据库中
         this.gvXw.DataSource = ds;
         this.gvXw.DataBind();                          //将数据绑定到GridView控件中
         for (int i = 0; i <= gvXw.Rows.Count - 1; i++)
         {
              DataRowView mydrv;                        //声明一个DataRowView的对象
              string gintro;                            //声明一个变量
              if (this.gvXw.PageIndex == 0)             //判断GridView控件中是否分页
              {
                   mydrv = ds.Tables["tb_xinwen"].DefaultView[i];
                   gintro = Convert.ToString(mydrv["title"]);
                   this.gvXw.Rows[i].Cells[0].Text = SubStr(gintro, 10);          //截取GridView控件单元格中的数据
              }
              else
              {
                   mydrv = ds.Tables["tb_xinwen"].DefaultView[i + (5 * gvXw.PageIndex)];
                   gintro = Convert.ToString(mydrv["title"]);                     //将数据转换为字符串类型
                        this.gvXw.Rows[i].Cells[0].Text = SubStr(gintro, 10);  //截取GridView控件单元格中的数据
                   }
              }
              con.Close();                              //关闭数据库连接
         }
```

在 fillgougao 自定义方法中，主要用来显示站内公告。在此方法中使用 OleDbDataReader 对象获取公告信息，并将公告信息绑定到 Label 标签上。实现代码如下：

例程03 代码位置：光盘\TM\03\物业\ WebSite3\ Default.aspx.cs

```
public void fillgougao()
    {
        //调用用户自定义的DB类的createDB()方法
❶       OleDbConnection con = DB.createDB();
        OleDbCommand cmd = new OleDbCommand(" select * from tb_zhannei", con);
❷       con.Open();//打开数据库连接
❸       OleDbDataReader sdr = cmd.ExecuteReader();
        //判断是否有数据
❹       sdr.Read();
        //从数据库中读取第一个数据放到Label标签中
        this.lblGgxx.Text = sdr.GetString(0);
        //关闭阅读器
❺       sdr.Close();
        //关闭数据库连接
        con.Close();
    }
```

代码贴士

❶ 自定义 createDB 方法：用于连接数据库。
❷ Open 方法：打开 Access 数据库。
❸ OleDbDataReader：提供从数据源读取数据行的只进流的方法。
❹ Read 方法：从给定的消息中读取内容，并创建包含反序列化消息的对象。
❺ Close 方法：关闭数据库的连接。

在 qianfeifillgv 自定义方法中，主要用来显示欠费通知。在此方法中使用 OleDbDataAdapter 对象获取用户欠费信息并将此信息填充到数据集中，将数据集中的信息绑定到 GridView 控件上将用户欠费信息显示出来。实现代码如下：

例程04　代码位置：光盘\TM\03\物业\WebSite3\ Default.aspx.cs

```
public void qianfeifillgv()
 {
❶   OleDbConnection con = DB.createDB();
❷   con.Open(); //打开数据库连接
    //调用用户自定义的DB类的createDB()方法
❸   OleDbCommand cmd = new OleDbCommand();
❹   OleDbDataAdapter sda = new OleDbDataAdapter("select top 10 * from tb_qianfei", con);
    //声明一个DataSet的ds对象，并将其实例化
❺   DataSet ds = new DataSet();
❻   sda.Fill(ds, "tb_qianfei");
❼   this.gvQf.DataSource = ds;
    //将数据绑定到GridView控件中
❽   this.gvQf.DataBind();
   //关闭数据库连接
❾   con.Close();
 }
```

代码贴士

❶ 自定义 createDB 方法：用于连接数据库。
❷ Open 方法：打开 Access 数据库。
❸ OleDbCommand：表示要对数据源执行的 SQL 语句或存储过程。
❹ OleDbDataAdapter：提供从数据源读取数据行的只进流的方法。
❺ DataSet 类：表示数据在内存中的缓存。
❻ Fill 方法：在 DataSet 中添加或刷新行以匹配使用 DataSet 名称的数据源中的行，并创建一个 DataTable。
❼ DataSource 属性：获取或设置对象，数据绑定控件从该对象中检索其数据项列表。
❽ DataBind 方法：将数据源绑定到被调用的服务器控件及其所有子控件。
❾ Close：关闭当前解决方案。

3.5.4　单元测试

在程序运行中，往往出现“应用程序中的服务器错误”，提示为“找不到类型或命名空间名称‘OleDbConnection’（是否缺少 using 指令或程序集引用）”，如图 3.24 所示。

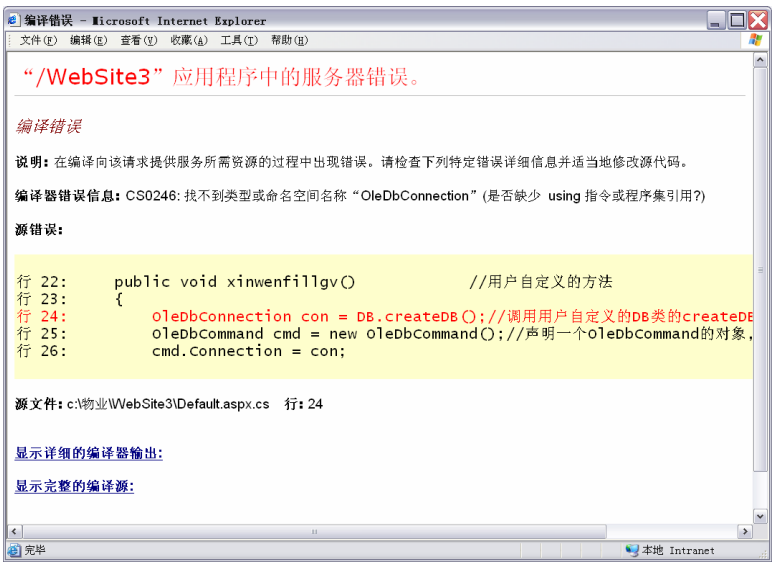

图 3.24　应用程序中的服务器错误

产生此类错误的主要原因，是由于在代码中用到了用户自定义控件中的类、属性及方法。解决办法很简单，只要在编写代码之前导入该自定义控件的命名空间即可。添加的代码如下：

```
using System.Data.OleDb;
```

3.6　欠费信息查询页

3.6.1　欠费信息查询页概述

任何用户都可以查询自己的相关费用（水费、电费、煤气费等）信息，如果用户查询到自己使用的水、电、煤气已欠费，则可将自己所欠的费用及时缴纳上。改进了以往物业人员挨家挨户上门通知的状况，大大提高了物业工作人员的工作效率。欠费信息查询页如图 3.25 所示。

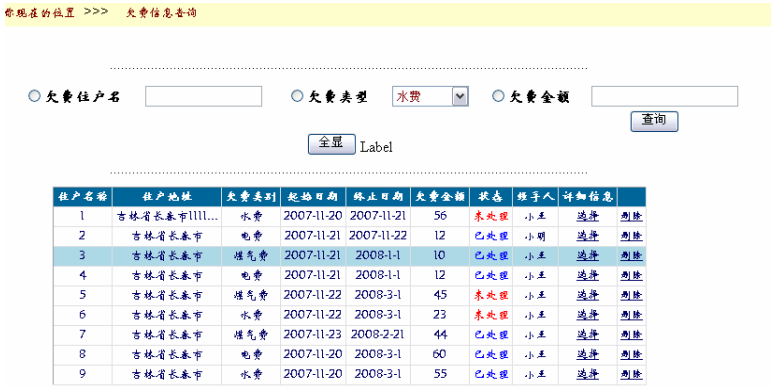

图 3.25　欠费信息查询页

3.6.2　欠费信息查询页技术分析

小区物业欠费信息查询页中，使用了 SELECT 语句，根据用户不同的需求，选择不同的 RadioButton 按钮，实现对“欠费住户名称”、“欠费类别”、“欠费金额”等进行不同的查询操作。小区物业欠费信息查询页中用到的代码如下：

例程05　代码位置：光盘\TM\03\物业\WebSite3\ qianfeixinxi \zhuhurzshenhe.aspx.cs

```
OleDbConnection con = DB.createDB();                    //调用用户自定义的DB类的createDB()方法
OleDbCommand cmd = new OleDbCommand();                  //声明一个OleDbCommand的 cmd对象，并将其实例化
cmd.Connection = con;                                   //设置cmd的连接属性
con.Open();                                             //打开与数据库的连接
DataSet ds = new DataSet();                             //声明一个DataSet的ds对象，并将其实例化
if (this.rbtnqfzh.Checked == true)                      //按欠费住户名查询
{
    OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_qianfei where name='" +
this.txtqfzh.Text + "'", con);
    sda.Fill(ds, "tb_qianfei");
    this.gvqf.DataSource = ds;
    this.gvqf.DataBind();                               //将数据绑定到GridView控件中
}
if (this.rbtnlx.Checked == true)                        //按欠费类型查询
{
    OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_qianfei where leibie='" +
this.ddllx.SelectedItem.Text.ToString() + "'", con);
    sda.Fill(ds, "tb_qianfei");
    this.gvqf.DataSource = ds;
    this.gvqf.DataBind();                               //将数据绑定到GridView控件中
}
if (this.rbtnqfje.Checked == true)                      //按欠费金额查询
{
    OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_qianfei where qianfei='" +
this.txtqfje.Text.ToString() + "'", con);
    sda.Fill(ds, "tb_qianfei");
    this.gvqf.DataSource = ds;
    this.gvqf.DataBind();//将数据绑定到GridView控件中
}
```

3.6.3　欠费信息查询页实现过程

本模块使用的数据表：tb_qianfei。

1．设计步骤

（1）在该网站中创建 qianfeixinxi 文件夹，用于存放基本信息 Web 窗体。

（2）在 qianfeixinxi 文件夹下创建一个 Web 窗体，将其命名为 qianfeicx.aspx，用于业主欠费信息的查询。

（3）在 Web 窗体中添加一个 Table 表格，用于页面的布局。

（4）在 Table1 表格中添加相关服务器控件，属性设置及其用途如表 3.6 所示。

表 3.6　欠费信息查询页面中用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
TextBox	txtxqmc	无	输入小区名称
	txtlymc	无	输入楼宇名称
Button	btnAdd	Text属性设置为“添加”	添加
	btnDelete	Text属性设置为“删除”	删除
GridView	gvzhzf	AllowPaging属性设置为True（允许分页） AutoGenerateColumns属性设置为False（取消自动生成） PageSize属性设置为5（每页显示5条数据）	显示业主住房信息

2．实现代码

在 page_Load 事件中判断管理员是否已经登录，如果 Session["name"]和 Session["pwd"]的值为空，说明管理员没有登录，否则说明管理员已经成功登录。

例程06　代码位置：光盘\TM\03\物业\WebSite3\ qianfeixinxi \qianfeicx.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["nameyh"] == null && Session["pwdyh"] == null)    //判断用户是否登录
        {
            if (Session["name"] == null && Session["pwd"] == null)    //判断管理员是否登录
            {
                Response.Redirect("~/denglu.aspx");                   //跳转到指定的页面中
            }
        }
        this.aa();                                                    //调用自定义的aa方法
    }
```

