
第 23 章 登录模块设计

登录模块能够配合注册模块让网站应用能够同用户进行信息交互，当用户在网站进行注册后，就需要登录模块进行用户登录，登录模块虽然看上去比较容易，但是要比注册模块复杂一些，如身份处理，这些复杂的地方需要使用 ASP.NET 内置对象。

23.1 学习要点

登录模块需要涉及到一些 ASP.NET 3.5 的基本知识，如果要仔细学习注册模块的开发，需要详细了解本书的一些章节知识，这些章节如下所示：

- ☐ ASP.NET 的网页代码模型。
- ☐ Web 窗体基本控件。
- ☐ 数据库基础。
- ☐ ADO.NET 常用对象。
- ☐ Web 窗体数据控件。
- ☐ ASP.NET 内置对象。

基本了解了以上章节的知识点后，就能够熟练学习和开发此模块。

23.2 系统设计

登录模块需与注册模块不同的地方在于登录模块面向的用户有两种情况，一种是用户已经注册了，另一种是用户还没有注册，对于没有注册的用户需要引导到注册页面，而对于没注册的非法用户必须进行登录限制。

23.2.1 模块功能描述

登录模块是配合注册模块的另一个非常重要的模块，相比之下，登录模块需要考虑更多的情况，例如用户是否注册，以及用户是否是合法用户，如果是合法用户忘记密码了怎么办，如果是非法用户，登录了多次是否要进行限制等等。登录模块的功能基本可以描述如图 23-1 所示。

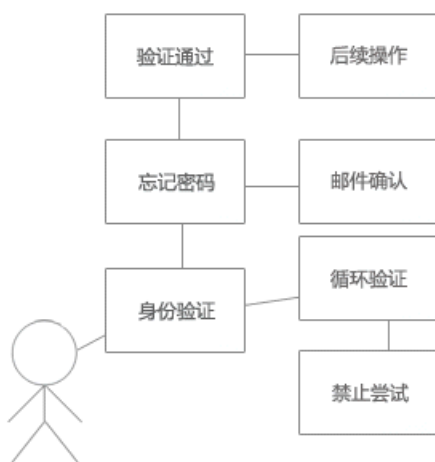


图 23-1 登录模块基本用户流程图

从登录模块可以看出，当用户进行身份验证后，可能会出现几种情况，包括验证通过、忘记密码和循环身份验证。如果用户是一个正常的用户，可以说一次就能够通过验证，那么这个用户就可以进行后续操作；如果用户已经是注册的用户，但是却忘记了密码，可以通过邮件确认进行密码的索要；如果用户是非法用户，在不断的进行尝试，那么就要禁止非法用户的不断尝试。从上述流程基本上可以规划以下几个页面：

- ❑ 登录页面：提供用户的主页面。
- ❑ 忘记密码页面：提供用户索取密码后提示的页面。
- ❑ 用户信息页面：提供用户登录成功后的个人信息页面。

在这其中最主要的是登录页面和忘记密码页面，其中很多的函数的实现都需要在这个页面实现，而其他页面主要是作为提示页面存在的。该模块需要使用 ASP.NET 内置对象对用户的操作进行保存和限制。

22.2.2 模块流程分析

在对业务进行了基本的划分之后，可以为模块进行基本的流程分析，包括这个模块中最基本的函数，以及这些函数在页面中是如何执行的。首先是登录模块需要提供哪些登录信息，登录模块中最重要的就是用户名和密码，登录模块通常情况下通过用户名和密码进行用户权限的判断。

如果用户登录成功，那么用户就是一个合法用户，可以进行后续的操作，如果用户登录失败，则需要让用户选择是否继续登录或者说明忘记密码，如果用户反复尝试则可以认为这个用户可能是非法用户，需要禁止该用户继续进行登录。在了解了基本的模块流程分析后，就可以进行函数和页面的划分，如图 23-2 所示。

