任务 4 留言簿

- 分析留言簿的页面功能
- 实现 DataList 的数据绑定和数据分页
- 实现留言簿中的各个页面

留言簿是企业网站中的一个基本功能，企业网站可以通过留言簿，获取浏览者的留言信息，并及时反馈给企业，加强企业与浏览者之间的互动与交流，不断改进企业的服务水平，更好地为顾客服务。

在留言簿任务中，主要实现一个功能较为简单的留言簿。首先说明留言簿的总体结构以及各页面的功能分析；然后介绍实现 DataList 数据绑定的两种方式，如何通过 DataPagedSource 类实现 DataList 数据的分页；最后说明如何逐步实现留言簿中的各个页面。

4.1 实训 1——页面功能分析

留言簿是一个功能较为单一、简单的网站，主要由 4 个页面组成，如图 4-1 所示。

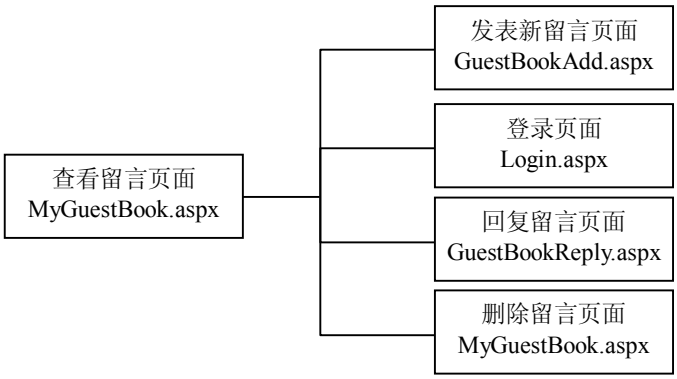

图 4-1 留言簿的总体结构

在查看留言页面 MyGuestBook.aspx 中，分页显示各种留言；如果需要发表新留言，进入发表新留言页面 GuestBookAdd.aspx 中，就可以发表新的留言。

在查看留言页面 MyGuestBook.aspx 中，如果管理员单击相关留言的回复链接，就会打开回复留言页面 GuestBookRelpy.aspx，从而针对指定的留言，发表回复信息。

在查看留言页面 MyGuestBook.aspx 中，如果管理员单击相关留言的删除链接，就会删除指定的留言。

这里需要说明的是，普通浏览者可以浏览查看留言页面 MyGuestBook.aspx，可以在发表新留言页面 GuestBookAdd.aspx 发表新留言，但只有管理员才能回复留言以及删除留言，管理员需要到登录页面 Login.aspx 中，输入正确的用户名和密码，才能进入回复留言页面 GuestBookRelpy.aspx 中回复留言，并在查看留言页面 MyGuestBook.aspx 中删除指定的留言。

因此，在留言簿网站中，实现的主要功能有：浏览者查看留言、浏览者发表新留言以及管理员回复留言和删除留言。

4.1.1 查看留言页面

查看留言页面 MyGuestBook.aspx 的运行界面，如图 4-2 所示。

图 4-2　查看留言界面

从图 4-2 中可以看出，留言通过 DataList 数据显示控件来显示，并且每页显示两条留言，在 DataList 数据显示控件的下方设置了专门的分页导航按钮。

4.1.2　发表新留言页面

发表新留言页面 GuestBookAdd.aspx 的运行界面，如图 4-3 所示。

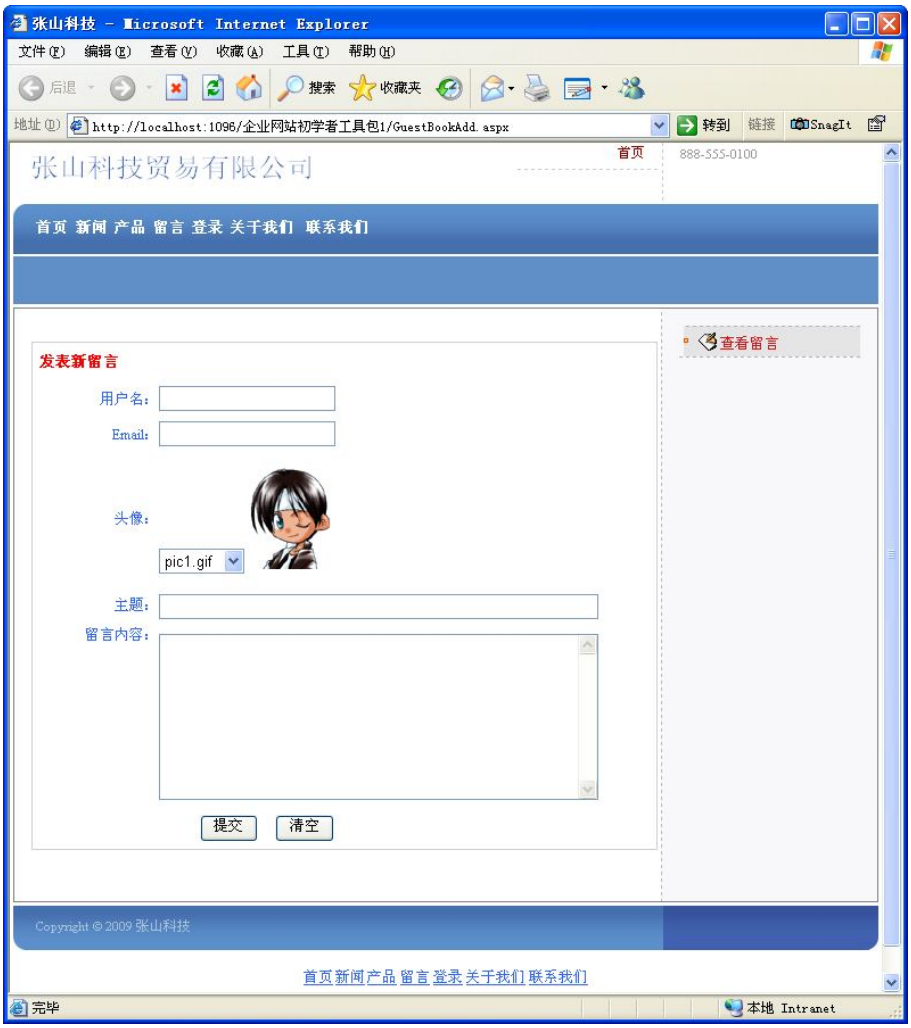

图 4-3　发表新留言界面

从图 4-3 中可以看出，浏览者要发表新留言，只需要填写用户名、Email 地址，选择自己喜欢的头像，输入留言的主题以及留言内容，然后单击“提交”按钮，即可发表新的留言。

4.1.3　登录页面

如图 4-4 所示是登录页面 Login.aspx 的运行界面。

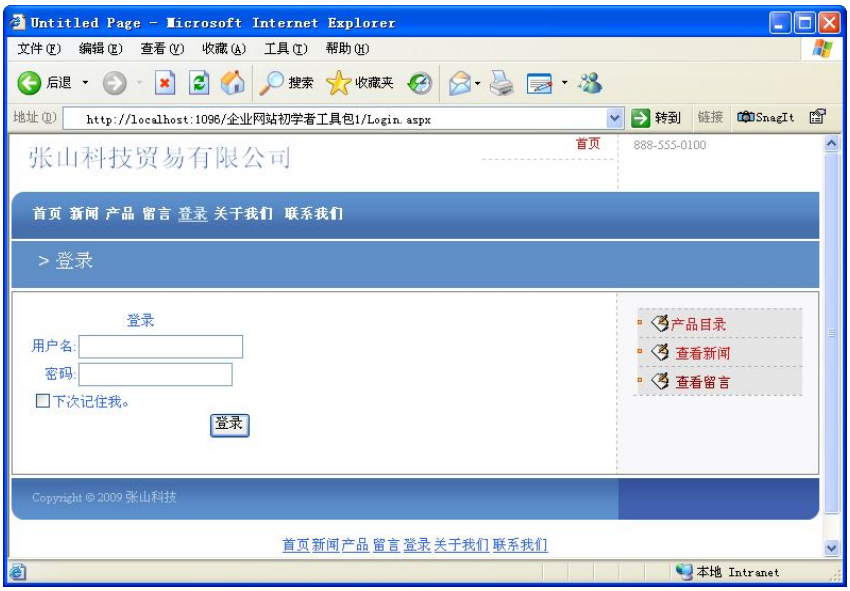

图 4-4　登录界面

在图 4-4 中，使用登录控件 Login 来验证注册用户，而注册用户的相关注册信息，如用户名、密码等，则是存储在名称为 AspNetDB 的数据库的相关数据表中。

