
第 28 章 制作一个 ASP.NET 留言本

在了解了一些基本的模块的开发之后就能够开发一些基本的应用，这些应用可以看作是很多的模块组成应用，在开发过程中可以应用现有的模块进行应用的开发。留言本是最基础 Web 应用，也是初学者最常学习的 Web 应用。

28.1 系统设计

系统设计在项目开发中是非常重要的，在系统设计中，需求分析也是最为重要的。需求分析规定了开发小组或团队以何种方式进行模块的开发和编码，也规定了客户最基本的需求，如果连客户最基本的需求都没有弄清楚，那么这个系统必然是失败的。

28.1.1 需求分析

需求分析是系统设计中最重要的一环，在任何系统的开发中都需要进行需求分析，虽然 ASP.NET 留言本是一个很小的项目，但是还是需要进行需求分析。需求分析并不因为项目的大小而有任何区别，需求分析更多的任务是告诉开发团队客户想要的是什么、客户需要的是什么、团队怎样进行模块划分和开发等等。

虽然在 ASP.NET 留言本开发中需求分析显得微不足道，但是随时保持编写需求分析是一个非常良好的习惯。需求分析是软件工程中的一个概念，指的是在建立一个新的或改变一个现存的电脑系统时描写新系统的目的、范围和定义时所要做的所有的工作。简单的说需求分析也就是分析客户要的是什么、怎么做、做完了怎么办。对于 ASP.NET 留言本项目而言，其需求分析可以编写如下：

1. 目录

需求分析通常情况下是一个单独的需求分析文档，需求分析文档的格式很像一本书或论文的格式，其示例目录如下所示。

- 1. 引言：通常是需求分析文档的引言，用户描述为何编写需求分析文档。
- 1.1 编写目的：编写目的用户描述为何编写需求分析文档。
- 1.2 项目背景：编写相应的项目背景。
- 1.3 定义缩写词和符号：编写在需求分析文档中定义的缩写词或符号等。
- 1.4 参考资料：用户描述在需求分析文档中所参考的资料。
- 2. 任务描述：定义任务，通常情况下用于描述完成何种任务。
- 2.1 开发目标：定义开发目标，包括为何要进行开发。
- 2.2 应用目标：定义应用目标，包括系统应用人员要实现什么功能，以及有哪些应用等。
- 2.3 软件环境：用于定义软件运行的环境。
- 3. 数据描述：用户进行数据库中数据设计开发的描述。

这里只是简单的介绍了需求分析文档编写中的目录的一些基本格式，需求分析文档通常是一个单独的文档，而需求分析文档需要解决客户需要的是什么，如何进行协调开发等。对于不同的项目其需求分

析文档其目录并不限于此格式，对于小型的项目的需求分析可以灵活更改。

2. 引言

在需求分析文档中，通常需要编写引言用户描述为何编写需求分析文档和需求分析文档的作用，对于 ASP.NET 留言本而言，其引言可以编写如下：

在对客户现有的应用模块的调查和了解的基础上，用户希望能够在现有的应用中加上留言本的功能以便能够及时的和用户进行信息反馈和调查。

此规格说明书在详细的调查了客户现有的应用模块和基本的操作流程后进行编写，对留言本功能进行了详细的规划、设计，明确了软件开发中应具有的功能、性能使得系统的开发人员和维护人员能够详细了解软件是如何开发和进行维护的，并在此基础上进一步提出概要设计说明书和完成后续设计与开发工作。本规格说明书的预期读者包括客户、业务或需求分析人员、测试人员、用户文档编写者、项目管理人员等。

3. 项目背景

项目背景用于描述该项目在何种背景或何种条件下进行开发的，以及为何要进行现有的项目的开发或升级，ASP.NET 留言本的项目背景可以编写如下：

由于现在信息化的迅猛发展，原有的软件项目已经不能满足现今越来越多的需求，更多的厂商都将软件应用基于互联网进行开发和使用。相对于原有的 C/S 软件开发而言，基于互联网的软件开发具有部署快、成本低、维护性低的特性，对于企业而言可以使用基于互联网的应用进行信息的发布和反馈。

对于原有的系统而言，用户必须下载客户端才能够与企业内部数据进行通信，这样难免会造成使用不便和安全性的问题，因为用户需要进行软件下载。如果用户并没有连接到网络，就不能够及时的了解用户的信息也无法下载现有的程序，如果用户将现有的程序进行反编译等操作也会造成安全性的问题。

随着互联网的发展，越来越多的用户已经可以使用互联网进行信息交互，也促成了越来越多的基于浏览器的应用程序，企业可以使用服务器/客户端的开发模型进行系统的开发，ASP.NET 留言本就是为了解决信息交互复杂和交互困难的问题的而诞生的。为了解决现有的企业中企业与用户信息反馈困难等情况，让企业能够更加方便的同用户进行信息交互，在征求了多方意见的情况下进行此 ASP.NET 留言本的开发，以便解决现有的企业难题。

4. 任务描述

任务描述用于描述客户的任务，以及基本的如何完成任务的描述，ASP.NET 留言本的任务描述可以编写为如下所示：

为了加强现有的企业和用户之间的信息交互，也解决企业和用户的沟通不便的情况，现开发基于 .NET 平台的留言本应用程序，用户能够使用留言本进行信息的反馈和调查，能够及时的获取用户的相关意见或信息的数据。

注意：任务描述作为章节的概述，在软件规格说明书中，该章节的其他章节将需要对概述进行更详细的说明。

5. 开发目标

ASP.NET 留言本的开发目标是为了加强现有的企业和用户之间的信息交互，解决企业和用户的沟通不便的情况，进行企业和用户之间的数据整合和交互。

6. 应用目标

ASP.NET 留言本的应用目标是为了能够让企业能够获取用户的信息，这些信息包括用户的意见、反馈的信息以及用户数据等，同时企业也能够通过留言本进行基础的意见调查。

需求分析是系统设计中最为重要的一部分，如果在系统设计初期需求分析设计的非常好也就方便了后期的开发和维护。

28.1.2 系统功能设计

ASP.NET 留言本是企业内部的一个信息交互平台，用户可以在相应的主题的留言本之内进行信息发布和反馈，用户还能够通过留言本进行信息的交互。在留言本的开发过程中需要确定基本的系统功能，这些基本的系统功能包括如下：

1. 留言信息浏览

用户可以在相应的留言页面进行留言信息的浏览，包括对企业产品的意见以及功能反馈等，在留言页面中按照用户的习惯可以进行按回复查看，按时间查看等选项。用户还能够通过导航栏进行不同留言板的跳转。管理员可以在留言信息浏览页面进行信息回复，可以对用户的疑问和意见进行反馈，管理员还能够删除不良的留言和屏蔽相关用户等操作。

2. 注册登录功能

在用户进行留言之前，必须进行注册和登录等操作，如果用户没有登录就不能够进行留言操作，用户登录或注册后可以通过留言索引自己的留言并进行留言修改或增加。

3. 用户留言索引

登录的用户可以索引自己的留言，对于自己较早的留言能够进行查看，这样就方便了用户进行信息整合，管理员也能够通过用户的索引相应的用户信息并进行用户管理。

4. 管理员留言管理

管理员对于不良的留言进行删除、屏蔽等操作，当用户进行了不良信息发布后管理员能够在留言页面进行删除操作。

28.1.3 模块功能划分

当介绍了系统所需实现的功能模块后并执行了相应的功能模块的划分和功能设计，可以编写相应的模块操作流程和绘制模块图，ASP.NET 留言本总体模块划分如图 28-1 所示。

图 28-1 描述了系统的总体模块功能划分，其中包括前面章节中讲到的留言信息浏览、用户信息注册、用户登录操作、用户留言索引以及管理员留言管理等操作，其中可以将用户注册、登录、信息浏览和索引等操作进行划分，如图 28-2 所示。

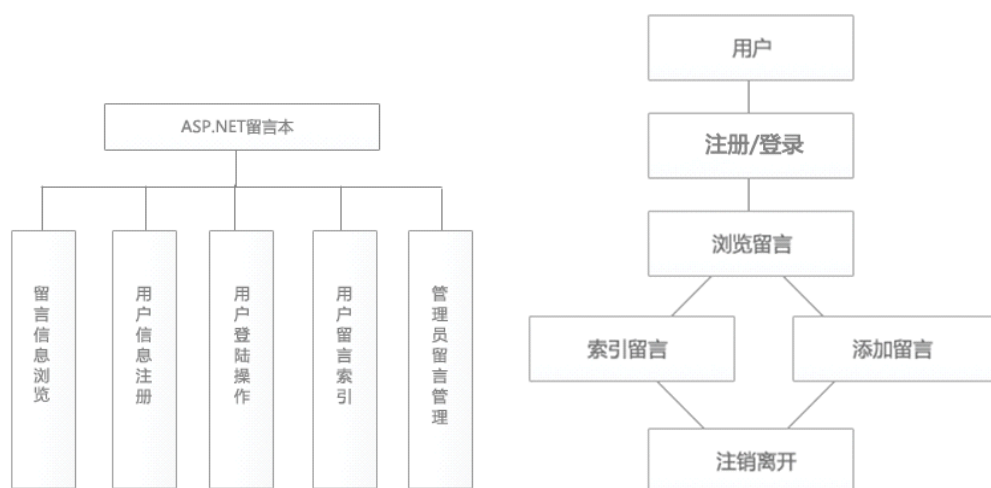


图 28-1 系统总体模块功能划分

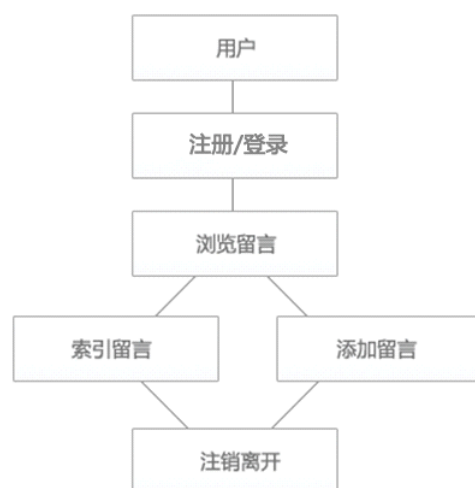


图 28-2 用户操作模块流程图

用户在进行页面访问时，可以呈现相关的留言信息，当用户进行留言时必须登录，如果用户事先没有任何账号信息可以进行注册，注册完成后会跳转到登录页面进行登录操作，如果用户已经存在账号就能够直接登录进行操作。

在用户注册或登录后就可以进行留言的索引和留言的添加，留言的索引能够方便用户查询长时间之前的自己的留言信息，例如用户进行留言后一个月再次访问企业网站，就会很难搜索到自己的留言，而通过索引能够方便的索引到自己的留言信息。如果用户没有任何留言可以选择添加留言，添加后的留言能够提交给管理员进行审核并回复。

对于管理员而言，管理员需要查看留言并进行留言的管理，在管理员管理之前也需要进行登录操作，以验证管理员身份的正确性和权限。当管理员验证通过后可以进行管理操作，管理员操作流程图如 28-3 所示。

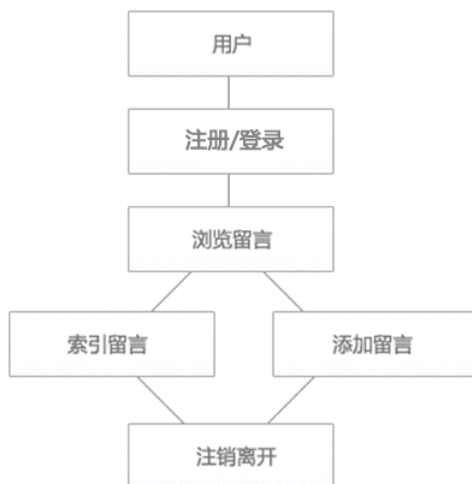


图 28-3 管理员操作模块流程图

在管理员进行管理之前同样需要进行身份验证，否则会造成系统的安全性问题，管理员只有在身份验证之后才能够进行管理回复和留言的删除操作。

28.2 数据库设计

在 ASP.NET 留言本的功能模块描述中，可以看出在数据库的设计中包括多个表，这些表包括留言表、留言分类表、用户信息表、管理员信息表等表，这些表用于实现前面小节中系统设计所规划的系统功能。

28.2.1 数据库的分析和设计

在前面的系统设计中已经详细的了解了系统的功能，在数据库的设计中，需要充分的了解系统的功能并进行合理的抽象再进行数据库设计，数据库设计图如图 28-4 所示。

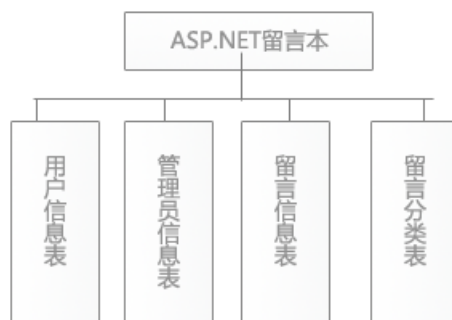


图 28-4 数据库设计图

其中初步的为数据库中的表进行设计，这里包括四个表，作用分别如下：

- ❑ 用户信息表：用于存放用户的信息并进行用户信息的管理。
- ❑ 管理员信息表：用于存放管理员的信息并在管理员登录时进行数据验证。
- ❑ 留言信息表：用于存放留言信息。
- ❑ 留言分类表：用于进行留言分类。