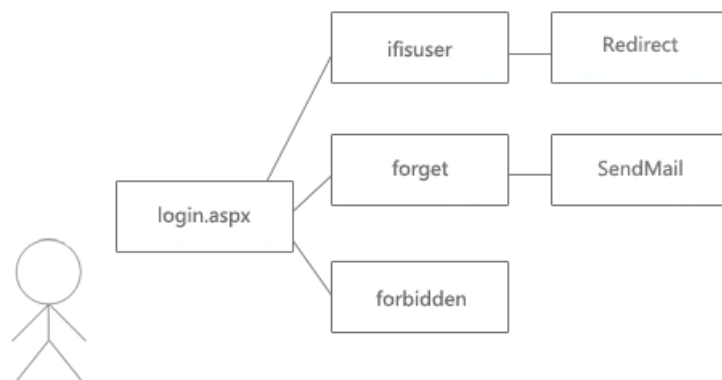


图 23-2 基本页面的函数分析

正如图 23-2 所示，这里主要起到作用的就是 login.aspx 页面，这个页面主要包括三个函数 ifisuser、forget 和 forbidden，分别作为判断用户是否为正常用户，以及判断用户是否忘记密码和非法用户等操作。在用户正常登录后，可以使用 Redirect 方法进行页面跳转，如果用户忘记了密码，需要使用发送邮件函数进行邮件发送，如果用户是非法用户，则需要禁止用户的登录。

23.3 数据库设计

对于登录表同样需要进行数据库设计，而登录表的数据库设计比较简单，只需要一个简单的用户表就能够进行登录设计。通常情况下注册模块和登录模块是一起协调合作的，登录模块读取用户表的信息而注册模块用于数据的索引和插入。

23.3.1 数据库设计分析

对于数据库设计分析，只需要简单的进行用户信息表的设计就可以了，但是这里需要使用用户信息表中的邮箱信息进行验证，所以数据库中表的字段可以归纳如下：

- ❑ 用户名：用户的用户名，用于登录使用。
- ❑ 密码：用户的密码，用于登录中输入密码。
- ❑ email：用户的 E-mail，用于发送邮件，如果用户忘记了密码就可以发送到该邮件。
- ❑ QQ/MSN：用户的 QQ 或 MSN，用于连接。
- ❑ 是否通过：用户的情况，用户保存用户信息，判断用户是否已经被通过。

这里最主要的字段是 email 和 password，这两个字段用于发送邮件到用户和判断用户是否被通过。如果用户忘记了密码，可以封锁该用户的用户信息然后发送邮件到用户的邮箱中，通过激活提示用户密码。

23.3.2 数据库表的创建

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。登录模块的数据库设计比较简单，这里创建一个 Login 数据库并创建一个表，如图 23-3 所示。



	列名	数据类型	允许空
	bh	int	<input type="checkbox"/>
	username	nvarchar(50)	<input checked="" type="checkbox"/>
	password	nvarchar(50)	<input checked="" type="checkbox"/>
	email	nvarchar(50)	<input checked="" type="checkbox"/>
	msn	nvarchar(50)	<input checked="" type="checkbox"/>
	passed	nvarchar(50)	<input checked="" type="checkbox"/>
	ask	nvarchar(50)	<input checked="" type="checkbox"/>
	answer	nvarchar(50)	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

图 23-3 数据库表结构

正如图 23-3 中所示，表为用户的基本信息创建了字段，这些字段的意义分别为：

- ☐ id: 用于标识用户的 ID 号，并为自动增长的主键。
- ☐ username: 用于标识用户名。
- ☐ password: 用于标识用户密码。
- ☐ email: 用于标识用户 E-mail 信息。
- ☐ msn: 用于标识用户的 MSN 等信息。
- ☐ passed: 用于标识用户是否通过审核。
- ☐ ask: 用于保存用户提示信息的问题。
- ☐ answer: 用于保存用户提示信息的答案。

上述字段描述了相应的字段在实际应用中的意义，创建表的 SQL 语句如下所示。

```
USE [Login]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Login](
    bh [int] IDENTITY(1,1) NOT NULL,
    username [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    password [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    email [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    msn [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    passed [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    ask [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    answer [nvarchar](50) COLLATE Chinese_PRC_CI_AS NULL,
    CONSTRAINT [PK_Login] PRIMARY KEY CLUSTERED
(
    bh ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY = OFF,
ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY]
```