注册用户登录成功后，就可以以管理员身份访问回复留言页面 GuestBookReply.aspx；否则就会提示用户输入正确的用户名、密码。

4.1.4　回复留言页面

回复留言页面 GuestBookReply.aspx 的运行界面，如图 4-5 所示。

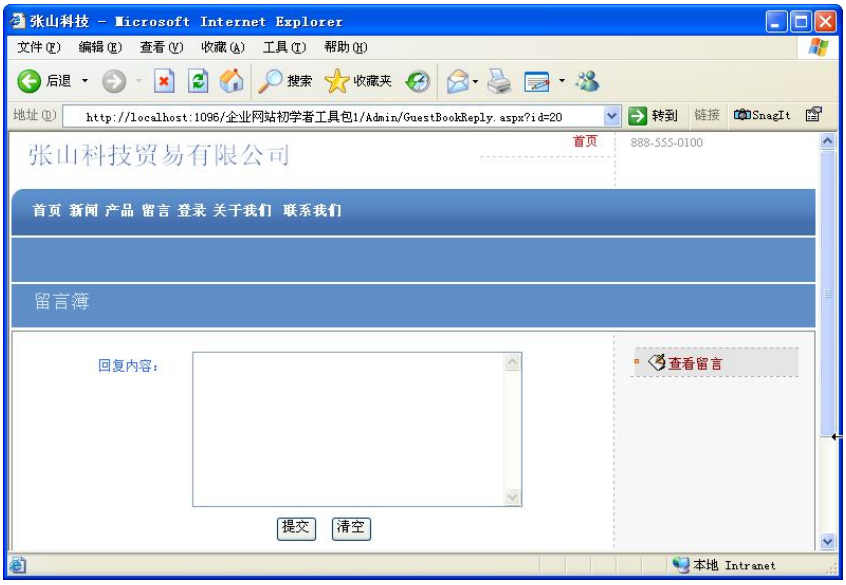

图 4-5　回复留言界面

从图 4-5 中可以看出，管理员只要输入回复内容，然后单击“提交”按钮，即可针对指定的留言，发表回复内容，并将回复内容显示在该条留言的下方。

4.2　实训 2——DataList 的数据绑定和数据分页

为实现上述的留言簿网站，下面说明在使用 DataList 控件时，实现数据绑定的两种方式，以及如何通过 DataPagedSource 类实现 DataList 数据的分页。

4.2.1　DataList 的数据绑定

在实现 DataList 的数据绑定时，可以使用页面代码绑定方式，也可以使用后置代码表示方式。

1．页面代码绑定

打开“实训 2 开始版本”中的 DataList 网站，拖放 SqlDataSource 控件到 Default.aspx 页面，后面的步骤与任务 3 的 3.1.3 节中的设置数据源 SqlDataSource 控件内容完全一样，然后拖放 DataList 数据显示控件到 Default.aspx 页面的指定位置，单击 DataList 控件右上方的智能标记，打开如图 4-6 所示的任务菜单。

图 4-6　DataList 任务菜单

在图 4-6 中，选择上面已经配置好的数据源 SqlDataSource1，此时 DataList 数据显示控件就会绑定该数据源中的数据，该页面的运行界面如图 4-7 所示。

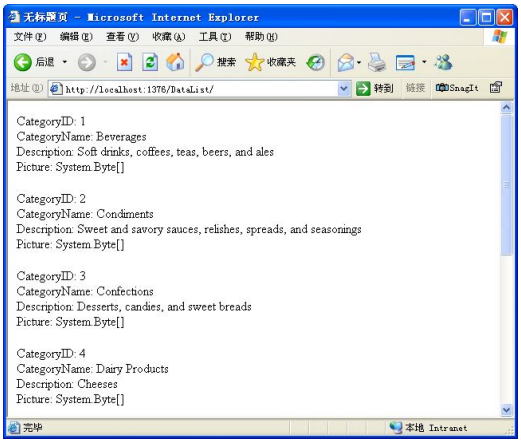

图 4-7　DataList 的运行界面

上述页面显示了数据表 Categories 中的数据，实现的实际上是一个单层架构，DataList 数据显示控件绑定了数据源控件 SqlDataSource1，而数据源控件的设置是通过 HTML 语言嵌入浏览者页面中，也就是说，在显示页面中书写了与数据库绑定的代码。

2．后置代码绑定

在 Default.aspx 页面中，复制 DataList 数据显示控件的设置代码到 DataList.aspx 页面中，并在页面 DataList.aspx 的后置代码中，添加如代码 4-1 所示的后置绑定代码。

代码 4-1 是页面 DataList.aspx 中后置代码绑定的实现代码。

代码 4-1　后置代码绑定的实现代码

```
 1: public partial class DataList : System.Web.UI.Page
 2: {
 3:   protected void Page_Load(object sender, EventArgs e)
 4:   {
 5:     DataList1.DataSource = GetList();
 6:     DataList1.DataBind();
 7:   }
 8:
 9:   private List<Categories> GetList()
10:   {
11:     List<Categories> list = new List<Categories>();
12:
13:     SqlConnection myConnection = new SqlConnection(
                        ConfigurationManager.ConnectionStrings[
                            "SQLConnectionString"].ConnectionString);
14:
15:     string mySql = "select * from Categories  ";
16:     SqlCommand myCommand = new SqlCommand(mySql, myConnection);
17:
18:     myConnection.Open();
19:     SqlDataReader myReader = null;
20:
21:     myReader = myCommand.ExecuteReader();
22:
23:     while (myReader.Read())
24:     {
25:       Categories temp = new Categories((int)myReader["CategoryID"],
                             (string)myReader["CategoryName"],
                             (string)myReader["Description"]);
26:       list.Add(temp);
27:     }
28:
29:     myReader.Close();
30:     myConnection.Close();
```

```
31:
32:     return list;
33:   }
34: }
```

在上述代码中，第 9 行到第 33 行所定义的 GetList()方法，返回数据表 Categories 的实例化对象列表 List<Categories>。其中的代码采用 ADO.NET 数据访问技术，获得数据表 Categories 的相关查询记录，可以划分为以下 5 个步骤：

第 1 个步骤是获得数据库连接对象 myConnection（代码第 13 行），其中通过读取配置文件 Web.config 中的数据库连接字符串，构造实例化的数据库连接对象 myConnection。

第 2 个步骤是构造 SQL 语句实例化对象 myCommand（代码第 15 行、第 16 行），其中传入了查询 SQL 语句 mySql。

第 3 个步骤是打开数据库连接，执行数据操作（代码第 18 行到第 21 行），这里执行的是 SQL 语句的查询操作。

第 4 个步骤是处理结果对象 myReader（代码第 23 行到第 27 行）。在循环语句中，通过结果对象 myReader 中的 Read()方法，遍历数据库的相关信息，构造业务类 Categories 的实例化对象 temp，并保存到列表 list 中。

第 5 个步骤是关闭 myReader 对象以及数据库连接 myConnection 对象，以便系统释放资源，提高系统的运行效率。

代码第 5 行、第 6 行设置数据显示控件 DataList1 的数据源和数据绑定。

采用后置绑定代码绑定数据的 DataList.aspx 页面的运行界面，如图 4-8 所示。

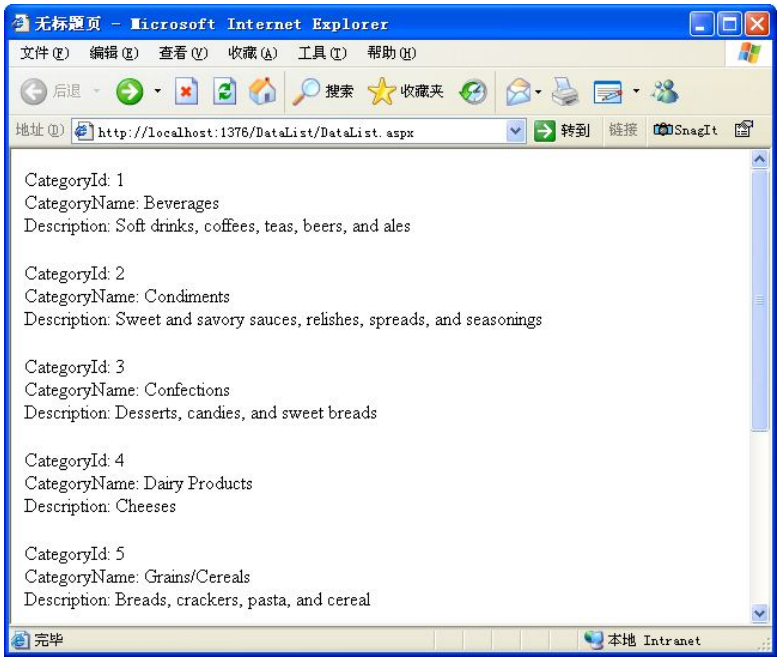

图 4-8　DataList 的运行界面

这里需要说明的是，通过页面代码绑定数据方式实现的 Default.aspx 页面使用了

SqlDataSource 数据源控件和数据显示控件 DataList，实际上是一个单层架构；而通过后置代码绑定数据方式实现的 DataList.aspx 页面并没有使用 SqlDataSource 数据源控件，但使用了同样代码的数据显示控件 DataList，后置代码所起的作用相当于一个 SqlDataSource 数据源控件。这种后置代码绑定数据方式，将数据的访问从页面中转移到后置代码中，便于页面功能的扩展。

4.2.2 DataList 的数据分页

数据显示控件 DataList 本身没有提供分页功能，但开发者可以通过使用 DataPagedSource 类为数据显示控件 DataList 提供分页功能。

1．DataPagedSource 类

DataPagedSource 类位于命名空间 System.Web.UI.WebControls，其主要功能是为相关的数据显示控件，如 DataList 控件，提供数据的分页功能。