其中留言信息表和留言的分类表用于描述留言项目，一个企业网站不只包含一个留言页面，当企业有多个产品时，需要考虑到留言模块的扩展性，使用留言分类可以进行相应的功能的扩展。在 ASP.NET 留言本中最为重要的就是留言信息表和留言分类表，其中留言信息表的字段可以归纳如下。

- ❑ 留言编号：用于标识留言本的编号进行索引，为自动增长的主键。
- ❑ 留言标题：用户留言的标题。
- ❑ 留言名称：用户的名称。
- ❑ 留言时间：用户留言的时间。
- ❑ 留言内容：用户留言的内容。
- ❑ 回复标题：管理员回复留言的标题。
- ❑ 管理员名称：管理员的名称。
- ❑ 回复时间：管理员回复留言的时间。
- ❑ 回复内容：管理员回复的内容。
- ❑ 所属留言分类：用户留言所属的分类。
- ❑ 所属用户：留言所属的用户 ID。

上述表用于描述用户留言信息，用户留言表中的信息为最主要的数据，在进行留言呈现时呈现的就

是此数据表中的数据。同时为了能够将留言数据进行分类，需要创建留言分类表，其字段可以描述如下所示。

- ❑ 分类编号：用于标识留言本分类的编号，为自动增长的主键。
- ❑ 分类名称：用于描述分类的名称，例如“客户服务”等。

留言分类和留言表一起描述留言项目，这样能够增加留言本系统的扩展性，管理员可以创建多个留言分类进行留言管理。例如现在企业有一个“皮鞋”产品，可以创建一个主题为“皮鞋”的留言本，但是如果企业有一天多了一个产品，那么就需要重新制作留言本，重新制作的留言本在数据和管理上都需要重新操作。而如果将一个留言本进行分类处理，当有新产品出现时就能够很好的扩展。

在进行留言前，用户必须要登录，如果没有登录就必须要注册，这里可以使用注册模块进行注册功能的实现，注册模块中用户表的字段可以归纳如下。

- ❑ 用户名：用于保存用户的用户名，当用户登录时可以通过用户名验证。
- ❑ 密码：用于保存用户的密码，当用户使用登录时可以通过密码验证。
- ❑ 性别：用于保存用户的性别。
- ❑ 头像：用于保存用户的个性头像。
- ❑ QQ/MSN：用于保存用户的 QQ/MSN 等信息。
- ❑ 个性签名：用于展现用户的个性签名等资料。
- ❑ 备注：用于保存用户的备注信息。
- ❑ 用户情况：用于保存用户的状态，可以设置为通过审批和未通过等。

用户在注册后就能够进行登录，登录后的用户将能够通过留言页面进行信息发布和反馈。在用户发布信息后，管理员可以对这些信息进行管理，同样管理员在进行管理时也需要进行登录操作，其字段可以归纳如下。

- ❑ 管理员编号：用于标识管理员信息，为自动增长的主键。
- ❑ 管理员用户名：用于标识管理员用户名。
- ❑ 管理员密码：用于标识管理员的密码，通常情况下和管理员用户名一起进行身份验证。

管理员要进行操作前需要进行登录并通过数据进行身份验证，验证通过后才能够进行相应的数据更改，在对数据库进行基本的分析后，就能够创建相应的数据表进行数据存储。

28.2.2 数据表的创建

创建表可以通过 SQL Server Management Studio 视图进行创建也可以通过 SQL Server Management Studio 查询使用 SQL 语句进行创建。

1. 事务表

在创建数据表之前首先需要创建 `guestbook` 数据库，创建完成后就能够进行其中的表的创建。在 ASP.NET 留言本中最为重要的是留言表，留言表 and 留言分类表可以统称为事务表，其中留言表结构如图 28-5 所示。



	列名	数据类型	允许空
	id	int	<input type="checkbox"/>
	title	nvarchar(50)	<input checked="" type="checkbox"/>
	name	nvarchar(50)	<input checked="" type="checkbox"/>
	time	datetime	<input checked="" type="checkbox"/>
	[content]	nvarchar(MAX)	<input checked="" type="checkbox"/>
	reptitle	nvarchar(50)	<input checked="" type="checkbox"/>
	admin	nvarchar(50)	<input checked="" type="checkbox"/>
	reptime	datetime	<input checked="" type="checkbox"/>
	repcontent	nvarchar(MAX)	<input checked="" type="checkbox"/>
	classid	int	<input checked="" type="checkbox"/>
	userid	int	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

图 28-5 留言表结构

从数据库中可以看出留言表中的字段信息，这些字段意义如下所示：

- ☐ id: 用于标识留言本的编号进行索引，为自动增长的主键。
- ☐ title: 用户留言的标题。
- ☐ name: 用户的名称。
- ☐ time: 用户留言的时间。
- ☐ content: 用户留言的内容。
- ☐ reptitle: 管理员回复留言的标题。
- ☐ admin: 管理员的名称。
- ☐ reptime: 管理员回复留言的时间。
- ☐ repcontent: 管理员回复的内容。
- ☐ classid: 用户留言所属的分类。
- ☐ userid: 留言所属的用户 ID。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [guestbook]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[gbook](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [title] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [name] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [time] [datetime] NULL,
    [content] [nvarchar](max) COLLATE Chinese_PRC_CI_AS NULL,
    [reptitle] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [admin] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [reptime] [datetime] NULL,
    [repcontent] [nvarchar](max) COLLATE Chinese_PRC_CI_AS NULL,
    [classid] [int] NULL,
    [userid] [int] NULL,
    CONSTRAINT [PK_gbook] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
```

//创建 gbook 表

```
) ON [PRIMARY]
```

gbook 表为用户留言表，其中所有的留言数据都会存放在此表中。为了提高留言本程序的可扩展性，需要创建留言分类表进行留言程序的分类，其中留言分类表字段如下所示。

- ❑ id: 用于标识留言本分类的编号，为自动增长的主键。
- ❑ classname: 用于描述分类的名称，例如“客户服务”等。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [guestbook]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[gbook_class](
    [id] [int] IDENTITY(1,1) NOT NULL,
    [classname] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    CONSTRAINT [PK_gbook_class] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

3. 验证表

验证表包括管理员验证表和用户注册表。用户注册表通常进行用户的验证包括注册和登录等操作，用户也能够通过注册表进行索引，而管理员验证表用于验证管理员的信息，验证后的管理员可以进行回复、删除等数据操作。管理员表字段如下所示。

- ❑ id: 用于标识管理员信息，为自动增长的主键。
- ❑ username: 用于标识管理员用户名。
- ❑ password: 用于标识管理员的密码，通常情况下和管理员用户名一起进行身份验证。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [guestbook]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[admin](                                //创建 admin 表
    [id] [int] IDENTITY(1,1) NOT NULL,
    [admin] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [password] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    CONSTRAINT [PK_admin] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

管理员表用于管理员的身份验证，以及确定操作人员的权限，对于用户而言需要一个表将用户的信

息进行存储和读取，其字段如下所示。

- ❑ **id**: 用于标识用户 ID，为自动增长的主键。
- ❑ **username**: 用于保存用户的用户名，当用户登录时可以通过用户名验证。
- ❑ **password**: 用于保存用户的密码，当用户使用登录时可以通过密码验证。
- ❑ **sex**: 用于保存用户的性别。
- ❑ **pic**: 用于保存用户的个性头像。
- ❑ **IM**: 用于保存用户的 QQ/MSN 等信息。
- ❑ **information**: 用于展现用户的个性签名等资料。
- ❑ **others**: 用于保存用户的备注信息。
- ❑ **ifisuser**: 用于保存用户的状态，可以设置为通过审批和未通过等。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [guestbook]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Register](                                //创建注册表
    [id] [int] IDENTITY(1,1) NOT NULL,
    [username] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [password] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [sex] [int] NULL,
    [picture] [nvarchar](max) COLLATE Chinese_PRC_CI_AS NULL,
    [IM] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    [information] [nvarchar](max) COLLATE Chinese_PRC_CI_AS NULL,
    [others] [nvarchar](max) COLLATE Chinese_PRC_CI_AS NULL,
    [ifisuser] [int] NULL,
    CONSTRAINT [PK_Register] PRIMARY KEY CLUSTERED
(
    [id] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

上述查询语句创建了注册表以存储用户的注册信息。在进行留言时，系统将使用用户注册表进行用户的身份验证，验证通过后的用户可以进行留言、索引等操作。

28.2.3 数据表关系图

系统数据库中需要进行约束，需要约束的表包括用户表、留言表和留言分类表，其约束可以使用 SQL Server Management Studio 视图进行编写，如图 28-6 所示。在创建数据表关系图后系统将弹出对话框进行关系的保存，如图 28-7 所示。

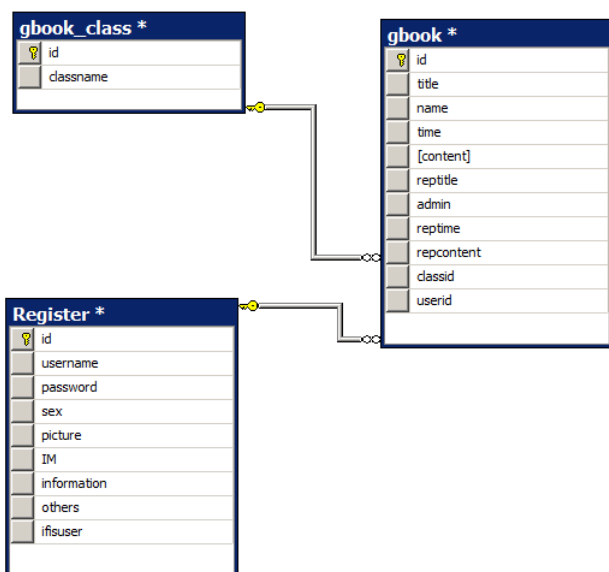


图 28-6 数据表关系图



图 28-7 保存关系

在进行关系的保存之后系统表就会被关系进行更改，其中的数据库创建的 SQL 语句也会相应的更改。作为数据库设计人员可以一开始设计毫无关系的数据表，然后在 SQL Server Management Studio 视图状态中进行关系的设计，而无需编写复杂的数据表创建 SQL 语句。

注意：当进行了数据表的更改后，其数据库中的结构和数据库中表的一些信息就会被更改，为了保证其数据的完整性和规范性，必须按照规范进行数据操作。

28.3 系统公用模块的创建

在系统开发中，为了保证其系统的可扩展性和可维护性，通常将需要经常使用的部分创建成为系统的公用模块，系统的公用模块可以被系统中的任何页面或者类库进行调用，当需要进行更改时，可以修改通用模块进行低成本维护。

28.3.1 创建 CSS

CSS 作为页面布局的全局文件，可以进行 ASP.NET 留言本全局的布局的样式控制，通过使用 CSS 能够将页面代码和布局代码相分离，这样就能够方便的进行系统样式维护。右击现有项目，在下拉菜单中选择【添加】选项，然后在【添加】选项的下拉菜单中单击【新建项】选项以创建 CSS 样式表，如图 28-8 所示。

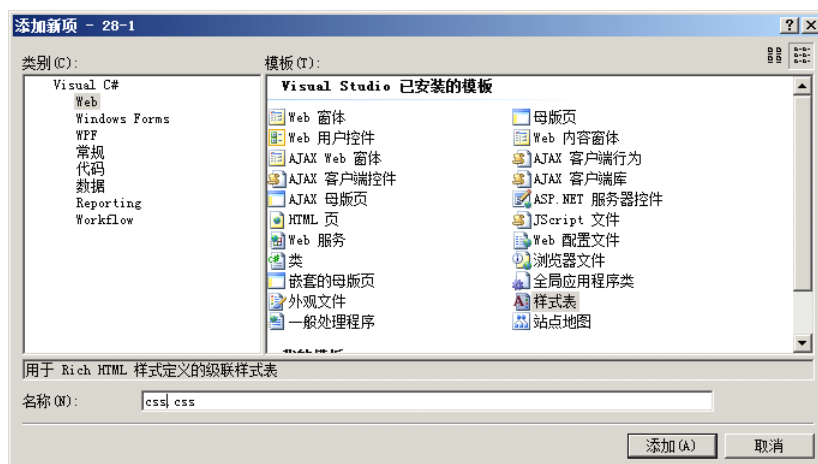


图 28-8 创建样式表

样式表可以统一存放在一个文件夹中，该文件夹能够进行样式表的统一存放和规划，以便系统可以使用不同的样式表。虽然样式表能够存放在系统的任何位置，但是为了文件的整洁，以及文件系统的可维护性建议存放在统一的文件夹中。在 CSS 文件中可以编写代码进行样式控制，示例代码如下所示。