//创建 Login 表

上述代码创建了一个数据库并将 ID 设为自动增长的主键，该数据库用于保存用户的基本信息，本模块通常不会更改数据库的信息，只是对数据库进行调用而已。所以在调用之前必须插入若干新数据，示例代码如下所示。

```
INSERT
    NTO
    [Login]
    (username,password,email,msn,passed,ask,answer)
```

```
alues
('guojing','123321','soundbbg@live.cn','hellome@hotmail.com',1,"你  
你好吗?","我很好")
```

上述代码在数据库中插入了一条用户名为 guojing，密码为 123321 的用户信息，并且这个用户的邮箱为 soundbbg@live.cn，当用户忘记密码时，就会通过这个邮箱发送确认信息。

23.4 界面设计

登录界面也能够吸引用户眼球，在登录界面也可以进行广告推广，因为一个网站的良好表现能够让用户大量的在登录页面停驻，在登录页面进行良好的设计可以使登录页面具有广告效应也能够提高用户体验。

23.4.1 基本界面

由于登录模块可能要考虑到很多的扩展，包括广告位之类的，登录页面也可以单独进行一个页面的制作，这些页面包括基本的 TextBox 和 Label 控件用于呈现基本的页面信息，示例代码见光盘中源代码\第 23 章\23-1\23-1\Default.aspx 所示。

上述代码在页面中使用了三个 Label 控件，用于显示用户登录必须的信息，包括指引用户如何填写相应的名称，以及提示是否存在该用户，该页面还包括两个 TextBox 控件用于用户填写相关的信息，并且为了验证用户是否输入正确，在页面中使用了验证控件对用户输入进行控制，示例代码如下所示。

```
<asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"  
    ControlToValidate="TextBox1"  
    ErrorMessage="用户名不能为空"></asp:RequiredFieldValidator>  
<asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"  
    ErrorMessage="密码不能为空"></asp:RequiredFieldValidator>
```

在注册控件已经说明了，验证控件能够验证用户是否输入的是合法的信息，如果用户输入的信息不合法或者输入的信息为空，那么就不应该让操作继续进行，而需要让用户再次进行信息输入。在没有 CSS 样式控制的情况下，使用了表格进行基本的布局，如图 23-4 所示。

用户名	<input type="text"/>	用户名不能为空
密码	<input type="password"/>	密码不能为空
<input type="button" value="登陆"/> 还没有注册? 忘记密码? 你已经被禁止登陆		

图 23-4 基本界面布局

23.4.2 创建 CSS

为了更好的为页面进行页面布局，可以使用 CSS 进行页面的样式控制。在登录页面中，可以为页面和控件进行样式控制，CSS 示例代码如下所示。

```
body //定义全局  
{  
    font-size:12px;  
    font-family:Geneva, Arial, Helvetica, sans-serif;
```

```

        margin:0px 0px 0px 0px;
        background:gray;
    }
    .top
    {
        background:white;
        margin:0px auto;
        margin-top:50px;
        padding-top:10px;
        padding-bottom:10px;
        padding-left:10px;
        width:490px;
        font-size:18px;
    }
    .login
    {
        background:white;
        margin:0px auto;
        width:500px;
    }
    .end
    {
        background:#f9fbfd;
        margin:0px auto;
        width:480px;
        text-align:center;
        padding:10px 10px 10px 10px;
    }

```

上述代码定义了全局页面的字体大小和字体属性，并定义了头部样式、登录主样式和底部样式，定义完成后如图 23-5 所示。



图 23-5 CSS 样式控制后的页面

上述页面的布局非常鲜明，让用户一下就知道登录窗口在哪里，但是这个布局并不方便扩展，也不方便广告位的布局。这里不详细讲解如何进行广告位布局，只是介绍如何对登录页面进行样式布局。

23.4.3 发送密码页面

对于登录控件而言，需要两个提示页面，这两个提示页面包括发送密码页面和错误信息页面。发送密码页面主要是用于发送忘记密码的用户的密码到用户的邮箱中，这样用户就能够获取相应的信息以登录网站；而错误信息页面主要是用于提示用户输入的次数超过限定的次数，禁止用户再次输入。在这两个页面中，需要进行事务处理的页面只有发送密码页面，发送密码页面需要向指定的用户的邮箱发送邮件，而在发送邮件前，必须让用户输入用户名才能够发送。

