# 第 29 章　制作一个 ASP.NET 校友录系统

在现在的网络应用中，用户是网络应用的中心，如现今最风靡的校内网都是把用户放到了网络应用的第一位。而校内网的成功和风靡在很大程度上是因为它是一个真实的社交网络，校友录系统也是利用了真实的社交网络进行设计和开发的。

## 29.1　系统设计

在编写校友录系统前，首先需要确定校友录系统所需要的一些功能模块和适用场景，例如校友录是以何种形式呈现给用户的，如何判断这个用户是不是一个真实的用户等等，这些功能都是需要在开发初级进行设计和规划的。

### 29.1.1　需求分析

在上一章 ASP.NET 留言本中，通过一个简单的 ASP.NET 留言本项目对需求分析进行介绍，需求分析是在系统设计中一个最为重要的组成部分，良好的需求分析设计能够极大的方便在后续过程中的软件开发以及软件维护。

**1. 目录**

需求分析通常情况下是一个单独的需求分析文档，为了模拟在软件开发过程中的顺序，以及软件开发的步骤，这里模拟基本的需求分析文档并为相关的部分进行描述。

- 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。

对于需求分析文档而言，其格式很像论文或软件开发说明书，所以在需求分析文档前通常会有一个目录方便客户和开发人员进行文档的阅读。上述目录描述了引言、编写文档的目录、项目背景等，当客户进行文档的翻阅时可以很方便的进行文档的查询。

**2. 引言**

对于 ASP.NET 校友录系统而言，其作用是为了增加同学之间的友情，在需求分析文档的引言部分可以简单的编写为何要开发该系统以及相应的背景。引言编写如下所示：

随着互联网的发展，越来越多的交流社区应用被广泛的接受，这些社区的存在都是为了能够加强人

与人之间的交流。在针对现有的系统进行调查，拟开发一套校友录系统进行校友联络，这样不仅方便校友之间的联络，也能够加强老校友和新校友的感情。

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，对校友录系统以及其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清楚的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**

由于互联网的迅猛发展，越来越多的用户希望在互联网上能够即时的，快速的与家人或朋友进行联络，相对于传统的 C/S（客户端/服务器）模式的软件开发而言，其成本较高、难以维护，虽然能够即时的与家人和朋友发送消息，但是无法与家人和朋友分享生活和照片等。

而由于互联网的发展，越来越多的用户已经能够适应基于浏览器的应用程序，即 Web 应用，也有越来越多的用户尝试在 Web 服务上进行自己的应用，包括 QQ 空间、博客、个人日志等，都是基于浏览器的应用程序。

为了解决 C/S 模式的应用程序中日志、照片、音乐等难以交互的情况，现开发 ASP.NET 校友录系统用于进行校友之间的交流和通信，方便校友与校友之间进行通信。校友与校友之间不仅能够分享日志，还能够进行身边信息的分享，这样就加强了人与人之间的交互。

**4．任务描述**

任务描述用于描述客户的任务，以及基本的讲述如何完成任务的描述，ASP.NET 校友录系统的任务描述可以编写为如下所示：

为了解决传统的 C/S 应用程序中程序的信息交互不够的问题，并加强用户与用户之间的信息交互，现开发基于.NET 平台的校友录应用程序，用户能够使用校友录进行信息的通信和分享，不仅能够加强校友与校友之间的感情，也能够增强现有的社交。

**5．开发目标**

ASP.NET 校友录系统的开发目标是为了加强现有的用户和用户之间的信息交互，解决传统的用户和用户沟通不便和沟通内容不够丰富的问题，进行用户和用户之间的数据整合和交互。

开发 ASP.NET 校友录系统可以为现有学校所使用，也可以被班级或个人进行使用，适用性广泛，不仅能够在大型应用中使用，同样也能够适用于小型应用。

**6．应用目标**

ASP.NET 校友录是为了能够让校友之间进行真实的交互，用于加强校友与校友之间的感情，同时也能够收集校友的信息。

## 29.1.2　系统功能设计

ASP.NET 校友录是学校内的一个交流平台，用于校友与校友之间的信息交互，校友能够在校友录系统进行注册，注册完毕后管理员审核相应的用户并进行相应的用户操作，当用户的审核通过后，用户就能够在校友录中进行新鲜事的分享。在 ASP.NET 校友录系统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．用户注册功能**

当用户访问 Web 页面时需要进行注册，如果用户不进行注册就不能够发表和回复留言，也不能够分享相应的信息。管理员可以配置是否需要进行登录才能够查看校友录的内容，如果管理员设置需要登录查看，则用户不登录就不能够查看相应的内容。

**2．用户登录功能**

用户注册之后就需要实现用户的登录，登录的用户可以进行信息的发表、回复以及相应内容的分享。登录的用户的操作也会被记录在日志中，用户可以通过自己的 ID 进行校友录中的功能或文章的索引。

**3．用户日志功能**

用户注册和登录后就能够在校友录中进行日志分享，发表关于自己觉得的最新事件，其他人能够查阅该日志并进行相应的日志操作。

**4．用户留言功能**

用户可以查看校友录中日志并进行相应的评论，不仅如此，用户还能够在回复中发布表情，进行文字处理等操作让留言功能更加丰富，用户还能够在校友录系统中对校友录的日志进行评分。

**5．管理员审核功能**

当用户注册后，需要对用户进行身份的审核，管理员可以审核已知的用户的身份，如果用户不是校友录系统的指定用户，则管理员可以不允许用户进行身份验证和登录，以确保校友录系统中的用户的身份都是真实的。

**6．文章管理功能**

管理员需要对校友发布的相应的信息进行管理，如果校友发布了反动、黄色、淫秽等文章，管理员有权进行修改、屏蔽和删除等操作。

**7．留言管理功能**

管理员需要对校友发布的相应的留言进行管理，如果校友发布了反动、黄色、淫秽或广告的留言，管理员可以进行相应的留言的删除操作。

**8．用户管理功能**

当用户进行了非法操作或者用户注册后发布了太多的反动、黄色、淫秽等内容，管理员可以将用户进行删除，在删除的同时系统数据库中的数据也会被删除。

**9．板报/公告等功能**

管理员在校友录系统中还可以进行板报、公告等发布和管理，让页面看上去更像学生时代课堂的样子，这样提高了用户友好度也能够及时的将相应的信息反馈给校友，以便校友能够获取该校友录活动等最新消息。

### 29.1.3　模块功能划分

ASP.NET 校友录系统中的模块非常的多，这些模块包含最基本的注册、登录等模块，还包括文章管理、用户管理、用户管理等模块，这些模块都在不同程度上进行系统的协调。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 校友录总体模块划分如图 29-1 所示。

图 29-1 描述了 ASP.NET 校友录系统的总体的模块划分，用户在校友录系统中需要进行注册登录等操作。对于用户而言，用户在 ASP.NET 校友录中必须要进行注册和登录操作，如果用户不进行登录操

作就无法进行 ASP.NET 校友录中校友的信息的查看，ASP.NET 校友录中用户的模块流程图如图 29-2 所示。

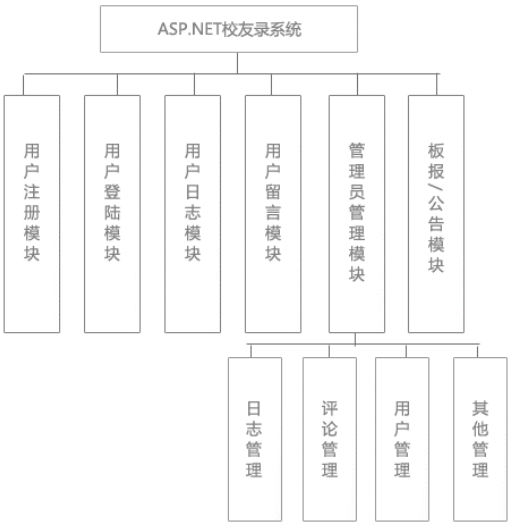


图 29-1　ASP.NET 校友录系统模块划分

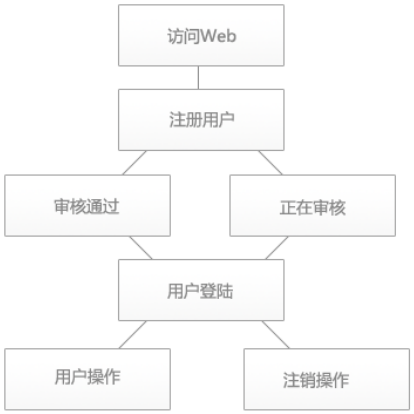


图 29-2　用户登录模块流程图

用户访问 Web 应用并能够在 Web 应用中进行注册，在用户注册后，并不能够立即进行相应的操作，如果用户没有被管理员审核，那么用户只能对校友录中的数据和信息进行查看，并不能进行修改等操作，如果管理员对用户进行了身份审核并通过相应的用户，则说明用户是一个可以被认为是真实的用户，那么用户就能够执行相应操作。

对于管理员而言，管理员不仅能够作为用户的一部分进行用户的活动，包括编写日志等，还应该具备管理功能，这些管理功能包括用户的审核、帖子的审核和用户的管理等等，管理员模块流程图如图 29-3 所示。

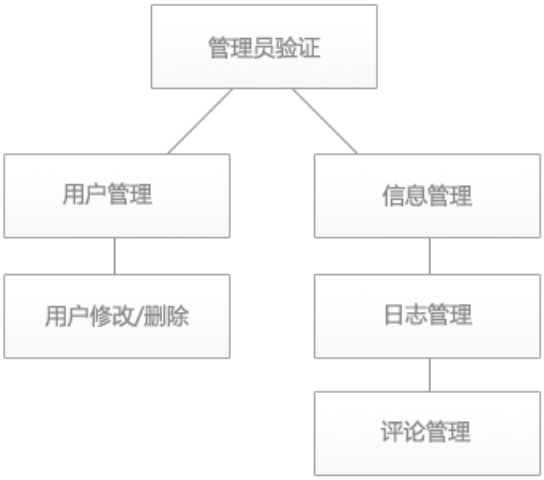


图 29-3　管理员模块流程图

正如图 29-3 所示，管理员在进行操作时同样需要对管理员进行身份验证，由于管理员也是用户的一部分，所以在进入后台管理时，需要判断用户是否有该权限进行管理，如果没有就不允许用户进行操作，如果有管理权限，管理员就能够在后台进行相应的管理操作。

对管理员进行身份验证后，管理员主要进行两大块管理，一个是用户管理，另一个是信息管理。对于用户管理而言，管理员主要是进行用户的删除、积分等操作，而对于信息管理而言，主要是用于不良的日志、评论进行修改和删除管理。

注意：由于管理员是用户的一部分，而一个校友录可以有多个管理员，这些管理员可以是用户，所以在数据库设计中需要额外的字段进行描述。

# 29.2 数据库设计

ASP.NET 校友录比 ASP.NET 留言本更加的复杂，在数据库设计上也更加复杂，不同的表之间还包含着连接。在这些数据表中，单个表或多个表都用来描述校友录的相应功能，在数据库设计中，还需要考虑到数据的约束和完整性约束以便数据库的维护。

## 29.2.1 数据库分析和设计

在前面的系统设计中已经非常仔细对功能和模块进行划分并对相应的用户（校友、管理员）进行了模块流程分析，在进行了模块划分和流程分析后就能够对数据库进行设计。从模块中可以看出 ASP.NET 校友录包含了更多的功能，这些功能都能够让校友用户在网站上分享自己的照片、音乐、视频等，所以在数据库的设计上，其表的数量和表与表之间的关系也比原有的模块或系统更加复杂。针对现有的模块以及模块流程图可以归纳数据库中相应的表，数据库设计图如图 29-4 所示。

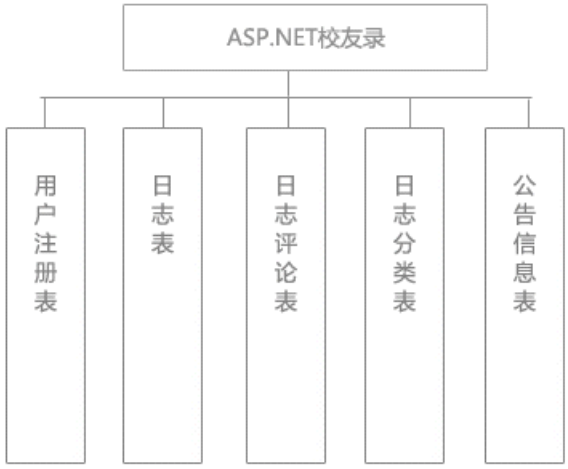


图 29-4 数据库设计图

其中初步的为数据库中的表进行设计，这里包括四个表，分别作用如下：

- 用户注册表：用于存放用户的注册信息，以便登录时使用。
- 日志表：用户可以发布相应的日志，这些日志都存放在日志表中。
- 日志评论表：用户可以对相应的日志进行评论。
- 日志分类表：用户可以选择自己喜欢的分类进行日志发布，但日志分类由管理员管理。
- 公告信息表：管理员可以在校友录中发布最新的信息。

其中用户在发布日志时可以选择相应的分类，例如选择“最近心情”或“好歌欣赏”等，用户还能

够进行相应的分类日志的索引。在 ASP.NET 校友录系统中最为重要的就是日志表和与之相关的表，用户在校友录系统中主要通过日志进行信息交换和分享。其中日志表的字段可以归纳如下。