```
body
{
    font-size:12px;
    font-family:Geneva, Arial, Helvetica, sans-serif;
    margin:0px 0px 0px 0px;
}
```

上述代码只是定义了一个 body 标签的样式，在后续开发过程中，如果需要进行样式控制可以随时更改此文件。

28.3.2 使用 SQLHelper

SQLHelper 是一个数据库操作的封装，使用 SQLHelper 类能够快速的进行数据的插入、查询、更新等操作而无需使用大量的 ADO.NET 代码进行连接。使用 SQLHelper 类为开发人员进行数据操作提供了极大的便利。在现有的系统中，在解决方案管理器中可以选择添加现有项添加现有的类库的引用，也可以通过自行创建类进行引用。在这里用户可以无需自行创建 SQLHelper 类，本书提供了 SQLHelper 类常用的精简版本，开发人员能够使用 SQLHelper 类进行高效的开发。

注意：本书提供的 SQLHelper 类在“源代码/附-SQLHELPER 类”文件夹中可以找到。

在使用现有的 SQLHelper 类之前，需要进行项目的创建，以便进行类的维护。在 SQLHelper 类中包含诸多静态方法，可以无需创建对象就能够使用 SQLHelper 类进行数据的插入、查询、更新、删除等操作，使用 SQLHelper 类也无需进行 ADO.NET 数据连接。SQLHelper 类是一个类，而不是项目，在 Visual Studio 2008 中可以创建新的类库项目进行分层开发，右击【解决方案管理器】选项，在下拉菜单中选择【添加新项目】选项，在弹出窗口中选择【类库】选项，如图 28-9 所示。

为了方便和容易阅读，新建类库项目的名称也被称为 SQLHelper，创建类库项目后可以删除自行创建的 Class1.cs 文件，在项目中添加现有项目，在计算机中找到需要的 SQLHelper 类即可。在使用 SQLHelper 类时，还需要添加命名空间 System.Collections 的引用，创建完成后的 SQLHelper 类库如图 28-10 所示。

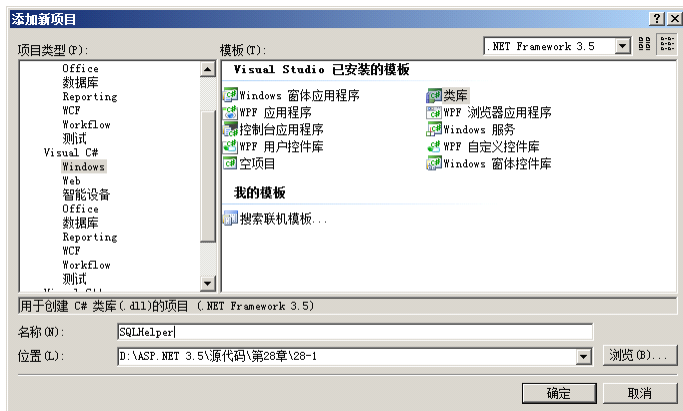


图 28-9 创建类库项目

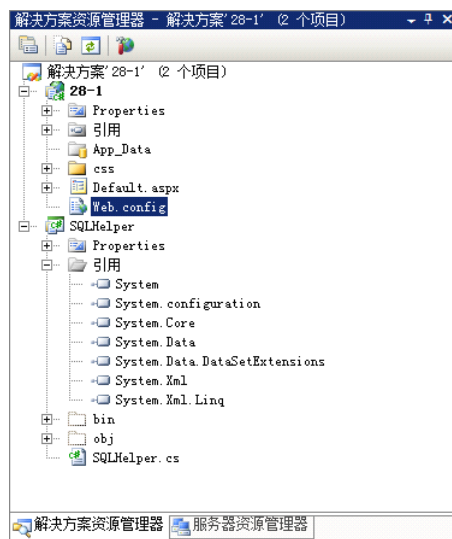


图 28-10 使用 SQLHelper

注意：为了更加方便的使用 SQLHelper，其类库的命名和命名空间的命名可以直接使用 SQLHelper 命名，通常情况下无需修改，如果开发人员需要修改，则需要类库的属性和代码中修改相应的名称。

28.3.3 配置 Web.config

Web.config 文件为系统的全局配置文件，在 ASP.NET 中 Web.config 文件提供了自定义可扩展的系统配置，在 Web.config 文件中的<appSettings/>配置节可以配置自定义信息。当使用 SQLHelper 类进行数据辅助操作时，其中包含连接字符串代码，示例代码如下所示。

```
private static readonly string database = ConfigurationManager.AppSettings["database"].ToString();
private static readonly string uid = ConfigurationManager.AppSettings["uid"].ToString();
private static readonly string pwd = ConfigurationManager.AppSettings["pwd"].ToString();
private static readonly string server = ConfigurationManager.AppSettings["server"].ToString();
private static readonly string condb = "server=" + server + ";database=" + database + ";uid=" + uid
+ ";pwd=" + pwd + "";
```

ConfigurationManager.AppSettings 能够获取 Web.config 文件中<appSettings/>配置节的相应的字段的值，上述代码分别获取<appSettings/>配置节中 database、uid、pwd、server 的值进行数据连接。在 Web.config 文件中，可以配置<appSettings/>配置节以使用 SQLHelper，示例代码如下所示。

```
<appSettings>
  <add key="server" value="(local)"/> //编辑 server 项
  <add key="database" value="guestbook"/> //编辑 guestbook 项
  <add key="uid" value="sa"/> //编辑 uid 项
  <add key="pwd" value="sa"/> //编辑 pwd 项
</appSettings>
```

在配置了 Web.config 中<appSettings/>配置的信息后，SQLHelper 类就能够使用<appSettings/>配置节中的字段和值。

注意：在使用 SQLHelper 类时，也可以自定义数据库连接字符串，在 SQLHelper 类中直接使用。当需要在项目中使用 SQLHelper 类时，还需要添加 SQLHelper 的引用。

28.4 系统界面和代码实现

在进行了系统设计和数据库设计之后就能够进行编码的实现，编码实现包括系统界面的编码实现和逻辑编码的实现，系统界面代码可以使用 CSS 进行全局样式控制，而逻辑编码实现需要在页面中进行逻辑控制。

28.4.1 留言板用户控件

在留言板的数据显示中，可以编写用户控件进行数据显示并在相应的页面中使用该用户控件，用户控件能够极大的方便开发人员进行开发维护。

用户控件是使用现有的服务器控件进行控件的制作，相比于自定义控件而言，用户控件更加容易和方便的进行开发和使用。自定义控件使用现有的服务器控件进行组合，而使用现有的数据源控件能够方便的进行分页、更新、删除等操作，所以在留言板中可以选择用户控件进行数据呈现。

右击现有项目，选择【添加】选项，在下拉菜单中单击【新建项】选项，在弹出窗口中选择【Web 用户控件】选项进行用户控件的创建，创建完成后可以进行用户控件的布局和样式控制并拖动数据源控件和数据绑定控件进行数据呈现和操作，示例代码见光盘中源代码\第 28 章\28-1\28-1\control\GbookList.aspx。

其中，代码使用了 DataList 数据控件进行数据呈现，DataList 控件可以不使用数据源控件中的更新、删除、插入等操作进行数据操作，在留言本控件中，DataList 控件只需要执行数据的呈现和分页即可。分页代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    PagedDataSource objPds = new PagedDataSource();           //使用分页类
    objPds.DataSource = this.SqlDataSource1.Select(new DataSourceSelectArguments());
    objPds.AllowPaging = true;                                   //设置是否分页
    objPds.PageSize = 20;                                       //设置分页个数
    int CurPage;                                                //设置页码
    Label2.Visible = false;                                     //隐藏导航
    Label4.Visible = false;                                     //隐藏导航
    if (Request.QueryString["Page"] != null)                   //获取传递参数
    {
        CurPage = Convert.ToInt32(Request.QueryString["Page"]); //获取参数
    }
    else
    {
        CurPage = 1;                                           //设置默认页数
    }
    objPds.CurrentPageIndex = CurPage - 1;                     //设置页码
    Label2.Visible = true;                                       //显示导航
    Label4.Visible = true;                                       //显示导航
    Label3.Text = "<a href='\"Gbook.aspx?cid=\" + Request.QueryString[\"cid\"] + \"'>首页</a>";
    Label2.Text = "<a href='\"Gbook.aspx?page=\" + Convert.ToString(CurPage + 1) + \" \"&cid=\" + Request.QueryString[\"cid\"] + \"'>下一页</a>"; //显示分页
    Label4.Text = "<a href='\"Gbook.aspx?page=\" + Convert.ToString(CurPage - 1) + \" \"&cid=\" + Request.QueryString[\"cid\"] + \"'>上一页</a>"; //显示分页
```

```

        if (CurPage == 1)                                //是否只有一页
        {
            Label4.Visible = false;                        //屏蔽下一页
        }
        if (objPds.IsLastPage)                            //是否最后一页
        {
            Label2.Visible = false;                        //屏蔽上一页
        }
        DataList1.DataSourceID = "";                      //清空绑定
        DataList1.DataSource = objPds;                    //选择数据源
        DataList1.DataBind();                             //数据重绑定
    }

```

使用数据分页类能够对 ListView 控件进行分页操作,当数据库中的信息过多时用户可以使用分页进行留言的查看,当管理员进行留言本的回复时,也可以使用分页操作对原来未进行回复的操作进行选择 and 回复。在创建自定义控件时,其数据源控件可以不必支持数据更新、插入和删除等操作。另外,在 SqlDataSource 数据源控件中还使用了 Web.config 中的数据连接字符串进行数据呈现和操作的支持。

28.4.2 管理员登录实现

管理员登录的实现可以使用前面开发的登录模块进行管理员身份的验证开发,这里也可以简单的使用服务器控件进行管理员登录的实现。管理员登录页面 HTML 代码见光盘中源代码\第 28 章\28-1\28-1\admin\login.aspx。

页面代码中包含了一个样式类为 admin_login 的 DIV 层,该层用于管理员登录界面中控件的存放和样式控制,其中 CSS 样式控制代码如下所示。

```

.admin_login
{
    margin:100px auto;
    width:500px;
    border:1px solid #ccc;
    background:#f0f0f0;
}

```

上述代码定义了该层的背景颜色、边框粗细、宽度以及外间距,布局后如图 28-11 所示。

图 28-11 管理员登录页面布局

当管理员单击【Login】按钮时会进行身份验证,如果管理员身份正确则会给管理员分配一个 Session 对象,在其他的页面中需要使用 Session 对象进行身份验证。示例代码如下所示。

```

protected void Button1_Click(object sender, EventArgs e)
{
    string strsql = "select * from admin where admin='" + TextBox1.Text + "' and password='"
        + TextBox2.Text + "'";                                //使用 SQL 语句
    SqlDataReader sdr = SQLHelper.SQLHelper.ExecReader(strsql); //使用 SqlDataReader
}

```



```

        if (sdr.Read())                                //判断是否有数据
        {
            Session["admin"] = TextBox1.Text;          //给予 Session
            Response.Redirect("../default.aspx");        //页面跳转
        }
        else
        {
            Label1.Text = "无法登录,用户名或密码错误"; //提示错误
        }
    }
}

```

上述代码使用了 `SqlDataReader` 类和 `SQLHelper` 类进行数据查询，从上述代码可以看出使用 `SQLHelper` 类减少了大量的代码，使用 ADO.NET 进行数据操作还需要进行数据连接、创建适配器、进行数据集填充和数据集行数的判断，而使用 `SQLHelper` 类精简了该过程，极大的简化了开发人员的开发。

28.4.3 用户注册登录实现

用户在留言之前需要进行登录，如果用户没有账号就需要在登录之前进行注册操作，注册操作可以使用注册模块进行数据操作，这里只需要简单的进行数据插入即可，注册页面 HTML 核心代码见光盘中源代码\第 28 章\28-1\28-1\login.aspx。

其中的页面代码实现了用户进行注册时所必要的控件以便用户能够进行快速注册，在用户注册时，用户名和密码为必填项，而其他为选填项目，当用户单击按钮控件进行注册时，会执行相应的注册事件进行数据库操作，示例代码如下所示。

```

protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        string strsql =
            "insert into register (username,password,sex,picture,IM,information,others,ifisuser) values ('" +
            TextBox1.Text + "','" + TextBox2.Text + "','" + DropDownList1.Text + "','" +
            TextBox3.Text + "','" + TextBox4.Text + "','" + TextBox5.Text + "','" +
            TextBox6.Text + "',1)"; //编写 INSERT 语句
        SQLHelper.SQLHelper.ExecNonQuery(strsql); //执行数据操作
        if (!String.IsNullOrEmpty(Request.QueryString["id"])) //选择跳转
        {
            Response.Redirect("Gbook.aspx?id=" + Request.QueryString["id"]);
        }
        else
        {
            Response.Redirect("default.aspx"); //跳转到默认页
        }
    }
    catch
    {
        Response.Redirect("default.aspx"); //错误也跳转到默认页
    }
}

```

上述代码执行数据操作只用了一条语句，并没有进行 ADO.NET 中的数据连接、`Command` 对象的创

建以及操作，只使用了一行代码就简化了数据操作。当用户进行注册时，可以在留言页面进行注册，也可以在首页进行注册，如果在留言页面进行注册跳转，希望注册后还能够直接跳回留言页面，如果用户在首页进行注册，注册完毕后应跳转回首页然后进行留言本的选择。