注意：在用户忘记密码后，必须让用户输入用户信息，然后在数据库中查询相应的用户的邮箱的信息，而不是直接让用户填写邮箱。

发送页面示例代码见光盘中源代码\第 23 章\23-1\23-1\Mail.aspx 所示。其中，代码创建了一个发送邮件页面，当用户填写用户名后，系统会在数据库中查找相应的用户名的用户信息，查找完成后会发送相应的信息到用户的邮箱中，如果用户邮箱正确或者用户提示信息正确，那么系统会发送信息到相应邮箱，如果用户邮箱不正确或者用户提示信息不正确，系统则不会将密码信息发送到用户邮箱。页面布局完成后如图 23-6 所示。

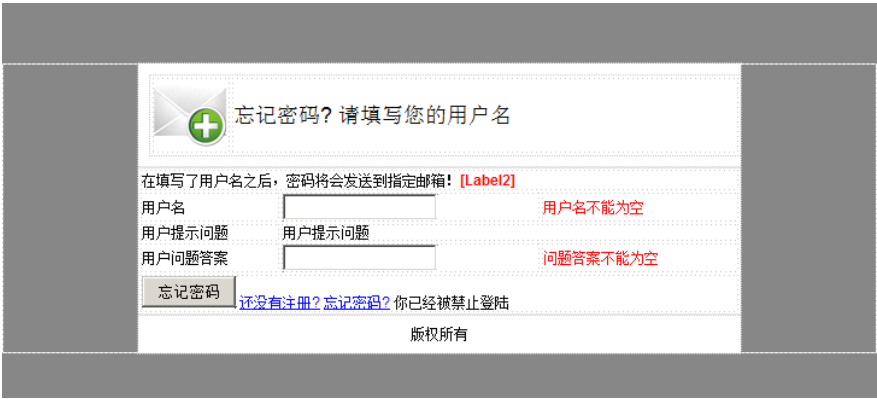


图 23-6 发送密码页面

发送密码页面需要进行业务处理，在发送密码时，必须填写用户名和用户提示问题以及答案，才能够保证此用户是一个安全合法的用户。

23.5 代码实现

在完成基本的 CSS 页面布局后，就需要进行代码实现，登录模块的代码实现比较复杂，不仅需要查询相应的用户是否是合法用户，当用户忘记密码后，还需要通过邮件进行密码的索取，所以在代码实现中还需要实现邮件发送等功能。

23.5.1 登录代码实现

在用户进行登录时，必须验证用户是否已经登录，如果已经登录则不需要再次登录，如果没有登录，则允许用户进行登录操作，当用户单击【登录】按钮时，首先会验证用户是否填写信息，如果没有填写则提示用户填写，如果已经填写了，则判断用户是否是合法用户。

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

```

{
    string str = "server=(local);database=login;uid=sa;pwd=sa";           //连接数据库
    SqlConnection con = new SqlConnection(str);                         //创建连接
    con.Open();                                                         //打开连接
    string strsql =
        "select * from login where username='"+TextBox1.Text+"' and password='"+TextBox2.Text+"'";
    SqlDataAdapter da = new SqlDataAdapter(strsql, con);                 //创建适配器
    DataSet ds = new DataSet();                                         //创建数据集
    int count=da.Fill(ds, "table");                                     //填充数据集
    if (count > 0)                                                      //登录成功
    {
        Session["name"] = TextBox1.Text;                               //赋予 Session
        Session["password"] = TextBox2.Text;                           //赋予 Session
        Session["login"] = "yes";                                       //赋予 Session
    }
    else
    {
        Label3.Text = "登录失败";                                       //登录失败
    }
}

```

当需要判断一个用户是否为合法用户时，只需要在数据库中查询出该用户即可，如果查询出该用户，则说明这个用户是存在的；如果查询不出该用户，则说明这个用户是不存在的。查询用户可以使用 ADO.NET 的 DataSet 对象，示例代码如下所示。

```

"select * from login where username='"+TextBox1.Text+"' and password='"+TextBox2.Text+"'";
SqlDataAdapter da = new SqlDataAdapter(strsql, con);                   //创建适配器
DataSet ds = new DataSet();                                           //创建数据集
int count=da.Fill(ds, "table");                                       //填充数据集

```

上述代码使用 DataSet 对象和 SqlDataAdapter 对象进行数据填充，DataSet 对象的 Fill 方法会返回受影响的行数，当执行查询语句时，如果返回受影响的行数大于 0，则说明存在这个用户，如果受影响的行数小于等于 0，则说明不存在该用户。