- 日志 ID：日志的 ID，为自动增长的主键。
- 日志标题：日志的标题，用于显示日志标题的信息。
- 日志作者：日志的作者，用于显示是谁发布了日志。
- 日志发布时间：日志发布时间，用于显示日志发布的日期。
- 日志内容：日志内容，用于呈现日志的内容，包括音乐、图片等信息。
- 日志打分：日志打分，对于其他用户而言可以为该日志进行评分。
- 日志所属分类：日志所属分类，用于显示日志所属于的分类。
- 日志阅读次数：用于表示阅读被访问的次数。
- 日志所属用户 ID：日志所属用户 ID 用于标识该日志所属的用户信息。

日志表能够描述日志的基本信息，而日志分类表和日志所属用户表用户描述整个日志的其他信息，这些信息是日志的分类、日志发布作者的个性签名等等。日志分类表可以规划如下。

- 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“阳光男孩”等。

一个日志可以有一个分类进行描述，当对日志的分类进行描述后，用户可以通过索引相应的分类的日志，例如有某个用户对“阳光男孩”这个分类特别感兴趣，那么用户就能够索引这个分类的所有文章，而暂时关闭对其他文章的浏览。注册模块在前面的章节中都有设计，这里同样需要注册模块，注册模块的字段可以描述如下所。

- 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。
- 用户权限：用户区分是管理员还是普通用户。

与前面的用户注册不同的是，这里多了一个用户权限字段，由于管理员也能够进行普通的用户的操作，所以需要另一个字段进行用户权限的描述。当用户进行登录后，可以对相应的日志进行评论。同样，当管理员进行管理登录后，管理员可以对日志的评论进行删除，日志评论表字段如下所示。

- 评论 ID：用于标识评论，是自动增长的主键。
- 评论标题：用于表示评论的标题。
- 评论时间：用于表示评论的时间。
- 评论内容：用于表示评论的内容。
- 用户 ID：用于标识评论的用户 ID，可以通过该 ID 进行多表连接查询。
- 日志 ID：用于标识评论的所在的日志，可以通过该 ID 进行多表连接查询。

这些表就能够实现校友录的基本信息，在校友录首页就能够通过查询相应的数据进行校友录中的用户和数据的查看。

## 29.2.2 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**

在创建日志表之前首先需要创建 friends 数据库，创建完成后就能够进行其中的表的创建。在 ASP.NET 校友录系统中最为重要模块的就是日志模块，日志模块的表结构分别如图 29-5 和图 29-6 所示。

| 列名 | 数据类型 | 允许空 |
|---|---|---|
| id | int | |
| title | nvarchar(500) | ☑ |
| author | nvarchar(50) | ☑ |
| time | datetime | ☑ |
| [content] | nvarchar(MAX) | ☑ |
| marks | int | ☑ |
| classid | int | ☑ |
| userid | int | ☑ |
| hits | int | ☑ |
| | | |

图 29-5 日志表结构

| 列名 | 数据类型 | 允许空 |
|---|---|---|
| id | int | |
| classname | nvarchar(50) | ☑ |
| | | |

图 29-6 日志分类表结构

从数据库中可以看出留言表中的字段信息，日志表中的字段意义如下所示：

- id：日志的 ID，为自动增长的主键。
- title：日志的标题，用于显示日志标题的信息。
- author：日志的作者，用于显示是谁发布了日志。
- time：日志发布时间，用于显示日志发布的日期。
- content：日志内容，用于呈现日志的内容，包括音乐、图片等信息。
- marks：日志打分，对于其他用户而言可以为该日志进行评分。
- classid：日志所属分类，用于显示日志所属于的分类。
- hits：用于表示阅读被访问的次数。
- userid：日志所属用户 ID 用于标识该日志所属的用户信息。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [friends]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[diary](
        [id] [int] IDENTITY(1,1) NOT NULL,
        [title] [nvarchar](500) COLLATE Chinese_PRC_CI_AS NULL,
        [author] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
        [time] [datetime] NULL,
        [content] [nvarchar](max) COLLATE Chinese_PRC_CI_AS NULL,
        [marks] [int] NULL,
        [classid] [int] NULL,
        [userid] [int] NULL,
        [hits] [int] NULL,
 CONSTRAINT [PK_diary] 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]
```

同样日志分类表的字段如下所示。

- 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“阳光男孩”等。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [friends]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[diaryclass](
        [id] [int] IDENTITY(1,1) NOT NULL,
        [classname] [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
 CONSTRAINT [PK_diaryclass] 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]
```

在用户发布日志后，用户可以对相应的日志进行评论，评论表和日志表的连接也是非常重要的，在相应的日志下需要筛选出相应的日志的评论进行呈现，用户也可以在相应的日志中添加自己的评论，评论表字段可以归纳如下。

- id：用于标识评论，是自动增长的主键。
- title：用于表示评论的标题。
- time：用于表示评论的时间。
- content：用于表示评论的内容。
- userid：用于标识评论的用户 ID，可以通过该 ID 进行多表连接查询。
- diaryid：用于标识评论的所在的日志，可以通过该 ID 进行多表连接查询。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [friends]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[diarygbook](
        [id] [int] IDENTITY(1,1) NOT NULL,
        [title] [nvarchar](500) COLLATE Chinese_PRC_CI_AS NULL,
        [time] [datetime] NULL,
        [content] [nvarchar](max) COLLATE Chinese_PRC_CI_AS NULL,
        [userid] [int] NULL,
        [diaryid] [int] NULL,
```

```
 CONSTRAINT [PK_diary_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]
) ON [PRIMARY]
```

评论表需要同多个表进行连接，其中 userid 需要与注册表进行连接用于查询用户的信息，而 diaryid 用于同日志表进行连接查询所在的日志。在进行日志呈现时，同样需要连接日志评论表进行相应的评论的筛选。

**2．验证表**

在用户注册中，增加了对管理员身份验证的字段，用户注册表字段如下所示。

- 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。
- userroot：用于验证用户是管理员还是普通用户。

创建数据表的 SQL 查询语句代码如下所示。

```
USE [friends]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Register](
      [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,
      [userroot] [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]
```

上述代码创建了一个可以进行身份判断的用户表，开发人员可以通过 userroot 字段进行管理员身份

的判断，其结构如图 29-7 所示。

| 列名 | 数据类型 | 允许空 |
|---|---|---|
| id | int | ☐ |
| username | nvarchar(50) | ☑ |
| password | nvarchar(50) | ☑ |
| sex | int | ☑ |
| picture | nvarchar(MAX) | ☑ |
| IM | nvarchar(50) | ☑ |
| information | nvarchar(MAX) | ☑ |
| others | nvarchar(MAX) | ☑ |
| ifisuser | int | ☑ |
| userroot | int | ☑ |
| | | ☐ |

图 29-7　注册表结构

**3．公告数据**

公告数据可以不使用数据库进行存储，在这里可以使用 txt 文档进行数据存储，这样不仅可以减轻数据库服务器的压力，也能够增加公告中文本的可扩展性。

注意：对于公告的数据直接存储在 txt 文档中，当首页需要调用公告时，可以直接从 txt 文档中读取数据进行 HTML 呈现。

# 29.3　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。在 ASP.NET 同学录系统中，包括一些数据约束用于保持数据库中数据的完整性，数据表关系图如图 29-8 所示。

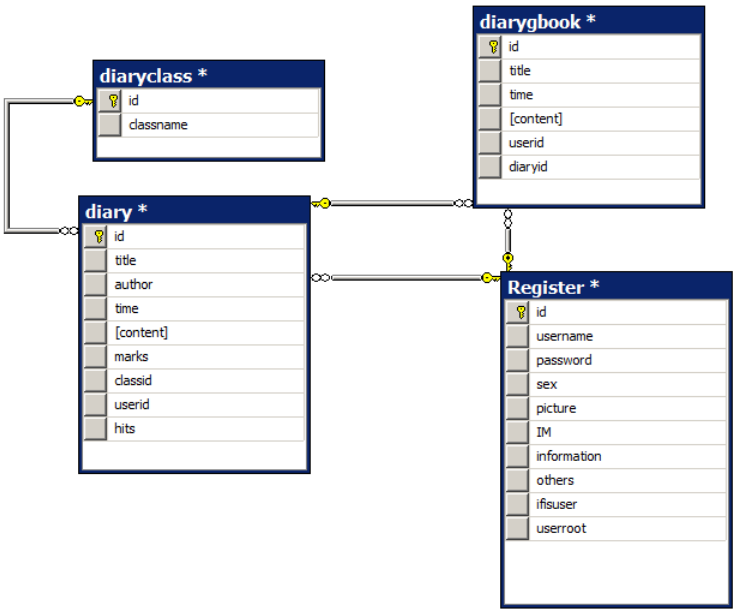


图 29-8　数据库关系图

图 29-8 中说明了数据库表之间的关系，进行表之间数据的约束。当进行数据插入时，就会判断数据库中表的约束情况和完整性，如果用户表没有任何数据而进行日志表中数据的插入是不被允许的。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，就不会是原来的简单的创建语句，而更改后的语句包含了键值的约束关系。当对数据库中的表之间进行约束，则在进行数据操作时就应该按照规范来进行插入、删除等操作。

# 29.4　系统公用模块的创建

在 ASP.NET 校友录系统中，用户能够分享自己的日志就需要使用 HTML 编辑器，HTML 编辑器是系统的公用模块，可以通过 HTML 编辑器进行富文本编写和呈现。同样为了简化数据操作，也可以使用 SQLHelper 类进行数据操作。

## 29.4.1　使用 Fckeditor

Fckeditor 是现在最热门的开源 HTML 编辑器，使用 Fckeditor 能够像 Word 一样进行页面排版和布局，Fckeditor 还能够使用表情、进行拼写检查等。

在留言本系统开发中，并没有使用 Fckeditor 进行文本提交是因为在留言本系统中不需要进行复杂的文字呈现，而对于校友录系统而言，用户可以分享自己的日志并且进行日志布局和排版，这就需要实现复杂的 HTML 代码进行页面呈现。Fckeditor 能够完成这一系列复杂的操作。Fckeditor 在“附-Fckeditor 编辑器”中，可以在项目中使用 Fckeditor 进行文本框制作和二次开发。

在项目中添加 Fckeditor 的引用，首先需要将 Fckeditor 文件夹拷贝到项目中。由于 ASP.NET 应用程序的关系，Fckeditor 并不会在解决方案管理器中呈现，单击【解决方案管理器】上方的【显示所有文件】小图标可以进行所有文件的显示，如图 29-9 所示。

显示所有文件后就能够看到 Fckeditor 编辑器文件夹，右击 Fckeditor 文件夹，选择【添加到项目】选项就能够将文件夹中的所有文件批量添加到项目中，如图 29-10 所示。

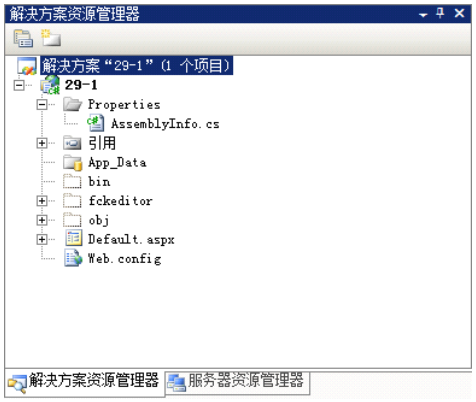


图 29-9　显示所有文件

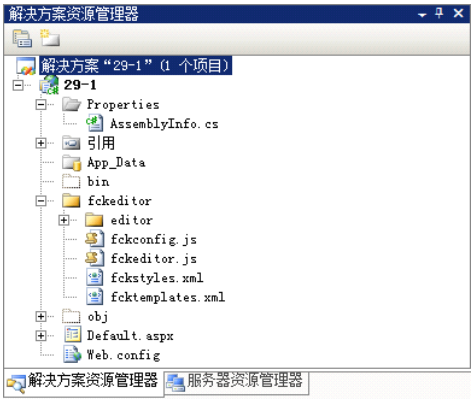


图 29-10　添加 Fckeditor

添加文件后还需要添加相应的 DLL 文件，以便在程序开发中使用 Fckeditor 编辑器进行文本框开发。右击现有项目，在下拉菜单中选择【添加现有项】选项，在标签中选择【浏览】选项卡，找到【附-Fckeditor 编辑器】目录中的 bin 目录并添加到项目中，如图 29-11 所示。

添加引用后就能够在开发中使用 Fckeditor 编辑器进行富文本编辑，开发人员还能够在工具栏中添加 Fckeditor 编辑器。单击【工具箱】的空白区域，单击右键，在下拉菜单中选择【选择项】选项，选择刚才添加的 DLL 文件。选择后单击【确定】按钮即可添加控件。控件添加完毕后就会在工具栏中呈

现相应的控件，如图 29-12 所示。

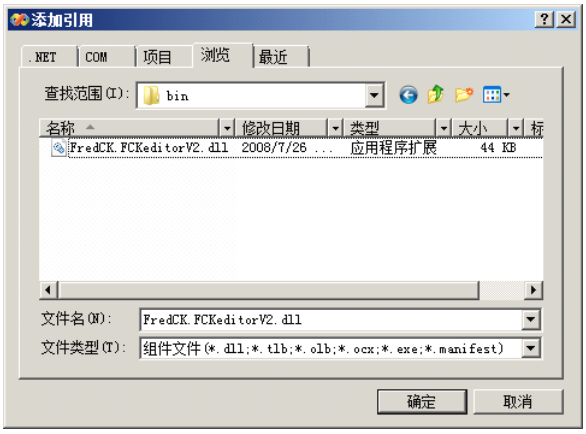


图 29-11　添加引用

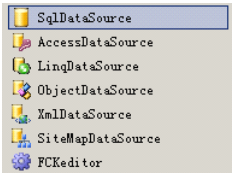


图 29-12　添加后的 Fckeditor 编辑器控件

开发人员能够将 Fckeditor 编辑器控件拖动到其他的区域，以适合自己的开发方式。在控件添加完成后，就可以向页面中添加控件，示例代码如下所示。