28.4.4 用户登录实现

用户登录实现同管理员登录相同，管理员登录时进行管理员用户名和密码的查询，如果存在数据则说明存在管理员，并赋予管理员相应的 Session 对象。对于用户登录而言，其代码基本相同，登录事件处理代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    string strsql = "select * from register where username='" + TextBox1.Text + "' and password='"
        + TextBox2.Text + "'"; //查询 SQL
    SqlDataReader sdr = SQLHelper.SQLHelper.ExecReader(strsql); //执行查询
    if (sdr.Read()) //判断用户
    {
        Session["username"] = TextBox1.Text; //给予权限
        Session["userid"] = sdr["id"].ToString(); //给予权限
        if (!String.IsNullOrEmpty(Request.QueryString["id"])) //选择跳转
        {
            Response.Redirect("Gbook.aspx?id=" + Request.QueryString["id"]); //跳转
        }
        else
        {
            Response.Redirect("default.aspx"); //默认跳转
        }
    }
    else
    {
        Label1.Text = "无法登录,用户名或密码错误"; //抛出异常
    }
}
```

上述代码执行了查询，将查询填充到 SqlDataReader 对象中，如果数据库中包含相应的查询数据，则 SqlDataReader 对象的 Read 方法会返回 true，否则返回 false。

用户登录界面可以同管理员界面一样，由于篇幅限制，这里就不再进行登录界面的样式布局和样式控制，直接使用管理员登录界面样式进行用户登录的样式控制。同样为了提高用户体验，如果用户在登录前已经打开了某个留言本，则在登录后能够直接跳转到该留言本，而如果用户在首页进行登录操作，则用户依旧会跳转回首页。

28.4.5 留言本界面布局

留言本界面是留言本项目中最为重要的页面，在该页面用户能够进行留言操作并可以查看管理员对自己留言的回复，而管理员能够在该页面进行数据管理和回复。留言本页面 HTML 核心代码见光盘中原代码第 28 章\28-1\28-1\Gbook.aspx。

留言本界面使用了 DIV+CSS 让页面的样式更加丰满、用户体验更加友好，在留言本界面中使用了

前面制作的用户控件，并且能够支持分页等操作。在页面以及自定义控件中定义了 CSS 样式，通过编写 CSS 样式文件能够进行控件和页面的样式控制，CSS 代码如下所示。

```
.gbook_main_title                                     //定义留言本标题界面
{
    margin:0px auto;
    margin-top:50px;
    width:800px;
    border:1px solid #ccc;
    background:white url(..images/top.png);
    height:200px;
}
.gbook_main                                           //定义留言本主界面
{
    margin:5px auto;
    width:800px;
    border:1px solid #ccc;
    background:white;
}
.gbook_banner                                         //定义留言本导航界面
{
    margin:5px auto;
    width:790px;
    border:1px solid #ccc;
    background:white;
    padding:5px 5px 5px 5px;
}
.left                                                 //定义留言本侧边界面
{
    width:200px;
    float:left;
}
.right                                                //定义留言本主界面
{
    width:579px;
    margin-left:10px;
    float:left;
    border-left:1px dashed #ccc;
    padding:5px 5px 5px 5px;
}
.copyright                                           //定义版权界面
{
    margin:5px auto;
    width:800px;
    border:1px solid #ccc;
    background:white;
    font-size:10px;
    text-align:center;
}
```

上述 CSS 代码定义了页面中的样式，能够让页面的样式更加友好。在留言本页面中，使用了前面制

作的用户控件，用户控件的模板中同样使用 CSS 进行了定义，通过编写相应的 CSS 代码能够控制控件中数据的呈现方式，CSS 代码如下所示。

```
.g_title //定义控件标题
{
    background:#f0f0f0;
    padding:5px 5px 5px 5px;
}
.g_content //定义控件内容界面
{
    padding:5px 5px 5px 5px;
}
.g_reply //定义控件回复界面
{
    padding:5px 5px 5px 5px;
}
.g_table //定义控件循环表格
{
    border:1px solid #ccc;
    margin:5px 5px 5px 5px;
    width:98%;
}
```

上述代码定义了留言本中用户控件的样式，在数据呈现的过程中，系统会将样式和页面代码整合在一起呈现给用户，留言本界面布局如图 28-12 所示。



图 28-12 ASP.NET 留言本主界面布局

28.4.6 留言功能实现

当用户单击【留言】按钮后，用户就能够进行留言操作，使用 SQLHelper 类只需要编写 INSERT 语句就能够实现数据插入，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
```

```

    {
        //编写执行插入的 INSERT 语句
        string strSql =
            "insert into gbook (title,name,time,content,replytitle,admin,replytime,replycontent,classid,userid)
            values ('" + TextBox2.Text + "','" + Session["username"].ToString() + "','" + DateTime.Now +
            "','" + TextBox1.Text + "','" + DateTime.Now + "','" + Request.QueryString["cid"] + "','" +
            Session["userid"].ToString() + "')";
        SQLHelper.SQLHelper.ExecNonQuery(strsql);
        Response.Redirect("Gbook.aspx?cid=" + Request.QueryString["cid"]);
    }
    catch
    {
        //编写错误处理
    }
}

```

在进行留言功能的实现前，首先需要判断用户是否注册或登录，如果用户没有注册或登录，那么用户就只能查看相应的留言而无权进行留言；如果进行注册并登录，则用户在留言本页面能够进行留言操作。所以在页面加载时必须对用户的身份进行验证判断，示例代码如下所示。

```

protected void Page_Load(object sender, EventArgs e)
{
    if (Session["username"] == null || Session["userid"] == null)
    {
        Panel1.Visible = false;
    }
}

```

上述代码在页面加载时被执行，如果页面加载时发现执行页面操作的用户并不是网站的用户就会隐藏留言区域，反之将会呈现留言区域以供用户进行留言操作。

28.4.7 回复功能实现

当管理员进行留言查看时，对于相应的留言可以选择对用户的留言进行回复，管理员能够通过页面中的回复超链接进行回复。留言页面 HTML 代码如下所示。

```

<body style="background:white url(..images/bg.png) repeat-x;">
    <form id="form1" runat="server">
        <div class="gbook_main">
            回复留言:<br />
            <asp:TextBox ID="TextBox1" runat="server" Height="150px" TextMode="MultiLine"
            Width="100%"></asp:TextBox>
            <br />
            <asp:Button ID="Button1" runat="server" Text="回复留言" />
        </div>
    </form>
</body>

```

当管理员单击【回复】超链接时，会跳转到该回复页面，该页面能够执行相应的留言的回复，当用户单击【回复】超链接时，必须在回复页面进行判断。如果是管理员，则允许进行回复，如果不是管理员，则跳转回留言页面，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null) //如果不是管理员
    {
        Response.Redirect("../Gbook.aspx?cid=" + Request.QueryString["cid"] + ""); //跳回页面
    }
}
```

如果身份验证后判断确实是管理员，则能够打开回复页面进行回复。当管理员回复后，单击按钮控件就能够进行回复并跳转到相应的页面，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        string strsql = "update gbook set repcontent=" + TextBox1.Text + ",reptime=" +
DateTime.Now
+ ",admin=" + Session["admin"].ToString() + " where id=" +
Request.QueryString["id"] + ""; //编写更新
SQLHelper.SQLHelper.ExecNonQuery(strsql); //执行更新
Response.Redirect("../Gbook.aspx?cid=" + Request.QueryString["cid"] + ""); //页面跳转
    }
    catch
    {
        //异常处理
    }
}
```

上述代码通过传递的参数进行更新操作，在用户控件中，每一列数据都包含回复和删除两个操作的超链接，这两个操作所在的页面需要通过获取控件中传递过来的参数进行数据的查询和操作，页面通过获取 cid 参数进行页面的跳转，而通过获取 id 的参数进行相应的数据处理。

28.4.8 删除功能的实现

在执行回复和删除操作时，页面通过获取 cid 参数进行页面的跳转，而通过获取 id 的参数进行相应的数据处理。删除页面并不需要进行数据的呈现或 HTML 的呈现，删除页面只需要进行数据的处理。当管理员执行删除操作时，同样需要进行身份验证，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null) //如果不是管理员
    {
        Response.Redirect("../Gbook.aspx?cid=" + Request.QueryString["cid"] + ""); //跳回页面
    }
}
```

如果验证通过，则能够执行删除操作，删除操作通过传递过来的 id 参数进行数据删除，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null) //管理员权限判断
    {
```



```

        Response.Redirect("../Gbook.aspx?cid=" + Request.QueryString["cid"] + ""); //页面跳转
    }
    else if
        (String.IsNullOrEmpty(Request.QueryString["cid"])
        || String.IsNullOrEmpty(Request.QueryString["id"])) //如果参数不为空
    {
        Response.Redirect("../default.aspx"); //页面跳转
    }
    else
    {
        try
        {
            string strsql = "delete from gbook where id=" + Request.QueryString["id"] + "";
            SQLHelper.SQLHelper.ExecNonQuery(strsql); //执行删除
            Response.Redirect("../Gbook.aspx?cid=" + Request.QueryString["cid"] + ""); //页面跳转
        }
        catch
        {
            Response.Redirect("../Gbook.aspx?cid=" + Request.QueryString["cid"] + ""); //页面跳转
        }
    }
}

```

在删除页面中可以直接进行数据删除而无需通过 HTML 呈现或页面显示进行数据删除，当数据删除后可以直接跳转到相应分类的页面。

注意：在执行删除操作时，需要判断传递过来的参数是否存在或安全，例如这里判断传递的 cid 和 id 的值是否存在，如果不存在则视用户的操作为非法操作，即跳转到安全的页面。

28.4.9 用户索引实现

当用户进行留言后，可能会在很长一段时间内不会再次进行网站的访问和登录，当经过很长的时间后，用户再次返回该网站进行访问，就会比较关心自己的留言是否被回复。而留言页面中的数据排序是根据留言 ID 进行倒序的，在经过很长一段时间后，用户的留言很可能就在很多页之后了。为了方便用户能够在网站中进行留言的检索，可以为用户创建索引，方便用户进行较早留言的查看，用户索引页面 HTML 代码见光盘源代码第 28 章\28-1\28-1\userindex.aspx。

上述代码同样使用 ListView 控件进行数据呈现和显示，而不同的是数据源控件的配置。在配置时，数据源控件需要获取传递的 uid 参数进行查询，数据源控件代码如下所示。

```

<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%= $ ConnectionStrings.guestbookConnectionString %>"
    SelectCommand="SELECT gbook.* FROM gbook,register WHERE (gbook.userid = @userid) AND
        (register.id=gbook.userid) ORDER BY gbook.id DESC">
    <SelectParameters>
        <asp:QueryStringParameter Name="userid" QueryStringField="uid" />
        <!--获取传递的参数进行查询--!>
    </SelectParameters>
</asp:SqlDataSource>

```

数据源控件查询的返回的值同留言本中查询的返回类型相同，但是索引页面查询的结果与留言本中查询的条件不同，留言本中的查询条件是按照留言本分类进行查询，而用户的索引是通过用户的 ID 号

进行查询。当数据填充 ListView 控件后，同样需要让索引页面的 ListView 控件支持分页操作，分页代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    ..... //前面基本相同
    .....
    Label3.Text = "<a href=\"UserIndex.aspx?uid=\" + Request.QueryString[\"uid\"] + \"\">首页</a>";
    Label2.Text = "<a href=\"UserInde.aspx?page=\" + Convert.ToString(CurPage + 1) + \"&uid=\"
        + Request.QueryString[\"uid\"] + \"\">下一页</a>"; //页面和参数不同
    Label4.Text = "<a href=\"UserInde.aspx?page=\" + Convert.ToString(CurPage - 1) + \"&uid=\"
        + Request.QueryString[\"uid\"] + \"\">上一页</a>"; //页面和参数不同
    if (CurPage == 1) //是否只有一页
    {
        Label4.Visible = false; //屏蔽下一页
    }
    if (objPds.IsLastPage) //是否最后一页
    {
        Label2.Visible = false; //屏蔽上一页
    }
    DataList1.DataSourceID = ""; //清空绑定
    DataList1.DataSource = objPds; //选择数据源
    DataList1.DataBind(); //数据重绑定
}
```

ListView 控件的分页操作代码基本相同，开发人员值需要将页面的从 Gbook.aspx 改为 UserIndex.aspx，然后将传递的参数更改即可。

28.5 用户体验优化

在基本的系统功能模块编写完毕后，还需要对现有的功能模块进行优化。优化不仅仅包括应用程序代码的优化，还包括系统的界面以及用户体验的优化。为了能够提升用户体验，这里就需要使用 AJAX 对系统进行功能进行优化。

28.5.1 AJAX 留言实现

在 ASP.NET 3.5 中，系统已经为 AJAX 提供了原生的支持，开发人员能够直接拖动控件进行 AJAX 应用程序的开发而无需复杂的功能实现。AJAX 应用有一个优点，就是能够让页面进行无刷新的数据交互，这样就能够提高用户体验度。

1. AJAX 留言页面

打开留言页面进行 AJAX 控件布局。在留言页面中，最主要的数据操作就是留言的实现，这里的留言实现所需要的控件都需要使用 AJAX 控件进行“包裹”和“承载”。留言控件已经存放在 Panel 控件中，用于不同身份权限的 HTML 呈现。同样，AJAX 控件也可以陈放在 Panel 控件中，示例代码见光盘源代码\第 28 章\28-1\28-1\Gbook.aspx。

其中，代码在 Panel 控件中加入了 AJAX 控件以保证页面数据能够局部刷新而不会造成页面的其他数据的刷新。在进行留言时，还需要对用户进行提示以告知用户正在留言。为了实现这样的功能，就需

要在 UpdatePanel 控件中加入 UpdateProgress 控件，示例代码如下所示。