在用户自定义方法 aa 中，查询欠费信息表(tb_qianfei)，并且利用 Fill 方法将数据填充到 GridView 控件中，然后利用 SubStr 方法将较长字符用“…”替换。实现的代码如下：

例程07　代码位置：光盘\TM\03\物业\WebSite3\ qianfeixinxi \qianfeicx.aspx.cs

```
public void aa()
    {
❶       OleDbConnection con = DB.createDB();                  //调用用户自定义的DB类的createDB()方法
❷       OleDbCommand cmd = new OleDbCommand();                //声明一个OleDbCommand的 cmd对象，并将其实例化
❸       con.Open();
        OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_qianfei", con);
        DataSet ds = new DataSet();                           //声明一个DataSet的ds对象，并将其实例化
        sda.Fill(ds, "tb_qianfei");
        this.gvqf.DataSource = ds;
        this.gvqf.DataBind();                                 //将数据绑定到GridView控件中
        for (int i = 0; i <= gvqf.Rows.Count - 1; i++)
        {
            DataRowView mydrv;
```

```
                string gintro;
                if (this.gvqf.PageIndex == 0)
                {
                    mydrv = ds.Tables["tb_qianfei"].DefaultView[i];
                    gintro = Convert.ToString(mydrv["address"]);
                    this.gvqf.Rows[i].Cells[1].Text = SubStr(gintro, 10);
                }
                else
                {
                    mydrv = ds.Tables["tb_qianfei"].DefaultView[i + (5 * gvqf.PageIndex)];
                    gintro = Convert.ToString(mydrv["address"]);
                    this.gvqf.Rows[i].Cells[1].Text = SubStr(gintro,10);
                }
            }
❹           con.Close();
        }
```

代码贴士

❶ OleDbConnection：表示 SQL Server 数据库的一个打开的连接。

❷ OleDbCommand：表示要对 SQL Server 数据库执行的一个 Transact-SQL 语句或存储过程。

❸ Open()：打开数据库。

❹ Close()：关闭数据库的连接。

上面的 aa 方法中，在显示“住户地址”内容时，调用了 SubStr 方法截取字符串。SubStr 方法的返回值为 String 类型，该方法有两个参数 sString 和 nLeng，其中 sString 表示要截取的字符串，nLeng 表示截取的长度。SubStr 方法关键代码如下：

例程08　代码位置：光盘\TM\03\物业\WebSite3\ qianfeixinxi \qianfeicx.aspx.cs

```
    public string SubStr(string sString, int nLeng)
      {
          if (sString.Length <= nLeng)
           {
                return sString;                                  //返回一个string 类型的变量
          }
            string sNewStr = sString.Substring(0, nLeng);       //截取字符串，从第一个字符开始截取，截取到第nLeng
个字符
            sNewStr = sNewStr + "...";                           //截取字符串后，其余的字符用“…”替换
            return sNewStr;                                      //返回一个string 类型的变量
      }
```

在欠费信息查询页中，利用 RadioButton 控件的 Check 属性的判断，实现不同的查询功能。实现的代码如下：

例程09　代码位置：光盘\TM\03\物业\WebSite3\ qianfeixinxi \qianfeicx.aspx.cs

```
    protected void Button1_Click(object sender, EventArgs e)
        {
            OleDbConnection con = DB.createDB();                 //调用用户自定义的DB类的createDB()方法
            OleDbCommand cmd = new OleDbCommand();               //声明一个OleDbCommand的 cmd对象，并将其实例化
```

```
                cmd.Connection = con;
                con.Open();
                DataSet ds = new DataSet();                          //声明一个DataSet的ds对象，并将其实例化
                if (this.rbtnqfzh.Checked == true)
                {
                    OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_qianfei where name='" +
this.txtqfzh.Text + "'", con);
                    sda.Fill(ds, "tb_qianfei");
                    this.gvqf.DataSource = ds;
                    this.gvqf.DataBind();                            //将数据绑定到GridView控件中
                }
                if (this.rbtnlx.Checked == true)
                {
                    OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_qianfei where leibie='" +
this.ddllx.SelectedItem.Text.ToString() + "'", con);
                    sda.Fill(ds, "tb_qianfei");
                    this.gvqf.DataSource = ds;
                    this.gvqf.DataBind();                            //将数据绑定到GridView控件中
                }
                if (this.rbtnqfje.Checked == true)
                {
                    OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_qianfei where qianfei='" +
this.txtqfje.Text.ToString() + "'", con);
                    sda.Fill(ds, "tb_qianfei");
                    this.gvqf.DataSource = ds;
                    this.gvqf.DataBind();                            //将数据绑定到GridView控件中
                }
            }
```

在 btnqx_Click 单击事件中，调用用户自定义 aa 方法，实现将所有的欠费信息绑定到 GridView 控件中。实现的代码如下：

例程10 代码位置：光盘\TM\03\物业\WebSite3\ qianfeixinxi \ qianfeicx.aspx.cs

```
protected void btnqx_Click(object sender, EventArgs e)
{
    this.aa();
}
```

在 gvqf_RowDataBound 的单击事件中，添加一个属性，实现高亮显示的功能。然后判断用户欠费的状态，如果是"已处理"，将字体颜色改成蓝色；如果用户的欠费状态是“未处理”，将字体颜色改成红色。实现代码如下：

例程11 代码位置：光盘\TM\03\物业\WebSite3\ qianfeixinxi \qianfeicx.aspx.cs

```
protected void gvqf_RowDataBound(object sender, GridViewRowEventArgs e)
{
            if (e.Row.RowType == DataControlRowType.DataRow)
            {
                e.Row.Attributes.Add("onMouseOver",
"Color=this.style.backgroundColor;this.style.backgroundColor='lightBlue'");
```

```
                e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=Color;");
                if (e.Row.Cells[6].Text == "已处理")
                {
                    e.Row.Cells[6].Text = "<font color=blue>已处理</font>";     //改变字体的颜色
                }
                else
                {
                    e.Row.Cells[6].Text = "<font color=red>未处理</font>";     //改变字体的颜色
                }
                e.Row.Cells[3].Text = Convert.ToDateTime(e.Row.Cells[3].Text).ToShortDateString();
                e.Row.Cells[4].Text = Convert.ToDateTime(e.Row.Cells[4].Text).ToShortDateString();
    }
```

3.7　管理员登录页设计

3.7.1　管理员登录页概述

物业人员通过管理员登录模块，进入网站后台进行相关信息的维护及管理等操作。系统登录模块主要用于对进入系统的用户进行安全性检查，以防止非法用户进入系统。在登录时只有合法的用户才能进入系统，同时，系统根据登录用户的级别，给予不同的操作权限。管理员登录页面运行效果如图 3.26 所示。

图 3.26　管理员登录页

3.7.2　管理员登录页技术分析

在管理员登录页面中，物业工作人员输入员工姓名、密码，并选择相应的权限，单击“登录”按钮时系统与数据库中的数据进行比较，如果相同则进入管理员的操作界面，否则会弹出提示框，提示用户输入的用户名或者密码错误。

在实现管理员登录页时，主要利用了以下技术，下面进行详细介绍。

（1）利用聚合函数 COUNT 返回组中相关数据的个数

COUNT 函数用于返回组中项目的数量。管理员登录页中利用 COUNT 函数实现根据员工输入的用户名、用户密码、权限在数据表中进行查询，如果查询有该记录 COUNT(*)返回 1，否则返回 0。

COUNT 函数的语法形式如下：

```
COUNT({[ALL|DISTINCT]expression}|*)
```

参数说明如下。

- ☑ ALL：该参数是默认设置，如果没有参数，系统对所有的值进行聚合函数运算。
- ☑ DISTINCT：指定 COUNT 返回唯一非空值的数量。
- ☑ expression：一个表达式。在 SQL Server 中其类型是除 uniqueidentifier、text、image 或 ntext 之外的任何类型。不允许使用聚合函数和字查询。
- ☑ *：指定应该计算所有行以返回表中的总数。COUNT(*)不需要任何参数，而且不能与 DISTINCT 一起使用。COUNT(*)返回指定表中行的数量而不消除副本，它对每行分别进行计算，包括含有空值的行。