注意：在验证用户时一定要同时进行用户名和密码的判断，如果不这样判断，很可能非法用户会猜出用户名就能够进行登录。

如果查询出的结果大于 0，则说明用户是合法用户，可以为用户赋予 ASP.NET 内置对象，以保存用户状态，示例代码如下所示。

```

Session["name"] = TextBox1.Text;                                       //赋予 Session
Session["password"] = TextBox2.Text;                                   //赋予 Session
Session["login"] = "yes";                                              //赋予 Session

```

上述代码当用户登录成功时，给每个用户一个 Session 对象，如果在一定时间内不进行操作或者用户关闭了浏览器进程，系统就会注销该用户。为了保证用户无法重复多次进行登录，可以在登录页面添加一个计数器，这里可以使用一个 Label 控件进行计数控制，Label 控件可以设置为不可见，初始值为 0，示例代码如下所示。

```
<asp:Label ID="Label4" runat="server" Text="0" Visible="False"></asp:Label>
```

在执行登录代码时，首先要判断该控件的值。这里设置登录 4 次后就无法登录了，示例代码如下所示。

```

if (Convert.ToInt32(Label4.Text) < 4)
{
    //登录操作
}

```



```

//判断操作如下
if (count > 0)                                     //判断登录次数
{
    Session["name"] = TextBox1.Text;               //赋予 Session
    Session["password"] = TextBox2.Text;           //赋予 Session
    Session["login"] = "yes";                      //赋予 Session
}
else
{
    Label3.Text = "登录失败";                      //提示登录失败
    int times = Convert.ToInt32(Label4.Text);      //登录次数
    Label4.Text = (times + 1).ToString();          //登录次数加一
}
}
else
{
    Label3.Text = "您已经被禁止登录,请稍后再登录"; //静止登录
}

```

上述代码首先会判断计数器中的值是不是小于 4，如果小于 4，则可以进行登录操作，否则就会禁止用户登录。在登录失败时，必须让计数器的值加 1，否则计数器的值永远小于 4。

23.5.2 邮件发送页面

在用户需要索取自己的密码时，系统对用户进行邮件发送功能的实现和使用，这样就保证了用户信息的机密性，而用户可以在自己的邮箱中获取密码。邮件发送示例代码如下所示。

```

protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    string str = "server=(local);database=login;uid=sa;pwd=sa"; //创建连接字符串
    SqlConnection con = new SqlConnection(str);                 //创建连接对象
    con.Open();                                                  //打开连接
    string strsql = "select * from login where username='" + TextBox1.Text + "'"; //编写 SQL 语句
    SqlDataAdapter da = new SqlDataAdapter(strsql, con);         //创建适配器
    DataSet ds = new DataSet();                                 //创建数据集
    int count = da.Fill(ds, "table");                            //填充数据集
    if (count > 0)                                               //查找用户
    {
        Label5.Text = ds.Tables["table"].Rows[0]["ask"].ToString(); //提示用户信息
        Label2.Text = "";                                          //清空错误信息
    }
    else
    {
        Label2.Text = "没有这个用户";                            //提示用户信息
    }
}

```

当用户填写用户名并失去焦点时，系统会在数据库中查询相关的用户信息，如果包括该用户，则会提示这个用户的提问信息；如果没有这个用户，则提示没有这个用户，如图 23-7 所示。