```
<asp:UpdateProgress ID="UpdateProgress1" runat="server">
    <ProgressTemplate>
        
    </ProgressTemplate>
</asp:UpdateProgress>
```

上述代码在 UpdateProgress 控件中插入了“等待”图片，该图片会在系统执行过程中呈现。当用户在留言时，局部的数据会发送到数据库进行数据发送和回传，在这个过程中，无疑是需要时间的。使用一个“等待”图片或者一段“等待”字样能够提示用户，从而提高用户体验。UpdateProgress 控件如图28-13所示。

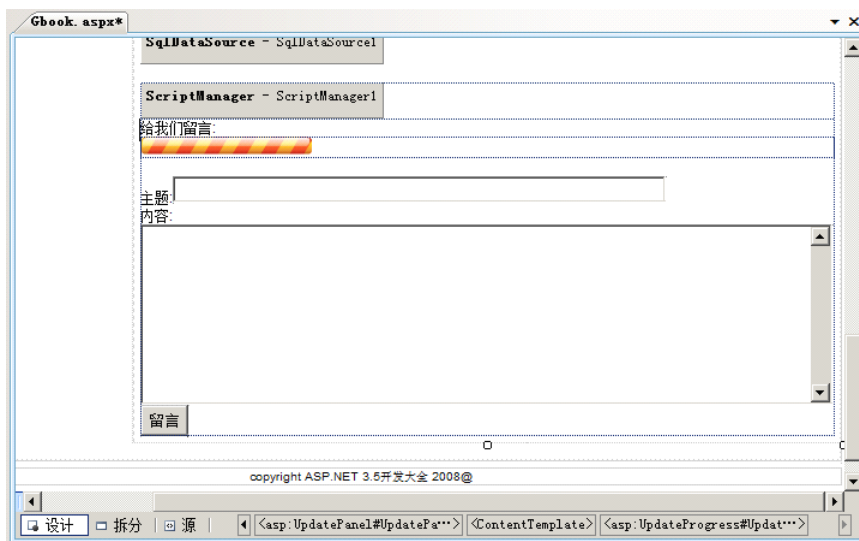


图 28-13 UpdateProgress 控件的使用

在留言过程中，如果留言本的数据量非常的小，那么留言本中的 UpdateProgress 控件就不会呈现在用户面前。因为当留言本的数据非常小时，其数据的交互和页面的往返过程也会变得非常的简单，这样就会造成 UpdateProgress 控件无法呈现。

为了让 UpdateProgress 控件呈现在用户面前，开发人员可以使用线程进行控制。在用户单击【留言】按钮时，系统会停止 3 秒以便模拟大量数据时等待的过程。示例代码如下所示。

```
try
{
    System.Threading.Thread.Sleep(3000);           //系统休眠
    string strsql
    = "insert into gbook (title,name,time,content,replytitle,admin,replytime,replycontent,classid,userid)
    values ('" + TextBox2.Text + "','" + Session["username"].ToString() + "','" + DateTime.Now +
    "','" +
    + TextBox1.Text + "','" + DateTime.Now + "','" + Request.QueryString["cid"] + "','" +
    Session["userid"].ToString() + "')";           //编写 SQL 语句
    SQLHelper.SQLHelper.ExecNonQuery(strsql);       //执行 SQL 语句
    Response.Redirect("Gbook.aspx?cid=" + Request.QueryString["cid"]); //页面跳转
}
```

上述代码通过使用 System.Threading.Thread.Sleep 方法进行系统休眠以达到系统等待的目的，从而能够让 UpdateProgress 控件呈现在浏览器当中以提示用户操作正在执行。

注意：在该代码段中使用 `System.Threading.Thread.Sleep` 方法进行系统休眠在实际的应用程序开发中是不可取的，这里使用 `System.Threading.Thread.Sleep` 方法进行系统休眠是为了模拟大量数据时页面的信息提示。

在对页面进行了 AJAX 控件的布局后，还需要对操作进行修改。这里值得一提的是，当使用了 AJAX 控件执行数据操作，AJAX 控件在执行数据的过程中的状态都是“激活”的，如图 28-14 所示。

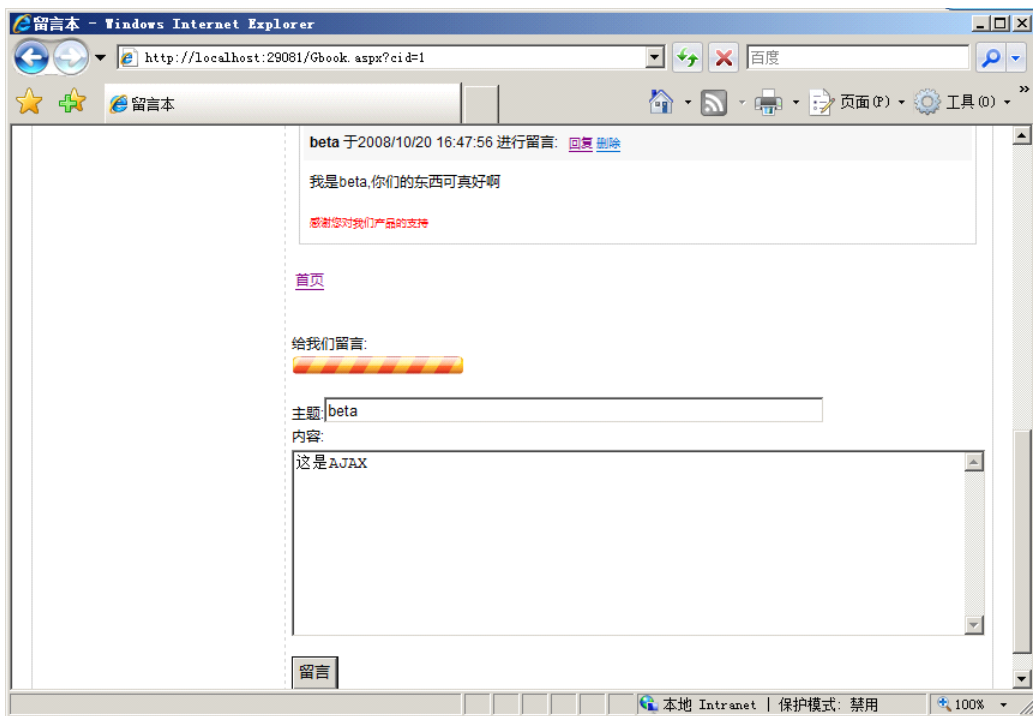


图 28-14 激活的控件状态

从图 28-14 中可以看出，当用户单击了【留言】按钮后，系统会执行相应的数据操作进行数据的增加，但是其中【留言】按钮依旧可以点击。在“等待”过程中，如果用户重复的单击【留言】按钮，则会造成数据的重复操作，当用户在执行过程中单击多个【留言】按钮时，系统也会添加同样多的数据，这不仅会造成数据冗余还会降低用户体验。

2. AJAX 留言页代码控制

为了解决以上问题，在 AJAX 留言页面中需要通过代码控制进行数据冗余的防止。这里可以暂时归纳成以下操作步骤：

- ❑ 用户单击前：用户单击前控件的状态是“激活”的，这也就是说用户可以单击按钮控件进行数据的操作。
- ❑ 用户单击时：用户单击时控件的状态应该是“非激活”的，这也就是说如果用户正在提交留言，那么就应该屏蔽控件的信息防止用户重复按键。
- ❑ 用户单击后：当用户单击后，其控件状态应该恢复成“激活”状态，这样用户就能够再次进行数据的提交。

从以上步骤可以看出，用户在执行单击时，相应的控件应该是“非激活”状态。这里最主要的是按钮控件。当用户单击【留言】按钮时，相应的数据会执行并产生页面回传，在这个过程中，应该使用操作屏蔽【留言】按钮防止用户再次进行提交，示例代码如下所示。

```
StringBuilder sb = new StringBuilder();
```

```
//声明字符串
```

```

sb.Append("if (typeof(Page_ClientValidate) == 'function')
{ if (Page_ClientValidate() == false) { return false; }};");           //编写 javascript 脚本
sb.Append("this.disabled  = true;");                                   //编写函数体
sb.Append(this.ClientScript.GetPostBackEventReference(Button1, ""));    //判断是否回传
sb.Append(";");
sb.Append("SetUIStyle();");                                             //设置 UI 的样式
Button1.Attributes.Add("onclick", sb.ToString());                      //控件添加属性

```

上述代码在代码页面中为控件添加了属性。当用户单击【留言】按钮并触发了相应的数据插入事件时，在回传过程中【留言】按钮的激活状态都会被修改为“非激活”，以阻止用户进行重复的单击操作，如图 28-15 所示。

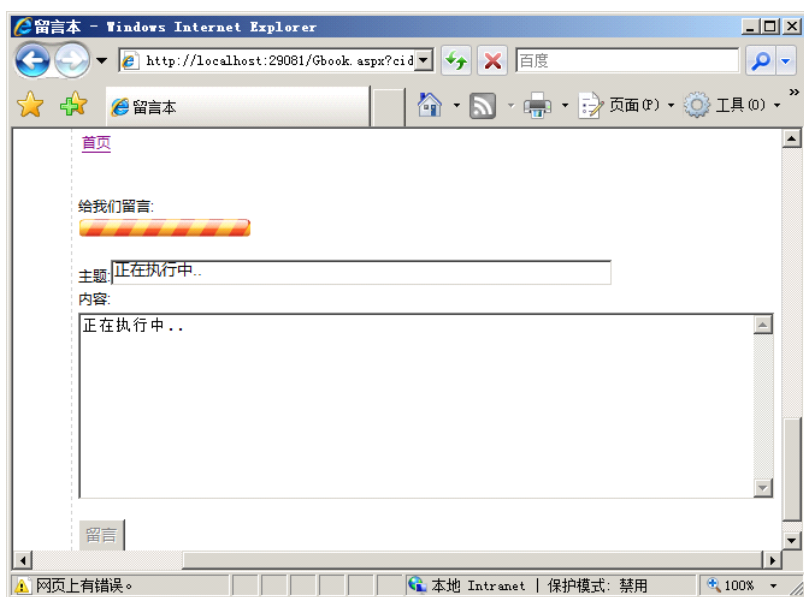


图 28-15 控件激活状态

从图 28-15 可以看出，当用户单击【留言】按钮后，【留言】按钮变灰，即无法单击。如果用户多次单击【留言】按钮，系统也会将多次单击认定为是一次单击，从而只执行一次数据操作。这样不仅避免了数据冗余，同样还提高了用户体验。

注意：上述代码段可以直接编写在 Page_Load 事件中。当页面初次被加载时，就需要对相应的控件进行属性的添加。这样在页面的执行中，该控件都会具备相应的属性。

28.5.2 AJAX 数据重绑定

在优化了 AJAX 留言功能之后可以发现当用户单击【留言】控件后，系统会跳转到该页面以便数据的重绑定，这样同样造成了页面全局刷新。当用户单击【留言】控件后，系统希望页面能够不跳转而重新加载页面即可执行数据的重绑定，这里同样需要使用 AJAX 进行局部刷新。

为了能够让数据能够进行局部刷新，这里需要将留言显示控件放置在 UpdatePanel 控件中，示例见光盘中源代码\第 28 章\28-1\28-1\Gbook.aspx。

将留言显示控件放置在 UpdatePanel 控件中之后，就能够进行页面的局部刷新。值得注意的时，页面数据的重绑定的过程比较简单，只需要将控件进行重新加载即可。示例代码如下所示。

```

protected void Button1_Click(object sender, EventArgs e)
{

```

```

try
{
    System.Threading.Thread.Sleep(3000); //模拟过程
    string strsql
    = "insert into gbook (title,name,time,content,replytitle,admin,replytime,replycontent,classid,userid)
    values ('" + TextBox2.Text + "','" + Session["username"].ToString() + "','" + DateTime.Now +
    "','" + TextBox1.Text + "','" + DateTime.Now + "','" + Request.QueryString["cid"] + "','" +
    Session["userid"].ToString() + "')"; //生成 SQL 语句
    SQLHelper.SQLHelper.ExecNonQuery(strsql); //执行 SQL 语句
    GbookList1.Reload(); //执行重加载方法
    //Response.Redirect("Gbook.aspx?cid=" + Request.QueryString["cid"]);
}
catch
{
    //编写错误处理
}
}

```

上述代码使用了 GbookList 控件的重加载方法，该方法必须要在 GbookList 控件中进行编写实现才能够调用。GbookList 控件中的 Reload 方法编写如下所示。