COUNT(*)：返回组中查询结果统计数量，这些项目包括 NULL 值和副本。

COUNT(ALL expression)：对组中的每一行都计算 expression 并返回非空值的查询结果统计数量。

COUNT(DISTINCT expression)：对组中的每一行都计算 expression 并返回唯一非空值的查询结果统计数量。

本实例中利用 SQL 语句的关键代码如下：

```
select count(*) from tb_zhuce where name='" + this.txtworkname.Text + "'and pwd='" + this.txtworkpwd.Text + "'and zhiwu='"+this.ddlQx.Text+"'
```

（2）利用 Session 对象来保存管理员的姓名和密码

当管理员成功登录后，在 Session 对象中保存管理员登录的姓名和密码，如果管理员没有退出该网站则 Session 中始终保存当前使用该网站的管理员的姓名和密码。当进入后台中的其他页时，只需判断 Session 里存的值是否为空，如果不为空，说明该使用者是管理员，便可直接访问。否则说明两点，一是还没有登录，二是登录的权限不是管理员。这样防止在 IE 地址栏中输入相应地址，非法访问该网页，大大提高了网站的安全性。

3.7.3 管理员登录页实现过程

本模块使用的数据表：tb_zhuce。

1. 设计步骤

（1）在网站中创建一个 Web 窗体，将其命名为 denglu.aspx，用于管理员登录。

（2）在 Web 窗体中添加一个 Table 表格，用于页面的布局。

（3）在 Table1 表格中添加相关服务器控件，属性设置及其用途如表 3.7 所示。

表 3.7　管理员登录中用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
TextBox	txtworkname	无	输入登录用户名
	txtworkpwd	PasswordChar属性设置为“*”（当用户）	输入登录用户密码
DropDownList	ddlQx	Items属性添加两个成员，分别是“管理员”、“普通员工”	用于显示管理员的权限
ImageButton	imbtn	ImageUrl属性设置为“~/image/login_10.jpg”	用于显示“进入”信息的文字按钮

2．实现代码

引入命名空间。代码如下：

```
using System.Data.OleDb;
```

在“登录”按钮中，通过从文本框中传来的姓名、密码及权限查询注册表中的信息，用 ExecuteScalar() 方法执行查询语句，并返回查询的结果集中第一行的第一列。并将返回来的结果转化成整型赋值给变量 a，如果变量大于 0，登录成功跳转到管理员主页；否则弹出一个“很遗憾，用户名或密码错误”的对话框。代码如下：

例程12　代码位置：光盘\TM\03\物业\ WebSite3\ denglu.aspx.cs

```
protected void btnDl_Click(object sender, EventArgs e)
{
    this.lblXx.Visible = false;
❶   OleDbConnection con = DB.createDB();                    //调用连库操作类里的createDB()方法
    OleDbCommand cmd = new OleDbCommand();                  //声明一个OleDbCommand对象，并将其实例化
    cmd.Connection = con;
❷   con.Open();                                             //打开数据库连接
    //通过员工姓名、密码、权限查询tb_zhuce数据库，以找出相应的信息
❸   cmd.CommandText = "select count(*) from tb_zhuce where name='" + this.txtworkname.Text + "'and pwd='" +
this.txtworkpwd.Text + "'and zhiwu='" + this.ddlQx.Text + "' ";
    Session["name"] = this.txtworkname.Text.ToString();     //把员工输入的用户名存储到Session变量中
    Session["pwd"] = this.txtworkpwd.Text.ToString();       //把员工输入的用户密码存储到Session变量中
    int a = Convert.ToInt32(cmd.ExecuteScalar());
    if (a > 0)
    {
      Response.Redirect("~/guanliyuan/guanliindex.aspx");   //登录成功跳转到管理员主页
    }
   else
    {
❹     Response.Write("<script language=javascript>alert('很遗憾，用户名或密码错误！');</script>");
    }
    }
❺   con.Close();                                            //关闭数据库连接
}
```

代码贴士

❶ 自定义 createDB 方法：用于连接数据库。

❷ Open 方法：打开 Access 数据库。

❸ CommandText 属性：获取或设置针对数据源运行的文本命令。
❹ Write 方法：将对象序列化为消息体。
❺ Close 方法：关闭数据库的连接。

3.8 值班员工页设计

3.8.1 值班员工页概述

在管理员主页的导航栏中单击“值班员工”超链接进入值班员工页面，并且根据小区物业员工的在岗状态显示员工的信息。通过对该页的查看，方便物业对值班员工的管理。值班员工页面的运行效果如图 3.27 所示。

图 3.27 值班员工页面

3.8.2 值班员工技术分析

小区物业值班员工信息页面中，主要用到了 Table 控件动态添加行和单元格功能。在页面的单元格中存放员工信息的用户自定义控件（yuangong.ascx），根据实际页面布局的需要将 Table 控件布局为两行 3 列。

本页面中用到的 Table 控件说明如下。

☑ TableRow

TableRow 类表示 Table 控件中的行。

向表动态添加行的语法格式如下：

```
TableRow tRow = new TableRow();
Table1.Rows.Add(tRow);
```

☑ TableCell

TableCell 类表示 Table 控件中的单元格。

向表中动态添加完行后，再向行中添加单元格。语法格式如下：

```
TableCell tCell = new TableCell();//创建一个TableCell对象
tRow.Cells.Add(tCell);
```

注意：Table 控件可作为 TableRow 控件的父控件。表支持一个名为 Rows 的属性，该属性是 TableRow 对象的集合。可以通过管理该集合（在其中添加或删除项），指定表中的行。而 TableRow 控件又支持一个名为 Cells 的集合，该集合包含 TableCell 对象。

3.8.3　值班员工页实现过程

本模块使用的数据表：tb_yuangong l。

1．设计步骤

（1）在网站中创建一个 Web 窗体，将其命名为 zhibanyuangong.aspx，用于显示值班员工的信息。

（2）在 Web 窗体中添加一个 Table 表格，用于页面的布局。

（3）在 Table1 表格中添加相关服务器控件，属性设置及其用途如表 3.8 所示。

表 3.8　值班员工页面中用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
A Label	lblzz	Text属性设置为“我们的终旨：全心全意为人民服务！”	显示“我们的终旨：全心全意为人民服务！”这段文字
Table	Table1	无	将用户自定义控件绑定到Table控件中，以显示员工的信息

2．实现代码

引入命名空间。具体代码如下：

```
using System.Data.OleDb;
```

在 page_load 单击事件中，根据传过来的变量 id 的值查询员工信息表，通过 OleDbDataReader 类的 Read 方法，将数据从数据库中读取出来加以显示，实现通过变量传来的值读取并显示对应员工的信息。代码如下：

例程13　代码位置：光盘\TM\03\物业\WebSite3\ zhibanyuangong.aspx.cs

```
public partial class zhibanyuangong : System.Web.UI.Page
{
    private int numrow = 3;                                    //设置表格中一行显示的个数
    protected void Page_Load(object sender, EventArgs e)
    {
❶       if (Session["name"] == null && Session["pwd"] == null)    //判断管理员是否登录
```

```
            {
❷             Response.Redirect("~/guanliyuan/guanliyuandenglu.aspx");          //将该页跳转到指定的页面中
            }
            //调用用户自定义的DB类的createDB()方法
❸           OleDbConnection con = DB.createDB();
            //声明一个OleDbCommand的cmd对象，并将其实例化
❹           OleDbCommand cmd = new OleDbCommand();
            cmd.Connection = con;
            //打开数据库连接
❺           con.Open();
❻           cmd.CommandText = "select top 6 id from tb_yuangong where state='在岗'order by id Desc";
            //声明一个变量
            int i = 0;
            //声明一个TableRow对象
❼           System.Web.UI.WebControls.TableRow tr = new TableRow();
❽           OleDbDataReader sdr = cmd.ExecuteReader();                          //通过ExecuteReader()执行Select语句
            while (sdr.Read())                                                  //判断是否还有记录
            {
                i = i + 1;
❾               System.Web.UI.WebControls.TableCell tc = new TableCell();     //声明一个TableCell对象
                yuangong yg = (yuangong)this.LoadControl("~/yuangong.ascx"); //声明一个员工类型的对象
                yg.id1 = sdr.GetString(0);//从数据库中读取第一条记录赋值给变量id1
                tc.Controls.Add(yg);     //将声明的用户自定义控件的对象填充到单元格中
                tr.Controls.Add(tc);                  //将声明的单元格对象填充到行中
                if (i % numrow == 0)
                {
                    this.Table1.Rows.Add(tr);         //向表动态添加行
                    tr = new TableRow();              //声明一个TableRow对象
                }
                if (i % numrow != 0)
                {
                    this.Table1.Rows.Add(tr);         //向表动态添加行
                }
            }           sdr.Close();                  //关闭阅读器
            con.Close();                              //关闭数据库连接
        }
        }
```

代码贴士

❶ Session：Session 是 ASP 和 ASP.NET 中用于保持状态的基于 Web 服务器的方法。Session 允许通过将对象存储在 Web 服务器的内存中在整个用户会话过程中保持任何对象。

Session 通常用于执行以下操作：

☑ 存储需要在整个用户会话过程中保持其状态的信息，例如登录信息或用户浏览 Web 应用程序时需要的其他信息。

☑ 存储只需要在页重新加载过程中或按功能分组的一组页之间保持其状态的对象。

❷ Response 对象：提供对当前页的输出流的访问。可以使用此类将文本插入页中、编写 Cookie 等。

Redirect 方法：配置响应以将客户端重定向到指定的 URL。

❸ OleDbConnection：到数据源的连接是打开的。

❹ OleDbCommand：要对数据源执行的 SQL 语句或存储过程。

❺ Open 方法：在指定视图中打开解决方案。
❻ CommandText 属性：获取或设置针对数据源运行的文本命令。
❼ TableRow 类：Table 控件中的行。
❽ OleDbDataReader：提供从数据源读取数据行的只进流的方法。
❾ TableCell：Table 控件中的单元格。