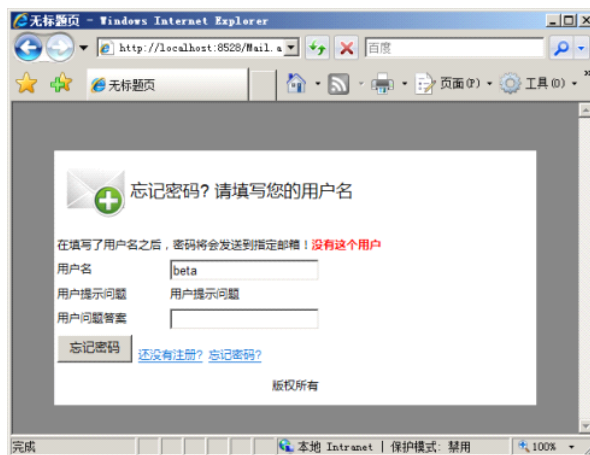


图 23-7 搜索用户信息

当用户填写完用户名和用户提示问题答案后，系统会判断用户答案是否正确，如果用户的答案是正确的，就会发送邮件到用户邮箱；如果用户答案不正确，则会提示用户再次输入答案，示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    string str = "server=(local);database='login';uid='sa';pwd='sa'"; //创建连接字符串
    SqlConnection con = new SqlConnection(str); //创建连接对象
    con.Open(); //打开连接
    string strSql = "select * from login where username='" + TextBox1.Text + "'"; //配置 SQL 语句
    SqlDataAdapter da = new SqlDataAdapter(strsql, con); //创建适配器
    DataSet ds = new DataSet(); //填充数据集
    int count = da.Fill(ds, "table"); //获取数据
    if (count > 0) //如果存在用户
    {
        if (TextBox2.Text != ds.Tables["table"].Rows[0]["answer"].ToString()) //对比问题
        {
            Label2.Text = "提示问题答案回答出错,请再次输入答案.."; //出现错误
        }
        else
        {
            SendUserMail(ds.Tables["table"].Rows[0]["email"].ToString(), //发送邮件
                ds.Tables["table"].Rows[0]["password"].ToString()); //实现邮件发送
        }
    }
    else
    {
        Label5.Text = "没有这个用户"; //声明没有用户
    }
}
```

上述代码会判断用户回答的问题是否和本身用户在注册时设置的问题相同，如果相同，就执行 SendUserMail 函数。SendUserMail 函数实现代码如下所示。

```
private bool SendUserMail(string recevie, string password)
{
    try
```

```

    {
        System.Net.Mail.SmtpClient client = new System.Net.Mail.SmtpClient();
        client.Host = "SMTP 服务器"; //SMTP 服务器信息
        client.UseDefaultCredentials = false;
        client.EnableSsl = false;
        client.Credentials = new System.Net.NetworkCredential("邮件发送邮箱", "发送邮箱密码");
        client.DeliveryMethod = System.Net.Mail.SmtpDeliveryMethod.Network;
        System.Net.Mail.MailMessage message =
        new System.Net.Mail.MailMessage("邮件发送邮箱", recevie);
        message.Subject = "获取密码信息"; //邮件的标题
        message.Body = "您的密码为:"+password; //邮件的密码
        message.BodyEncoding = System.Text.Encoding.UTF8; //邮件的编码形式
        message.IsBodyHtml = true; //邮件内容的形式
        try
        {
            client.Send(message); //发送邮件
            return true; //返回真
        }
        catch (Exception ex) //抛出异常
        {
            return false; //返回假
        }
    }
    catch //不存在邮件服务器
    {
        return false; //返回假
    }
}

```

上述代码实现了邮件的发送功能，这个函数的参数为接受者的地址和密码。在使用这个函数时，SMTP 服务器信息可以编写服务器所需要的邮件服务器 SMTP 服务器，例如 126 邮箱的 SMTP 服务器就是 SMTP.126.COM。编写完成 SMTP 服务器之后，只需要在上述代码中编写发送邮箱地址和发送邮箱密码就能够实现邮箱的发送。

注意：如果邮箱需要使用 SSL 安全证书才能够进行发送和接受，那么就需要配置 System.Net.Mail.SmtpClient 对象的 EnableSsl 为 true。例如 Gmail 就需要配置 EnableSsl。

23.5.3 根据不同的用户显示不同的内容

为了方便不同的用户显示不同的内容，可以使用 ASP.NET 提供的内置对象进行编程和判断，例如在登录时，如果登录成功，则系统会为用户配置一个 Session 内置对象。Session 内置对象寄宿在用户浏览器进程内，如果用户浏览器进程关闭或者用户长时间没有操作，则 Session 内置对象就会注销。

在 Session 生命周期内，可以使用 Session 内置对象对不同的用户进行页面编程，这样就能够实现不同的用户显示不同的内容。例如当用户登录后，会跳转到一个个人界面，这个界面可能是通用界面，但是需要不同的用户在当前界面操作。创建一个个人界面，当不同的用户访问该界面时显示的效果也不同，该页面为 logined.aspx，示例代码见光盘源代码\第 23 章\23-1\23-1\Logined.aspx 所示。

其中，代码在页面中添加了一个 Label 控件和一个 Image 控件，这两个控件分别对不同的用户呈现不同的效果。例如当用户“soundbbg”登录后，则系统应该提示说“感谢您 soundbbg 的登录”，而如果

是用户“wujunmin”登录，则系统应该提示“感谢您 wujunmin 的登录”而不是原来的提示信息，同时也可以为相应的用户显示不同的用户头像，示例代码如下所示。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (String.IsNullOrEmpty(Session["name"].ToString()))           //判断 Session
    {
        Response.Redirect("default.aspx");                          //页面跳转
    }
    else
    {
        Label1.Text = Session["name"].ToString();                  //获取 Session
        if (Session["name"].ToString() == "guojing")                //执行编程
        {
            Image1.ImageUrl = "mail.png";                          //获取图像
        }
    }
}
```