DataPagedSource 类中的主要属性，如表 4-1 所示。

表 4-1 DataPagedSource 类中的主要属性列表

属性名称	说明
AllowPaging	允许在数据显示控件中，启用分页功能
CurrentPageIndex	当前页的索引，第 1 页的索引为 0
DataSource	设置需要分页的数据源
PageCount	页面总数
PageSize	每页所显示的记录数
IsFirstPage	当前页是否为首页
IsLastPage	当前页是否为最后一页

2．DataList 的数据分页

要实现为数据显示控件 DataList 提供分页功能，只需要将 DataList 的数据源绑定到 PagedDataSource 类的实例化对象即可，很显然，此时实现数据显示控件 DataList 绑定数据的方式，应该是后置代码绑定数据方式。

下面是实现 DataList 数据分页后置代码绑定的关键代码。

```
PagedDataSource pds= new PagedDataSource();
DataList1.DataSource = pds;
DataList1.DataBind();
```

在上述代码中，将 DataList1 数据显示控件的数据源，绑定到 PagedDataSource 类的实例化对象 pds，使得 DataList1 控件具有分页功能，至于具体的分页功能，则需要设置上述表 4-1 中的相关分页属性。

代码 4-2 是 DataListWithPager 类的实现代码。

代码 4-2 DataListWithPager 类的实现代码

```
1: public partial class DataListWithPager : System.Web.UI.Page
2:
3:   PagedDataSource pds;
4:
5:   protected void Page_Load(object sender, EventArgs e)
6:   {
7:     pds= new PagedDataSource();
8:     pds.AllowPaging = true;
9:     pds.PageSize = 3;
10:    pds.DataSource = GetList();
11:
12:    if (!IsPostBack)
13:    {
14:      MessageTotal.Text = GetPageCount().ToString();
15:
16:      PageTotal.Text = pds.PageCount.ToString();
17:      PageCurrent.Text = (pds.CurrentPageIndex + 1).ToString();
18:
19:      Binding();
20:
21:      for (int i = 1; i <= pds.PageCount; i++)
22:        DropDownList1.Items.Add(i.ToString());
23:    }
24:  }
25:
26:  private List<Categories> GetList()
27:  {
28:    ……
29:  }
30:
31:  private  int GetPageCount()
32:  {
33:    SqlConnection myConnection = new SqlConnection(
                  ConfigurationManager.ConnectionStrings[
                     "SQLConnectionString"].ConnectionString);
34:
35:    string mySql = "select count(*) from Categories ";
36:    SqlCommand myCommand = new SqlCommand(mySql, myConnection);
37:
38:    myConnection.Open();
39:
40:    int tempInt = Convert.ToInt32(myCommand.ExecuteScalar());
41:
42:    return tempInt;
43:  }
```

```
44:
45:   public void Binding()
46:   {
47:     DataList1.DataSource = pds;
48:     DataList1.DataBind();
49:
50:     Previous.Enabled = true;
51:     Next.Enabled = true;
52:
53:     if (pds.CurrentPageIndex == 0)
54:       Previous.Enabled = false;
55:
56:     if (pds.CurrentPageIndex == pds.PageCount - 1)
57:       Next.Enabled = false;
58:   }
59:
60:   protected void Previous_Click(object sender, EventArgs e)
61:   {
62:     pds.CurrentPageIndex = Convert.ToInt32(PageCurrent.Text) - 2;
63:
64:     PageCurrent.Text = (pds.CurrentPageIndex + 1).ToString();
65:     Binding();
66:   }
67:
68:   protected void Next_Click(object sender, EventArgs e)
69:   {
70:     pds.CurrentPageIndex = Convert.ToInt32(PageCurrent.Text);
71:
72:     PageCurrent.Text = (pds.CurrentPageIndex + 1).ToString();
73:     Binding();
74:   }
75:
76:   protected void First_Click(object sender, EventArgs e)
77:   {
78:     pds.CurrentPageIndex = 0;
79:     PageCurrent.Text = "1";
80:     Binding();
81:   }
82:
83:   protected void Last_Click(object sender, EventArgs e)
84:   {
85:     PageCurrent.Text = PageTotal.Text;
86:     pds.CurrentPageIndex = pds.PageCount - 1;
87:     Binding();
88:   }
```

```
89:
90:   protected void Go_Click(object sender, EventArgs e)
91:   {
92:     pds.CurrentPageIndex = Convert.ToInt32(DropDownList1.SelectedValue)
                                        - 1;
93:     Binding();
94:   }
95: }
```

在上述代码中，第 5 行到第 24 行所定义的 Page_Load()方法，主要设置实例化对象 pds，以便使用 DataList 控件实现数据分页。第 8 行设置允许使用 pds 对象分页，第 9 行设置分页的记录数为 3，第 10 行设置数据源对象。第 14 行显示记录的总数量，第 16 行显示记录页面的总数量，第 17 行则显示当前的页面数，需要说明的是，当前页面 CurrentPageIndex 是从零开始计数的。

代码第 31 行到第 43 行所定义的 GetPageCount()方法，获得 Categories 数据表中的数据记录总数。

代码第 45 行到第 58 行所定义的 Binding()方法，实现 DataList1 控件的数据绑定，实现 DataList1 控件的数据分页。

代码第 60 行到第 66 行主要实现“上一页”按钮的单击操作；代码第 68 行到第 74 行主要实现“下一页”按钮的单击操作；代码第 76 行到第 81 行主要实现“首页”按钮的单击操作；代码第 83 行到第 88 行主要实现“尾页”按钮的单击操作；代码第 90 行到第 94 行主要实现 GO 按钮的单击操作。

DataList 数据分页的运行界面，如图 4-9 所示。

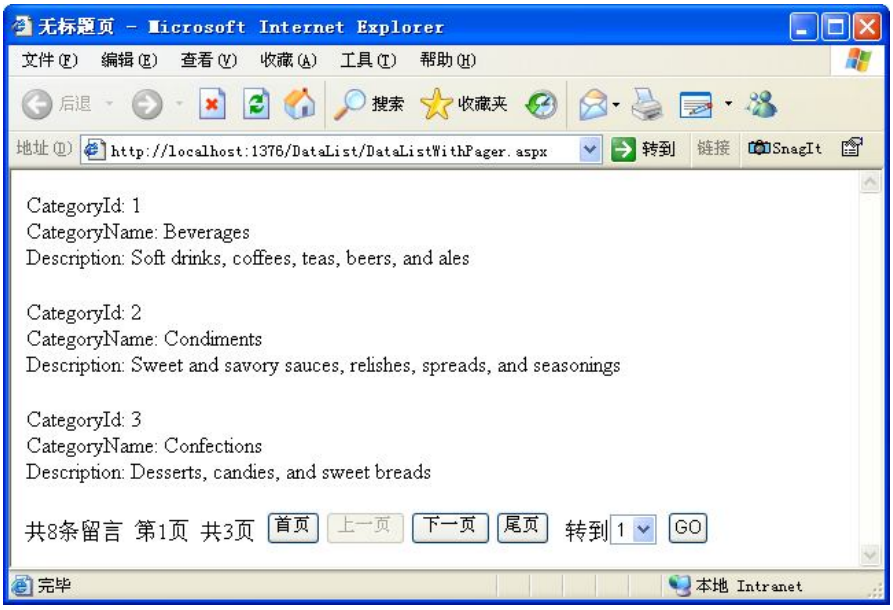

图 4-9　DataList 数据分页的运行界面

4.3 实训 3——留言簿的实现

要实现上述的留言簿，首先需要设计相关的数据库，然后针对数据库中的数据表以及留言簿的功能，构建业务类和中间数据访问层，最后针对各个页面的功能，在后置绑定代码文件中开发相关代码。

4.3.1 数据库设计

数据库设计包括数据表的设计以及存储过程的设计。

1．数据表

数据表 GuestBook 的结构，如图 4-10 所示。

GuestBook

列名	数据类型	允许 Null
ID	int	☐
UserName	varchar(20)	☑
Subject	varchar(50)	☑
Email	varchar(50)	☑
Message	varchar(3000)	☑
Reply	varchar(3000)	☑
PostTime	datetime	☑
ImageUrl	varchar(50)	☑
		☐

图 4-10　GuestBook 数据表的结构

在数据表 GuestBook 中包括 8 个字段，分别是留言 ID、用户名 UserName、主题 Subject、电子邮件 Email、留言内容 Message、回复内容 Reply、发表留言时间 PostTime 以及头像地址 ImageUrl。

其中留言 ID 被设置为主键。

2．存储过程

存储过程是一个被命名的存储在数据库服务器上的 SQL 语句的集合，是对业务逻辑的一种封装，它被越来越广泛地运用在实际的 Web 应用程序中。

在留言簿 Web 应用程序中，封装了 3 个存储过程，分别实现留言的查询、修改和添加以及删除功能，详细说明如表 4-2 所示。

表 4-2　留言簿中定义的存储过程

存储过程名称	说明
sprocGuestBookSelectSingleItem	查询指定编号的留言
sprocGuestBookUpdateSingleItem	修改或者添加新的留言
sprocGuestBookDeleteItem	删除指定编号的留言

4.3.2　构建业务类

留言簿中包括一个数据表 GuestBook，因此需要构建业务类 GuestBook 以及列表类 GuestBookList。

1．GuestBook 类

GuestBook 类的 UML 类图，如图 4-11 所示。

图 4-11　GuestBook 类的 UML 类图

从图 4-11 中可以看出，GuestBook 类中设置了 8 个属性，对应数据表 GuestBook 中的各个字段，但需要说明的是，业务类中各个属性的设置需要根据业务的需求来设定，有时需要添加一些属性，有时需要减少一些属性。