```
<body>
    <form id="form1" runat="server">
    <div>
        <FCKeditorV2:FCKeditor ID="FCKeditor1" runat="server">
        </FCKeditorV2:FCKeditor>
    </div>
    </form>
</body>
```

上述代码使用了 Fckeditor 编辑器控件进行富文本操作，运行后如图 29-13 所示。

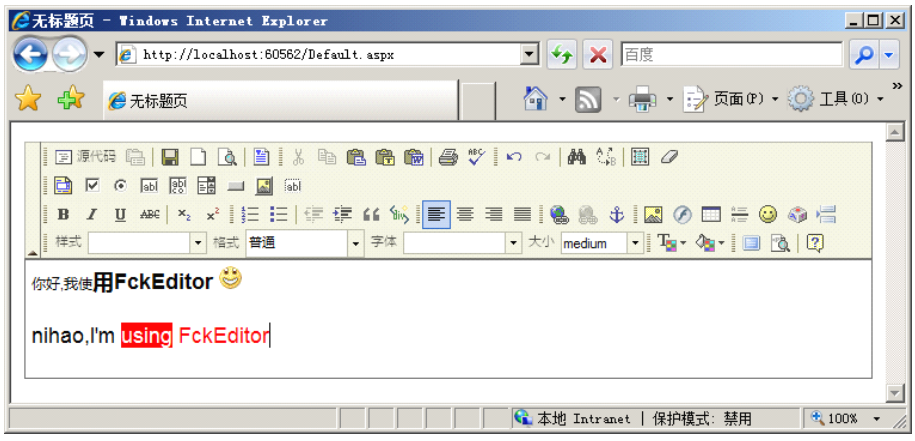


图 29-13　Fckeditor 编辑器

使用 Fckeditor 编辑器可以更快的进行富文本的编辑，如果开发人员从头开发 HTML 编辑器会花费大量的时间，使用 Fckeditor 编辑器能够进行样式的布局、文本格式化等操作而无需从头进行开发。对于 Fckeditor 编辑器而言，Fckeditor 编辑器是免费和开源的，开发人员能够免费的下载 Fckeditor 编辑器并进行二次开发，极大的简化富文本功能的开发。

### 29.4.2 使用 SQLHelper

SQLHepler 是一个数据库操作的封装，使用 SQLHepler 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遍历，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。

在前面的 ASP.NET 留言本中详细的讲解了如何使用 SQLHepler 类进行数据操作，使用 SQLHepler 类能够无需自己创建 ADO.NET 对象进行复杂的数据操作。

### 29.4.3 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，这里同样可以通过配置<appSettings/>配置节配置自定义信息，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
    <add key="look" value="false"/>                        //编辑 look 项
</appSettings>
```

上述代码对配置文件 Web.config 进行了相应的配置，<appSettings/>配置节的配置信息能够在程序中通过 ConfigurationManager.AppSettings 获取，在 SQLHelper 类中就使用 ConfigurationManager.AppSettings 进行相应的自定义配置节的参数值的获取。这些配置节的相应的意义如下所示。

- server：server 项，用于配置数据库服务器的服务器地址。
- database：database 项，用于配置数据库服务器的数据库名称。
- uid：uid 项，用于配置数据库服务器的用户名。
- pwd：pwd 项，用于配置数据库服务器的密码。
- look：look 项，用于配置用户是否需要登录才能进行查看。

在配置了 Web.config 中<appSettings/>配置的信息后，不仅 SQLHelper 类能够进行相应参数的获取，在应用程序中也能够获取 Web.config 中<appSettings/>配置节的参数值。

## 29.5 系统界面和代码实现

在ASP.NET校友录系统中使用了Fckeditor以及SQLHelper简化了HTML 编辑器的开发和数据操作，在系统界面编写和代码实现上也更加容。，对于校友录系统而言，具有比较多的页面，这些页面用于注册、登录、发布日志和管理。

### 29.5.1 用户注册实现

在用户进行校友录系统登录前必须进行注册，对于注册而言，本书的前面的模块章节以及 ASP.NET 留言本项目都有比较详细的介绍，这里就不在做过多的介绍，用户注册只需要将数据插入到数据库即可，注册页面 HTML 核心代码见光盘中源代码\第 29 章\29-1\29-1\register.aspx。

上述代码进行了用户注册页面的基本布局，当用户打开校友录页面时，系统会提示用户必须要进行登录操作，如果用户没有用户惟一则必须先进行注册，注册页面如图 29-14 所示。

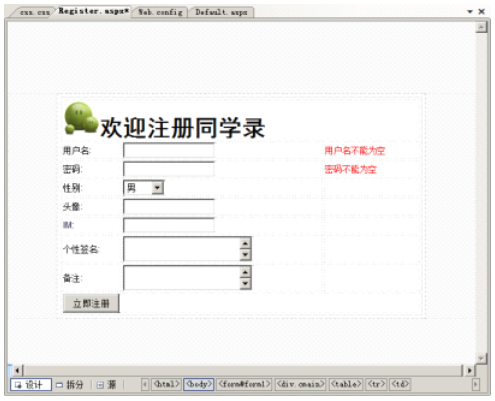


图 29-14　注册页面

当用户进行注册时，需要将数据插入到数据库中，使用 SQLHelper 类能够简化数据操作，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,userroot)
          values ('" + TextBox1.Text + "','" + TextBox2.Text + "','" + DropDownList1.Text + "','" +
          TextBox3.Text + "','" + TextBox4.Text + "','" + TextBox5.Text + "','" + TextBox6.Text + "',0,0)";
        SQLHelper.SQLHelper.ExecNonQuery(strsql);                    //执行 SQL 语句
        Response.Redirect("login.aspx");                             //注册后跳转到登录页面
    }
    catch
    {
        Response.Redirect("default.aspx");                           //出错后跳转到首页
    }
}
```

当用户执行注册后，如果注册成功系统就会跳转到登录页面进行登录操作，如果没有注册成功（抛出异常），则系统会认定用户执行了非法操作，会跳转到首页。在进行注册时，默认情况下 ifisuser 字段为 0，用户注册后并不能够立即通过，需要管理员进行身份验证。

注意：在进行注册时首先需要进行查询，查询是否已经有现有的用户，这里可以参考注册模块，由于前面已经讲解了很多关于注册的操作，这里就不再详细讲解如何实现。

## 29.5.2　用户登录实现

用户登录操作在前面的章节中讲的非常的多，并且在模块篇中还详细的介绍了用户登录模块的开发，这里可以使用简单的登录模块进行登录操作即可而无需实现复杂的登录控制。登录页面 HTML 核心代码见光盘中源代码\第 29 章\29-1\29-1\login.aspx。

用户注册完成后就会跳转到登录页面，登录页面能够给用户配置相应的 Session 对象以存储用户状态，登录界面布局后如图 29-15 所示。

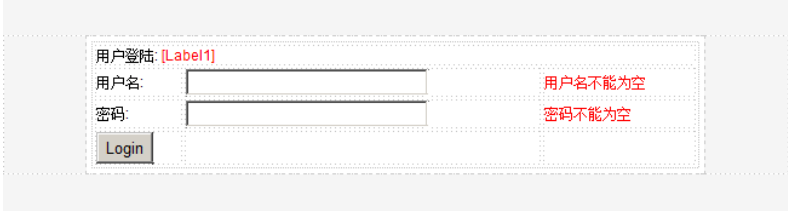


图 29-15　登录界面布局

当用户单击【Login】按钮时进行登录，使用 SQLHelper 类能够快速进行查询，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    string strsql = "select * from register where username='" + TextBox1.Text + "' and password='" +
                    TextBox2.Text + "'";                                    //编写 SQL
    SqlDataReader sdr = SQLHelper.SQLHelper.ExecReader(strsql);             //执行查询
    if (sdr.Read())
    {
        Session["username"] = TextBox1.Text;                                //用户名
        Session["userid"] = sdr["id"].ToString();                           //用户 ID
        Session["admin"] = sdr["userroot"].ToString();                      //管理员判断
        Response.Redirect("friends.aspx");                                  //页面跳转
    }
    else
    {
        Label1.Text = "无法登录,用户名或密码错误";                          //提示错误登录
    }
}
```

当用户进行登录后，系统会为用户赋予三个 Session 对象，这个三个对象的意义为用户名、用户 ID 和管理员判断。在用户表中，其字段 userroot 用于判断是否为管理员，如果 userroot 的值为 0，说明不是管理员，否则说明该用户是一个管理员用户。

### 29.5.3　校友录页面规划

校友录页面是校友录系统中最丰富的页面，大部分用户都会在校友录页面中停留较长的时间，在校友录页面不仅要呈现相应的日志，还需要呈现黑板报、公告、管理员等信息，在开发校友录页面时，首先需要进行页面规划。页面规划如图 29-16 所示。


图 29-16　校友录页面规划

正如图 29-16 所示，校友录页面的基本规划可以分为 5 块，这 5 块内容如下所示。

- 1：显示头部信息，包括 logo 等。
- 2：背景，用户填充背景颜色。
- 3：显示公告，管理员等。
- 4：显示最新发布的日志。
- 5：显示最新加入的同学，最热门的日志等等。

为了提高用户的友好度，可以将管理员命名为“值日生”，而将用户命名为“同学”，更加有校园的感觉。

## 29.5.4　自定义控件实现

在进行了页面规划后，就需要使用自定义控件进行相应的数据成呈现，例如呈现值日生、最新加入的同学等。在页面进行数据呈现的自定义控件并没有多大的难度，但是自定义控件能够方便开发和维护，当进行管理员显示的开发时，只需要修改相应的自定义控件即可。

**1．值日生控件**

值日生控件用于显示校友录系统的管理员，管理员在校友录系统中被称为“值日生”，这样具有更好的友好度，值日生控件实现代码如下所示。

```
namespace DiaryAdmins
{
    [ToolboxData("<{0}:Myadmins runat=server></{0}:Myadmins>")]                    //控件呈现形式
    public class Myadmins : WebControl
```

```
    {
        protected override void RenderContents(HtmlTextWriter output)
        {
            try
            {
                StringBuilder str = new StringBuilder();                                    //使用 String
                string strsql = "select * from register where userroot=1 order by id desc"; //创建 SQL 语句
                SqlDataReader sdr = SQLHelper.SQLHelper.ExecReader(strsql);                 //查询内容
                while (sdr.Read())                                                          //遍历对象
                {
                    str.Append("<span style=\"color:white\"><a href=\"userindex.aspx?uid=" + sdr["id"] + "\">"
                                + sdr["username"] + "</span></a><br/>");                    //输出 HTML
                }
                output.Write(str);                                                          //呈现 HTML
            }
            catch
            {
                output.Write("");                                                           //输出空
            }
        }
    }
}
```

上述代码编写了值日生控件，当使用值日生控件，能够将校友录的管理员呈现在页面中。

**2．加入校友控件**

加入校友控件用于呈现最新加入的校友，校友能够关注最新加入的校友并和他成为好友，加入校友控件代码如下所示。

```
namespace AddFriends
{
    [ToolboxData("<{0}:NewFriends runat=server></{0}:NewFriends>")]                         //控件呈现形式
    public class NewFriends : WebControl
    {
        protected override void RenderContents(HtmlTextWriter output)
        {
            try
            {
                StringBuilder str = new StringBuilder();                                    //使用 String
                string strsql = "select top 10 * from register where userroot=0 order by id desc";    //编写 SQL
                SqlDataReader sdr = SQLHelper.SQLHelper.ExecReader(strsql);                 //执行查询
                while (sdr.Read())                                                          //遍历对象
                {
                    str.Append("<span style=\"color:white\"><a href=\"userindex.aspx?uid=" + sdr["id"] + "\">"
                                + sdr["username"] + "</span></a><br/>");                    //输出 HTML
                }
                output.Write(str);                                                          //呈现 HTML
            }
            catch
            {
                output.Write("");                                                           //输出空串
            }
```

```
        }
    }
}
```

上述代码与值日生控件不同的是，值日生控件通过遍历用户表中的userroot 为 1 的用户，而校友控件是遍历用户表中 userroot 为 0 的用户，userroot 字段用于辨别用户身份，可以通过 userroot 字段进行筛选。

### 29.5.5 校友录页面实现

29.5.3 中的图片可以作为页面布局的规范，页面布局人员能够使用该图片作为页面布局的蓝本进行页面布局，校友录页面布局头部 HTML 代码如下所示。

```
<div class="top">
    <img src="images/logo.png" />
</div>
```

在校友录界面头部布局实现中，需要使用 logo 进行页面呈现，这里可以使用 HTML 控件进行图片呈现。在显式了 logo 之后，就需要呈现 banner 标签。banner 标签的样式在 CSS 文件中进行编写，在实际的校友录系统中，可以直接使用，示例代码如下所示。

```
<div class="banner">
</div>
```

在实现了banner标签后，就需要实现校友录页面中最重要的页面，即标签center内的内容。标签center中包括 main_board、main_site 以及 main_right 标签，示例代码见光盘中源代码\第 29 章\29-1\29-1\friends.aspx。

在编写了校友录页面的主窗体后，就能够编写校友录底部信息，示例代码如下所示。在end 标签中，开发人员能够进行版权的声明和编写。

```
<div class="end">
    校友录系统由 xx 开发完成
</div>
```

在校友录页面中使用了 GridView 控件和自定义控件，GridView 控件主要是用于呈现日志数据，其排序方式是按照最后回复时间进行排序，而自定义控件包含值日生控件和加入校友控件，用户呈现相应的用户数据。