上述代码可以使用 Session 对象进行判断和编程，如果 Session 对象为空，则说明用户并没有登录或者用户为非法用户，那么就必须跳转到登录页面进行登录。如果用户是合法用户，并且用户是一些例如 VIP 等用户，就需要对特定的用户进行编程以呈现不同的样式。

在用户登录后，不仅可以使 Session 对象进行用户信息和权限的判断，也可以使用 Cookie 对象对用户信息和权限进行判断。使用 Session 对象和 Cookie 对象能够非常方便的进行用户信息的获取和存储。网站应用中，使用 Session 对象和 Cookie 对象是最常用的用户信息的获取和存储的方法，开发人员能够根据不同的使用 Session 对象和 Cookie 对象的值进行相应的编程。

23.6 实例演示

编写完成后就能够运行模块，这个模块包含登录、用户密码发送和用户个人页面，运行后用户登录页面如下图 23-8 所示。

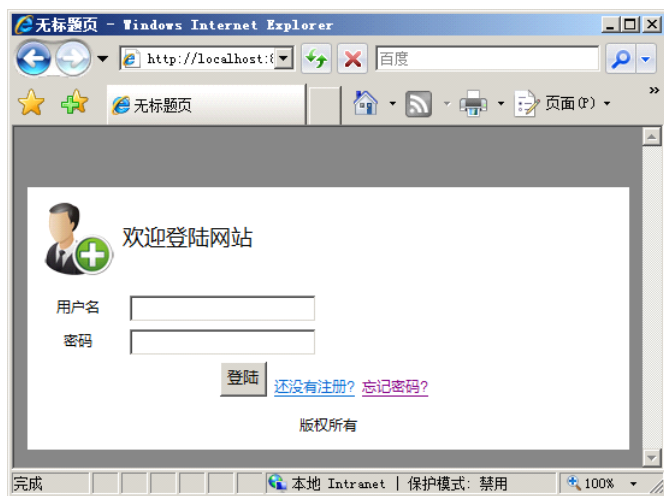


图 23-8 登录页面

当用户进行登录操作后，计数器就会开始计数，如果用户登录失败，用户还有3次机会能够再次进行登录。如果用户多次登录都没有成功，这就说明用户可能忘记了密码或者用户是一个非法用户，对于这样的用户必须禁止再次登录，如图 23-9 所示。

如果用户登录成功，则可以在相同的页面进行不同的用户信息的呈现，如果用户是 VIP 用户或者某个特殊的用户，就可以使用编程的方法为用户进行相应的页面编程，如图 23-10 所示，其中用户登录成功，登录成功后的用户头像已经被程序编制成 mail.png，否则是默认的头像显示。

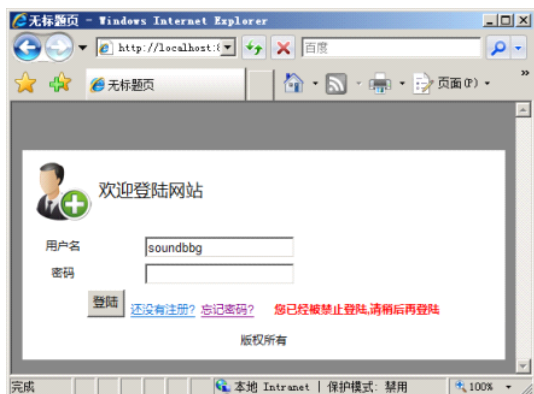


图 23-9 禁止再次登录



图 23-10 用户登录成功

如果用户多次登录都出现错误被禁止登录，用户可以单击忘记密码来让系统发送密码到用户邮箱中，在向用户邮箱中发送邮件前，必须要判断用户是否为合法用户，否则任何人都能够发送邮件到相应的邮箱中，如图 23-11 和图 23-12 所示。