代码 4-3 是 GuestBook 类的实现代码。

代码 4-3　GuestBook 类的实现代码

```
1: public class GuestBook
2: {
3:   private int id = -1;
4:   private string userName = string.Empty;
5:   private string subject = string.Empty;
6:   private string email = string.Empty;
7:   private string message = string.Empty;
8:   private string reply = string.Empty;
```

```
 9:  private string postTime = string.Empty;
10:  private string imageUrl = string.Empty;
11:
12:  public int Id
13:  {
14:    get { return id; }
15:    set { id = value; }
16:  }
17:
18:  public string UserName
19:  {
20:    get { return userName ; }
21:    set { userName  = value; }
22:  }
23:
24:  public string Subject
25:  {
26:    get { return subject ; }
27:    set { subject = value; }
28:  }
29:
30:  public string Email
31:  {
32:    get { return email ; }
33:    set { email = value; }
34:  }
35:
36:  public string Message
37:  {
38:    get { return message ; }
39:    set { message = value; }
40:  }
41:
42:  public string Reply
43:  {
44:    get { return reply ; }
45:    set { reply = value; }
46:  }
47:
48:  public string PostTime
49:  {
50:    get { return postTime ; }
51:    set { postTime = value; }
52:  }
53:
54:  public string ImageUrl
```

```
55:   {
56:     get { return imageUrl ; }
57:     set { imageUrl  = value; }
58:   }
59: }
```

上述代码的实现比较简单，主要设置了 GuestBook 类中 8 个属性的读写器，并且在第 3 行到第 10 行设置了这些属性的默认值。其中需要说明的是，私有变量 id 的默认值被设置为-1，在实现数据的更新或者添加操作时，只要判断该 id 变量的默认值是否为-1，如果是-1，则执行添加数据的操作；否则就执行更新数据的操作。

2．GuestBookList 类

GuestBookList 类的 UML 类图，如图 4-12 所示。

图 4-12　GuestBookList 类的 UML 类图

从图 4-12 中可以看出，GuestBookList 类继承于泛型列表 List<GuestBook>，它只包括一个构造函数。

代码 4-4 是 GuestBookList 类的实现代码。

代码 4-4　GuestBookList 类的实现代码

```
1: public class GuestBookList: List <GuestBook >
2: {
3:   public GuestBookList()
4:   {
5:   }
6: }
```

从上述代码中可以看出，GuestBookList 类的实现代码非常简单，该列表是一个 GuestBook 类的实例化泛型列表。

4.3.3　构建中间数据访问层

根据留言簿的功能，在构建中间数据访问层时，不仅需要实现留言数据表中数据的各种基本操作，还需要实现专门的方法以及设置相关的分页数据信息，GuestBookManager 类的 UML 类图，如图 4-13 所示。

图 4-13　GuestBookManager 类的 UML 类图

从图 4-13 中可以看出，GuestBookManager 类中设置了数据的各种基本操作方法，如 Delete()方法、GetList()方法和 Save()方法；实现了专门的 Reply()方法，以便提交管理员的回复留言；设置了 GetPageCount()方法和 SetPage()方法，获得或者设定分页数据信息。

GuestBookManager 类的实现代码，如代码 4-5 所示。

代码 4-5　GuestBookManager 类的实现代码

```
1: public class GuestBookManager
2: {
3:   public static GuestBookList GetList()
4:   {
5:     GuestBookList tempList = null;
6:
7:     SqlConnection myConnection = new SqlConnection(
                ConfigurationManager.ConnectionStrings[
                "SQLConnectionString"].ConnectionString);
8:
9:     string mySql="select * from GuestBook  ";
10:    SqlCommand myCommand = new SqlCommand(mySql, myConnection);
11:
12:    myConnection.Open();
13:    SqlDataReader myReader = null;
14:
15:    myReader = myCommand.ExecuteReader();
16:
17:    if (myReader.HasRows)
18:    {
19:      tempList = new GuestBookList();
20:
21:      while (myReader.Read())
22:        tempList.Add(FillDataRecord (myReader ));
23:    }
24:
```

```
25:     myReader.Close();
26:     myConnection.Close();
27:
28:     return tempList;
29:
30:   }
31:
32:   public static int Save(GuestBook myGuestBook)
33:   {
34:     int result = 0;
35:
36:     SqlConnection myConnection = new SqlConnection(
                ConfigurationManager.ConnectionStrings[
                "SQLConnectionString"].ConnectionString);
37:
38:     SqlCommand myCommand = new SqlCommand(
                        "sprocGuestBookUpdateSingleItem", myConnection);
39:     myCommand.CommandType = CommandType.StoredProcedure;
40:
41:     if (myGuestBook.Id == -1)
42:       myCommand.Parameters.AddWithValue("@id", DBNull.Value);
43:     else
44:       myCommand.Parameters.AddWithValue("@id",myGuestBook.Id );
45:
46:     myCommand.Parameters.AddWithValue("@email", myGuestBook.Email );
47:     myCommand.Parameters.AddWithValue("@userName",
                                    myGuestBook.UserName );
48:     myCommand.Parameters.AddWithValue("@subject",
                                    myGuestBook.Subject) ;
49:     myCommand.Parameters.AddWithValue("@message",
                                    myGuestBook.Message );
50:     myCommand.Parameters.AddWithValue("@reply", myGuestBook.Reply );
51:     myCommand.Parameters.AddWithValue("@imageUrl",
                                    myGuestBook.ImageUrl );
52:     myCommand.Parameters.AddWithValue("@postTime",
                                    System.DateTime.Now   );
53:
54:     DbParameter returnValue = myCommand.CreateParameter();
55:     returnValue.Direction = ParameterDirection.ReturnValue;
56:     myCommand.Parameters.Add(returnValue );
57:
58:     myConnection.Open();
59:
60:     myCommand.ExecuteNonQuery();
61:     result = Convert.ToInt32(returnValue.Value  );
62:
```

```
63:     myConnection.Close();
64:
65:     return result;
66:   }
67:
68:   public static bool Delete(int id)
69:   {
70:     int result = 0;
71:
72:     SqlConnection myConnection = new SqlConnection(
                ConfigurationManager.ConnectionStrings[
                "SQLConnectionString"].ConnectionString);
73:
74:     SqlCommand myCommand = new SqlCommand("sprocGuestBookDeleteItem",
                                        myConnection);
75:     myCommand.CommandType = CommandType.StoredProcedure;
76:     myCommand.Parameters.AddWithValue("@id", id);
77:
78:     myConnection.Open();
79:
80:     result = myCommand.ExecuteNonQuery();
81:
82:     myConnection.Close();
83:
84:     return result > 0;
85:   }
86:
87:   public static bool Reply(int id, string reply)
88:   {
89:     bool result;
90:     SqlConnection myConnection = new SqlConnection(
                ConfigurationManager.ConnectionStrings[
                "SQLConnectionString"].ConnectionString);
91:
92:     SqlCommand myCommand = new SqlCommand();
93:
94:     myCommand.Connection = myConnection;
95:     myCommand.CommandText=" update GuestBook set reply=@reply
                                    where id=@id";
96:
97:     myCommand.Parameters.AddWithValue("@id", id);
98:     myCommand.Parameters.AddWithValue("@reply", reply );
99:
100:    myConnection.Open();
101:
102:    if (Convert.ToInt32(myCommand.ExecuteNonQuery()) > 0)
```

```
103:      result  = true;
104:    else
105:      result  = false;
106:
107:    myConnection.Close();
108:
109:    return result ;
110:   }
111:
112:   public static PagedDataSource SetPage()
113:   {
114:     PagedDataSource pagedDS = new PagedDataSource();
115:     pagedDS.AllowPaging = true;
116:     pagedDS.PageSize = 2;
117:     pagedDS.DataSource =GetList ();
118:
119:     return pagedDS ;
120:   }
121:
122:   public static int GetPageCount()
123:   {
124:     SqlConnection myConnection = new SqlConnection(
                ConfigurationManager.ConnectionStrings[
                "SQLConnectionString"].ConnectionString);
125:
126:     string mySql = "select count(*) from GuestBook ";
127:     SqlCommand myCommand = new SqlCommand(mySql, myConnection);
128:
129:     myConnection.Open();
130:
131:     int tempInt =Convert.ToInt32 ( myCommand.ExecuteScalar());
132:
133:     return tempInt;
134:
135:   }
136:
137:   private static GuestBook FillDataRecord(IDataRecord myDataRecord)
138:   {
139:     GuestBook myGuestBook = new GuestBook();
140:
141:     myGuestBook.Id = myDataRecord.GetInt32(
                                myDataRecord.GetOrdinal ("Id") );
142:     int indexUserName = myDataRecord.GetOrdinal("UserName");
143:     if (myDataRecord.IsDBNull(indexUserName))
144:       myGuestBook.UserName = "";
145:     else
```

```
146:      myGuestBook.UserName = myDataRecord.GetString(indexUserName);
147:
148:    int indexSubject = myDataRecord.GetOrdinal("Subject");
149:    if ( myDataRecord.IsDBNull (indexSubject )  )
150:      myGuestBook.Subject ="" ;
151:    else
152:      myGuestBook.Subject = myDataRecord.GetString(indexSubject );
153:
154:    int indexEmail = myDataRecord.GetOrdinal("Email");
155:    if (myDataRecord.IsDBNull(indexEmail))
156:      myGuestBook.Email  = "";
157:    else
158:      myGuestBook.Email  = myDataRecord.GetString(indexEmail);
159:
160:    int indexMessage = myDataRecord.GetOrdinal("Message");
161:    if (myDataRecord.IsDBNull(indexMessage))
162:      myGuestBook.Message  = "";
163:    else
164:      myGuestBook.Message  = myDataRecord.GetString(indexMessage);
165:
166:    int indexReply = myDataRecord.GetOrdinal("Reply");
167:    if (myDataRecord.IsDBNull(indexReply))
168:      myGuestBook.Reply  = "";
169:    else
170:      myGuestBook.Reply  = myDataRecord.GetString(indexReply);
171:
172:    myGuestBook.PostTime  = myDataRecord.GetValue (
                          myDataRecord.GetOrdinal("PostTime")).ToString ();
173:    myGuestBook.ImageUrl  = myDataRecord.GetString(
                           myDataRecord.GetOrdinal("ImageUrl"));
174:
175:    return myGuestBook;
176:   }
177: }
```