注意：在使用自定义控件时，可能会提示 SQLHelper 类异常，开发人员可以不予理会。开发人员也能够自己编辑异常处理进行错误信息处理。

### 29.5.6 日志发布实现

在日志发布页面，用户能够使用 HTML 编辑器进行富文本编辑，这样就提高了交互性。对于用户而言，也能够使用 HTML 编辑器编写更多丰富的内容，包括音乐分享和文件上传。日志发布页面只需要将数据插入到相应的表即可，日志发布页面示例 HTML 核心代码见光盘中源代码\第 29 章\29-1\29-1\new.aspx。

上述代码使用了同日志显示页面相同的CSS 和样式进行布局，在一些相同的应用中使用相同的样式和布局能够提高用户的熟悉程度，让用户能够尽快适应。当用户填写了相应的日志之后，就能够进行日志的提交，日志提交代码如下所示。

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

```
        try
        {
            string strsql = "insert into diary (title,author,time,content,marks,classid,userid,hits) values ('" +
                            TextBox1.Text + "','" + Session["username"].ToString() + "','" + DateTime.Now +
                            "','" + FCKeditor1.Value + "',0," + DropDownList1.Text + ",'" +
                            Session["userid"].ToString() + "',0)";                          //编写 SQL 语句
            SQLHelper.SQLHelper.ExecNonQuery(strsql);                                       //执行插入
            Response.Redirect("friends.aspx");                                              //页面跳转
        }
        catch
        {
            Label3.Text = "出现错误,请检查日志";                                            //提示错误信息
        }
    }
```

在页面载入时，首先需要进行用户的身份的判断才能够进行相应的操作，如果用户没有登录则不允许进行日志操作，在载入时进行判断需要使用 Page_Load 方法，示例代码如下所示。

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (Session["username"] == null || Session["userid"] == null)                       //如果未登录
        {
            Response.Redirect("login.aspx");                                                //跳转到登录页
        }
    }
```

如果用户没有登录或者登录超时，则会跳转到登录页面重新进行登录操作。

### 29.5.7 日志修改实现

日志修改页面基本同日志添加页面相同，这里就不再重复 HTML 代码，日志修改页面中的控件基本同日志添加页相同，而在日志修改页面中需要使用控件进行传递的参数的存放，当需要进行修改等操作时可以使用传递的控件进行日志修改。

在页面加载时，需要通过传递的参数进行日志的查询和控件中数据的填充，而当用户进行修改时，需要判断用户是否是作者，否则不运行修改功能，日志页面加载时实现代码如下所示。

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            if (Session["username"] == null || Session["userid"] == null)                   //判断状态
            {
                Response.Redirect("login.aspx");                                            //页面跳转
            }
            else
            {
                string strsql = "select * from diary where id='" + Request.QueryString["id"] + "'";
                SqlDataReader sdr = SQLHelper.SQLHelper.ExecReader(strsql);                 //编写 SQL
                if (sdr.Read())                                                             //存在该文章
                {
                    if (sdr["userid"].ToString() == Session["userid"].ToString()
                    || Session["admin"].ToString() == "1")                                  //判断权限
```

```
                    {
                        TextBox1.Text = sdr["title"].ToString();                    //控件值初始化
                        Label1.Text = sdr["author"].ToString();                     //控件值初始化
                        Label2.Text = sdr["time"].ToString();                       //控件值初始化
                        FCKeditor1.Value = sdr["content"].ToString();               //控件值初始化
                        DropDownList1.Text = sdr["classid"].ToString();             //控件值初始化
                    }
                    else
                    {
                        Response.Redirect("error/cmodi.aspx?id=" + sdr["id"]);      //跳转错误
                    }
                }
                else
                {
                    Response.Redirect("login.aspx");                                //登录跳转
                }
            }
        }
    }
```

当页面加载时会判断用户是否登录，如果用户没有登录则会跳转到登录页面，否则进行数据库查询判断该文章是否为当前用户所能够操作的文章，如果不能够操作文章，则跳转到错误页面。当用户进行修改时，执行 UPDATE 语句对数据库中的数据进行更改，示例代码如下所示。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        try
        {
            string strsql = "update diary set title='" + TextBox1.Text + "',content='" + FCKeditor1.Value + "'
w
                                                here id='" + Request.QueryString["id"] + "'";    //更新语句
            SQLHelper.SQLHelper.ExecNonQuery(strsql);                                            //执行更新
            Response.Redirect("news?id=" + sdr["id"]);                                           //页面跳转
        }
        catch
        {
            Label3.Text = "出现错误,请检查日志";                                                  //抛出异常
        }
    }
```

当用户进行日志更新时，只需要进行 UPDATE 语句的执行就能够执行更新，更新完成后可以跳转到相应的页面。

注意:管理员也能够进行日志的修改,通过判断 userroot 字段的值进行管理员权限判断,如果 userroot 的值为 1 则说明该用户是管理员，可以无条件的对日志进行更改。

## 29.5.8 管理员日志删除

对于删除页面而言，同前面的章节一样无需实现 HTML 页面的呈现，只需要进行相应的逻辑实现即可，示例代码如下所示。

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

```
        if (Session["admin"] == null)                                          //判断有没有登录
        {
            Response.Redirect("friends.aspx");                                 //跳转到校友录
        }
        else
        {
            if (Session["admin"].ToString() == "1")                            //判断是不是管理员
            {
                string strsql = "delete from diary where id='" + Request.QueryString["id"] + "'";
                SQLHelper.SQLHelper.ExecNonQuery(strsql);                      //执行删除操作
                Response.Redirect("friends.aspx");                             //页面跳转
            }
            else
            {
                Response.Redirect("errors/cdelete.aspx?id=" + Request.QueryString["id"]);
            }
        }
    }
```

当用户进行删除操作时，页面需要对用户进行身份验证。如果用户是管理员，则允许执行操作，否则会跳转到错误页面，并提示不运行进行相应的操作。

### 29.5.9 日志显示页面

在校友发布了日志后，就能够进行日志显示，其他的校友也能够进行相应的页面进行日志的查看和评论。校友进行日志查看，也能够对自己的日志进行编辑处理，管理员能够对校友的相关日志进行查看、删除、编辑等操作。

在日志显示时，包括多个小模块进行数据显示，这些小模块包括日志显示模块、评论显示模块以及评论模块。日志显示页面 HTML 代码基本同日志发布页面相同，而各个模块根据其显示的效果而不同。日志显示模块 HTML 代码见光盘中源代码\第 29 章\29-1\29-1\shownew.aspx。

其中，页面代码使用了 DataList 控件和 SqlDataSource 控件进行日志显示，通过传递的页面参数 id 进行数据查询并呈现在相应的 DataList 控件中。当用户评论后，日志显示页面还需要对相应的评论进行显示，评论显示页面 HTML 代码见光盘中源代码\第 29 章\29-1\29-1\shownew.aspx。

显式评论代码用于呈现用户的评论并按照一定的 HTML 格式输出。当用户阅读日志后并希望能够进行相应的评论，可以在页面的评论模块中进行评论操作，评论模块 HTML 代码见光盘中源代码\第 29 章\29-1\29-1\shownew.aspx。当用户填写完成标题和内容后就能够通过按钮控件进行评论的提交，评论提交代码如下所示。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        string strsql = "insert into diarygbook (title,time,content,userid,diaryid) values
                        ('"+TextBox1.Text+"','"+DateTime.Now+"','"+TextBox2.Text+"','"
                        +Session["userid"].ToString()+"','"+Request.QueryString["id"]+"')";      //编写 SQL
        SQLHelper.SQLHelper.ExecNonQuery(strsql);                                                //执行 SQL
        Response.Redirect("shownew.aspx?id=" + Request.QueryString["id"]);                       //页面跳转
    }
```

当用户进行留言操作后会跳转到同样的页面并呈现用户的留言信息。

### 29.5.10 用户索引页面

当用户发布日志后，用户可以通过索引页面索引自己的日志并查看相关的日志。不仅是一个用户，而且校友和管理员都能够进行用户索引的查看，用户索引页面 HTML 核心代码见光盘中源代码\第 29 章\29-1\29-1\userindex.aspx。

管理员可以在用户索引页面进行用户的管理，用户也能够在用户索引页面进行相应的用户的信息的查看，例如一个校友用户对他的另一个校友感兴趣，可以点击用户名跳转到用户索引页面查看该用户曾经发布的信息。

### 29.5.11 管理员用户删除

由于数据库中数据表的约束关系，在删除用户时需要对多个表进行操作，用户删除页面同样不需要呈现相应的 HTML 代码而可以直接执行逻辑处理，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["admin"] == null)                                          //判断是否登录
    {
        Response.Redirect("friends.aspx");                                 //页面跳转
    }
    else
    {
        if (Session["admin"].ToString() == "1")                            //判断是否为管理员
        {
            string strsql = "delete from diary where userid='" + Request.QueryString["uid"] + "'";
            string strsql1 = "delete from diarygbook where userid='" + Request.QueryString["uid"] + "'";
            string strsql2 = "delete from register where id='" + Request.QueryString["uid"] + "'";
            SQLHelper.SQLHelper.ExecNonQuery(strsql);                      //删除用户日志
            SQLHelper.SQLHelper.ExecNonQuery(strsql1);                     //删除用户留言
            SQLHelper.SQLHelper.ExecNonQuery(strsql2);                     //删除用户信息
            Response.Redirect("friends.aspx");
        }
        else
        {
            Response.Redirect("errors/cdelete.aspx?id=" + Request.QueryString["id"]);
        }
    }
}
```

在进行删除用户操作时，首先删除用户的日志中的数据，清空其发布的日志，清空后再删除用户的评论数据，清空其对日志发布的评论，清空后才能够进行用户信息的删除。

注意：当数据库中表与表之间包含约束关系，由于数据完整性的原因，如果操作不按照数据规范进行操作，则系统会抛出异常。

## 29.6 用户体验优化

在前面的小结中只是将应用程序所需要的功能简单的实现了，而简单的实现并不能够满足现今越来

越丰富的应用程序要求。用户体验优化也是现在应用程序开发的必经阶段，提高用户体验有助于快速的加入和使用应用程序。

### 29.6.1 超链接样式优化

超链接样式是用户体验优化中一个非常简单却非常重要的部分。超链接显示着不同连接之间的样式，用户能够通过超链接进行跳转。单击【F5】快捷键运行现有的应用程序，如图 29-17 所示。


图 29-17 应用程序运行

从图 29-17 中可以看出，其超链接文本样式为默认文本样式，这样就显得非常的不友好。在进行样式控制时，可以在 CSS 文件中进行超文本连接样式的控制。为了能够更好的配合校友录系统，这里将超文本链接样式设置为蓝色链接样式，示例代码如下所示。

```
a:link
{
text-decoration: none;
color: #3b5888;
}
a:active
{
text-decoration: none;
color: #3b5888;
}
a:visited
{
text-decoration: none;
color: #3b5888;
}
/*设置超链接鼠标移动样式*/
a:hover
{
text-decoration:underline;
color:White;
background: #3b5888;
```

```
}
```

上述代码使用了超链接文本控制样式进行样式控制。其中包括了 a:link、a:active、a:visited 和 a:hover。在 CSS 层叠样式表中，样式是能够继承的。在校友录应用程序开发中，可以定义一个全局超链接样式表，另外，也可以为单独的某个样式进行超链接文本样式控制。示例代码如下所示。

```
.main_right a:link
{
text-decoration: none;
color: #3b5888;
}
.main_right a:active
{
text-decoration: none;
color: #3b5888;
}
.main_right a:visited
{
text-decoration: none;
color: #3b5888;
}
.main_right a:hover
{
text-decoration:underline;
color:#3b5888;
font-weight:bolder;
background:white;
}
```

main_right 是样式表中用于控制侧边栏的样式。在校友录系统中，系统希望右侧的边栏的超链接样式与全局的超链接样式不同，那么开发人员就能够通过继承的方法将相应的层中的样式的超链接文本样式进行覆盖，从而呈现另一种超链接文本样式。

在定义了一个全局超链接文本样式后，全局的超链接文本样式都会被更改成全局样式，如图 29-18 所示。而局部定义的超链接文本样式会被局部样式覆盖，如图 29-19 所示。

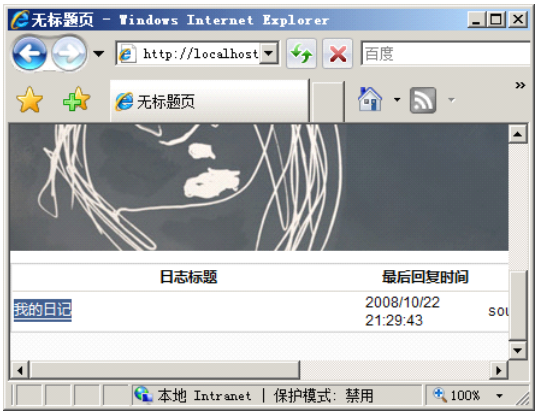


图 29-18　全局样式


图 29-19　局部样式

## 29.6.2 默认首页优化

在前面的代码实现中，并没有制作默认首页，这样就会导致用户访问网站时找不到网站页面。默认首页对网站整体应用而言是非常重要的，当用户访问网站时，默认首页会首先展示在用户面前。在默认首页加载时，应该首先跳转到校友录页面。示例代码如下所示。