注意：LoadControl()方法根据指定的虚拟路径从文件加载 Control 对象。

3.9　业主住房信息管理页设计

3.9.1　业主住房信息管理页概述

业主住房信息管理页是小区物业内部管理网必不可少的一部分，在该模块中涉及了业主住房信息的添加、修改、删除和查询几项内容。其运行效果如图 3.28 所示。

图 3.28　业主住房信息管理页

3.9.2　业主住房信息管理页技术分析

业主住房信息管理页中，主要用到了一个典型功能——如何将数据控件中的日期以编程的方式转换为指定格式。

在开发网站的过程中，如开发新闻网站、论坛、聊天室等，都需要获取日期。有一些网站将数据控件中的日期以编程的方式转换为指定格式显示出来，以美化网站。下面将以获得系统当前日期为例，向读者介绍如何获取系统日期并将系统日期用不同的格式显示出来。运行效果如图 3.29 所示。

图 3.29　获取当前系统日期

获取当前系统日期并将其转换为短日期格式，代码如下：

```
❶ Label1.Text = DateTime.Now.ToShortDateString();
```

获取当前系统日期并将其转换为长日期格式，代码如下：

```
❷ Label2.Text = DateTime.Now.ToLongDateString();
```

代码贴士

❶ ToShortDateString()：将日期的格式转换为其等效的短日期字符串表示形式。

❷ TolongDateString()：将日期的格式转换为其等效的长日期字符串表示形式。

Now：它包含系统的当前日期和时间。

说明：代码中的 DateTime 对象中的 Now 属性，可以用 DateTime 对象中的 Today 属性代替，DataTime.Now 属性与 DataTime.Today 属性都是用来获取当前系统时间。

3.9.3　业主住房信息管理页实现过程

本模块使用的数据表：tb_zhuhu。

1．设计步骤

（1）在该网站中创建 zhuhuxinxi 文件夹，用于存放基本信息 Web 窗体。

（2）在 zhuhuxinxi 文件夹下创建一个 Web 窗体，将其命名为 zhuhuxinxiguanli.aspx，用于业主住房信息管理。

（3）在 Web 窗体中添加一个 Table 表格，用于页面的布局。页面中的主要控件及说明如表 3.9 所示。

表 3.9　业主住房信息管理页用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
TextBox	txtxqmc	无	输入小区名称
	txtlymc	TextMode属性设置为“”	输入楼宇名称
Button	btnAdd	Text属性设置为“添加”	添加
	btnDelete	Text属性设置为“删除”	删除
GridView	gvzhzf	AllowPaging属性设置为True（允许分页） AutoGenerateColumns属性设置为False（取消自动生成） PageSize属性设置为5（每页显示5条数据）	显示业主住房信息
LinkButton	lbtnzfcx	Text属性设置为“业主住房信息查询”	业主住房信息查询
	lbtnzfsh	Text属性设置为“业主住房信息审核”	业主住房信息审核

2．实现代码

引入命名空间。代码如下：

```
using System.Data.OleDb;
```

在 page_load 事件中主要是判断业主或管理员是否已登录，如果没有登录，业主跳转到业主登录页中，管理员跳转到管理员登录页中，并且调用用户自定义的 aa 方法，将业主住房信息绑定到 GridView 控件中，加以显示。实现代码如下：

例程14　代码位置：光盘\TM\03\物业\ WebSite3\ zhuhuxinxi \zhuhuxinxiguanli.aspx.cs

```
        protected void Page_Load(object sender, EventArgs e)
        {
❶       if (Session["name"] == null && Session["pwd"] == null)          //判断用户名和密码是否为空
          {
❷             Response.Redirect("~/default2.aspx");                     //将页面跳转到网站首页
          }
❸          this.Label3.Text = System.DateTime.Now.ToLongTimeString();  //显示当前系统时间
           this.aa();                                                   //调用用户自定义aa方法
        }
```

代码贴士

❶ Session：Session 是 ASP 和 ASP.NET 中用于保持状态的基于 Web 服务器的方法。Session 允许通过将对象存储在 Web 服务器的内存中在整个用户会话过程中保持任何对象。

Session 通常用于执行以下操作：

☑ 存储需要在整个用户会话过程中保持其状态的信息，例如登录信息或用户浏览 Web 应用程序时需要的其他信息。

☑ 存储只需要在页重新加载过程中或按功能分组的一组页之间保持其状态的对象。

Session 的作用就是它在 Web 服务器上保持用户的状态信息供在任何时间从任何页访问。因为浏览器不需要存储任何这种信息，所以可以使用任何浏览器，即使是像 PDA 或手机这样的浏览器设备。

❷ Redirect：配置响应以将客户端重定向到指定的 URL。

❸ DateTime：获取事件跟踪发生的日期和时间。

ToLongTimeString()：将此实例的值转换为其等效的长时间字符串表示形式。

在 aa 方法中，主要是用户自定义的方法，用于查询业主住房信息，将业主住房信息填充到数据集中，通过调用 DataBind()方法将业主住房信息绑定到 GridView 控件中，加以显示。实现的代码如下：

例程15　代码位置：光盘\TM\03\物业\WebSite3\zhuhuxinxi \zhuhuxinxiguanli.aspx.cs

```
        public void aa()
        {
❶          OleDbConnection con = DB.createDB();
❷          con.Open();                                                  //打开数据库连接
           OleDbCommand cmd = new OleDbCommand();
           OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_zhuhu", con);
           DataSet ds = new DataSet();
           sda.Fill(ds, "tb_zhuhu");                                    //填充数据集
           this.GridView2.DataSource = ds;                              //获取数据源
           this.GridView2.DataBind();                                   //绑定GridView控件
           con.Close();                                                 //关闭数据库连接
        }
```

代码贴士

❶ createDB(): 用于连接数据库。

❷ Open(): 打开 Access 数据库。

在“添加”按钮中，用 INSERT 语句添加业主住房信息，并且执行 ExecuteNonQuery()方法执行 INSERT 语句。成功插入业主住房信息后，将弹出一个“恭喜您，信息添加成功！！”对话框。实现的代码如下：

例程16 代码位置：光盘\TM\03\物业\ WebSite3\ zhuhuxinxi \zhuhuxinxiguanli.aspx.cs

```
        protected void Button1_Click(object sender, EventArgs e)
        {
❶           OleDbConnection con = DB.createDB();
❷           OleDbCommand cmd = new OleDbCommand();                    //声明一个OleDbCommand对象
            cmd.Connection = con;
❸           con.Open();                                               //打开数据连接
❹             cmd.CommandText = "insert into tb_zhuhu values('"+ this.txtxqmc.Text +"','"+ this.txtlymc.Text +"','"+
this.txtlydy.Text +"','"+ this.txtfh.Text +"','"+ this.DropDownList1.SelectedItem.Text.ToString() +"','"+ this.txtjzmj.Text +"','"+
this.DropDownList2.SelectedItem.Text.ToString() +"','"+ this.txtfwbz.Text +"','"+
this.DropDownList3.SelectedItem.Text.ToString() +"','"+ this.txtqrsj.Text +"','"+ this.txtzfmc.Text +"','"+ this.txtsfz.Text +"','"+
this.txtgzdw.Text +"','"+this.txtlxdh.Text+"','"+ this.txtlxdz.Text +"')";       //执行SQL语句
              cmd.ExecuteNonQuery();                                  //执行OLEDB语句并返回受影响的行数
              Response.Write("<script language=javascript>alert('恭喜您，信息添加成功！！');</script>");
❺             con.Close();                                            //关闭数据库连接
              this.aa();                                              //调用自定义aa方法
        }
```

代码贴士

❶ OleDbConnection: SQL Server 数据库的一个打开的连接。

❷ OleDbCommand: 要对 SQL Server 数据库执行的一个 Transact-SQL 语句或存储过程。

❸ Open(): 打开数据库。

❹ CommandText: 获取或设置针对数据源运行的文本命令。

❺ Close(): 关闭数据库的连接。

在“住户住房信息审核”按钮中，实现的主要功能是将页面跳转到业主住房信息审核页面中。实现的代码如下：

例程17 代码位置：光盘\TM\03\物业\ WebSite3\ zhuhuxinxi \zhuhuxinxiguanli.aspx.cs

```
    protected void LinkButton3_Click(object sender, EventArgs e)
    {
        Response.Redirect("zhuhurzshenhe.aspx");  //将页面跳转到zhuhurzshenhe.aspx页
    }
```

在“住户住房信息查询”按钮中，实现的主要功能是将页面跳转到业主住房信息查询页面中。实现的代码如下：

例程18 代码位置：光盘\TM\03\物业\WEBSITE3\ zhuhuxinxi \zhuhuxinxiguanli.aspx.cs

```
        protected void LinkButton1_Click(object sender, EventArgs e)
        {
```

```
            Response.Redirect("zhuhucx.aspx");      //将页面跳转到zhuhucx.aspx页
        }
```

在“检验住房名称”按钮中，根据输入的住房名称来查找业主住房信息表，并且通过执行ExecuteScalar()进行查询，并返回查询所返回的结果集中第一行的第一列。将返回的结果转换为整型格式，并将其值传递给变量 i 中，并加以判断，如果变量 i 的值大于 1，则弹出一个“住房名称存在，请再添加一个！”对话框，否则弹出一个“住房名称不存在！”对话框。实现的代码如下：