在上述代码中，第 3 行到第 30 行所定义的 GetList()方法，实现的主要功能是返回一个 GuestBookList 类的实例化对象 tempList；其中的代码采用 ADO.NET 技术访问数据库，对指定的数据表 GuestBook 进行查询，并通过第 22 行将查询的数据记录转换为 GuestBook 类的实例化业务对象。

第 32 行到第 66 行所定义的 Save ()方法，实现的主要功能是对留言簿数据进行更新或者添加操作。其中的代码采用 ADO.NET 技术访问数据库，使用了存储过程，第 42 行设置了一个空值 Id，说明是对数据表的添加操作；第 44 行获得了一个 ID，说明是对数据表的更新操作；第 46 行到第 52 行对存储过程中的相关参数赋值。

第 68 行到第 85 行所定义的 Delete()方法，实现的主要功能是对留言簿中的指定数据进行删除操作。其中的代码采用 ADO.NET 技术访问数据库，调用了存储过程 sprocGuestBookDeleteItem，

最后的返回值是一个布尔类型的值，如果为真，则表示删除操作成功；如果为假，则表示删除操作失败。

第 87 行到第 110 行所定义的 Reply()方法，实现的主要功能是提交管理员的回复留言。针对指定留言簿中的 id，对该留言添加回复内容 reply。

第 112 行到第 120 行所定义的 SetPage ()方法，主要设置 pagedDS 对象，以便实现 DataList 控件的数据分页。

第 122 行到第 135 行所定义的 GetPageCount()方法，获得留言簿中的留言数量，以便进行分页。

第 137 行到第 176 行所定义的 FillDataRecord()方法，则是一个私有的帮助方法，实现的主要功能是将数据表 GuestBook 中所读取的字段数据转换为 GuestBook 类的实例化对象 myGuestBook。为防止 myGuestBook 对象的相关属性为空值，代码中添加了许多条件语句加以判断。

这里需要说明的是，查询、更新等操作的对象是 GuestBook 类的实例化对象，而不是数据表中的字段。

4.3.4 查看留言页面的实现

在查看留言页面 MyGuestBook.aspx 中，使用了 DataList 数据显示控件，并采用后置代码数据绑定的方式，通过 DataPagedSource 类实现 DataList 数据的分页。

代码 4-6 是 DataList 控件的设置代码。

代码 4-6 DataList 控件的设置代码

```
1: <asp:DataList ID="DataList1" runat="server"   Style="position:
          relative"  Width="100%" >
2:   <ItemTemplate>
3:     <table border="0" cellpadding="0" cellspacing="0"
             style="border-top: #e8e8e8 1px solid;left: 0px; width: 550px;
             position: relative; top: 0px; height: 32px">
4:       <tr>
5:         <td style="background-image: url(images/showbj.bmp);
                  width: 21px; height: 27px; border-left: #e8e8e8 1px solid;"
                  align="center"></td>
6:         <td align="right" colspan="2" style="border-right: #e8e8e8 1px
                  solid; background-image: url(images/showbj.bmp);
                  width: 550px; height: 27px">
7:           <div style="left: 2px; width: 550px; position: relative;
                   top: 0px; height: 26px;text-align: left">
8:             <asp:Label ID="Subject" runat="server" Style="position:
                     relative" Text='<%# Eval("subject")%>'  ></asp:Label>
9:           </div>
10:        </td>
11:      </tr>
```

```
12:    </table>
13:
14:    <table border="0" cellpadding="0" cellspacing="0" style="left: 1px;
           width: 550px;position: relative; top: 0px; border-right: #e8e8e8
           1px solid; border-left: #e8e8e8 1px solid; height: 100%;">
15:      <tr>
16:        <td style="width: 125px; border-right: #e8e8e8 1px solid;"
               align="center">
17:          <div style="width: 100px; position: relative; height: 100px">
18:            <img alt="a"  style="position: relative"
                   src='images/face/<%#DataBinder.Eval(Container.DataItem,
                   "imageUrl")%>' />
19:          </div>
20:          <asp:Label ID="lblUserPic" runat="server" Style="position:
                 relative" Text='<%# Eval("userName")%>'  ></asp:Label></td>
21:        <td style="width: 431px; border-bottom: #e8e8e8 1px solid;
                 height: 121px">
22:          <div style="left: 3px; width: 450px; position: relative;
                   top: 0px;  text-align: left; height: 82px;">
23:            <%# Eval("message")%>
24:            <br />————————————————<br />
25:            管理员回复：<%# Eval("Reply")%>
26:          </div>
27:          <div style="left: 2px; width: 440px; position: relative;
                   height: 25px;text-align: right">
28:            <img src="images/posttime.gif" alt="" style="position:
                   relative; border-top-style: none; border-right-style:
                   none; border-left-style: none; border-bottom-style: none;
                   left: 0px; top: 3px;" /> 
29:            发布时间：<%#DataBinder.Eval(Container.DataItem,
                           "postTime", "{0:D}")%>
30:            <img src="images/004.gif" style="position:
                           relative; top: 2px;" alt=""/>
31:            <asp:LinkButton ID="Reply" runat="server"
                   CommandArgument='<%# Eval("ID")%>'
                   OnCommand="Reply_Command" Style="position: relative;
                   top: 0px">回复</asp:LinkButton>
32:            <img alt="a"  src="images/delete.gif"
                   style="border-top-style: none; border-right-style: none;
                 border-left-style: none; position: relative;
                     border-bottom-style: none; top: 5px;" />
33:            <asp:LinkButton ID="Delete" runat="server"
                    CommandArgument='<%# Eval("ID")%>'
                 OnCommand="Delete_Command"  >删除</asp:LinkButton> 
34:          </div>
35:        </td>
```

```
36:       </tr>
37:     </table>
38:
39:     <table border="0" cellspacing="0" style="border-top: #e8e8e8 1px
              solid; left: 1px; width: 550px; position: relative;
              top: 0px; height: 13px">
40:       <tr>
41:         <td style="background-image: url(images/showbj.bmp);
                  width: 114px; border-left: #e8e8e8 1px solid;"></td>
42:         <td style="background-image: url(images/showbj.bmp);
                  width: 2172px"></td>
43:         <td style="background-image: url(images/showbj.bmp);
                  width: 133px; border-right: #e8e8e8 1px solid;"
                  align="right"></td>
44:       </tr>
45:     </table>
46:   </ItemTemplate>
47:   <SeparatorTemplate>
48:     <br />
49:   </SeparatorTemplate>
50: </asp:DataList>
```

在上述代码中，第 2 行到第 46 行设置了 DataList 控件的项目模板，即显示绑定数据的模板，其中包括 3 个部分，由 3 个表格组成。

第 3 行到第 12 行所设置的表格，主要显示留言主题的蓝色背景条，其中第 8 行通过数据绑定表达式<%# Eval("subject")%>，来读取留言的主题内容。

第 14 行到第 37 行所设置的表格，主要显示留言的头像、留言内容、留言回复内容以及相关命令链接等。其中第 18 行通过<%#DataBinder.Eval(Container.DataItem,"imageUrl")%>数据绑定表达式，读取留言头像的路径，显示留言的头像；第 20 行通过<%# Eval("userName")%>数据绑定表达式，读取用户名称，以便显示留言头像下的用户名；第 23 行到第 25 行则分别显示留言内容以及管理员的回复内容；第 29 行设置发布时间；第 30 行、第 31 行设置“回复”的图像和链接按钮；第 32 行、第 33 行设置“删除”的图像和链接按钮。