```
public partial class _default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        Response.Redirect("friends.aspx");                                    //跳转到校友录页面
    }
}
```

当系统载入首页时，首先会执行首页的 Page_Load 事件。上述代码在 Page_Load 事件中进行了页面跳转。当用户打开并访问页面时，页面即会跳转到“friend.aspx”页面。

## 29.6.3 导航栏编写

导航栏用于用户操作的指引。当用户进入 Web 系统时，通常需要通过导航栏进行应用程序功能的查找。校友录系统的导航栏同样需要编写事务逻辑判断以便不同身份权限的用户查看的信息是不相同的。在校友录系统中，包括以下两种用户权限。

- 普通校友：该用户是普通校友，该用户能够进行日志的发表和用户信息的索引。
- 校友录管理员：该用户是校友录管理员，不仅如此，管理员也是校友录中校友用户的一份子，该用户也能够进行日志的发布和留言的发布。

在导航栏中进行逻辑事务判断并编写成为 js 文件，通过其他文件的引用能够在多个不同页面中进行相同的功能的实现。js 文件示例代码如下所示。

```
<%
    if (Session["admin"].ToString() == "1")
    {
%>
    document.write('<div style="margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;border:1px dashed #ccc">');
    document.write('<img alt="" src="../images/groups.png" style="width: 16px; height: 16px" />
    你好:<% Response.Write(Session["username"].ToString()); %> <br/>
    <img alt="" src="../images/gift.png" style="width: 16px; height: 16px" /> 你的身份是
    <a href="admin/default.aspx"> <span style="color:Red">管理员</span></a> ');
    document.write('<br/><img alt="" src="../images/list.png" style="width: 16px; height: 16px" />
    <a href="../logout.aspx">注销</a> ');
    document.write('</div>');
<%
    }
    else
    {
%>
    document.write('<div style="margin:5px 5px 5px 5px;padding:5px 5px 5px 5px;border:1px dashed #ccc">');
    document.write('<img alt="" src="../images/groups.png" style="width: 16px; height: 16px" />
    你好:<% Response.Write(Session["username"].ToString()); %> <br/><img alt="" src="../images/gift.png"
    style="width: 16px; height: 16px" /> 你的身份是<span style="color:Red">普通用户</span> ');
```

```
    document.write('<br/><img alt="" src="../images/list.png" style="width: 16px; height: 16px" />
    <a href="../logout.aspx">注销</a> ');
    document.write('</div>');
<%
    }
%>
```

上述代码不仅简单的实现了导航栏的编写，还实现了用户信息的简约查看，如图 29-20 所示。如果用户是管理员，则会以管理员的导航样式形式进行呈现，如图 29-21 所示。



图 29-20　用户信息导航栏　图 29-21　管理员导航样式

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="main_right">
            <script src="js/banner.aspx" type="text/javascript"></script><br/>
            <cc2:NewFriends ID="NewFriends1" runat="server" />
        </div>
```

在需要使用导航的页面添加 js 调用，则相应的页面就能够呈现 js 导航信息。js 导航并不局限于网站页面或功能的导航，在很多情况下，js 导航还能够制作用户控制面板、网站页头、网站页尾等通用模块。

注意：js 导航不仅仅能够制作带有逻辑的页面引用，js 还能够编写 HTML 进行页面题头、题尾等通用模块的编写，这样不仅能够在多个页面中使用，也方便了系统的维护。但如果在网站中大量的使用 js 页面进行逻辑判断，也可能会造成性能问题。

## 29.6.4　AJAX 留言优化

AJAX 能够提高应用程序的用户体验，在前面的留言本章节中就使用了 AJAX 用于无刷新应用程序的实现。在校友录系统的实现中，同样可以使用 AJAX 进行无刷新实现，示例代码见光盘中源代码\第 29 章\29-1\29-1\shownew.aspx。

上述代码将留言进行呈现，为了能够使用 AJAX 进行无刷新功能的实现，需要将此控件防止在 AJAX 的局部更新控件中。在进行了数据绑定控件的模板配置后，还需要配置数据源，示例代码见光盘中源代码\第 29 章\29-1\29-1\shownew.aspx 中数据源的配置。

其中，页面代码进行了数据源的配置。值得注意的是，数据在 AJAX 应用中也是非常重要的。当用户执行数据更新时，数据绑定控件还需要通过数据源重绑定进行数据更新。当用户单击【留言】按钮后，系统会执行数据库插入操作并跳转到当前页面进行页面重加载。使用 AJAX 进行页面局部更新就不需要进行页面跳转。

在应用程序代码控制中，值需要进行数据绑定控件的数据重绑定即可。在数据绑定控件中，大部分的数据绑定控件都具有 DataSourceID 属性。该属性用于指定当前用户使用的数据源，在指定了数据源之后，数据并不会自动进行绑定，还需要使用 DataBind 方法进行数据绑定，示例代码如下所示。

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

```
            string strsql = "insert into diarygbook (title,time,content,userid,diaryid) values
              ('"+TextBox1.Text+"','"+DateTime.Now+"','"+TextBox2.Text+"','"+Session["userid"].ToString()+"','"+
            Request.QueryString["id"]+"')";                                          //生成 SQL 语句
            SQLHelper.SQLHelper.ExecNonQuery(strsql);                                //执行 SQL 语句
            DataList2.DataSourceID = "SqlDataSource2";                               //配置数据源属性
            DataList2.DataBind();                                                    //数据源重绑定
        }
```

上述代码使用了 DataBind 方法进行数据重绑定。当执行了数据重绑定后，数据绑定控件将能够直接进行数据呈现而无需再次页面跳转进行页面呈现，如图 29-22 所示。

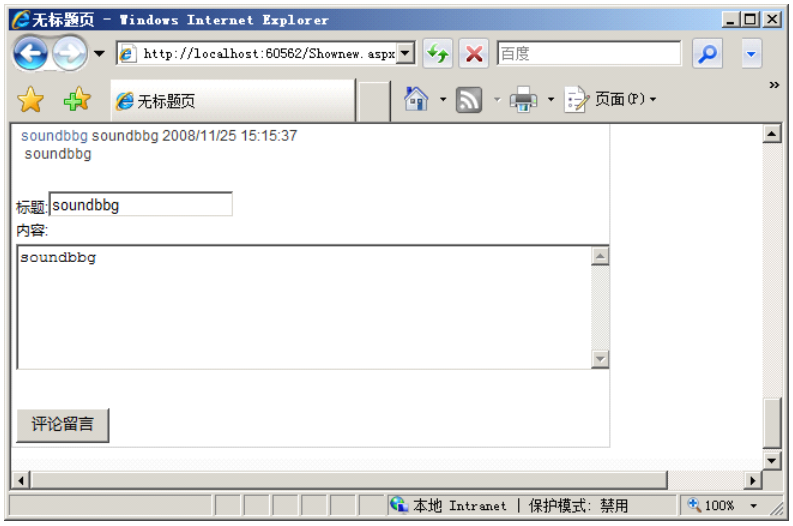


图 29-22　无刷新页面重绑定

值得注意的是，在使用 AJAX 控件进行留言等数据操作时，其控件的状态依旧会保持原有的数据状态，例如图 29-22 中所示。当向某个日志进行评论后，文本框控件中的文本并没有被清楚。在执行操作时还需要进行文本清除，示例代码如下所示。

```
            TextBox1.Text = "";
            TextBox2.Text = "";
```

注意：由于 ASP.NET 对 AJAX 的封装，AJAX 应用程序的开发已经非常容易，但是在 AJAX 应用程序开发时，还需要注意 AJAX 运行过程中的控件状态。

## 29.6.5　优化留言表情

在很多应用程序中，留言和聊天的文本中都会出现表情，如 QQ 就可以使用表情，如图 29-23 所示。同样，在 Web 应用程序中也可以使用表情，最常见的就是论坛、博客和新闻评论了，如图 29-24 所示。在校友录系统中，只有正文能够使用表情，而该表情是通过 Fckeditor 进行实现的。为了让校友录系统更加丰富，可以使用 C#和 js 共同实现表情功能。

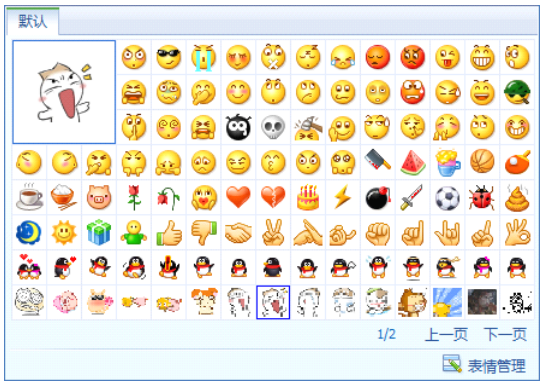


图 29-23　QQ 表情


图 29-24　网站应用表情

在制作和实现表情功能时，首先需要考虑表情是如何呈现的。从现有的应用中可以看出，当单击了表情之后，表情会以一种字符的形式呈现在相应的控件中（主要是文本框控件），如图 29-25 所示。当用户单击发布按钮进行留言发布后，该字符会被转义成表情图片进行呈现。所以，表情功能实现的第一步就是表情的呈现。


图 29-25　表情呈现

### 1．表情呈现和选择

正如图 29-25 所示，微笑的表情是字符“: )”。当用户单击【笑脸】表情时，首先笑脸表情会转义成字符串“: )”呈现在文本框控件中。表情可以使用图片按钮控件进行呈现，当单击相应的【表情】按钮时，会触发方法进行文本框中文本的呈现。

值得注意的是，这种方法当有多个表情时会使代码变得非常的复杂和冗长，不仅如此，这种方法也增加了页面的控件数量。当页面载入时，ASP.NET 托管程序首先会将控件进行转义和呈现并生成新的页面模型，当控件的数量增大时，难免会页面性能问题。为了解决这个问题，可以使用 JavaScript 进行按钮控件的事件模拟。

右击现有项，在下拉菜单中选择【新建文件夹】选项，在新建文件夹选项中选择【添加新项】，选项，在弹出菜单中选择【JScript】文件，如图 29-26 所示。

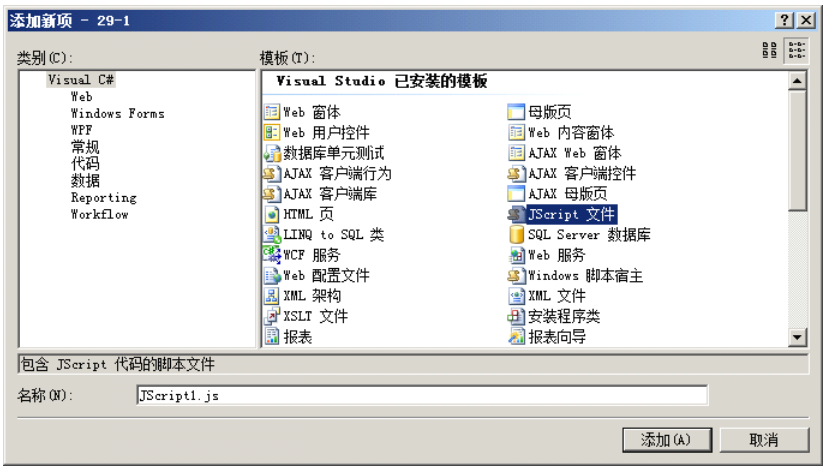


图 29-26　新建 JScript 文件

JScript 文件用于编写 JavaScript 函数，该函数能够在其他页面进行调用。添加表情函数示例代码如下所示。

```
function add_smile(smile)
 {
  var str=document.getElementById("TextBox2");
  str.value+=smile;
 }
```

上述代码使用了 JavaScript 创建了一个添加表情函数，其过程非常简单。该函数拥有一个参数，这个参数是表情的字符，如“: )”字符串。当执行了函数时，该函数首先会查找 ID 为 TextBox2 的文本框，查找后会将传递的字符串添加到相应的文本框控件中。编写了函数后就需要在页面中进行函数的引用，示例代码如下所示。

```
<script src="js/JScript1.js" type="text/javascript"></script>
```

上述代码声明了一个 JavaScript 页面的引用，当引用了该 JavaScript 页面后，该页面的标签就能够使用该页面提供的函数。在表情的呈现过程中，使用图片控件或图片按钮控件都是不合适的，这里可以直接使用 HTML 图片并通过使用 add_smile 函数实现表情，示例代码如下所示。

```
<img src="smiles/0.gif" onclick="add_smile(':)')"/>
<img src="smiles/1.gif" onclick="add_smile(':s')"/>
<img src="smiles/2.gif" onclick="add_smile(':>')"/>
<img src="smiles/3.gif" onclick="add_smile(':-)')"/>
<img src="smiles/4.gif" onclick="add_smile(':->')"/>
<img src="smiles/5.gif" onclick="add_smile(':<')"/>
<img src="smiles/6.gif" onclick="add_smile(';)')"/>
<img src="smiles/7.gif" onclick="add_smile(':o')"/>
<img src="smiles/8.gif" onclick="add_smile(':zz')"/>
```

上述代码呈现了若干表情，并编写了 HTML 控件的 onclick 事件。该事件通过传递参数添加到文本框控件中。例如当单击 URL 路径为“smiles/0.gif”的图片时，会触发 add_smile(‘:)’)事件，该事件会传递一个“:)”字符串到文本框 TextBox2 中，如图 29-27 所示。