图 23-11 不存在用户

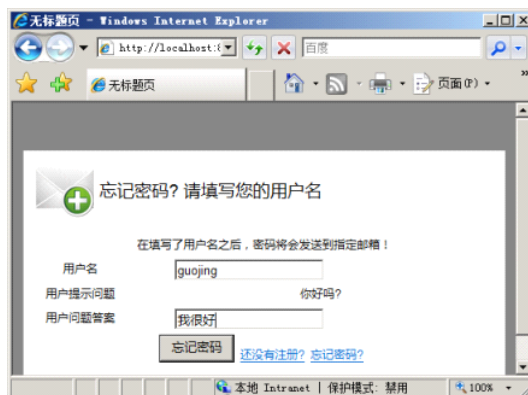


图 23-12 存在一个用户

在发送邮件到一个用户邮箱之前，必须要检测是否包含这个用户，正如图 23-11 所示，如果不存在这个用户，那么系统就会提示不存在该用户。如果存在这个用户，则系统会提示一个用户提示问题，如图 23-12 所示，其中就提示了问题“你好吗”。用户必须在问题答案文本框中输入答案信息，如果答案正确的话就能够发送邮件到相应的邮箱，如果答案不正确就会提示用户答案不正确，需要再次填写答案并确认，如图 23-13 所示。

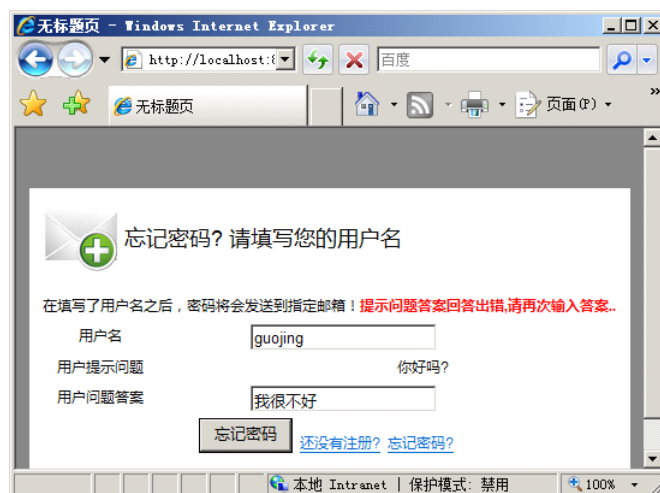


图 23-13 提示用户答案错误

如果用户答案错误，当用户单击【忘记密码】按钮时，就需要提示用户回答的答案错误，如果用户答案正确，就可以发送相应的邮件到用户的用户信箱中，用户可以通过查看用户信箱获取自己的密码。登录成功的用户，系统会跳转到 `logged.aspx`，用户可以在该页面进行相应的操作，开发人员可以在该页面进行不同用户权限的获取和判断甚至是为某个或某些用户进行编程，这样就能够方便的利用 ASP.NET 提供的内置对象对用户进行编程操作。

23.7 小结

本章通过编程的方法实现了登录模块的编写，登录模块通常包括用户登录页面、忘记密码页面以及个人信息页面。登录模块通常情况下是和注册模块一起使用的，因为登录模块所需要的数据库也是注册模块所需要使用的数据库，当用户注册后，注册信息会存放在数据库中，登录模块只是在数据库中索取相应的信息，而不会对其中的信息进行操作。本章还巩固了：

- ❑ ASP.NET 的网页代码模型。
- ❑ Web 窗体基本控件。
- ❑ 数据库基础。
- ❑ ADO.NET 常用对象。
- ❑ Web 窗体数据控件。
- ❑ ASP.NET 内置对象。

通过本章的学习能够巩固和强化对本书中的这些章节的理解。