第 39 行到第 45 行所设置的表格，主要显示一个空白的蓝色背景条，以便美化、分割留言界面。

第 47 行到第 49 行设置了数据记录间的分割模板，其中设置了一个空白行（代码第 48 行），以便各留言之间留有一个空白行，美化留言界面。

代码 4-7 是分页界面的设置代码。

代码 4-7　分页界面的设置代码

```
1: 共<asp:Label ID="MessageTotal" runat="server" Style="position:
       relative" Text="Label"></asp:Label>条留言 
2: 第<asp:Label ID="PageCurrent" runat="server" Style="position:
       relative" Text="Label"></asp:Label>页 
3: 共<asp:Label ID="PageTotal" runat="server" Style="position:
```

```
        relative" Text="Label"></asp:Label>页 
4: <asp:Button ID="First" runat="server" Style="position: relative"
        Text="首页" OnClick="First_Click" />
5: <asp:Button ID="Previous" runat="server" Style="position: relative"
        Text="上一页" OnClick="Previous_Click" />
6: <asp:Button ID="Next" runat="server" Style="position: relative"
        Text="下一页" OnClick="Next_Click" />
7: <asp:Button ID="Last" runat="server" Style="position: relative"
        Text="尾页" OnClick="Last_Click" />
8:   转到<asp:DropDownList ID="DropDownList1" runat="server"
        Style="position: relative"></asp:DropDownList> 
9: <asp:Button ID="Go" runat="server" Style="position: relative"
        Text="GO" OnClick="Go_Click" />
```

在上述代码中，第 1 行设置了留言总数；第 2 行设置了留言的当前索引；第 3 行设置了留言的页面总数；第 4 行到第 7 行分别设置了留言的首页、上一页、下一页和尾页的页面导航按钮；第 8 行、第 9 行设置了一个下拉列表框选择页面，然后单击 Go 按钮即可转到指定的留言页面。

查看留言页面 MyGuestBook.aspx 的运行界面，如图 4-2 所示。

代码 4-8 是后置代码绑定的实现代码。

代码 4-8　后置代码绑定的实现代码

```
 1: public partial class MyGuestBook : System.Web.UI.Page
 2: {
 3:   PagedDataSource pds ;
 4:
 5:   protected void Page_Load(object sender, EventArgs e)
 6:   {
 7:     pds = GuestBookManager.SetPage();
 8:
 9:     if (!IsPostBack)
10:     {
11:       MessageTotal.Text = GuestBookManager.GetPageCount().ToString();
12:       PageTotal.Text = pds.PageCount.ToString();
13:       PageCurrent.Text = (pds.CurrentPageIndex + 1).ToString() ;
14:
15:       Binding();
16:
17:       for (int i = 1; i <= pds.PageCount; i++)
18:         DropDownList1.Items.Add(i.ToString());
19:     }
20:   }
21:
22:   public void Binding()
23:   {
24:     DataList1.DataSource = pds;
```

```
25:     DataList1.DataBind();
26:
27:     Previous.Enabled = true;
28:     Next.Enabled = true;
29:
30:     if (pds.CurrentPageIndex == 0)
31:       Previous.Enabled = false;
32:
33:     if (pds.CurrentPageIndex == pds.PageCount-1)
34:       Next.Enabled = false;
35:   }
36:
37:   protected void Delete_Command(object sender, CommandEventArgs e)
38:   {
39:     int id = Convert.ToInt32(e.CommandArgument);
40:
41:     if (User.Identity.IsAuthenticated)
42:     {
43:       if (GuestBookManager.Delete(id))
44:         Response.Write("<script>alert('删除成功! ');
                            window.location=window.location;</script>");
45:       else
46:         Response.Write("<script>alert('删除失败! ');
                            window.location=window.location;</script>");
47:     }
48:     else
49:     {
50:       Response.Write("<script>alert('对不起,只有管理员才允许删除留言,
                           如果你是管理员,请先登录! ');
                           window.location.href='login.aspx';</script>");
51:     }
52:   }
53:
54:   protected void Reply_Command(object sender, CommandEventArgs e)
55:   {
56:     string id = e.CommandArgument.ToString();
57:     Response.Redirect("Admin/GuestBookReply.aspx?id=" + id + "");
58:   }
59:
60:   protected void Previous_Click(object sender, EventArgs e)
61:   {
62:     pds.CurrentPageIndex = Convert.ToInt32(PageCurrent.Text)-2;
63:
64:     PageCurrent.Text =(pds.CurrentPageIndex+1).ToString();
65:     Binding();
66:   }
```

```
67:
68:   protected void Next_Click(object sender, EventArgs e)
69:   {
70:     pds.CurrentPageIndex=Convert.ToInt32(PageCurrent.Text);
71:
72:     PageCurrent.Text = (pds.CurrentPageIndex+1).ToString ();
73:     Binding();
74:   }
75:
76:   protected void First_Click(object sender, EventArgs e)
77:   {
78:     pds.CurrentPageIndex = 0;
79:     PageCurrent.Text = "1";
80:     Binding();
81:   }
82:
83:   protected void Last_Click(object sender, EventArgs e)
84:   {
85:     PageCurrent.Text = PageTotal.Text;
86:     pds.CurrentPageIndex = pds.PageCount -1;
87:     Binding();
88:   }
89:
90:   protected void Go_Click(object sender, EventArgs e)
91:   {
92:     pds.CurrentPageIndex =Convert.ToInt32(DropDownList1.SelectedValue)-1;
93:     Binding();
94:   }
95: }
```

在上述的后置代码中，第 22 行到第 35 行所定义的 Binding()方法，是其中的关键代码。第 24 行设置了数据显示控件 DataList1 的数据源为 PagedDataSource 类的实例化对象 pds，第 25 行设置了 DataList1 的数据绑定，因此通过 pds 对象可以实现数据显示控件 DataList1 的分页功能。第 27 行到第 34 行根据对象 pds 的当前索引状态，分别设置分页导航按钮的可用或不可用状态。

第 37 行到第 52 行所定义的 Delete_Command()方法，主要实现删除指定留言的功能。其中第 41 行判断是否是通过注册验证的用户，如果是管理员，第 43 行则调用 GuestBookManager.Delete()方法，删除指定的留言；否则执行第 50 行代码，给用户提示。

第 54 行到第 58 行所定义的 Reply_Command()方法，主要实现打开回复留言页面的功能，由于回复留言页面 GuestBookReply.aspx 只有管理员才能访问，因此只有登录验证通过的管理员才能进入回复留言页面 GuestBookReply.aspx。

第 60 行到第 88 行分别实现页面导航按钮的相关跳转功能；第 90 行到第 94 行所定义的 Go_Click()方法，主要实现显示指定页面的留言。

4.3.5 发表新留言页面的实现

发表新留言的页面是GuestBookAdd.aspx，浏览者打开该页面，在其中填写相关留言信息，单击“提交”按钮，即可发表新的留言。

代码4-9是发表新留言GuestBookAdd.aspx用户界面的设置代码。

代码4-9 发表新留言GuestBookAdd.aspx用户界面的设置代码

```
<table border="0" cellpadding="0" cellspacing="0" style="width: 100%;
    position: relative;height: 94%; border-right: activeborder 1px solid;
    border-top: activeborder 1px solid; left: 0px; border-left:
    activeborder 1px solid; border-bottom: activeborder 1px solid;
    top: 0px;">
  <tr>
    <td colspan="2" style="font-weight: bold; color: red; height: 32px"
        align="left">
        发表新留言</td>
  </tr>
  <tr>
    <td style="width: 111px; text-align: right; height: 29px;">
       用户名：</td>
    <td style="text-align: left; height: 29px;">
      <asp:TextBox ID="UserName" runat="server" Style="position:
          relative"></asp:TextBox></td>
  </tr>
  <tr>
    <td style="width: 111px; text-align: right; height: 29px;">
      Email：</td>
    <td style="text-align: left; height: 29px;">
      <asp:TextBox ID="Email" runat="server" Style="position:
          relative"></asp:TextBox></td>
  </tr>
  <tr>
    <td style="width: 111px; height: 110px; text-align: right">
      头像：</td>
    <td style="height: 110px; text-align: left" valign="middle">
      <asp:DropDownList ID="ddlPic" runat="server" Style="position:
          relative; top: 0px;" AutoPostBack="True"
          OnSelectedIndexChanged="ddlPic_SelectedIndexChanged">
      </asp:DropDownList>
      <asp:Image ID="Image" runat="server" Style="position:
          relative; left: 0px;" /></td>
  </tr>
  <tr>
    <td style="width: 111px; text-align: right; height: 29px;">
```

```
28:       主题：</td>
29:     <td style="text-align: left; height: 29px;">
30:       <asp:TextBox ID="Subject" runat="server" Style="width: 380px;
                position: relative" Width="367px"></asp:TextBox></td>
31:   </tr>
32:   <tr>
33:     <td style="width: 111px; text-align: right" valign="top">
34:       留言内容：</td>
35:     <td style="text-align: left">
36:       <asp:TextBox ID="Message" runat="server" Height="140px"
              Style="position: relative" TextMode="MultiLine" Width="380px">
              </asp:TextBox></td>
37:   </tr>
38:   <tr>
39:     <td style="width: 111px; height: 32px; text-align: right"
            valign="top"></td>
40:     <td style="height: 32px; text-align: left">
41:                  
42:       <asp:Button ID="Add" runat="server" Style="position: relative"
              Text="提交" OnClick="Add_Click" Width="51px" />   
43:       <asp:Button ID="Clear" runat="server" Style="position: relative"
              Text="清空" OnClick="Clear_Click" Width="52px" /></td>
44:   </tr>
45: </table>
```