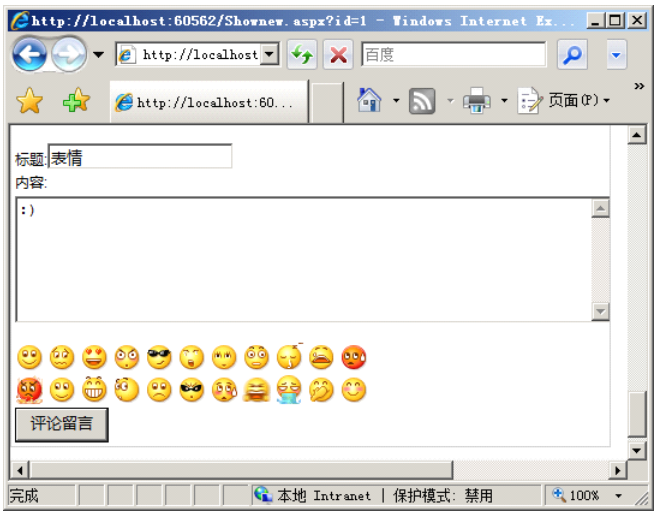


图 29-27　表情呈现和选择

**2．表情转义**

如果直接单击【评论留言】按钮，系统将会进行数据插入。但是值得注意的是，“:)”字符串并不会以表情的形式呈现，如果需要以表情的形式呈现，在执行数据插入前，还需要将表情转换成 HTML 代码，核心代码如下所示。

```
public string opFormatsmiles(object input)
{
    string data = input.ToString();                                          //获取传递的参数
    data = data.Replace("&amp;", "&");                                       //替换特殊符号
    data = data.Replace("&quot;", "/");                                      //替换特殊符号
    data = data.Replace("&qapos;", "'");                                     //替换特殊符号
    data = data.Replace("&lt;", "<");                                        //替换特殊符号
    data = data.Replace("&gt;", ">");                                        //替换特殊符号
    data = data.Replace(":o)", "<img src=\"smiles/13.gif\" ale=\"大笑\"/>");   //替换成图片
    data = data.Replace(":)", "<img src=\"smiles/0.gif\" ale=\"我得意的笑\"/>");  //替换成图片
    data = data.Replace(":s", "<img src=\"smiles/1.gif\" ale=\"委屈的很\"/>");   //替换成图片
    data = data.Replace(":>", "<img src=\"smiles/2.gif\" ale=\"色色的笑\"/>");   //替换成图片
    data = data.Replace(":-)", "<img src=\"smiles/3.gif\" ale=\"啊哦..呜呜..\"/>");  //替换成图片
    data = data.Replace(":->", "<img src=\"smiles/4.gif\" ale=\"嘿嘿..\"/>");   //替换成图片
    data = data.Replace(":<", "<img src=\"smiles/5.gif\" ale=\"我哭哭了..\"/>");  //替换成图片
    data = data.Replace(";)", "<img src=\"smiles/6.gif\" ale=\"媚眼..\"/>");    //替换成图片
    data = data.Replace(":o", "<img src=\"smiles/7.gif\" ale=\"有点小小的惊讶\"/>");  //替换成图片
    data = data.Replace(":zz", "<img src=\"smiles/8.gif\" ale=\"睡觉觉咯..\"/>");  //替换成图片
    data = data.Replace(":(", "<img src=\"smiles/9.gif\" ale=\"大哭特哭..\"/>");   //替换成图片
    data = data.Replace(":...", "<img src=\"smiles/10.gif\" ale=\"..\"/>");    //替换成图片
    data = data.Replace(":xx", "<img src=\"smiles/11.gif\" ale=\"我恼火的很\"/>");  //替换成图片
    data = data.Replace(":p", "<img src=\"smiles/12.gif\" ale=\"笑笑\"/>");     //替换成图片
    data = data.Replace(":ma", "<img src=\"smiles/14.gif\" ale=\"惊讶\"/>");    //替换成图片
    return data;
}
```

上述代码将相应的字符串进行转换，从而呈现相应的表情的 HTML 图片。例如“:)”字符串会在应用程序执行时被替换成字符串“<img src=\"smiles/0.gif\" ale=\"我得意的笑\"/>”。当页面呈现时，该字符串会以 HTML 的形式呈现，这样用户看到的字符串就是一个表情而不是一个文本字串。在执行数据

插入之前，还需要通过该函数进行转换，示例代码如下所示。

```
string strsql = "insert into diarygbook (title,time,content,userid,diaryid) values ('" + TextBox1.Text + "','" +
DateTime.Now + "','" + opFormatsmiles(TextBox2.Text) + "','" + Session["userid"].ToString() + "','" +
Request.QueryString["id"] + "')";                                    //转换后添加
SQLHelper.SQLHelper.ExecNonQuery(strsql);                            //执行 SQL 语句
```

上述代码在插入数据前使用了 opFormatsmiles 方法进行文本中字符串的替换，替换完成后就能够以 HTML 的形式呈现在留言信息中，如图 29-28 所示。

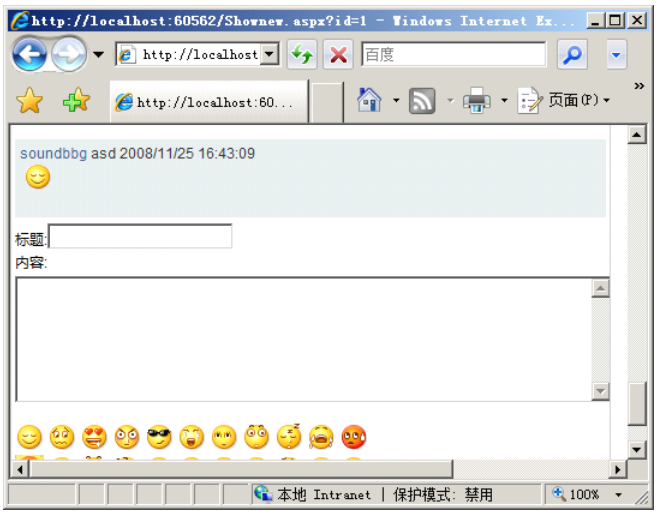


图 29-28　表情显示

注意：在表情功能的实现中，还可以不进行替换直接将表情字串添加到数据库中。当呈现留言或评论数据时，可以使用编程控制数据的呈现方式，即在呈现表情时进行字符串替换。

# 29.7　高级功能实现

在前面的功能实现的章节中，只是制作了基本的应用开发所需要的功能，其中还有板报、管理员管理、后台留言管理以及关键字过滤等功能没有实现。虽然这些功能的实现并不复杂，但是这些功能都是健壮的应用程序所必备的。

## 29.7.1　后台管理页面实现

虽然管理员能够在前台进行数据的管理，但是前台的数据管理往往非常不方便。复杂的前台管理功能不仅影响了用户体验，并且还暴露了管理功能和管理路径。后台管理页面不仅能够保护管理功能防止非法用户进行管理的尝试，后台还为管理员提供了统一的管理界面和管理工具，管理员能够在后台管理页面方便的进行数据管理。

单击【开始】菜单，在菜单中选择【程序】选项，找到【Microsoft Expression】选项并在 Microsoft Expression 选项中选择【Microsoft Expression Web 2】选项打开 Microsoft Expression Web 2 用于框架集的制作。在制作框架集之前首先需要确定框架的作用，这里可以创建一个【横幅和目录】形式的框架用于系统的管理，如图 29-29 所示。

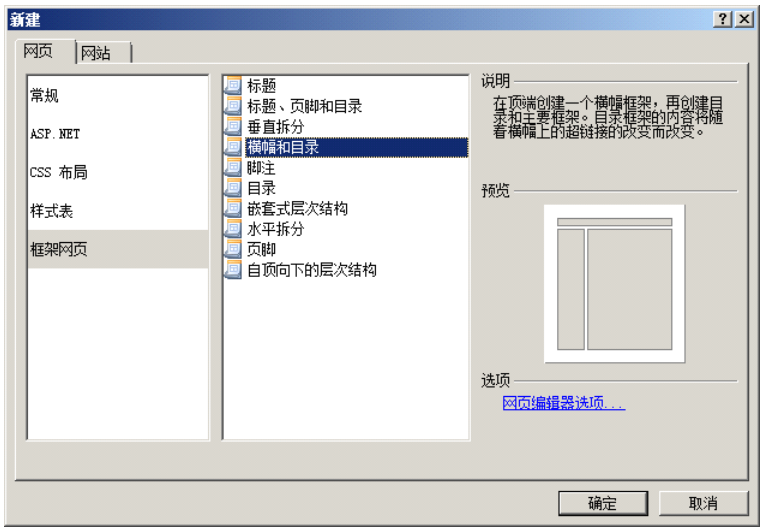


图 29-29　选择框架集

【横幅和目录】形式的框架包括三个窗口，从图 29-29 中可以看出，这三个串口分别位于主窗口的上方、下方左侧和下方右侧。这里分别命名为 head.aspx、left.aspx 和 center.aspx 并将框架集保存为 default.aspx。这三个框架集的作用分别如下所示。

- head.aspx：用于呈现头部信息，通常情况下该框架集用于美工作用，并不呈现实际的作用。
- left.aspx：侧边栏用于快捷方式的存放，开发人员能够选择侧边栏进行相应的功能的设置。
- center.aspx：中间部分用于呈现主工作区，当开发人员在侧边栏中选择了相应的快捷方式后，主工作区用于呈现工作所必须呈现的界面。

在确定了三个框架的基本作用后，可以编写相应的页面进行框架的呈现，示例代码如下所示。

```
<body style="background:white url('images/bg.png') repeat-x;">
<p><img alt="" height="96" src="images/logo.png" width="379" /></p>
</body>
```

上述代码实现了框架集头部代码，该代码用于实现头部布局。为了让管理人员方便的进行系统的管理和操作，可以在侧边栏使用 TreeView 控件进行导航，示例代码见光盘中源代码\第 29 章\29-1\29-1\admin\left.aspx。

其中，页面代码编写了一个 TreeView 控件用于后台系统的导航，TreeView 控件包括管理首页、日志管理、用户管理和退出管理几个大块。这几个模块的作用如下所示。

- 管理首页：主要是用于全局配置，包括关键字管理等。
- 日志管理：包括日志的修改和删除，管理员能够在后台管理日志并进行日志的删除操作。
- 用户管理：用户管理包括用户密码的修改、信息的修改以及用户的删除。
- 退出管理：注销管理主要用于管理员的退出操作。

在确定了基本的管理模块后就可以针对管理模块进行后台页面的开发。

## 29.7.2　日志管理

虽然管理员能够在前台页面进行日志的管理操作，但是前台的操作毕竟十分有限，在后台管理页面中，由于已经确认了管理员的身份，则管理员能够对数据进行修改和删除操作。在进行修改和删除操作时，日志数据中的任何字段都能够被管理员修改。

日志管理页面需要展示日志数据，并提供管理员快捷的进行日志的修改和删除操作。在日志管理页

面，可以使用 GridView 控件用于数据呈现。GridView 控件能够创建自定义连接进行高级的数据操作，这里可以自行创建【修改】选项和使用系统默认的【删除】选项，示例代码如下所示。

```
<asp:HyperLinkField DataNavigateUrlFields="id"
    DataNavigateUrlFormatString="dmodi.aspx?id={0}" Text="修改">
    <ItemStyle Width="25px" />
</asp:HyperLinkField>
<asp:HyperLinkField DataNavigateUrlFields="id"
        DataNavigateUrlFormatString="del.aspx?id={0}" Text="删除">
    <ItemStyle Width="25px" />
</asp:HyperLinkField>
```

上述代码只是 GridView 控件的一部分，用于呈现自定义修改超链接和删除超链接。在代码中，系统创建了【修改】选项和【删除】选项，【修改】选项是自定义选项，当管理员单击【修改】超链接时，系统会跳转到 dmodi.aspx 页面并进行数据的呈现。管理员能够在 dmodi.aspx 页面进行数据的修改和更新。当管理员单击【删除】超链接时，系统会删除相应的新闻信息。

日志管理的数据源配置不需要使用自动生成 SQL 语句选项进行智能的 SQL 操作的支持，因为在数据源操作的过程中，其中的【修改】选项和【删除】选项所需要实现的数据操作都是通过自定义模块进行实现。日志管理页面的数据源只需要连接数据即可，示例代码如下所示。

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:friendsConnectionString %>"
    SelectCommand="SELECT * FROM [diary] ORDER BY [id] DESC">
</asp:SqlDataSource>
```

上述代码只使用了 SqlDataSource 控件的 SelectCommand 属性进行数据的呈现，如图 29-30 所示，管理员只需要对其中的数据进行查看和筛选即可。

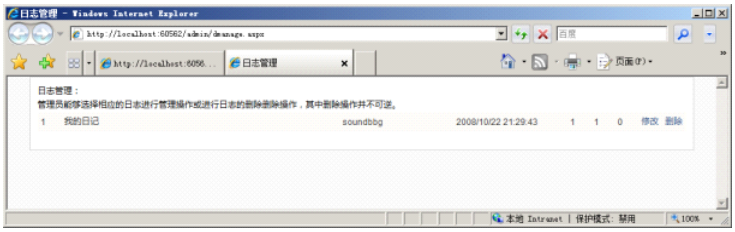


图 29-30　日志管理页面

## 29.7.3　日志修改和删除实现

在前台页面中，已经实现了日志的修改和删除。在后台页面中，日志的修改和删除操作也基本相同。另外，管理员在修改日志时具备比前台修改更多的权限，包括前台不能够修改的字段，管理员也能够在后台修改。

**1. 日志修改实现**

后台管理页面中的日志修改可以修改不同的用户、不同的字段，相比之下，从后台进行日志修改能够更加方便的进行多个保密字段的修改，日志修改页面 HTML 核心代码见光盘中源代码\第 29 章\29-1\29-1\admin\dmodi.aspx。

在前台页面中，管理员能够修改用户的基本信息，但是无法修改用户日志的发布时间和阅读次数。在后台系统中，管理员能够修改用户日志的发布时间以便能够修正用户的日志信息。另外，管理员还能够修改阅读次数，示例代码见光盘中源代码\第 29 章\29-1\29-1\admin\dmodi.aspx。

由于用户发布的日志是基于 Fckeditor 编辑器进行发布的，所以在后台日志管理页面中，同样需要使用 Fckeditor 进行日志修改，示例代码如下所示。