例程19　代码位置：光盘\TM\03\物业\ WebSite3\ zhuhuxinxi \zhuhuxinxiguanli.aspx.cs

```
        protected void Button1_Click1(object sender, EventArgs e)
        {
❶         OleDbConnection con = DB.createDB();
❷         OleDbCommand cmd = new OleDbCommand();
          cmd.Connection = con;
❸         con.Open();                                          //打开数据库连接
❹         cmd.CommandText = "select * from tb_zhuhu where zfname='" + this.txtzfmc.Text + "'";
          int i = Convert.ToInt32(cmd.ExecuteScalar());
            if (i > 1)
            {
                this.txtzfmc.Text = "";                        //清空文本框
                this.txtzfmc.Focus();                          //获取焦点
                Response.Write("<script language=javascript>alert('住房名称存在，请再添加一个！');</script>");
            }
            else
            {
                Response.Write("<script language=javascript>alert('住房名称不存在！');</script>");
            }
❺       con.Close();
        }
```

代码贴士

❶ OleDbConnection：SQL Server 数据库的一个打开的连接。

❷ OleDbCommand：要对 SQL Server 数据库执行的一个 Transact-SQL 语句或存储过程。

❸ Open()：打开数据库。

❹ CommandText：获取或设置针对数据源运行的文本命令。

❺ Close()：关闭数据库的连接。

3.10　业主投诉信息审核页

3.10.1　业主投诉信息审核页概述

业主投诉信息审核页面主要实现对业主投诉信息的审核，主要审核业主的投诉信息是否已处理，如果未处理，则通知相关部门进行处理；如果已处理，则设置为已处理状态。对于已处理的投诉信息可适当删除。业主投诉信息审核页面的运行效果如图 3.30 所示。

图 3.30　业主投诉信息审核页

3.10.2　业主投诉信息审核页技术分析

由于小区物业内部管理网涉及的信息量很大，如果将已处理和未处理的欠费信息混在一起，看上去会很乱，因此本程序将信息表“状态”列中的“未处理”和“已处理”的字体颜色分别设置成红色和蓝色，以给人一目了然的感觉。页面中信息的显示是通过使用 GridView 控件实现的，为了将不同的状态信息以不同的颜色显示，需要处理 GridView 控件的 RowDataBound 事件，在该事件中判断表格行如果为数据行，则读取“状态”单元格的信息，根据不同的状态设置不同的字体颜色。主要代码如下：

例程20　代码位置：光盘\TM\03\物业\ WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
protected void gvqf_RowDataBound(object sender, GridViewRowEventArgs e)
{
    if (e.Row.RowType == DataControlRowType.DataRow)
    {
        if (e.Row.Cells[6].Text == "已处理")
        {
            e.Row.Cells[6].Text = "<font color=blue>已处理</font>"; //改变文字颜色
        }
        else
        {
            e.Row.Cells[6].Text = "<font color=red>未处理</font>";    //改变文字颜色
        }
    }
}
```

3.10.3　业主投诉信息审核页实现过程

本模块使用的数据表：tb_tousu。

1．设计步骤

（1）在该网站中创建 tousuxinxi 文件夹，用于存放基本信息 Web 窗体。

（2）在 tousuxinxi 文件夹下创建一个 Web 窗体，将其命名为 tousushenhe.aspx，用于业主投诉信息审核。

（3）在 Web 窗体中添加一个 Table 表格，用于页面的布局。

（4）在 Table1 表格中添加相关主要的服务器控件，属性设置及其用途如表 3.10 所示。

表 3.10　业主投诉信息审核页用到的主要控件

控 件 类 型	控 件 名 称	主要属性设置	控 件 用 途
LinkButton	txtxqmc	Text属性设置为“业主投诉信息审核”	链接业主投诉信息页
	txtlymc	Text属性设置为“业主投诉信息审核”	链接业主投诉查询页
GridView	gvzhzf	AllowPaging属性设置为True（允许分页） AutoGenerateColumns属性设置为False（取消自动生成） PageSize属性设置为5（每页显示5条数据）	显示小区业主投诉信息

2．实现代码

在 page_Load 事件中，利用 Session 对象判断是否登录，并且调用自定义 aa 方法。代码如下：

例程21　代码位置：光盘\TM\03\物业\WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
protected void Page_Load(object sender, EventArgs e)
{
        if (Session["name"] == null && Session["pwd"] == null)
        {
            Response.Redirect("~/guanliyuan/guanliyuandenglu.aspx"); //跳转到指定的页面中
        }
        this.aa();                                                    //调用自定义aa方法
}
```

在自定义 aa 方法中，利用 OleDbDataAdapter 的 fill 方法，实现将投诉信息填充到 GridView 控件中。实现的代码如下：

例程22　代码位置：光盘\TM\03\物业\ WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
    private void aa()
    {
❶       OleDbConnection con = DB.createDB();          //调用用户自定义的DB类的createDB()方法
❷        con.Open();
❸        OleDbCommand cmd = new OleDbCommand(); //声明一个OleDbCommand的cmd对象，并将其实例化
         OleDbDataAdapter sda = new OleDbDataAdapter("select * from tb_tousu", con);
         DataSet ds = new DataSet();                  //声明一个DataSet的ds对象，并将其实例化
         sda.Fill(ds, "tb_tousu");
         this.GridView1.DataSource = ds.Tables[0].DefaultView;
         this.GridView1.DataKeyNames = new string[] { "id" };
         this.GridView1.DataBind();                   //将数据绑定到GridView控件中
❹        con.Close();
    }
```

代码贴士

❶ OleDbConnection：SQL Server 数据库的一个打开的连接。

❷ Open()：打开数据库。

❸ OleDbCommand：对 SQL Server 数据库执行的一个 Transact-SQL 语句或存储过程。
❹ Close()：关闭数据库的连接。

在 GridView1_RowDataBound 事件中，主要是实现改变单元格中信息的功能。实现的代码如下：

例程23　代码位置：光盘\TM\03\物业\ WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
        protected void GridView1_RowDataBound(object sender, GridViewRowEventArgs e)
        {
            if (e.Row.RowType == DataControlRowType.DataRow)
            {
                e.Row.Attributes.Add("onMouseOver",
"Color=this.style.backgroundColor;this.style.backgroundColor='lightBlue'");
                e.Row.Attributes.Add("onMouseOut", "this.style.backgroundColor=Color;");
                if (e.Row.Cells[4].Text == "已处理")
                {
                    e.Row.Cells[4].Text = "<font color=blue>已处理</font>";   //改变文字的颜色
                }
                else
                {
                    e.Row.Cells[4].Text = "<font color=red>未处理</font>";    //改变文字的颜色
                }
                //弹出提示信息
                ((LinkButton)(e.Row.Cells[6].Controls[0])).Attributes.Add("onclick", "return confirm('确定删除吗？')");
                e.Row.Cells[2].Text = Convert.ToDateTime(e.Row.Cells[2].Text).ToShortDateString();
            }
    }
```

在 GridView1_RowDeleting 事件中，利用 OleDbCommand 的 ExecuteNonQuery()方法，实现将相应记录删除的功能。主要代码如下：

例程24　代码位置：光盘\TM\03\物业\ WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
        protected void GridView1_RowDeleting(object sender, GridViewDeleteEventArgs e)
        {
            string id = this.GridView1.DataKeys[e.RowIndex].Value.ToString();
❶           OleDbConnection con = DB.createDB();
❷           con.Open();                                          //打开数据库连接
❸           OleDbCommand cmd = new OleDbCommand("delete from tb_tousu where id='" + id + "'", con);
            cmd.ExecuteNonQuery();
            this.aa();                                           //调用aa方法
            this.GridView1.EditIndex = -1;
❹           this.Close();                                        //关闭数据库连接
    }
```

代码贴士

❶ OleDbConnection：SQL Server 数据库的一个打开的连接。
❷ Open()：打开数据库。
❸ OleDbCommand：对 SQL Server 数据库执行的一个 Transact-SQL 语句或存储过程。
❹ Close()：关闭数据库的连接。

在 GridView1_PageIndexChanging 中，实现分页的功能。实现的代码如下：

例程25　代码位置：光盘\TM\03\物业\WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
protected void GridView1_PageIndexChanging(object sender, GridViewPageEventArgs e)
{
    GridView1.PageIndex = e.NewPageIndex;
    this.aa();                                    //调用aa方法
}
```

在 GridView1_SelectedIndexChanging 事件中，利用 OleDbCommand 的 ExecuteScalar()方法查询投诉信息表，并且修改投诉信息表中状态。实现的代码如下：

例程26　代码位置：光盘\TM\03\物业\ WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
    protected void GridView1_SelectedIndexChanging(object sender, GridViewSelectEventArgs e)
    {
        //获取主键的值
        string id = this.GridView1.DataKeys[e.NewSelectedIndex].Value.ToString();
❶       OleDbConnection con = DB.createDB();   //调用用户自定义的DB类的createDB()方法
❷       con.Open();
❸       OleDbCommand cmd = new OleDbCommand("select zhuangtai from tb_tousu where id='" + id + "'", con);
        string count = cmd.ExecuteScalar().ToString();
        if (count == "已处理")
        {
            count = "未处理";                      //将"未处理"赋值给变量Count
        }
        else
        {
            count = "已处理";                      //将"已处理"赋值给变量Count
        }
❹       cmd.CommandText = "update tb_tousu set zhuangtai='" + count + "'where id='" + id + "'";
        cmd.ExecuteNonQuery();
❺       con.Close();                               //关闭数据库连接
        this.aa();                                 .//调用自定义的aa方法
}
```

代码贴士

❶ OleDbConnection：SQL Server 数据库的一个打开的连接。

❷ Open()：打开数据库。

❸ OleDbCommand：要对 SQL Server 数据库执行的一个 Transact-SQL 语句或存储过程。

❹ CommandText：获取或设置针对数据源运行的文本命令。

❺ Close()：获取或设置针对数据源运行的文本命令。

在 LinkButton1_Click 事件中，利用 Response 对象的 Redirect 方法，实现将页面跳转到 tousuguanli.aspx 中。实现代码如下：

例程27　代码位置：光盘\TM\03\物业\WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
protected void LinkButton1_Click(object sender, EventArgs e)
{
```

```
        Response.Redirect("tousuguanli.aspx");                    //跳转到指定的页面中
}
```