```

public void Reload()
{
    PagedDataSource objPds = new PagedDataSource(); //设置分页数据
    objPds.DataSource = this.SqlDataSource1.Select(new DataSourceSelectArguments());
    objPds.AllowPaging = true; //设置分页属性为 true
    objPds.PageSize = 20; //设置分页数
    int CurPage; //设置页码
    Label2.Visible = false; //重新清空统计
    Label4.Visible = false; //重新清空统计
    if (Request.QueryString["Page"] != null) //判断传递参数
    {
        CurPage = Convert.ToInt32(Request.QueryString["Page"]); //参数的显式转换
    }
    else
    {
        CurPage = 1; //设置默认页面
    }
    objPds.CurrentPageIndex = CurPage - 1; //设置分页索引
    Label2.Visible = true; //开始统计信息
    Label4.Visible = true; //开始统计信息
    Label3.Text = "<a href='\"Gbook.aspx?cid=\" + Request.QueryString[\"cid\"] + '\">首页</a>";
    Label2.Text = "<a href='\"Gbook.aspx?page=\" + Convert.ToString(CurPage + 1) + \"&cid=\" +
    Request.QueryString[\"cid\"] + '\">下一页</a>"; //设置分页
    Label4.Text = "<a href='\"Gbook.aspx?page=\" + Convert.ToString(CurPage - 1) + \"&cid=\" +
    Request.QueryString[\"cid\"] + '\">上一页</a>"; //设置分页
    if (CurPage == 1) //判断是否为第一页
    {
        Label4.Visible = false;
    }
    if (objPds.IsLastPage) //判断是否为最后一页

```



```

    {
        Label2.Visible = false;
    }
    DataList1.DataSourceID = "";
    DataList1.DataSource = objPds;
    DataList1.DataBind();
}

```

//清空绑定数据源
//重新绑定数据源
//数据源重绑定

上述代码实现了用户控件的数据重绑定，当执行该方法时，数据绑定控件中的数据将会被“刷新”并呈现在客户端浏览器中。在使用该用户控件的页面中，可以使用该用户控件的 **Reload** 方法进行数据刷新。当用户单击【留言】按钮时，AJAX 控件能够完成页面的局部刷新，从而实现了用户无需进行页面跳转即可进行数据查看的功能，如图 28-16 所示。

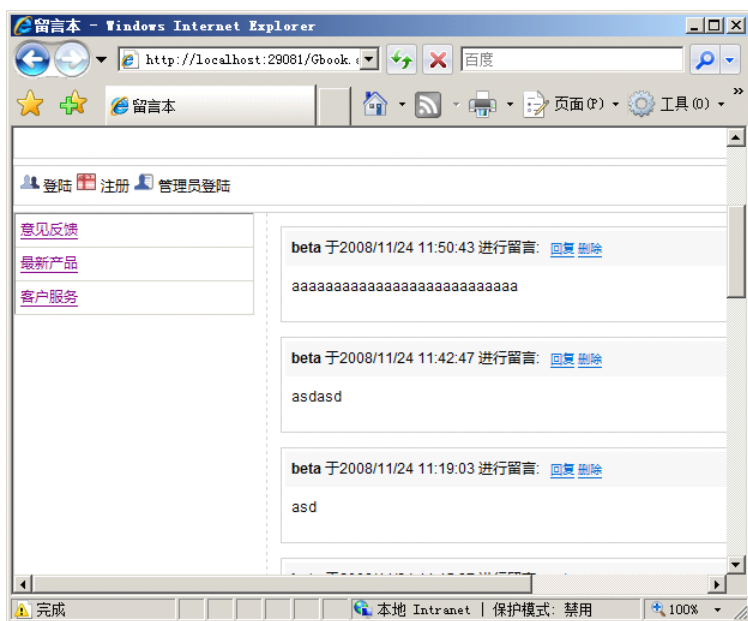


图 28-16 无刷新数据重绑定

注意：在执行无刷新数据重绑定时，依旧推荐能够使用 **UpdateProgress** 控件进行执行状态的呈现以提高用户体验。

28.5.3 系统导航实现

导航在系统中非常的重要，良好的导航能够方便的用户选择【登录】、【注册】等操作而无需手动的编写地址，这样能够提高网站的用户体验。对于留言本系统而言，留言本需要进行登录操作才能够执行数据操作，另外，留言本系统中还包括不同的用户权限，例如用户和管理员。对于不同的用户权限所需要展示的导航也不尽相同。

对于不同的权限，系统希望权限不同的用户所看到的导航也不尽相同。在前面的代码实现中，实现了固定的导航，实现了登录、注册、管理员登录等操作。值得注意的是，如果用户是一个管理员，那么这个导航应该是合理的，而如果用户不是管理员而是一个普通用户，【管理员登录】选项是不应该被呈现的。系统导航可以规划如下：

- 管理员：管理员导航包括【管理员名称】选项、【管理员注销】选项。
- 用户：用户导航包括【用户名称】选项、【用户索引导航】选项、【用户注销】选项。

❑ 未登录用户：未登录用户包括【登录】选项、【注册】选项和【管理员登录】选项。

在 Web 开发中，导航基本是每个页面都需要的，在每个页面手动的添加导航是非常不现实的。在应用程序编写过程中，可以使用动态页面进行数据的呈现并通过 JS 的方式进行调用。右击当前项目，选择【新建文件夹】选项，在项目根目录中创建【js】文件夹。在 js 文件夹中创建 banner.aspx 用于导航。删除 banner.aspx 文件中生成的代码而只保留声明代码，示例代码如下所示。

```
<%@ Page
Language="C#" AutoEventWireup="true" CodeBehind="banner.aspx.cs" Inherits="_28_1.js.banner" %>
```

上述代码是一段声明代码，开发人员能够在代码下使用 javascript 脚本的形式用于页面的数据的呈现，示例代码如下所示。

```
<%@ Page
Language="C#" AutoEventWireup="true" CodeBehind="banner.aspx.cs" Inherits="_28_1.js.banner" %>
<%
    if (Session["admin"] != null)
    {
%>
        document.write('
        你好:<% Response.Write(Session["admin"].ToString()); %>&nbsp;');
        document.write('
        <a href="../admin/logout.aspx">注销</a>&nbsp;');
    <%
    }
    else if (Session["username"] != null && Session["userid"] != null)
    {
%>
        document.write('
        你好:<a href="../personal.aspx?uid=<% Response.Write(Session["userid"].ToString()); %>">
        <% Response.Write(Session["username"].ToString()); %></a>&nbsp;');
        document.write('
        <a href="../userindex.aspx?uid=<% Response.Write(Session["userid"].ToString()); %>">
        我的留言</a>&nbsp;');
        document.write('
        <a href="../admin/logout.aspx">注销</a>&nbsp;');
    <%
    }
    else if (Session["username"] == null && Session["userid"] == null && Session["username"] == null)
    {
%>
        document.write('
        <a href="../login.aspx">登录</a>&nbsp;');
        document.write('
        <a href="../register.aspx">注册</a>&nbsp;');
        document.write('
        <a href="../admin/login.aspx">管理员登录</a>&nbsp;');
    <%
    }
%>
```

上述代码制作了一个用于呈现导航的 js，通过调用此 js 文件能够呈现不同的导航，示例代码如下所示。

```
<div class="gbook_banner">
  <script src="js/banner.aspx" type="text/javascript"></script>
</div>
```

上述代码为留言本页面中的导航页面，在该页面使用制作的 js 文件能够快速的呈现相应的导航并实现逻辑判断。当有多个页面需要使用该导航时，可以直接使用该 js 文件而不需要额外的在每个页面中进行逻辑判断和事务处理，这样就能够有效降低应用程序之间的耦合度。

注意：在编写 js 文件的过程中，需要以 document.write 的方式进行 HTML 标签以及数据的输出。任何动态页面都能够作为 js 文件的进行编写和调用。

该导航实现了不同的用户权限的不同视图呈现，对于不同的用户而言，其导航也不应该相同。而针对不同的用户提示用户进行不同的操作是非常必要的，例如这里的用户进行【我的索引】选项的选择，就会跳转到该用户的索引页面而不需用户手动编写。

28.5.4 侧边栏界面优化

侧边栏也是导航的另一种形式，但是在留言本应用程序中，侧边栏用于不同的留言本之间的跳转。开发人员能够优化侧边栏界面以便用户能够更加方便的进行留言本的跳转和索引。现有的侧边栏界面如图 28-17 所示。

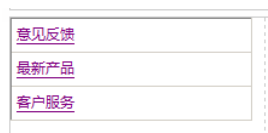


图 28-17 现有侧边栏界面

从现有的侧边栏界面可以看出，其界面是非常不友好的。由于这里使用了 GridView 控件，开发人员能够自动套用格式以便数据的呈现，如图 28-18 所示，开发人员还能够自定义控件以便数据的呈现，如图 28-19 所示。

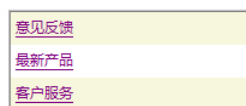


图 28-18 自动套用格式



图 28-19 自定义格式

开发人员可以使用不同的格式以便侧边栏能够更加友好。从图 28-18 和图 28-19 中的侧边栏呈现形式可以看出，及时对侧边栏进行样式控制依旧不能对其中的超链接进行样式控制。因为超链接的样式控制需要在 CSS 文件中编写，示例代码如下所示。

```
a:link
{
    text-decoration: none;color: #000000;
}
a:active
{
    text-decoration: none;color: #000000;
}
a:visited
{
    text-decoration: none;color: #000000;
```

```
}
```

上述代码控制了超链接文本的样式。其中 `a:link` 的意义是链接的样式，`a:active` 是激活状态的超链接样式，而 `a:visited` 是已访问过的超链接文本样式。除了这几种超链接文本样式以外，还包括 `a:hover` 超链接样式，`a:hover` 是当鼠标移动到超链接文本上时，超链接文本的呈现样式。示例代码如下所示。

```
a:hover
{
    color:white;
    background:red;
}
```

上述代码则通过编写 `a:hover` 样式进行超链接文本样式的丰富。当超链接呈现在用户浏览器中时，其呈现方式首先以 `a:link` 样式呈现，如图 28-20 所示。当用户的鼠标放置在连接上时（并不是点击，而是放置），其样式会以 `a:hover` 的形式呈现，如图 28-21 所示。

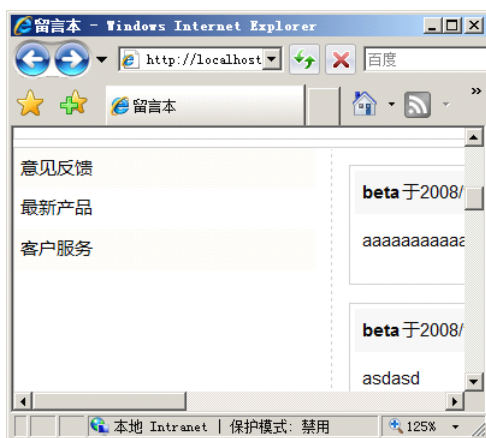


图 28-20 a:link 样式

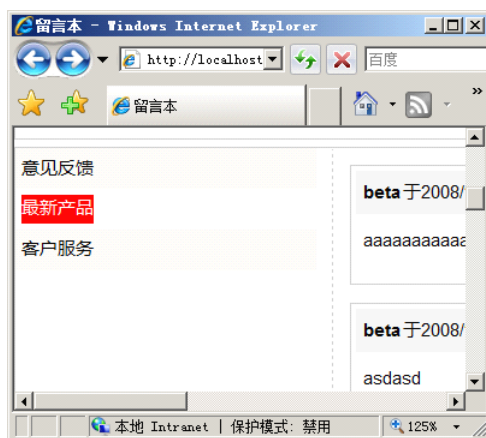


图 28-21 a:hover 样式

丰富侧边栏样式以及编写超链接文本样式能够让页面样式更加简约，编写后的样式能够提高用户体验的友好度并加强应用程序的交互性。

28.6 用户功能实现

用户功能是 Web 应用中另一个非常重要的功能，该功能能够让用户方便的查看自己注册的信息并通过修改模块进行用户信息的修改以便呈现不同的用户信息。用户功能的实现主要是为了能够更好的管理网站的注册用户，对于注册并不使用的用户，管理员还能够进行用户的修改、删除等操作。

28.6.1 用户信息界面

在用户登录后，用户能够通过不同的导航栏进行相应的操作。在 Web 应用中，用户可以索引自己的用户信息，另外用户还能够修改自己的信息。为了方便用户对自己信息的查看，这里可以制作用户信息界面。在该界面，用户不仅能够查看自己的用户信息，还能够查看自己留言，以及回复信息的统计。示例代码见光盘源代码第 28 章\28-1\28-1\personal.aspx。

其中，页面代码实现了用户信息的呈现。在该代码中，使用了多个 `Label` 标签控件以便呈现用户信息。在呈现用户信息时，页面的其他信息同样需要呈现，页面布局如图 28-22 所示。



图 28-22 个人信息页面布局

注意：在进行页面布局时，其样式可以直接使用个人留言索引页面的布局和样式控制。值得注意的是，对于多个不同功能却需要使用相同的布局样式的页面而言，可以通过 CSS 进行整体的样式控制。

为了能够方便的让用户查看统计信息，开发人员能够通过 SQL 语句进行数据库中信息的查询进行数据的呈现和统计。在用户信息页面，用户能够查看信息、查看统计和修改信息，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e) //页面加载执行
{
    if (!String.IsNullOrEmpty(Request.QueryString["uid"])) //获取传递的 uid
    {
        string uid = Request.QueryString["uid"]; //参数值获取
        string strsql = "select * from register where id='"+uid+"'"; //编写查询语句
        SqlDataReader da=
        SQLHelper.SQLHelper.ExecReader(strsql); //初始化适配器
        while (da.Read()) //查询数据
        {
            Label1.Text = da["username"].ToString(); //显示用户名
            if (da["sex"].ToString() == "1") //判断用户性别
                Label2.Text = "男"; //显示性别
            else if (da["sex"].ToString() == "2") //判断用户性别
                Label2.Text = "女"; //显示性别
            else
                Label2.Text = "保密"; //显示默认
            Label3.Text = da["im"].ToString(); //输出 im 值
            Label4.Text = da["information"].ToString(); //输出用户信息
            Label5.Text = da["others"].ToString(); //输出备注

            string strsql1 = "select count(*) as mycount from gbook where name='"+Label1.Text+"'";
            SqlDataReader da1 =
            SQLHelper.SQLHelper.ExecReader(strsql1); //查询统计信息
            while (da1.Read())
            {
                Label6.Text = da1["mycount"].ToString(); //输出统计信息
            }
        }
    }
}
```

```

    }
}
}