```
<td colspan="2">
    <FCKeditorV2:FCKeditor ID="FCKeditor1" runat="server" Height="300px">
    </FCKeditorV2:FCKeditor>
</td>
```

当管理员修改了相应的选项后，可以单击【修改】按钮进行日志的修改。当单击【修改】按钮后，系统还需要进行字段的检查才能够进行数据更新，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    try
    {
        string strsql = "update diary set title='" + TextBox1.Text + "',content='" + FCKeditor1.Value + "',
        time='"+TextBox2.Text+"',hits='"+TextBox3.Text+"' where id='" +
        Request.QueryString["id"] + "'";                                    //更新数据库
        SQLHelper.SQLHelper.ExecNonQuery(strsql);                           //执行更新语句
        Response.Redirect("dmanage.aspx");                                  //跳转到管理页面
    }
    catch
    {
        Label3.Text = "出现错误,请检查日志";                                 //提示异常信息
    }
}
```

当管理员单击【修改】按钮进行数据更新时，管理员所填写的字段都会更新到数据库中。用户能够在前台的相应页面进行查看。

**2．日志删除实现**

日志删除实现的过程比较容易，但是同留言本系统一样，在执行数据的删除时同样要注意数据的约束性和完整性，删除操作代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    string strsql = "delete form diarygbook where diaryid='"+Request.QueryString["id"]+"'"; //删除评论
    string strsql1 = "delete from diary where id='"+Request.QueryString["id"]+"'";          //删除新闻
    SQLHelper.SQLHelper.ExecNonQuery(strsql);                                               //执行删除
    SQLHelper.SQLHelper.ExecNonQuery(strsql1);                                              //执行删除
    Response.Redirect("dmanage.aspx");                                                      //页面跳转
}
```

执行了上述代码后，系统会将日志以及与日志有关的评论全部删除。但是在执行删除操作时，首先需要删除与目的数据相关的所有其他数据后才能够删除目的数据。

注意：在这几章实例章节中都强调了数据完整性需求，当删除某个数据前，一定要检查数据的约束性和完整性，以便能够正确删除数据。

## 29.7.4　评论删除实现

评论删除功能的实现非常简单，可以直接使用系统数据源控件提供的删除功能即可实现评论的删除。示例代码见光盘中源代码\第 29 章\29-1\29-1\admin\gmanage.aspx。

其中，页面代码配置了 GridView 控件以便该控件能够进行数据的呈现和删除功能的选择。由于该

控件需要数据源支持删除操作，则数据源必须支持智能的生成插入、更新、删除的 SQL 语句，示例代码见光盘中源代码\第 29 章\29-1\29-1\admin\gmanage.aspx 中数据源的配置。

其中，页面中数据源控件的代码实现了数据的插入、更新和删除所需要使用的 SQL 语句的生成。当数据绑定控件执行了删除操作时，会触发数据源控件的 DeleteCommand 属性进行数据删除。

### 29.7.5 板报功能实现

在校友录的每个页面中，都有一个板报用于呈现相应的板报信息。校友能够通过板报了解最近的校友录的最新动态，包括日志、管理员信息以及一些周边八卦等。在设计数据库时，并没有为板报功能专门设计数据库信息，所以板报可以通过 js 进行实现。在板报功能制作直线，首先需要知道板报功能是如何工作的，板报功能的实现需要两个文件，这两个文件分别为 txt 文件和以.aspx 为结尾的 js 文件，这两个文件的作用如下所示。

- txt 文件：用于存放数据，显示板报内容。
- js 文件：用于调用 txt 文件中的板报数据。

txt 文件用于存放数据。在板报功能模块中，板报主要用于存放字符串数据，而且字符串数据通常只是若干字符而已，最多显示一些图片，所以板报只需要打开 txt 文件进行文件内容的增删即可。当板报管理页面加载时，首先需要加载 txt 文本文件的内容，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        try
        {
            StreamReader aw = File.OpenText(Server.MapPath("banbao.txt"));      //打开文本文件
            TextBox1.Text = aw.ReadToEnd();                                     //读文本文件
            aw.Close();                                                         //关闭文本对象
        }
        catch
        {
            TextBox1.Text = "公告文本文件读取错误";
        }
    }
}
```

在载入页面文件后，管理员能够在文本框中填写板报文本字段，在填写了之后，单击【保存公告】按钮就能够进行板报的发布，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    StreamWriter sw1 = File.CreateText(Server.MapPath("banbao.txt"));           //创建文本文件
    sw1.Write(TextBox1.Text);                                                   //输出文本内容
    sw1.Close();                                                                //关闭输出对象
    Response.Redirect("manage.aspx");                                           //页面跳转
}
```

保存的板报数据会保存到文本文档“banbao.txt”中，当需要页面中读取“banbao.txt”文档的文本时，同样可以使用 StreamReader 类进行读取。另外，还可以使用 JS 进行文本文档中内容的读取，示例代码如下所示。

```
<%@ Page Language="C#"
AutoEventWireup="true" CodeBehind="banbao.aspx.cs" Inherits="_29_1.js.banbao" %>
document.write('<% Response.Write(str); %>');
```

上述代码输出了共有字符串 str，在页面逻辑代码实现中，可以从文本中读取字符串并赋值给 str 变量进行文本输出，示例代码如下所示。

```
public partial class banbao : System.Web.UI.Page
{
    public string str = "";                                                    //声明共有变量以便输出
    protected void Page_Load(object sender, EventArgs e)
    {
        try
        {
            StreamReader aw = File.OpenText(Server.MapPath("../admin/banbao.txt")); //读取文本
            str = aw.ReadToEnd();                                              //读取文本
            aw.Close();                                                        //关闭读取对象
        }
        catch
        {
            str = "暂时没有任何公告";                                          //抛出异常
        }
    }
}
```

上述代码声明了一个共有的字符串型变量 str，该共有变量能够在页面中直接进行输出。值得注意的是，由于该 js 文件保存在根目录的 js 文件夹下，所以读取文本的路径也应该随之改变。如果无法读取文本，但为了提高用户的体验度，系统就不应该输出异常信息，而应该直接输出“暂时没有任何公告”。在需要使用公告的页面可以通过 js 调用进行数据呈现，示例代码如下所示。

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

注意：在使用 JavaScript 形式呈现数据时，要过滤“’”等符号，因为 JavaScript 无法显示某些关键字或特殊符号，如果字符串中包含了这些符号，则可能无法呈现字符串。

## 29.7.6 用户修改和删除实现

管理员能够在前台进行用户信息的访问和用户索引的查看，在后台的操作中，管理员还能够对用户信息进行删除。删除用户信息时，为了保证用户数据的约束性和完整性，还需要对用户评论和用户数据中的数据进行删除。当页面初次被载入时，首先需要从数据库中读取相应的数据，示例代码如下所示。

```
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            string str = "select * from register where id='" + Request.QueryString["id"] + "'";     //执行查询
            SqlDataReader da = SQLHelper.SQLHelper.ExecReader(str);                                  //填充适配器
            while (da.Read())                                                                        //读取数据
            {
                Label1.Text = da["username"].ToString();                                             //填充控件
                TextBox2.Text = da["password"].ToString();                                           //填充控件
```

```
                DropDownList1.Text = da["sex"].ToString();                                  //填充控件
                TextBox3.Text = da["picture"].ToString();                                   //填充控件
                TextBox4.Text = da["im"].ToString();                                        //填充控件
                TextBox5.Text = da["information"].ToString();                               //填充控件
                TextBox6.Text = da["others"].ToString();                                    //填充控件
            }
        }
    }
```

上述代码当页面被载入时执行，首先会将数据库中的数据查询并呈现在用户控件中。当管理员进行用户信息的填写后，可以单击【修改】按钮进行数据更改，示例代码如下所示。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (String.IsNullOrEmpty(TextBox2.Text))                                    //判断密码
        {
            string str = "update register set sex='" + DropDownList1.Text + "',picture='" + TextBox3.Text +
            "',im='" + TextBox4.Text + "',information='" + TextBox5.Text + "',others='" + TextBox6.Text + "'
            where id='"+Request.QueryString["id"]+"'";                              //生成 SQL 语句
            SQLHelper.SQLHelper.ExecNonQuery(str);                                  //执行 SQL 语句
        }
        else
        {
            string str = "update register set password='"+TextBox2.Text+"',sex='" + DropDownList1.Text +
            "',picture='" + TextBox3.Text + "',im='" + TextBox4.Text + "',information='" + TextBox5.Text +
            "',others='" + TextBox6.Text + "' where id='"+Request.QueryString["id"]+"';   //生成 SQL 语句
            SQLHelper.SQLHelper.ExecNonQuery(str);                                  //执行 SQL 语句
        }
    }
```

在修改用户信息时，管理员可以填写用户的密码进行用户密码的更改，如果管理员不填写用户密码，那么在执行更新时不会更新用户的密码。管理员管理用户时，对于长期不上线的用户可以进行删除操作，删除用户信息还需要删除与用户相关的所有数据。在校友录系统中，与注册用户信息相关的数据包括评论数据和日志数据，在执行用户信息删除前首先要删除这些数据，示例代码如下所示。

```
    protected void Page_Load(object sender, EventArgs e)
    {
        string strsql1 = “delete from diarygbook where userid='" + Request.QueryString["uid"] + "'”;
        string strsql2 = " delete from diary where userid='"+Request.QueryString["uid"]+"'";
        string strsql3 = "delete from register where id='" + Request.QueryString["uid"] + "'";
        SQLHelper.SQLHelper.ExecNonQuery(strsql1);                                  //删除日志评论
        SQLHelper.SQLHelper.ExecNonQuery(strsql2);                                  //删除日志信息
        SQLHelper.SQLHelper.ExecNonQuery(strsql3);                                  //删除用户信息
    }
```

上述代码首先删除日志评论以保证日志数据的约束性和完整性，然后再删除日志以保证用户信息的约束性和完整性，当删除了以上数据后，最后删除用户信息。上述代码分别进行数据的删除，在执行删除时，还可以通过编写复杂的删除 SQL 语句进行数据的删除，示例代码如下所示。

```
    protected void Page_Load(object sender, EventArgs e)
    {
        string strsql1 = "delete from diarygbook,diary,register where diarygbook.userid=diary.userid and
        diarygbook.userid=register.id and diarygbook.userid='" + Request.QueryString["uid"] + "'";
        //上述代码进行复杂的 SQL 语句删除多个表
```

```
            SQLHelper.SQLHelper.ExecNonQuery(strsql1);                                    //执行数据删除
        }
```

当运行上述代码时，系统会删除用户所有相关的信息并保证了用户数据的约束性和完整性。系统管理页面如图 29-31 所示。

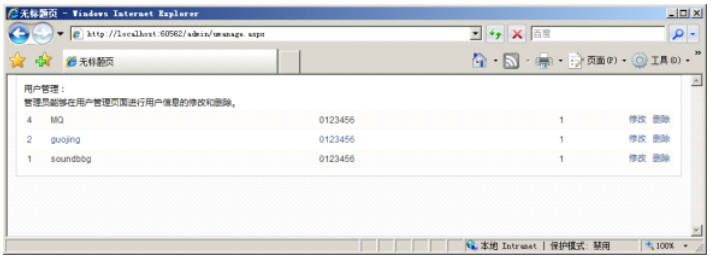


图 29-31　用户管理界面

## 29.7.7　用户权限管理

在数据库设计时，用户的权限通过 userroot 字段进行描述的。如果用户的 userroot 字段值为 0 时，那么这个用户就是一个普通的校友用户，如果 userroot 字段的值为 1 时，则用户会在系统中被判断为管理员。用户权限的管理就是用户信息的管理，如果需要将用户权限进行修改，可以直接使用用户修改功能进行实现。在修改模块中添加一个 DropDownList 控件用于权限的选择，示例代码如下所示。

```
        <asp:DropDownList ID="DropDownList2" runat="server">
            <asp:ListItem Value="0">普通用户</asp:ListItem>
            <asp:ListItem Value="1">管理员</asp:ListItem>
        </asp:DropDownList>
```

在用户权限管理中，还需要在页面加载时载入数据，示例代码如下所示。

```
DropDownList2.Text = da["userroot"].ToString();
```

在执行更新时，还需要将用户权限进行更新，示例代码如下所示。

```
            if (String.IsNullOrEmpty(TextBox2.Text))
            {
                string str = "update register set sex='" + DropDownList1.Text + "',picture='" + TextBox3.Text +
                "',im='" + TextBox4.Text + "',information='" + TextBox5.Text + "',others='" + TextBox6.Text +
                "',userroot='" + DropDownList2.Text + "' where id='" + Request.QueryString["id"] + "'";    //更新
                SQLHelper.SQLHelper.ExecNonQuery(str);                                    //执行 SQL
            }
            else
            {
                string str = "update register set password='" + TextBox2.Text + "',sex='" + DropDownList1.Text
                + "',picture='" + TextBox3.Text + "',im='" + TextBox4.Text + "',information='" + TextBox5.Text +
                "',others='" + TextBox6.Text + "',userroot='" + DropDownList2.Text + "' where id='" +
                Request.QueryString["id"] + "'";                                          //更新 SQL 语句
                SQLHelper.SQLHelper.ExecNonQuery(str);                                    //执行 SQL 语句
            }
```

当更改了 SQL 语句后，用户信息的更新也会更新用户权限。如果管理员希望升级某个校友用户的身份，管理员可以使用下拉菜单控件进行身份的选择，如图 29-32 所示。