在 LinkButton2_Click 事件中，实现将页面跳转到 tousuchaxun.aspx 中。实现代码如下：

例程28　代码位置：光盘\TM\03\物业\ WebSite3\ tousuxinxi\tousushenhe.aspx.cs

```
protected void LinkButton2_Click(object sender, EventArgs e)
{
        Response.Redirect("tousuchaxun.aspx");                    //跳转到指定的页面中
}
```

3.11　网站文件清单

为了帮助读者了解小区物业内部管理网的文件构成，笔者现以表格形式列出网站的文件清单，如表 3.11 所示。

表 3.11　网站文件清单

文件位置及名称	说　　明
WebSite3\App_Code\ DataBase.cs	数据库操作类
WebSite3\App_Data\ db_wygl.mdb	Access数据库文件
WebSite3\cheliangxinxi\ cheliangchaxun.aspx	网站后台停车信息查询页
WebSite3\cheliangxinxi\cheliangguanli.aspx	网站后台停车信息管理页
WebSite3\cheliangxinxi\cheliangshenhe.aspx	网站后台停车信息审核页
WebSite3\guanliyuan\guanliindex.aspx	网站后台管理员主页
WebSite3\guanliyuan\guanliyuandenglu.aspx	网站后台管理员登录页
WebSite3\qianfeixinxi\qianfeicx.aspx	网站后台欠费信息查询页
WebSite3\qianfeixinxi\qianfeiguanli.aspx	网站后台欠费信息管理页
WebSite3\qianfeixinxi\qianfeishenhe.aspx	网站后台欠费信息审核页
WebSite3\qianfeixinxi\qianfeixiangxi.aspx	网站后台欠费信息详细页
WebSite3\tousuxinxi\tousucx.aspx	网站后台业主投诉信息查询页
WebSite3\tousuxinxi\tousuguanli.aspx	网站后台业主投诉信息管理页
WebSite3\tousuxinxi\tousushenhe.aspx	网站后台业主投诉信息审核页
WebSite3\weihuxinxi\weihushenhe.aspx	网站后台设备维修信息审核页
WebSite3\weihuxinxi\weixuichaxun.aspx	网站后台设备维修信息查询页
WebSite3\weihuxinxi\weixuiguanli.aspx	网站后台设备维修信息管理页
WebSite3\xinwenxinxi\xinwencx.aspx	网站后台新闻信息查询页
WebSite3\xinwenxinxi\xinwenguanli.aspx	网站后台新闻信息管理页
WebSite3\xinwenxinxi\xinwenshenhe.aspx	网站后台新闻信息审核页
WebSite3\xinwenxinxi\xinwenxiangxi.aspx	网站后台新闻信息详细页

续表

文件位置及名称	说　明
WebSite3\yuangongxinxi\xiangxi.aspx	网站后台员工信息详细页
WebSite3\yuangongxinxi\yuangongcx.aspx	网站后台员工信息查询页
WebSite3\yuangongxinxi\yuangonginfo.aspx	网站后台员工信息管理页
WebSite3\zhuce\yuangongzhuce.aspx	网站前台注册信息页
WebSite3\zhuce\zhuce.aspx	网站前台注册协议信息页
WebSite3\zhuhuxinxi\zhuhucx.aspx	网站后台业主信息查询页
WebSite3\zhuhuxinxi\zhuhurzshenhe.aspx	网站后台业主信息审核页
WebSite3\zhuhuxinxi\zhuhuxinxiguanli.aspx	网站后台业主信息管理页
WebSite3\userControl\foot.ascx	网站脚标用户自定义控件
WebSite3\userControl \head.ascx	网站表头用户自定义控件
WebSite3\Default.aspx	网站首页
WebSite3\denglu.aspx	网站登录页
WebSite3\Web.Config	Web配置文件
WebSite3\yuangong.ascx	网站员工信息自定义用户控件
WebSite3\zhibanyuangong.aspx	网站值班员工信息页

3.12　Access 数据库操作技术专题

SQL 是结构化查询语言（Structured Query Language）的缩写，是 Access 用于后台编写查询操作的语言。其实，Access 中的所有查询都可以被认为是 SQL 查询，因为 Access 中的查询功能就是以 SQL 语句为基础实现的，而且某些查询只能通过 SQL 语句实现。下面将对简单的 SELECT 语句的查询、FROM 子句、使用 WHERE 子句设置查询条件、使用 ORDER BY 子句对查询结果排序、使用 GROUP BY 子句将查询结果分组、嵌套查询、多表查询和添加数据等进行详细介绍。

3.12.1　简单的 SELECT 语句的查询

SELECT 语句是 SQL 中最常用、最重要的语句。SELECT 语句的完整语法是：

```
SELECT select_list
[ INTO new_table ]
FROM table_source
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_by_expression [ASC| DESC ]]
```

SELECT 语句中的主要参数说明如下。

- ☑ select_list：查询的列或者表达式的列表，用逗号分隔。
- ☑ new_table：新的表名。
- ☑ table_source：要查询的表。如果是多个表，用逗号分隔。
- ☑ search_condition：查询条件。
- ☑ group_by_ expression：分组表达式。
- ☑ order_by_ expression：排序表达式。
- ☑ ASC：升序排列。
- ☑ DESC：降序排列

1．选择所有列

Select 语句后的第一个子句，即 Select 关键字开头的子句，用于选择进行显示的列。如果要显示数据表中的所有列的值时，Select 子句后用星号（*）表示。

例如查询 student 数据表的所有数据，SQL 语句如下：

```
Select *
from student
```

2．选择部分列

在查询表时，很多时候只显示所需要的列。这时在 Select 子句后分别列出各个列名称即可。例如显示 student 表中学号、年龄和出生日期等字段，SQL 语句如下：

```
Select 学号 , 年龄 , 出生日期
from student
```

各个列用逗号隔开，但逗号是英文状态下的逗号。

3．不显示重复记录

DISTINCT 关键字主要用来从 SELECT 语句的结果集中去掉重复的记录。如果用户没有指定 DISTINCT 关键字，那么系统将返回所有符合条件的记录，并组成结果集，其中包括重复的记录。

例如，在 grade 表中不显示学号重复的学生信息，SQL 语句如下：

```
Select distinct 学号
from grade
```

对多个列使用 DISTINCT 关键字时，查询结果只显示每个有效组合的一个例子，即结果表中没有完全相同的两个行。

例如在 grade 表中显示“学号”和“课程代号”的不同值，SQL 语句如下：

```
select  distinct 学号,课程代号
```

3.12.2 FROM 子句

FROM 子句指定 SELECT 语句查询及与查询相关的表或视图。在 FROM 子句中最多可指定 256 个表或视图，它们之间用逗号分隔。

1．从表和视图中选择

例如查询 course 表的信息，SQL 语句如下：

```
select  *  from  course
```

例如查询用户视图 newview 中的信息，SQL 语句如下：

```
select  *
from  newview
```

其中，course 是表的名称；newview 是视图的名称。

2．列以表的名称作为前缀

由于 FROM 子句指定的数据源可以是多个表，列名可能会有重复的，所以对不同子句中的列以表的名称作为前缀可使其变得更为直观，从而避免当两个或更多个表有相同的列名时所引起的混乱。

例如查询显示 student 表和 grade 表中的几个列，SQL 语句如下：

```
select  student.学号,
studnet.姓名,grade.学号,grade.课程成绩
from  student,grade
```

列以表的名称作为前缀主要用于多个表连接。为了直观，即使有一个表，其中的列也可以用表的名称作为前缀。

例如查询 course 表中所有列的信息，可以用如下 SQL 语句：

```
select  student.*
from  student
```

3.12.3　使用 WHERE 子句设置查询条件

WHERE 子句是用来选取需要检索的记录。因为一个表通常会有数千条记录，在查询结果中，用户仅需其中的一部分记录，这时就需要使用 WHERE 子句指定一系列的查询条件。下面是 WHERE 子句最简单的语法：

```
SELECT<字段列表>
FROM<表名>
WHERE<条件表达式>
```

例如在 student 表中，查询年龄>20 的学生，SQL 语句如下：

```
select *
from  作者
where  年龄>20
```

为了实现许多不同种类的查询，WHERE 子句提供了丰富的搜索条件，下面总结了 5 个基本的搜索条件。

1．比较查询条件

比较查询条件由比较运算符连接表达式组成，系统将根据该查询条件的真假来决定某一条记录是否满足该查询条件，只有满足该查询条件的记录才会出现在最终的结果集中。比较运算符有：>、<、>=、<=、=、<>、!>、!<。

例如，在 grade 表中，查询“课程成绩”大于 90 分的，SQL 语句如下：

```
select   *
from   grade
where   课程成绩 > 90
```

例如在 student 表中，查询“年龄”在 20~22 之间（包括 20 和 22）的所有学生，SQL 语句如下：

```
select   *
from   student
where  年龄>=20 and  年龄<=22
```