```

上述代码在页面被加载时执行。当页面加载时，该代码段会获取传递的用户 id 的信息进行数据查询。在查询完成后，该代码将查询的数据值填充到控件中以便能够呈现在客户端浏览器中。在查询了用户数据后，还需要通过用户的用户名进行数据库中留言数据的统计，以便用户能够快速查看自己的留言统计信息。

28.6.2 用户修改实现

当用户进行信息的查看后，如果需要修改自己的信息，则可以通过用户修改页面进行修改功能的实现。为了方便用户进行信息的修改，用户可以在个人信息页面进行用户信息的修改。另外，为了保证用户信息的安全性，在用户查看信息时，如果用户查看的信息是自己的信息，则允许用户对信息的修改，否则该用户不是可被允许的用户，则该用户不能够进行信息的修改。判断用户身份代码如下所示。

```

if (Session["username"] != null && Session["userid"] != null)           //判断是否登录
{
    if (Session["username"].ToString() == Label1.Text)                   //如果登录并且是当前用户
    {
        Label7.Text = "<a href=\"modi.aspx?uid=\" + Request.QueryString[\"uid\"] + \"\">
        修改资料</a>";                                                    //修改资料
    }
    else                                                                    //如果登录但不是当前用户
    {
        Label7.Text = "<a href=\"personal.aspx?uid=\" + Session[\"userid\"].ToString() + \"\">
        我的信息</a>";                                                    //可以选择跳转到“我的信息”
    }
}

```

上述代码在用户信息界面实现并在页面加载时执行。用户通常不允许直接跳转到用户修改页面，用户能够在自己的用户信息页面选择是否修改自己的信息。当一个用户登录后，该用户就具备了查看其他用户信息的权限，当用户查看其他用户信息时，该用户不能够修改其他用户的信息，如果当前用户访问的是自己的信息页面时，这个而用户就被认为是“可操作的”用户，即当前用户能够修改用户信息。

用户能够单击【修改资料】超链接跳转到修改资料页面中，在修改资料页面加载时，还需要进一步的进行用户权限的判断。如果当前用户是“可操作的”用户（包括管理员），则当前用户能够继续进行操作，示例代码如下所示。

```

if (Session["username"] != null && Session["userid"] != null)           //判断是否登录
{
    if (Session["username"].ToString() != Label1.Text)                   //判断是否是当前用户
    {
        Response.Redirect("default.aspx");                               //不是用户则跳转
    }
    else
    {
        Response.Redirect("default.aspx");                               //未登录用户跳转
    }
}

```

上述代码在执行加载时同样再次判断用户权限，如果用户权限正确，则允许从页面的呈现并允许用

户进行数据更新操作，示例代码如下所示。

```
string uid = Request.QueryString["uid"];           //获取传递的参数
string strsql = "select * from register where id='" + uid + "'"; //编写查询语句
SqlDataReader da =
SQLHelper.SQLHelper.ExecReader(strsql);           //创建和初始化适配器
while (da.Read())                                  //读取数据
{
    Label1.Text = da["username"].ToString();       //进行数据填充
    Label2.Text = da["password"].ToString();       //填充密码
    DropDownList1.Text = da["sex"].ToString();     //填充性别
    TextBox4.Text = da["im"].ToString();           //填充 im
    TextBox5.Text = da["information"].ToString(); //填充用户信息
    TextBox6.Text = da["others"].ToString();       //填充用户备注
}
```

当用户访问自己的页面时，用户可以进行数据的查看和更新，如图 28-23 所示，当用户单击【提交】按钮控件时，用户能够执行数据操作。

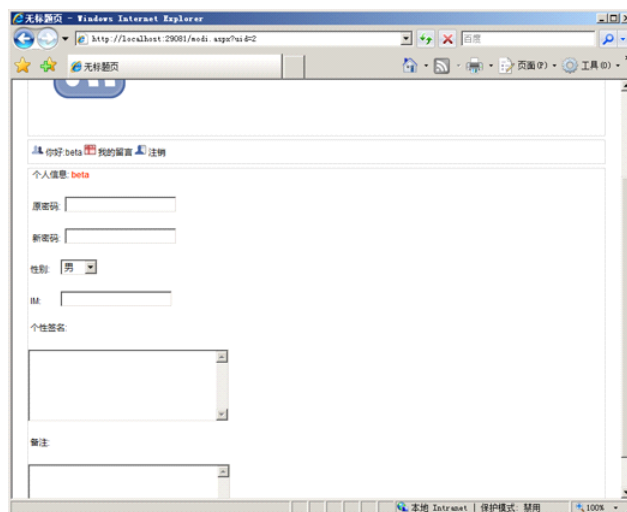


图 28-23 查看基本信息

正如图 28-23 所示，当用户进行注册时，往往不会填写注册的信息，如果用户希望在今后的操作中进行信息的更正，用户可以填写或清空相应的信息。值得注意的是，如果用户需要修改自己的信息，首先需要填写原密码进行身份的验证，如果填写的原密码信息等于数据库中的密码，则说明该用户能够进行修改操作，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text != Label2.Text)                //判断密码是否正确
    {
        Label3.Text = "原密码不正确,无法修改!";    //不正确则停止修改
    }
    else
    {
        if (!String.IsNullOrEmpty(TextBox2.Text)) //判断密码输入框
        {
            string strsql = "update register set password='" + TextBox2.Text + "',sex='" +
```

```

        DropDownList1.Text + ",im=" + TextBox4.Text + ",information=" + TextBox5.Text +
        ",others=" + TextBox6.Text + " where id=" + Request.QueryString["uid"] + """;
        SQLHelper.SQLHelper.ExecNonQuery(strsql); //如果输入密码则更新
        Response.Redirect("personal.aspx?uid="+Request.QueryString["uid"]); //页面跳转
    }
    else
    {
        string strsql = "update register set sex=" + DropDownList1.Text + ",im=" + TextBox4.Text
        + ",information=" + TextBox5.Text + ",others=" + TextBox6.Text + " where id=" +
        Request.QueryString["uid"] + """; //未输入密码则不更新
        SQLHelper.SQLHelper.ExecNonQuery(strsql); //执行 SQL 语句
        Response.Redirect("personal.aspx?uid=" + Request.QueryString["uid"]);
    }
}
}

```

在用户信息更改页面中，如果用户填写了【新密码】文本框，则在执行更新时会更改用户的密码。如果用户在更新过程中没有填写【新密码】文本框，则系统不会更改用户的密码，当用户再次登录时，依旧可以使用原密码进行信息查询和留言。

注意：在执行数据库更新操作时，页面需要使用 IsPostBack 属性进行判断，否则可能不会更新数据。

28.6.3 用户信息删除实现

用户能够查看自己的信息并修改自己的信息，但是用户无法对自己的信息进行删除。在留言本系统中，只有管理员能够对用户的信息进行删除。值得注意的是，为了保证用户信息的完整度，在删除用户信息时，还需要进行用户相关数据的删除。

在用户信息查看页面，管理员能够进行用户信息的删除。在用户信息查看页面被加载时，需要进行管理员权限的判断，如果一个用户的权限是管理员，那么在页面加载时就应该加载【删除用户】连接以便管理员进行删除操作，示例代码如下所示。

```

if (Session["admin"] != null)
{
    Label8.Text = "<a href='\"delete.aspx?uid="+Request.QueryString["uid"]+"\">删除用户</a>";
    //呈现连接
}

```

当管理员进行用户信息页面访问时，由于该用户是一个管理员，则该用户能够执行删除操作。删除操作在“delete.aspx”页面实现，示例代码如下所示。

```

namespace _28_1
{
    public partial class delete : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e) //页面加载事件
        {
            string strsql = "delete form register where id="+Request.QueryString["uid"]+""; //删除用户信息
            string strsql1 = "delete from gbook where userid="+Request.QueryString["uid"]+""; //删除留言
            SQLHelper.SQLHelper.ExecNonQuery(strsql1); //执行留言删除
            SQLHelper.SQLHelper.ExecNonQuery(strsql); //执行信息删除
            Response.Redirect("default.aspx"); //页面跳转
        }
    }
}

```

```
}  
}
```

当执行上述代码时，系统将删除用户的信息，以及和用户相关的留言信息以保证数据库系统的完整性。当删除用户信息后，用户的所有信息都会从数据库中删除，当该用户访问 Web 应用时，无法再使用原来的账号进行登录。

注意：在某些系统设计中，当需要进行数据库的删除时，需要判断数据库中数据的约束性和完整性。当进行某些数据的删除时，首先需要考虑数据的完整性，然后再执行删除操作。

28.6.4 用户注销

当用户登录完毕后并执行了相应的操作后，用户可以选择注销操作进行注销以保证用户信息的安全。在主要页面中，不仅是普通用户能够执行注销操作，管理员也同样能够执行注销操作。当执行了注销操作后，用户的所有信息将会被清除，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Session["username"] = null;           //清除用户信息
    Session["userid"] = null;             //清除用户信息
    Session["admin"] = null;              //清除管理员信息
    Response.Redirect("../default.aspx"); //页面跳转
}
```

用户注销的过程十分简单，从上述代码即可看出。当用户执行注销操作时，系统只需要将相应的 Session 对象的值赋值为 null 即可。该操作将删除用户的信息，当系统的某些页面被加载时，会判断用户未登录。

28.7 实例演示

在编码完成后还需要对现有的项目进行测试，在内部进行演示是一个很好的方法，在进行实例演示之前，需要准备一些数据，这些数据能够提供基本的数据操作所必备的元素从而能够为后面的应用程序运行提供基本保障。

28.7.1 准备数据源

由于篇幅的限制，ASP.NET 留言本中没有针对管理员和聊天室分类进行数据添加、数据操作等页面的制作，这些页面的制作在前面的模块中已经有所涉及，所以就不再重复讲解。在进行演示前，需要执行相应的数据操作以保证基本的数据是存在的。在执行操作前，需要添加管理员，管理员的添加可以在 SQL Server Management Studio 视图状态下添加也可以执行 SQL 语句进行添加。执行 SQL 语句添加管理员代码如下所示。

```
INSERT INTO admin (admin,password) VALUES ('admin','admin')
```

上述代码创建了一个名字为 admin，密码为 admin 的管理员，使用该用户名和密码能够进行页面操作。除了需要准备管理员数据以外，还需要准备留言分类数据，其 SQL 语句如下所示。

```
INSERT INTO gbook_class (classname) VALUES ('客户服务')
INSERT INTO gbook_class (classname) VALUES ('最新产品')
INSERT INTO gbook_class (classname) VALUES ('意见反馈')
```

上述代码使用 SQL 语句进行留言本分类数据的插入，当用户浏览不同的留言本时所看到的留言也不相同。

28.7.2 基本实例演示

当用户进行网站的访问时，可以选择相应的留言本分类进行留言本的访问，不同的分类其留言也不相同，如图 28-24 所示。

在用户选择了相应的留言本分类后，单击【Go!】按钮控件就能够访问相应的留言本，跳转到相应的页面，如图 28-25 所示。

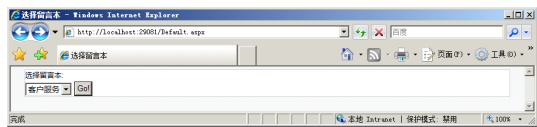


图 28-24 选择留言本



图 28-25 留言本页面

用户能够在左侧的导航栏中选择感兴趣的留言本进行留言查看，也可以查看当前选择的留言。正如图 28-25 所示，用户并没有进行登录操作，所以无法看到留言本中的留言控件进行留言操作，未登录的用户可以看见留言本中的留言并进行翻页。如果用户希望在留言本中进行留言，就需要进行登录操作，如果在登录操作前没有注册，则还需要进行注册，这里注册一个用户名和密码都为 beta 的用户，注册页面如图 28-26 所示。

单击【立即注册】按钮就能够进行注册操作，用户注册完成后依旧不能够进行留言操作，只有进行了登录操作的用户才能够进行留言，打开 login.aspx 页面进行登录操作，如图 28-27 所示。