上述代码实现的是一个 7 行 2 列的表格。表格的第 1 行设置了“发表新留言”（第 2 行到第 5 行）；表格的第 2 行设置了用户名（第 6 行到第 11 行）；表格的第 3 行设置了 Email（第 12 行到第 17 行）；表格的第 4 行设置了头像（第 18 行到第 25 行）；表格的第 5 行设置了主题（第 26 行到第 31 行）；表格的第 6 行设置了留言内容（第 32 行到第 37 行）；表格的第 7 行设置了“提交”、“清空”按钮（第 38 行到第 44 行）。

发表新留言页面 GuestBookAdd.aspx 的运行界面，如图 4-3 所示。

上述用户页面所对应的后置代码，如代码 4-10 所示。

代码 4-10　发表新留言的后置代码

```
 1: public partial class GuestBookAdd : System.Web.UI.Page
 2: {
 3:   protected void Page_Load(object sender, EventArgs e)
 4:   {
 5:     if (!IsPostBack)
 6:     {
 7:       for (int i = 1; i <= 10; i++)
 8:       {
 9:         this.ddlPic.Items.Add("pic" + i.ToString() + ".gif");
10:       }
11:       Image.ImageUrl = "images/face/" + ddlPic.SelectedValue;
12:     }
```

```
13:   }
14:
15:   protected void Add_Click(object sender, EventArgs e)
16:   {
17:     GuestBook guestBook = new GuestBook();
18:
19:     guestBook.Email = UserName.Text;
20:     guestBook.Message =Email.Text;
21:     guestBook.Subject  =Subject.Text;
22:     guestBook.Message  =Message.Text;
23:     guestBook.ImageUrl  =ddlPic.SelectedValue;
24:
25:     if ( GuestBookManager.Save (guestBook )> 0)
26:     {
27:       Response.Write("<script>alert('留言成功! ');
                              location.href='MyGuestBook.aspx';</script>");
28:     }
29:     else
30:     {
31:       Response.Write("<script>alert('留言失败! ');
                              window.location = window.location;</script>");
32:     }
33:   }
34:
35:   protected void ddlPic_SelectedIndexChanged(object sender, EventArgs e)
36:   {
37:     Image.ImageUrl = "images/face/" + ddlPic.SelectedValue;
38:   }
39:
40:   protected void Clear_Click(object sender, EventArgs e)
41:   {
42:     UserName.Text = "";
43:     Email.Text = "";
44:     Subject.Text = "";
45:     Message.Text = "";
46:   }
47: }
```

上述的代码比较简单，第 15 行到第 33 行所定义的 Add_Click()方法，实现留言内容的提交，其中第 25 行是关键语句，将提交内容保存到数据库中。

4.3.6 回复留言页面的实现

回复留言页面是 GuestBookReply.aspx，只有管理员才能打开该页面，在其中填写回复留言内容，单击“提交”按钮，即可针对指定留言发表回复内容。

代码 4-11 是回复留言页面 GuestBookReply.aspx 用户界面的设置代码。

代码 4-11　回复留言用户界面的设置代码

```
1: <table border="0" cellpadding="0" cellspacing="0" style="width: 550px;
        position: relative; text-align: center">
2:   <tr>
3:     <td>
4:       <table border="0" cellpadding="0" cellspacing="0"
              style="width: 100%; position: relative;height: 121%">
5:         <tr>
6:           <td style="height: 156px">
7:             <asp:Label ID="Label1" runat="server" Style="position:
                    relative; left: -17px; top: -125px;" Text="回复内容：">
                    </asp:Label>
8:             <asp:TextBox ID="txtReply" runat="server" Style="position:
                    relative" Height="141px" TextMode="MultiLine"
                    Width="306px"></asp:TextBox></td>
9:         </tr>
10:        <tr>
11:          <td style="height: 33px">
12:            <asp:Button ID="Reply" runat="server" Style="position:
                    relative" Text="提交" OnClick="Reply_Click" /> 
13:            <asp:Button ID="Clear" runat="server" Style="left: 10px;
                    position: relative" Text="清空" OnClick="Clear_Click" />
14:          </td>
15:        </tr>
16:      </table>
17:    </td>
18:  </tr>
19: </table>
```

上述的用户界面实现的是一个 2 行的表格（代码第 4 行到第 16 行）。表格的第 1 行设置的是回复内容（代码第 5 行到第 9 行）；表格的第 2 行设置的是“提交”、“清空”按钮（代码第 10 行到第 15 行）。

回复留言页面 GuestBookReply.aspx 的运行界面，如图 4-5 所示。

上述用户页面所对应的后置代码，如代码 4-12 所示。

代码 4-12　实现回复留言的后置代码

```
1: public partial class GuestBookReply : System.Web.UI.Page
2: {
3:
4:   protected void Reply_Click(object sender, EventArgs e)
5:   {
6:     string reply =txtReply.Text;
7:     int id = Convert.ToInt32(Request.QueryString["id"]);
8:
9:     if (GuestBookManager.Reply (id,reply ) )
10:    {
```

```
11:       Response.Write("<script>alert('回复成功！');
                              location.href='../MyGuestBook.aspx';</script>");
12:     }
13:     else
14:     {
15:       Response.Write("<script>alert('回复失败！');
                               location.href=location.href;</script>");
16:     }
17:   }
18:
19:   protected void Clear_Click(object sender, EventArgs e)
20:   {
21:     this.txtReply.Text = "";
22:   }
23: }
```

在上述代码中，第 9 行是关键代码，通过调用 GuestBookManager 类的 Reply()方法，实现回复留言保存到数据库中。

4.3.7　登录页面

在留言簿中，当浏览者访问回复留言页面 GuestBookReply.aspx 时，打开 Login.aspx 登录页面，只有登录用户才能访问回复留言页面 GuestBookReply.aspx，在其中填写回复留言内容，单击“提交”按钮，针对指定的留言发表回复内容。

1．创建需要被保护的文件夹

在留言簿中创建一个只能被注册用户访问的文件夹——Admin，在该文件夹中添加回复留言页面 GuestBookReply.aspx，使得只有注册用户才能访问该文件夹中的页面，如图 4-14 所示。

图 4-14　网站目录结构

2．创建注册用户

在 Visual Studio 2005 中，单击“网站”菜单下的“ASP.NET 配置”命令，打开如图 4-15 所示的网站管理工具。

图 4-15　网站管理工具

在图 4-15 中，单击“提供程序配置”链接，打开如图 4-16 所示的设置提供程序配置界面，确保当前应用程序所配置的提供程序为 AspNetSqlProvider。

图 4-16　设置提供程序

通过上述配置，表明存储网站注册用户以及用户访问规则的相关数据保存在 Visual Studio 2005 所附带的 SQL Server 2005 Express 数据库中，数据库的名称为 AspNetDB。

在图 4-16 中，单击“安全”选项卡，打开如图 4-17 所示的“选择身份验证类型”运行界面，在其中单击“选择身份验证类型”链接，就会打开“设置用户访问站点方式”的运行界面，如图 4-18 所示。

图 4-17　选择身份验证类型

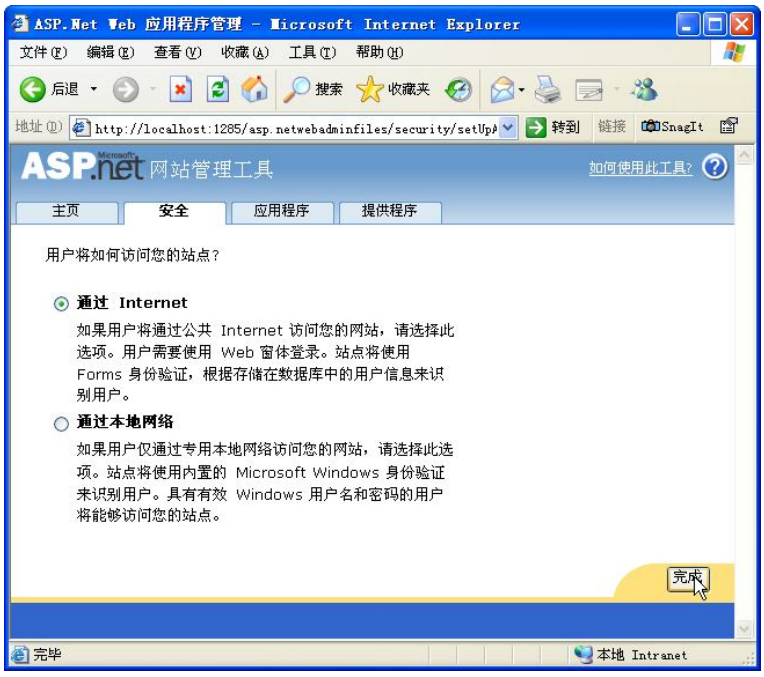

图 4-18　设置用户访问站点方式

在图 4-18 中，选择“通过 Internet”方式，然后单击右下方的“完成”按钮，即可完成对身份验证类型的设置。当需要身份验证时，使用登录页面的方式来实现，至于使用什么具体的登录页面，还需要开发者进一步设置。