2．范围查询条件

使用范围条件进行查询，用于返回位于两个给定的值之间的某个数据值。通常使用 BETWEEN…AND 和 NOT…BETWEEN…AND 来指定范围条件。

BETWEEN…AND 要包括两端的值，等价于比较运算符（>=...<=）。

例如在 grade 表中，显示年龄在 20~21 之间的学生信息，SQL 语句如下：

```
select   *
from   student
where  年龄  between 20 and 21
```

而 NOT…BETWEEN…AND 语句用于返回在两个指定值的范围以外的某个数据值，但并不包括两个指定的值。

例如在 student 表中，显示年龄不在 20~21 之间的学生信息，SQL 语句如下：

```
select   *
from   student
where  年龄    not between 20 and 21
```

3．列表查询条件

当测试一个数据值是否匹配一组目标值中的一个时，通常使用 IN 关键字来指定列表搜索条件。IN 关键字的格式是 IN（目标值 1，目标值 2，目标值 3，…），目标值的项目之间必须使用逗号分隔，并且括在括号中。

例如在 student 表中，查询“学号”是 B001,B003,B004 的学生信息，SQL 语句如下：

```
select   *
from   student
where    学号  in ('B001','B003', 'B004')
```

IN 运算符可以与 NOT 配合使用排除特定的行，以测试一个数据值是否不匹配任何目标值。

例如在 student 表中，查询学号不是 B001、B003 和 B004 的学生信息，SQL 语句如下：

```
select   *
from   student
where   学号  not   in ('B001','B003', 'B004')
```

4．模糊 LIKE 查询

有时用户对查询数据表中的数据了解得不全面，例如不能确定所要查询人的姓名只知道他姓李、查询某个人的联系方式只知道是以“3451”结尾等，这时就要使用 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 3.12 所示。

表 3.12　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]范围以外的任何单个字符

例如在 student 表中，查询姓“李”的学生信息，SQL 语句如下：

```
select *
from   student
where  姓名   like '李%'
```

例如在 student 表中，查询姓“刘”的并且名字只有两个字的学生信息，SQL 语句如下：

```
select *
from   student
where  姓名   like '刘_'
```

例如在 student 表中，查询联系方式以'3451'结尾并且开头数字位于 1~5 之间的学生信息，SQL 语句如下：

```
select *
from   student
where  联系方式      like     '[1-5]3451'
```

例如在 student 表中，查询联系方式以'3451'结尾，但不以 2 开头的学生信息。SQL 语句如下：

```
Select   *
from   student
where   联系方式   like   '[^2]3451'
```

3.12.4　使用 ORDER BY 子句对查询结果排序

使用 ORDER BY 子句可对查询返回的结果按一列或多列排序。语法如下：

```
[ ORDER BY {column_name [ ASC | DESC ] }  [ ,...n ] ]
```

其中 ASC 表示升序，DESC 为降序。

例如在 grade 表中，按照“课程成绩”升序排序，SQL 语句如下：

```
select *
from grade
order by 课程成绩
```

查询结果以降序排序，必须在列名后指定关键字 DESC。

例如在 grade 表中，按照“课程成绩”降序排序。SQL 语句如下：

```
Select  *
from  grade
order by 课程成绩 desc
```

3.12.5 使用 GROUP BY 子句将查询结果分组

GROUP BY 子句可以将表的行划分为不同的组，以便系统对满足条件的组返回结果，这样就可以控制想要看见的详细信息的级别。例如按照学生的性别分组、按照不同的学期分组等。

使用 GROUP BY 子句的注意事项：

- ☑ 在 SELECT 子句的字段列表中，除了聚集函数外，其他出现的字段一定要 GROUP BY 子句中有定义才行。例如“GROUP BY A,B”，那么“SELECT SUM(A),C”就有问题，因为 C 不在 GROUP BY 中，但是 SUM(A)还是可以的。
- ☑ SELECT 子句的字段列表中不一定要有聚集函数，但至少要用到 GROUP BY 子句列表中的一个项目。例如“GROUP BY A,B,C”，则“SELECT A”是可以的。
- ☑ 在 SQL Server 中 text、ntext 和 image 数据类型的字段不能作为 GROUP BY 子句的分组依据。
- ☑ GROUP BY 子句不能使用字段别名。

1．按单列分组

GROUP BY 子句可以基于指定某一列的值将数据集合划分为多个分组，同一组内所有记录在分组属性上具有相同值。

例如把 student 表按照“性别”这个单列进行分组，SQL 语句如下：

```
select  性别
from  student
group  by 性别
```

2．按多列分组

GROUP BY 子句可以基于指定多列的值将数据集合划分为多个分组。

例如在 student 表中，按照“性别”和“年龄”列进行分组，SQL 语句如下：

```
select 性别,年龄
from student
group by 性别,年龄
```

在 student 表中，首先按照“性别”分组，然后再按照“年龄”分组。

3．与 HAVING 一起使用

分组之前的条件要用 where 关键字，而分组之后的条件要使用关键字 having 子句。

例如在 student 表中，先按“性别”分组求出平均年龄，然后筛选出平均年龄大于 20 岁的学生信息，SQL 语句如下：

```
select   avg(年龄), 性别
from student
group   by  性别
having avg(年龄)>20
```

4．与聚集函数一起使用

当聚集函数与 GROUP BY 子句一起使用时，实际编程会有很大的意义。例如按性别统计年龄的平均值，按学期统计课程成绩的平均值等。

每个聚集函数都为每一组生成一个值，而不是对整个表生成一个值。

例如在 student 表中，按性别统计年龄的平均值，SQL 语句如下：

```
Select  性别,avg(年龄) as  平均年龄
from student
group by  性别
```

3.12.6　嵌套查询

嵌套查询是指在一个外层查询中包含有另一个内层查询，即一个 SELECT-FROM-WHERE 查询语句块可以嵌套在另一个查询块的 WHERE 子句中。其中外层查询称为父查询、主查询，而内层查询也称为子查询、从查询。

1．简单的嵌套查询

嵌套内层子查询通常作为搜索条件的一部分呈现在 WHERE 或 HAVING 子句中。例如把一个表达式的值和一个由子查询生成的一个值相比较，这个测试类似于简单比较测试。

子查询比较测试用到的运算符是：=、<>、<、>、<=、>=。子查询比较测试把一个表达式的值和由子查询产生的值进行比较，这时子查询只能返回一个值，否则错误。最后返回比较结果为 TRUE 的记录。

例如在 student 表中，查询“课程成绩”大于 96 分的学生信息，SQL 语句如下：

```
select   *
from   student
where  学号  = ( select  学号  from   grade where  课程成绩>96 )
```

子查询“select 学号　from　grade where 课程成绩>96”的查询结果是“B003”，仅仅就这一个值。

子查询过程：

（1）首先执行子查询，从 grade 表中查询出课程成绩>96 的学生学号为“B003”。

（2）然后把子查询的结果和外层查询的“学号”字段内容一一比较，从学生表中查询出学号是“B003”的学生信息。

2．带 in 的嵌套查询

带 in 的嵌套查询语法格式为：

```
WHERE  查询表达式  IN(子查询)
```

一些嵌套内层的子查询会产生一个值，也有一些子查询会返回一列值，即子查询不能返回带几行和几列数据的表，原因在于子查询的结果必须适合外层查询的语句。当子查询产生一系列值时，适合用带 in 的嵌套查询。

把查询表达式单个数据和由子查询产生的一系列的数值相比较，如果数值匹配一系列值中的一个，则返回 TRUE。

例如在 student 和 grade 表中，查询参加考试的学生信息，SQL 语句如下：

```
select  *
from  student
where  学号  in(select  学号  from  grade)
```

子查询生成 grade 表中“学号”列的数值，WHERE 子句检查主查询记录中的值是否与子查询结果中的数值匹配，如果匹配则返回 TRUE 值。由于主查询记录的“B006”的学号值与子查询结果的数值不匹配，所以查询结果不显示学号为“B006”的记录信息。

带 in 的内层嵌套还可以是多个值的列表。

例如查询年龄是 19、21、24 的学生信息，SQL 语句如下：

```
select *
from student
where  年龄  in (19 , 22 , 24)
```

3．带 some 和 any 的嵌套查询

SQL 支持 3 种定量比较谓词——some、any 和 all，都是用于判断是否任何或全部返回值都满足搜索要求。其中 some 和 any 谓词是存在量的，只注重是否有返回值满足搜索要求。这两种谓词含义相同，可以替换使用。some 的语法如下：

```
<表达式> {= 、<> 、!= 、> 、>= 、< 、<= 、!> 、!> 、!< } some (子查询)
```

例如在 studnet 表中，查询年龄小于平均年龄的所有学生信息，SQL 语句如下：

```
select *
from student
where  年龄<some
(select avg(年龄) from student)
```

some 是把每一行指定的列值与子查询的结果进行比较，如果哪行的比较结果为真，满足条件就返回该行。本实例嵌套的子查询结果就是一个数值，然后 student 学生表的每一条记录的年龄列值与之相比较，满足就返回行。

any 属于 SQL 支持的 3 种定量谓词之一，且和 some 完全等价，即能用 some 的地方完全可以使用

any。

下面用 any 解决上述示例。

例如在 student 表中，查询年龄小于平均年龄的所有学生的信息，SQL 语句如下：

```
select *
from student
where   年龄<any
(select avg(年龄) from student)
```

SQL 中定量谓词不支持反操作，也就是说，不能在 any 或者 some 前加 not 关键字。但可以用“<>”号表示否定。

例如在 student 表中，查询年龄不小于平均年龄的所有学生信息，SQL 语句如下：

```
select * from student   where   年龄  <> any   (select avg(年龄) from student)
```