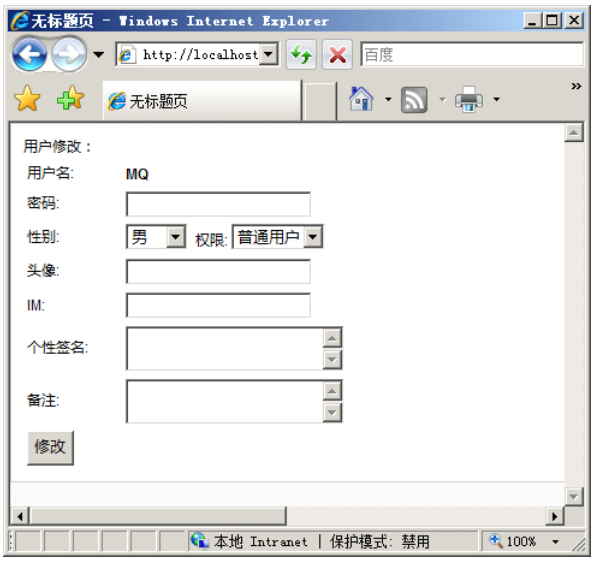


图 29-32　用户权限管理

### 29.7.8　权限及注销实现

在校友录系统登录中，无论是校友用户还是管理员都是从前台登录，并且校友用户和管理员都是使用同一个表，惟一能够区分校友用户和管理员的就是 userroot 字段。在后台登录中，可以使用标识 userroot 的 Session 对象进行权限的判断，示例代码如下所示。

```
if (Session["admin"] == null)                                    //判断是否登录
{
    Response.Redirect("../login.aspx");                          //未登录跳转
    if (Session["admin"].ToString() != "1")                      //判断是否为管理员
    {
        Response.Redirect("../friends.aspx");                    //非管理员跳转
    }
}
```

当管理员在前台登录后，管理员就能够管理员面板进入后台。当管理员完成管理并离开系统时，可以选择【退出管理】选项进行注销操作，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    Session["username"] = null;                                  //清空用户名信息
    Session["userid"] = null;                                    //清空用户 ID 信息
    Session["admin"] = null;                                     //清空管理员信息
}
```

注意：在后台管理中，所有需要且只需要管理员操作的页面都需要进行页面权限判断。

## 29.8　实例演示

在编码完成后还需要对现有的项目进行测试，测试时需要准备数据源并进行数据操作，同时还需要进行程序中的页面测试，对错误的布局和逻辑进行修正。在数据源的准备中，需要考虑到所有的应用场

景进行相应的数据插入，而在对页面进行测试时，不仅需要测试页面的显示，同样还需要对页面逻辑进行测试。

### 29.8.1 准备数据源

在 ASP.NET 校友录系统中，现有的代码并没有为校友录中日志的分类进行添加、删除管理等操作，可以使用 SQL 语句进行数据库中的数据添加，以便用户在校友录中添加日志时选择分类，添加分类 SQL 语句如下所示。

```
insert into diaryclass (classname) values ('生活日记')
insert into diaryclass (classname) values ('青青校园')
insert into diaryclass (classname) values ('天下驴友')
insert into diaryclass (classname) values ('社会时事')
```

上述代码在数据库中添加了 4 个分类，当用户进行日志发布时可以选择相应分类进行归类。

注意：由于篇幅限制，校友录系统并没有为日志分类的添加和删除进行开发，校友录系统的分类的系统开发在前面的新闻模块章节中都有所涉及。

### 29.8.2 实例演示

当用户进行 friend.aspx 页面访问时，系统会判断用户是否登录。如果用户没有登录，则会跳转到 login.aspx 登录页面进行登录，如果用户没有账户则需要进行注册。用户进行注册后就会跳转到登录页面，用户必须要进行登录才能够进行校友录系统的访问，如图 29-33 和图 29-34 所示。


图 29-33 用户注册页面

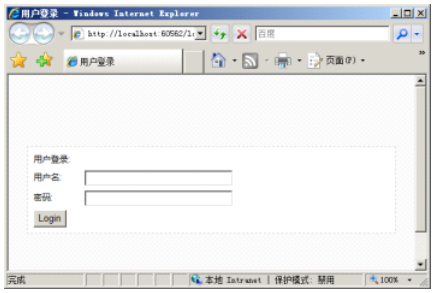


图 29-34 用户登录页面

当用户登录完成后，系统会跳转到校友录首页面，这里在校友录系统中还没有任何的日志，系统会提示用户发布日志，在侧边栏中会显示用户列表，用户列表控件的实现就是通过加入校友控件实现的，如图 29-35 所示。

当有多个用户注册时，侧边的用户列表就会显示多个用户，校友能够点击相应的用户进入用户索引。当用户进入 new.aspx 时，可以发布日志，如图 29-36 所示。

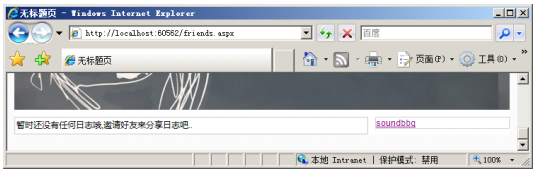

图 29-35　加入校友控件

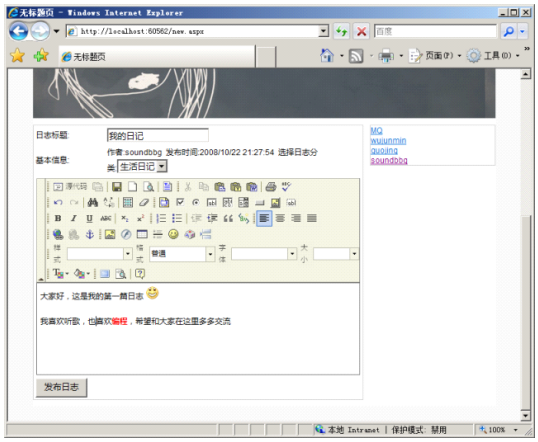

图 29-36　添加日志

发布日志时可以进行文本编辑进行富文本编写，用户还能够添加表情、进行编写检查等操作，这样就能够提高日志的可读性，丰满了日志。查看日志页面如图 29-37 所示。

当用户访问相应的日志时，还能够对日志进行评论，评论的内容会随着相应新闻而呈现在页面中，对于不同的新闻而言所呈现的留言内容也不同，如图 29-38 所示。

图 29-37　浏览日志

图 29-38　日志评论

对于感兴趣的用户，如图 29-38 中的 wujunmin，可以单击其用户名跳转到相应的索引页面，如果访问的用户为管理员，可以进行删除操作。管理员还能够在日志或校友录主界面进行相应的用户的访问，如图 29-39 所示。

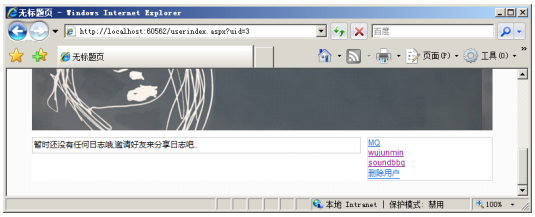

图 29-39　删除用户操作

管理员能够单击相应的用户跳转到用户界面并对用户进行删除操作，删除用户后会删除该用户的所有日志和留言，并删除用户相关的信息。管理员还能够修改和删除日志，删除日志后，与日志相应的评论也会被删除。

注意：无论是在系统开发还是系统测试，都需要遵守数据的完整性，否则可能造成大量的垃圾数据。

### 29.8.3 管理后台演示

管理员在前台登录后，可以通过用户面板进行后台的访问，如图 29-40 所示。


图 29-40 用户面板

单击红色的【管理员】超链接字样，系统将会跳转到后台页面。在后台界面，管理员能够进行日志管理、评论管理以及用户管理，如图 29-41 所示。

单击【日志管理】跳转到日志管理，管理员能够在日志管理页面进行日志的修改和删除。在修改日志时，管理员能够修改前台用户不能修改的字段，如图 29-42 所示。


图 29-41 管理侧边栏


图 29-42 日志管理

管理员能够在日志管理中进行日志的修改，如图 29-43 所示，修改完成后，管理员可以单击【修改】按钮进行数据更新。

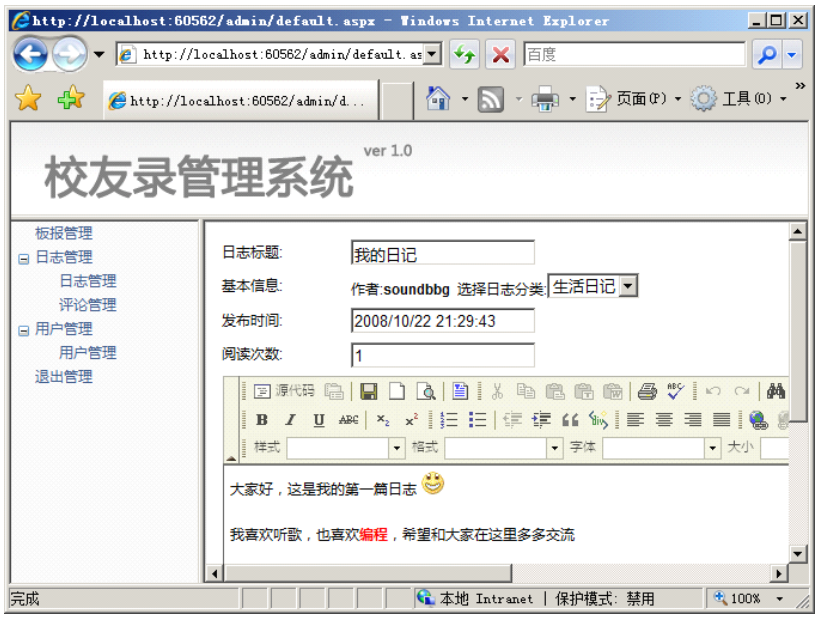


图 29-43　日志修改

单击【评论管理】超链接能够对用户评论进行管理。当前台校友发布了不良的信息时，可以通过【评论管理】选项管理评论并删除相应的评论，如图 29-44 所示。

单击【用户管理】超链接能够对用户进行管理。如果一个用户是非法用户或者用户并不是校友录中的相关用户（如这个校友录是给某个班级做的，而这个用户又不是这个班级的人），那么管理员就能够在【用户管理】选项中进行用户的删除操作。管理员还能够提升一个用户为管理员或将一个管理员降级为普通用户，如图 29-45 所示。

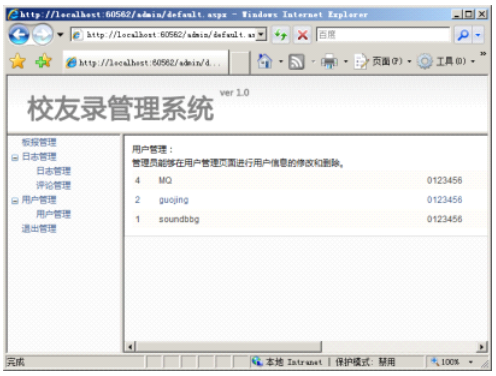


图 29-44　用户管理

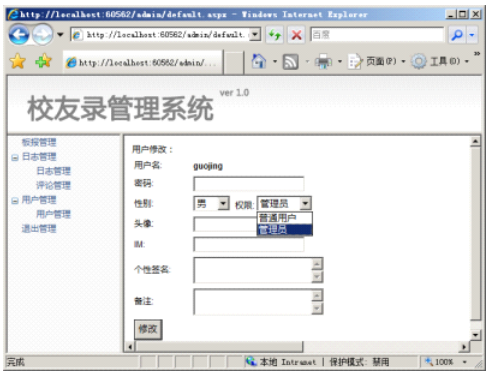


图 29-45　用户权限管理

当管理员操作完毕后，可以单击【退出管理】超链接进行注销操作，注销完成后，系统会跳出后台并提示管理员进行登录操作，如图 29-46 所示。


图 29-46　注销后台

注销后台后，管理员在前台的用户信息也会被注销，如果管理员希望在前台登录或者继续在后台进行管理，管理员还需要进行登录操作。

## 29.9　小结

本章通过开发 ASP.NET 校友录系统进行系统开发讲解，这其中包括了系统设计、模块划分、文档编写和数据设计等，由于篇幅的限制，在 ASP.NET 校友录系统中还有一些功能没有实现，但是这些功能在前面的模块章节中已经实现，对于开发人员而言已经不是很难的问题。

对于系统开发而言，其过程是非常复杂的，不仅从一开始的系统设计还是从开发中的界面设计和编码实现，都是非常复杂的过程，对于初学者而言可能是一个页面的代码实现，而在系统的开发中是将系统模块化，进行分层开发，这就对开发人员有了更高的要求。

本章不仅从系统规划入手，着手基本的分层开发，使用 SQLHelper 类库和自定义控件都能够方便维护中维护成本的降低，本章不仅包括 ASP.NET 校友录系统的开发，还包括开源 HTML 编辑器的使用。本章还包括：

- 数据表关系图：绘制了数据库关系图，保障数据库中的约束条件。
- 使用 Fckeditor：讲解了如何使用 Fckeditor 进行富文本编程。
- 校友录页面规划：讲解了在开发前如何对页面进行页面规划。
- 校友录页面实现：讲解了校友录页面的实现和自定义控件的实现。
- 日志发布实现：讲解了如何使用 Fckeditor 实现日志发布。
- 日志显示页面：讲解了如何进行多表查询进行日志显示。

校友录系统使用了前面模块章节中讲到的模块，这些模块包括注册模块、登录模块和新闻模块，将这些模块进行整合就能够开发出复杂的系统。但是在模块整合的过程中同样会遇到很多问题，这些问题还需要开发人员进行二次开发和完善。