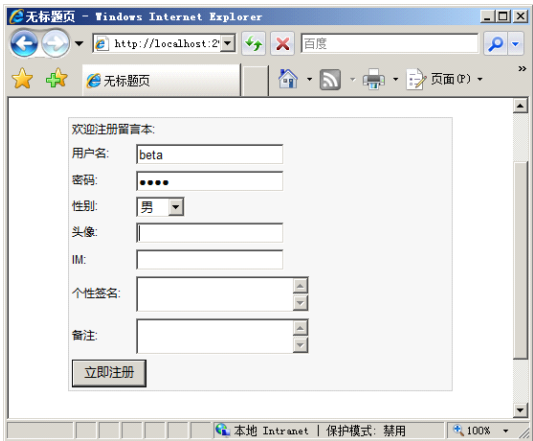


图 28-26 注册页面

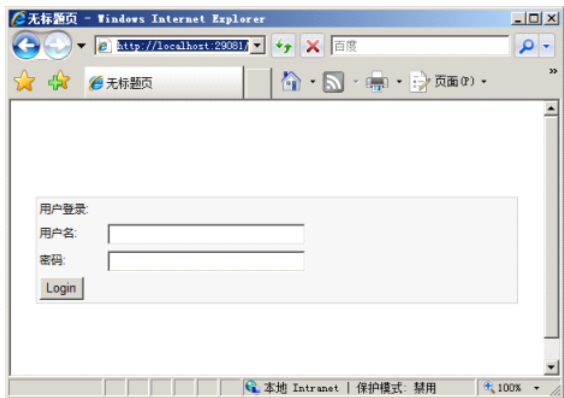


图 28-27 登录操作

登录完成后，系统会给登录的用户一个 Session 对象，以便进行留言。登录后再次选择留言本就能够进行留言了，如图 28-28 所示。

从图 28-28 中可以看出，登录后的用户可以进行留言，留言后会根据留言的 ID 进行倒序，以确保最新的留言数据在最前端。如果留言本中的留言需要管理员进行回复，管理员可以单击【回复】按钮进行回复操作。在进行回复操作之前，管理员需要进行登录，登录页面在 admin/login.aspx 中，可以使用 admin/admin 进行登录，登录完成后可以选择相应的聊天室进行回复，如图 28-29 所示。

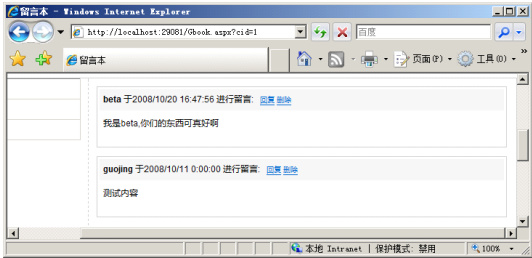


图 28-28 执行留言操作

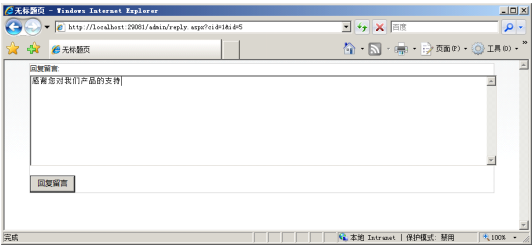


图 28-29 管理员回复

管理员发表回复后系统会跳转到相应的留言分类的页面，如图 28-30 所示。

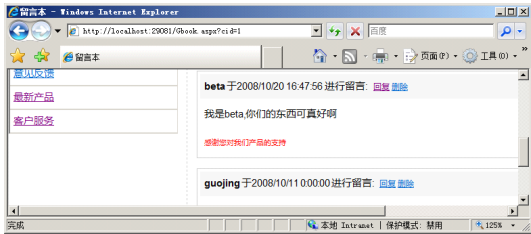


图 28-30 管理员回复

管理员同样可以在留言页面进行留言的删除操作。当执行删除操作后，系统会跳转到 delete.aspx 页面并通过参数进行删除操作的执行，当执行完毕后会跳转回留言页面，而留言页面的数据会被同步更新。在留言本页面中，可以选择【导航】选项进行不同留言本中留言的查看和管理，用户也能够通过导航在不同的分类的留言本内进行留言。

28.7.3 用户功能演示

上一小节演示了基本的用户注册、登录以及留言等操作，在这一节中将演示用户功能。当用户登录后，用户就可以选择相应的留言模块进行留言操作，如图 28-31 所示。



图 28-31 进行登录、注册选择

当用户选择登录并登录后，用户可以在导航栏中进行相应的用户操作，如图 28-32 所示。



图 28-32 登录后进行导航选择

当用户登录完成后，就能够在导航中进行用户信息的选择和留言的索引。单击【用户名】超链接能够进入用户信息页面。在用户信息页面，用户能够查看自己的用户信息，如图 28-33 所示。不仅如此，用户还能够通过相应的 id 信息访问其他用户的用户信息，如图 28-34 所示。

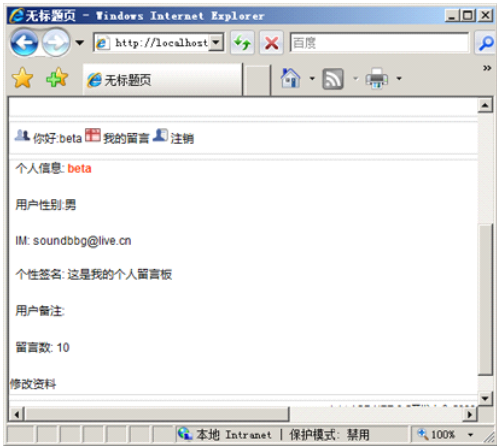


图 28-33 查看当前用户信息

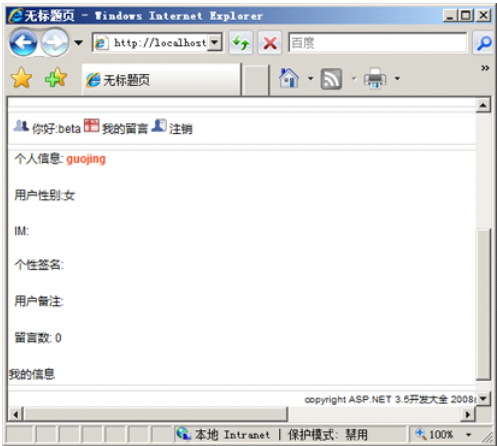


图 28-34 查看其他用户信息

正如图 28-33 和图 28-24 所示，当用户访问自己的信息时，其页面呈现的就是【修改资料】字样，当用户访问其他的用户信息时，则呈现【我的信息】字样。单击【修改资料】连接即可修改用户资料，如图 28-35 所示。

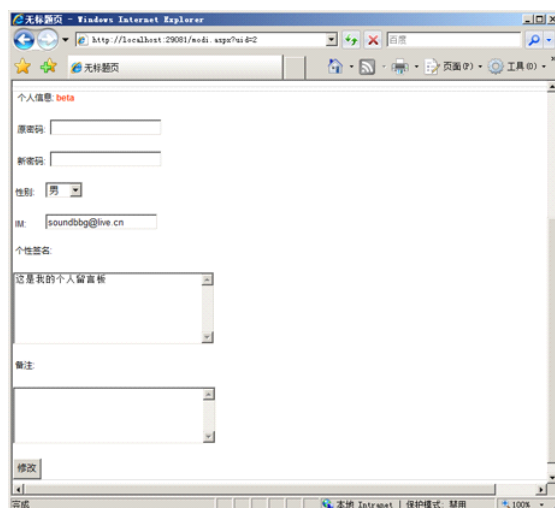


图 28-35 修改用户信息

用户可以在文本框控件中编写相应的信息进行数据的更新。在更新数据前，用户还需要填写【原密码】以便能够执行用户信息的更新验证。当用户填写了正确的原密码后，系统将允许用户执行更新操作，否则系统将提示用户错误信息并重新进行填写，如图 28-36 所示。

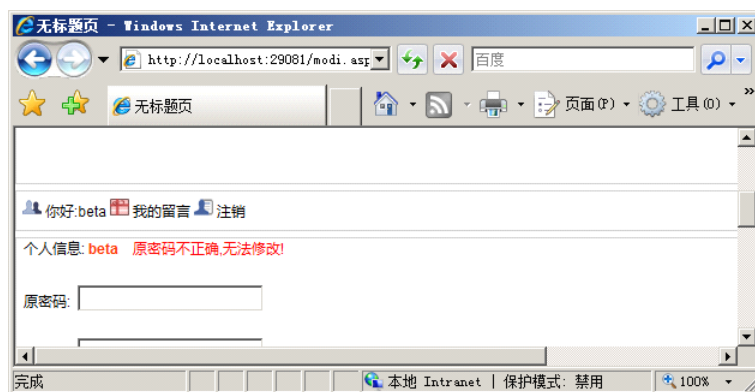


图 28-36 错误的修改密码

当用户修改密码后，用户必须使用新的密码进行登录，原密码将被更新。如果一个用户长时间没有进行更新或者留言等出道作，管理员能够将用户进行删除。管理员或用户可以单击【管理员登录】超链接进行管理员登录操作，登录后，管理员能够打开用户界面进行用户信息的删除，如图 28-37 所示。

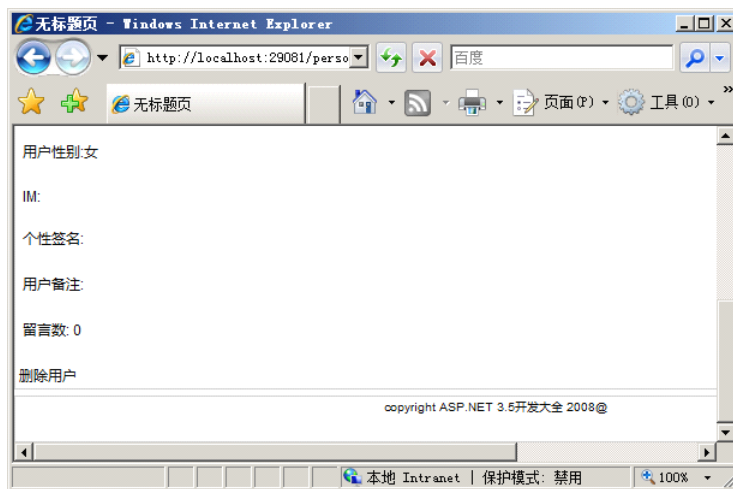


图 28-37 删除用户

在删除用户时，必须以管理员身份进行登录，如果不以管理员身份登录，则无法呈现【删除用户】超链接。当管理员单击【删除用户】超链接时，系统会跳转到删除页面并执行删除操作，在删除过程中，删除功能会删除用户的留言和用户的信息。在删除用户前，留言本的留言如图 28-38 所示。在执行了删除操作后，留言本的留言如图 28-39 所示。

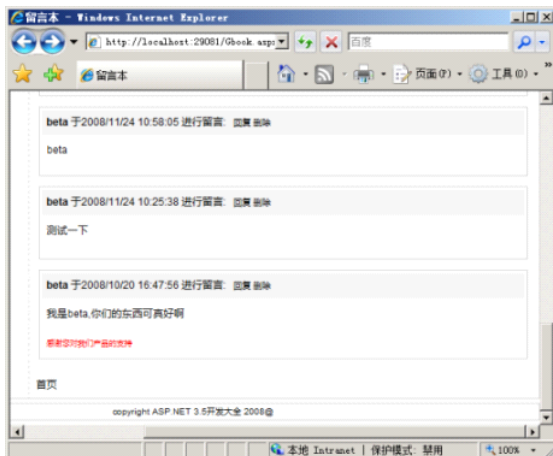


图 28-38 删除用户前



图 28-39 用户删除后

在执行删除后，用户的评论也会随之删除，这样不仅保证了数据的完整性，另外也保证了数据库的约束性。

注意：在执行信息删除操作时，首先执行的是约束条件的删除。例如这里首先必须删除用户的留言信息，当留言信息被删除完毕后，才能够进行用户信息的删除。如果不删除留言信息而进行用户信息的删除，则在数据库设计时的约束条件会被破坏，系统会抛出异常。

28.8 小结

本章通过留言本实例的讲解讲解了软件开发过程中的基础步骤，这些步骤包括系统设计、需求分析文档、数据库设计等等。软件项目管理在软件开发过程中是非常重要的，良好的软件项目管理能力能极

大的简化和协调开发人员对产品的开发。

本章还使用了 `SQLHelper` 类进行数据操作，`SQLHelper` 类简化了开发人员对数据的操作，只需要使用一行或几行代码就能够实现 ADO.NET 中的复杂代码实现，`SQLHelper` 类还能够不同的项目中使用，这些项目包括 Web 应用、类库和自定义控件。本章还包括：

- ❑ 功能模块划分：讲解了 ASP.NET 留言本中的功能模块的划分。
- ❑ 使用 `SQLHelper`：讲解了如何使用 `SQLHelper` 类进行数据操作。
- ❑ 配置 `Web.config`：讲解了如何使用 `Web.config` 配置文件配置可扩展元素。
- ❑ 留言板用户控件：讲解了如何开发留言板的用户控件。
- ❑ 留言功能实现：讲解了如何实现留言本需要的一系列操作。

留言本是系统开发中比较基础的系统，但是很多系统模块都是基于留言本的制作流程的，例如论坛、小型社区、博客等都是类似于留言本的。这些模块在很久之前都是从留言本系统中演化而来的，对于掌握的较熟练的读者可以将本章中的留言本系统结合前面章节中的新闻模块进行博客系统的开发。