4．带 all 的嵌套查询

all 谓词的使用方法和 any 或者 some 谓词一样，也是把列值与子查询结果进行比较，但是它不要求任意结果值的列值为真，而是要求所有列的查询结果都为真，否则就不返回行。

例如查询课程成绩不大于 90 分的课程信息，SQL 语句如下：

```
select *
from course
where  课程代号  <> all
(select  课程代号  from grade   where  课程成绩>90)
```

5．带 exists 的嵌套查询

exists 谓词只注重子查询是否返回行。如果子查询返回一个或多个行，谓词评价为真，否则为假。exists 搜索条件并不真正地使用子查询的结果，它仅仅测试子查询是否产生任何结果。

例如在 student 和 grade 表中，查询参加考试的学生信息，用 exists 完成嵌套查询，SQL 语句如下：

```
select *
from student
where   exists
(select   学号   from   grade where student.学号=grade.学号  )
```

exists 谓词子查询中的 select 子句中可使用任何列名，也可使用任意多个列，这种谓词只注重是否返回行，而不注重行的内容。用户可以规定任何列名或者只使用一个星号。例如上述例子的 SQL 语句和下面的 SQL 语句是完全等价的。实现的代码如下：

```
select *
from student
where   exists (select * from   grade where student.学号=grade.学号  )
```

not exists 的作用与 exists 正相反。如果子查询没有返回行，则满足 not exists 中的 where 子句。

例如查询没参加考试的学生信息，SQL 语句如下：

```
select   *
from   student
```

```
where  not exists
(select  *  from  grade  where  student.学号=grade.学号 )
```

3.12.7 多表查询

在程序开发过程中，不仅可以对单一数据表进行查询，还可以进行多表查询，从多个表中提取出需要的数据。

多表查询分为内连接查询、外连接查询以及合并查询。

1．内连接

内连接也叫连接，是最早的一种连接，还可以被称为普通连接或自然连接。内连接是从结果中删除其他被连接表中没有匹配行的所有行，所以内连接可能会丢失信息。

内连接的语法如下：

```
SELECT fieldlist
FROM table1 [INNER] JOIN table2
ON table1.column=table2.column
```

一个表中的行和另外一个表中的行匹配连接，表中的数据决定了如何对这些行进行组合，从每一个表中选取一行。

例如 student 表和 grade 表进行内连接查询，SQL 语句如下：

```
select  *
from  student
inner join  grade
on  student.学号=grade.学号
```

2．外连接

外连接则扩充了内连接的功能，会把内连接中删除原表中的一些保留下来，由于保留下来的行不同，因此把外连接又分为左外连接、右外连接和全外连接 3 种。

（1）左外连接

左外连接保留了第一个表的所有行，但只包含第二个表与第一表匹配的行。第二个表相应的空行被放入 NULL 值。

左外连接的语法如下：

```
SELECT fieldlist
FROM table1 left JOIN table2
ON table1.column= table2.column
```

例如把 student 表和 grade 表左外连接，第一个表 student 有不满足连接条件的，SQL 语句如下：

```
select  *
from student
left join  grade
on  student.学号=grade.学号
```

（2）右外连接

右外连接保留了第二个表的所有行，但只包含第一个表与第二个表匹配的行。第一个表相应空行被放入 NULL 值。

右外连接的语法如下：

```
SELECT fieldlist
FROM table1 right JOIN table2
ON table1.column=table2.column
```

例如把 grade 表和 course 表右外连接，第二个表 course 有不满足连接条件的行，SQL 语句如下：

```
select　*
from grade
right join　course
on　course.课程代号=grade.课程代号
```

3．合并查询

表的合并操作是将两个表的行合并到一单个表中，且不需要对这些行作任何更改。

在构造合并查询时必须遵循以下几条规则：

☑　所有查询中的列数和列的顺序必须相同。

☑　数据类型必须兼容。

☑　作为对所有 SELECT 语句的合并操作结果进行排序的 ORDER BY 子句，必须放到最后一个 SELECT 后面，但它所使用的排序列名必须是第一个 SELECT 选择列表中的列名。

例如合并查询“select　姓名, 年龄　from　student　where 年龄>21 ”和“select　学号,学期　from grade　where 学期=1　order by　年龄 ”，按“年龄”列排序，SQL 语句如下：

```
select　姓名,年龄　from　student　where 年龄>21　union　all　select　学号,学期　from　grade　where 学期=1 order by　年龄
```

可以把很多表进行合并，表的数量可达十多个，但仍要遵循合并表时的规则。

3.12.8　添加数据

要向表中添加新的记录，则需要使用 SQL INSERT 语句实现。其语法如下：

```
INSERT[INTO]　{table_name |view_name}[(column_list)] { DEFAULT VALUES |
Values_list |select_statement}
```

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。

1．为指定的列添加数据

用户为一个表插入数据时，由于某些列值未知，只能为部分列插入值，这时表名后的 column_list

列名表参数一定不能省略，指出为哪些列插入值，并且和 values 后面列值的顺序相对应。

例如在 student1 表中，向姓名、年龄、联系方式几个字段添加数据，SQL 语句如下：

```
insert into student1(姓名,年龄,联系方式)
values('小小葱',02,'1472')
```

对于没有指定数据的列会有下列 4 种情况：

（1）如果列设置了标识（IDENTITY）属性，那么将填入自动编号。

（2）如果列设置有默认值，则填入默认值。

（3）如果列“允许 NULL”，则填入 NULL。

列既设有默认值，也“允许 NULL”，则省略时，会填入默认值。

（4）如果前几项都不符合时，不输入任何数据，则会显示错误提示而取消操作。

2．为所有的列添加数据

用户可以给数据表的所有列都插入值，即 values 后要包含所有列的值。而表名后的 column_list 列名表参数有两种写情况：一种依然列出所有的列；另一种省略列名表。

例如，在 student1 表中，添加一条完整记录，即给表的所有列添加数据，并显示插入完数据的 student1 表信息，SQL 语句如下：

```
insert into student
values('B008','木葱葱','女',23,'1988-01-01','12345')
```

3．添加批量数据

INSERT 语句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。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 ]
```

例如给 grade 表的“学号”和“课程代号”列添加批量数据，SQL 语句如下：

```
insert    into grade(学号,课程代号)    select    姓名,年龄  from student
```

上例是为表的部分列添加了数据。

3.12.9　修改数据

修改数据用 UPDATE 语句。可以修改一个列或者几个列中的值，但一次只能修改一个表。

UPDATE 命令的语法如下：

```
UPDATA     { table_name |view_name    }    SET [{table_name|view_name} ]
    {column_list={variable|variable_and_column_list}
    [,{column_list2={variable|variable_and_column_list2}…
    [,{column_listn={variable|variable_and_column_listn} ] ]
       [WHERE clause]
```

经常用 UPDATE 语句修改数据表中某一行中的数据，用 WHERE 子句限定某一行。如果 set 子句后只为一个列赋值，那么就是修改表中某一行中的某一列的值。

例如在 student1 表中，把姓名是“李小葱”的学生年龄减 1，SQL 语句如下：

```
update student
set  年龄=年龄-1
where  姓名='李小葱'
```

UPDATE 语句中的 WHERE 子句是可选的。如果省略了 WHERE 子句，那么目标表的所有列都将被更新。

例如在 student 表中，将所有学生的年龄减 1，SQL 语句如下：

```
update student
set  年龄=年龄-1
```

3.12.10　删除数据

要从表中删除一个或多个记录，则需要使用 SQL DELETE 语句实现。DELETE 语句的语法如下：

```
DELETE   [ FROM ] { table_name | view_name} [ WHERE clause ]
```

1．用 WHERE 子句删除部分数据

DELETE 语句从一个表中删除所选的数据记录。其中 FROM 子句指定了包含记录的目标表；WHERE 子句指定了表中的哪些记录要被删除。

例如在 student 数据库中，删除 studet 表中所有女生的记录，SQL 语句如下：

```
delete from student
where  性别= '女'
```

2．删除表中所有数据

DELETE 语句中的 WHERE 子句是可选的，如果在 DELETE 语句中省略了 WHERE 子句，目标表中的所有记录都将被删除。

例如在 student 表中，删除表中所有记录，SQL 语句如下：

```
delete from student
```

student 表中的所有记录都被删除，但 student 表的定义和它的字段存储在数据库中，用 INSERT 语句可以为这个表添加新的记录。要想彻底删除这个表的定义，必须使用 DROP TABLE 语句。

例如彻底删除 student 表的所有记录和这个表的定义，必须使用 DROP TABLE 语句：

```
drop table student
```

3．删除表中多余的重复行数据

在关系数据库中，允许创建包含重复行的表。这表示在每一列中可以有相同数值的两个或更多的行。为了避免在表中出现重复行，通常给表添加一个主键，即不允许出现重复行。

允许在一个表中出现重复行，是为了表示数据方便，但是也给使用这个表的人容易造成误解，而

且 SQL Server 不允许直接用手动方式删除重复数据。下面就讲解如何删除有重复行的数据。

例如在 student 表中，删除 student 表中多余的重复行的数据，即保留有重复行数据其中的一行，SQL 语句如下（一次只能执行一行数据）：

```
select distinct  *  into temp from student
delete from student
insert into student  select * from  temp
```

3.13 本章小结

本章详细介绍了小区物业内部管理网的开发过程。通过学习本章，读者可以了解如何通过系统目标设计业务流程和数据库，如何合理组织系统文件结构。通过主要模块的实现过程，读者可以深入了解用户控件技术的应用，聚合函数 COUNT 的应用，利用 Table 控件动态添加行和单元格，将数据控件中的日期以编程的方式转换为指定格式显示出来。希望本章所介绍的内容对读者今后在开发项目的过程中会有所启发和帮助。