在图 4-17 中，单击“创建用户”链接，打开如图 4-19 所示的注册新帐户界面。

图 4-19　注册新帐户

这里需要说明的是，当开发者创建一个新的注册用户 test1 时，密码的长度必须是 8 位以上，而且其中必须包含非字母、非数字，否则新帐户就不会成功创建，建议初学者选取密码为“123456&”，然后单击“创建用户”按钮，就会成功创建一个新的注册用户，如图 4-20 所示，利用该注册用户，管理人员可以登录进入 Admin 文件目录下的回复留言页面 GuestBookReply.aspx。

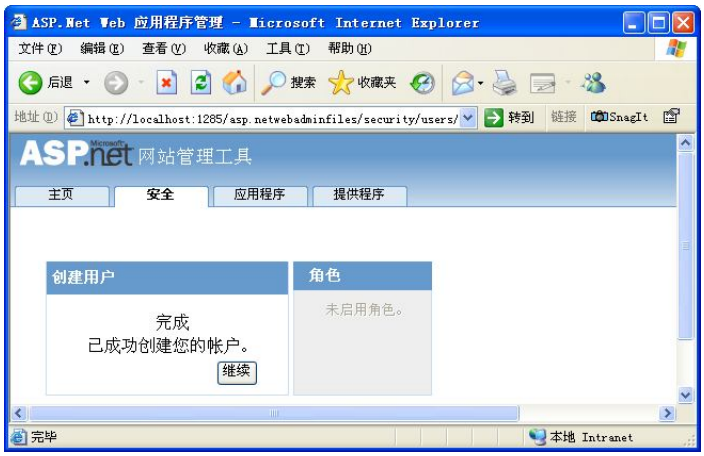

图 4-20　成功创建新帐户

3．设置访问规则

通过设置访问规则，可以设置需要保护的文件夹。在留言簿中，希望 Admin 目录只有注册用户才能访问，因此需要对 Admin 目录设置访问规则。

在图 4-17 所示的运行界面中，单击“创建访问规则”链接，打开如图 4-21 所示的“添加访问规则”界面。

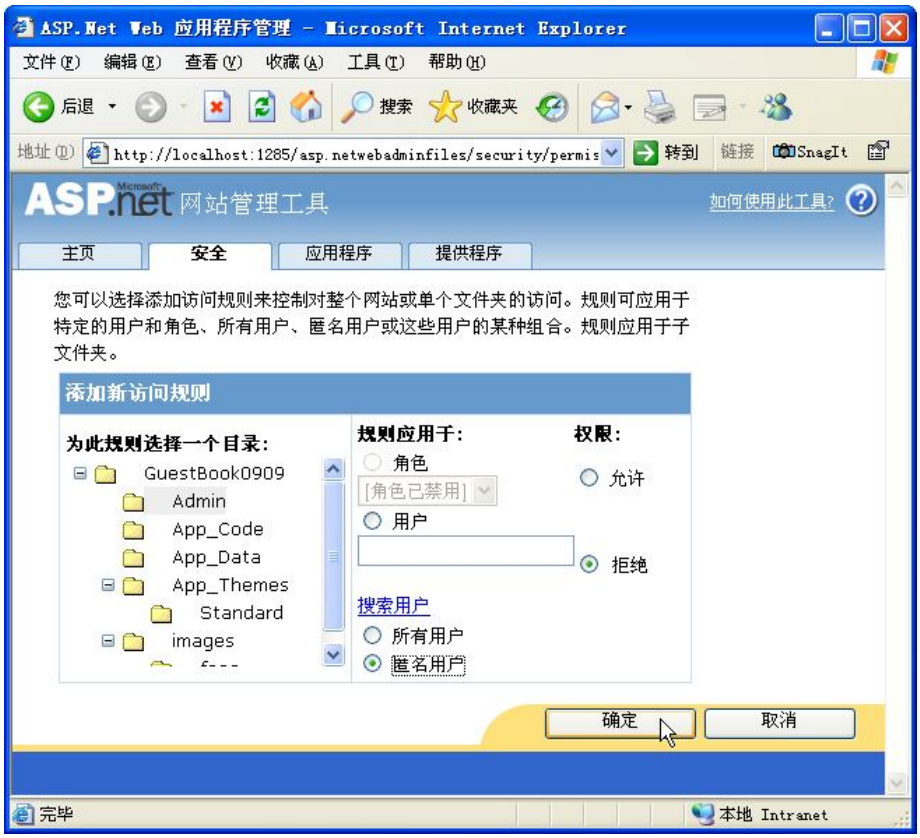

图 4-21　添加访问规则

在图 4-21 中，选择左边网站目录结构中的 Admin 文件夹，设置右边的权限为“拒绝”—“匿名用户”，然后单击“确定”按钮，即可完成访问规则的设置。

通过添加上述的访问规则，只有注册用户登录后，才能访问 Admin 文件夹下的所有页面，而非注册登录用户，也就是匿名用户是不允许访问 Admin 文件夹中的所有页面的。

此时在 Visual Studio 2005 开发工具界面右边的“解决方案资源管理器”中刷新 Admin 文件夹，将会发现该文件夹下添加了一个新的配置文件 web.config。

打开该配置文件，其配置代码如代码 4-13 所示。

代码 4-13　访问规则的配置文件代码

```
1: <?xml version="1.0" encoding="utf-8"?>
2:  <configuration
```

```
            xmlns="http://schemas.microsoft.com/.NetConfiguration/v2.0">
3:   <system.web>
4:     <authorization>
5:       <deny users="?" />
6:     </authorization>
7:   </system.web>
8:  </configuration>
```

在上述配置文件中，其中的关键代码是第 4 行到第 6 行，设置了权限访问规则，拒绝所有的匿名用户访问（第 5 行），由于该配置文件位于 Admin 文件夹中，因此只有注册登录用户才能访问 Admin 文件夹中的页面。

随着开发者技能的不断提高，开发者可以不再需要通过图 4-21 的可视化方式，来添加访问规则，而可以直接编写上述代码的配置文件来实现相关的访问规则。

4．登录页面的实现

当浏览者尝试访问 Admin 文件目录下的回复留言页面 GuestBookReply.aspx 时，由于设置了访问规则，即只有注册登录用户才能访问该页面，此时留言簿会自动转移到根目录下的 Login.aspx 页面，这是 ASP.NET 2.0 提供的默认设置。如果不存在 Login.aspx 页面，将会显示“无法找到资源”的错误信息页面。

在登录页面 Login.aspx 中，需要拖放“登录”工具箱中的 Login 控件，并需要设置 Login 控件的 DestinationPageUrl 属性为 Admin 文件目录下的回复留言页面 GuestBookReply.aspx，如图 4-22 所示。

图 4-22　设置 DestinationPageUrl 属性

当浏览者在 Login.aspx 页面中输入有效的用户名和密码，登录成功之后，留言簿就会进入到 Admin 文件目录下的回复留言页面 GuestBookReply.aspx；如果没有有效的用户名、密码，就不能进入回复留言页面 GuestBookReply.aspx。这样就保证了只有管理员，也就是具有有效的用户名和密码的用户，才能给相关用户回复留言。

4.4　任务小结

下面对如何实现留言簿这一工作任务作一个小结：

- 页面功能分析：介绍了留言簿的总体结构，说明组成留言簿 4 个页面的具体功能。
- DataList 数据绑定和数据分页：介绍了 DataList 数据绑定的两种方式，分别是页面代码绑定方式和后置代码绑定方式；说明了如何设置 DataPagedSource 类，为数据显示控件 DataList 提供分页功能，实现 DataList 的数据分页。
- 留言簿的实现：依据前面的页面功能分析，首先需要设计数据库，其中包括数据表以及存储过程的设计；然后根据二层架构的设计方法，创建业务类，构建中间数据访问层，最后实现留言簿中的各个页面。这里需要说明的是，这些页面的功能实现大都采用后置代码绑定的方式，并非严格遵循前面架构分析任务中所说的二层架构方式。

4.5　思考题

1．如何在数据显示控件 GridView 中实现数据绑定的两种方式？

2．使用数据显示控件 DataList，实现对 NorthWind 数据库中数据表 Contacts 数据的浏览，并添加数据分页功能。

4.6　工作任务评测单

学习情境 2	个人网站开发	班级	
任务 4	留言簿	小组成员	
任务描述	留言簿主要由 4 个页面组成，他们分别是查看留言页面 MyGuestBook.aspx、发表新留言页面 GuestBookAdd.aspx、回复留言页面 GuestBookReply.aspx 以及登录页面 Login.aspx，根据各页面的功能需求，需要设计数据库，创建业务类和中间数据访问层，最后分别实现这 4 个页面。		
任务分析	数据库设计：　　数据表的结构图： 存储过程的名称： 业务类和中间数据访问层的名称： 4 个页面的实现：		
任务实施	实施步骤：（并回答思考题） 1．业务类和中间数据访问层的 UML 类图		

<table>
<tr><td></td><td colspan="3">2．留言簿的目录结构图

3．使用数据显示控件 DataList，实现对 NorthWind 数据库中数据表 Contacts 数据的浏览，并添加数据分页功能的运行界面</td></tr>
<tr><td rowspan="3">工作评价</td><td>小组自评</td><td>分数：</td><td>签名：　　年　月　日</td></tr>
<tr><td>小组互评</td><td>分数：</td><td>签名：　　年　月　日</td></tr>
<tr><td>教师评价</td><td>分数：</td><td>签名：　　年　月　日</td></tr>
</table>