第 27 章　聊天模块设计

在一些网站应用中，常常需要在线聊天模块的设计，例如网站的客户服务。聊天模块的设计往往不需要使用到数据库，因为在线聊天是一种即时的行为，当用户离开在线聊天页面时，可能无需进行用户信息的保存。

27.1　学习要点

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

- Web 窗体基本控件。
- Web 窗体数据控件。
- ASP.NET 内置对象。
- 生成静态的概念
- 自定义控件和用户控件。
- ASP.NET 3.5 与 AJAX

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。

27.2　系统设计

聊天模块的作用在于能够实现在线同网站的其他人员聊天。聊天不仅可以与网站的其他用户进行聊天，也可以同网站的客服人员进行聊天。所以聊天模块不仅仅局限于单个用户和单个用户的聊天，可以是单个用户和单个用户，也可能是单个用户对多个用户。

27.2.1　模块功能描述

聊天模块能够在一定程度上加强用户与用户之间的信息沟通，另外聊天模块也能够加强客户与用户之间的交互，对用户的信息进行及时的反馈。聊天模块的功能并没有复杂的模块和过多的页面，对于聊天模块而言，只需要进行页面中的数据处理即可，也无需保存数据。

当用户选择进行在线聊天时，用户需要进行登陆操作，这个登陆操作无需从数据库中进行提取和验证，这里的用户登陆操作只是给用户一个名称，以便于在页面聊天中方便被识别。当用户登陆完成后就能够进入主登陆窗口进行聊天操作。用户能够同多个人进行群聊也可以同单个用户进行私聊，当用户进行群聊时，群聊的信息会发布到相应的群聊文本框中；而当用户进行私聊时，其信息并不会发布到相应的群聊文本框，而发布到私聊文本框。

对于聊天的用户而言，可以选择是否将聊天记录进行保存，如果将聊天记录进行保存，系统可以将用户的相应的聊天记录保存为 txt 文档存放在用户的个人电脑中，以保存相应的数据而无需进行数据库

的数据存储。同时，为了能够提高用户体验，对于聊天模块而言可以使用 AJAX 进行无刷新实现，聊天模块用户流程图如图 27-1 所示。

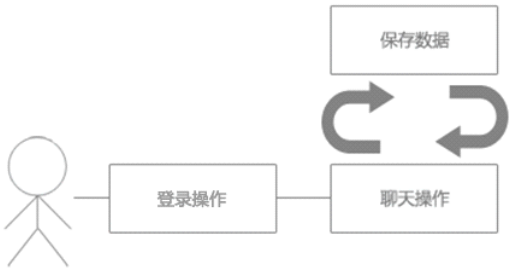


图 27-1　聊天模块功能描述

正如图 27-1 所示，用户在进行聊天操作之前首先需要进行登陆操作，在进行登陆操作后，系统会为用户分配一个 Session 对象用户描述用户的基本信息。当用户进行聊天时，其 Session 对象的相应值能够描述用户信息。

当用户执行完成聊天操作之后，可以选择是否保存数据并继续进行聊天。如果用户选择保存数据，用户的聊天信息全都保存为 txt 文档存放在本地，如果用户并不希望进行数据保存，可以继续聊天或直接关闭浏览器。从上述模块功能描述中可以规划成以下几个页面：

- 登陆操作页面：用于用户的登陆操作。
- 选择聊天室：用户选择喜爱的聊天室分类。
- 聊天操作页面：用于用户的聊天操作，可以进行单人聊天或多人聊天。

其中登陆操作主要是用于用户的登陆和 Session 对象的分配，而聊天操作页面就是一个在线聊天的主页面。聊天模块中并没有涉及到数据库的存储，而更多的是编程的技巧，聊天模块没有数据读取也就不存在数据库设计和性能了。

27.2.2　模块流程分析

聊天模块并不是网站应用中非常重要的模块，但是在很多业务情况下聊天模块也是非常必要的模块。特别是在用户咨询等情况下作为网站的客户服务就非常需要及时的信息获取和反馈，聊天模块能够非常好的完成这项任务。

在聊天模块的开发过程中，聊天模块并不涉及到数据的存储和读取，所以聊天模块属于即时模块，也就是说当用户关闭了浏览器之后除非用户保存聊天记录到文件中，否则用户的大部分信息全部都会丢失。当用户需要进行数据的存储或管理，可以选择保存聊天记录以保存操作的数据，操作数据将以 txt 文本文档的形式存储在用户的个人电脑中。从以上的流程分析中可以归纳出大部分用户在聊天模块中执行的操作的顺序，在流程分析中可以归纳如下：

- 用户登陆：用户登陆并进行信息初始化。
- 访问聊天页面：用户访问相应的聊天页面进行聊天操作。
- 页面初始化：聊天页面能够初始化用户信息，包括有多少个用户在此聊天室。
- 执行聊天操作：用户和一个用户或多个用户进行聊天操作。
- 存储聊天信息：用户可以选择存储聊天信息或直接离开。
- 离开聊天室：离开聊天室后需要刷新聊天室信息。

上述流程如图 27-2 所示。

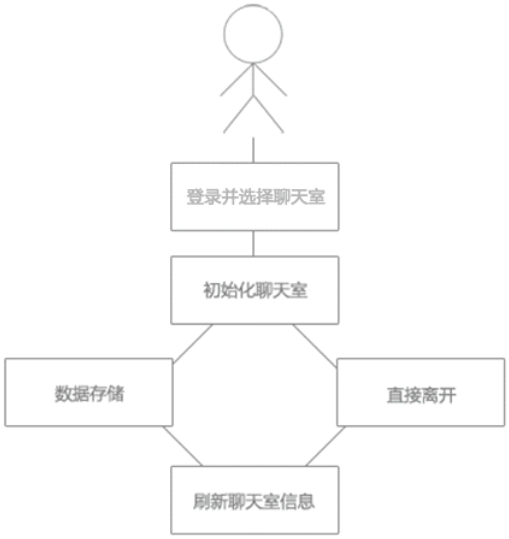


图 27-2　聊天操作流程图

在用户进入聊天室之前，首先需要初始化聊天室信息，例如已经在此页面的用户数量和用户的遍历。当用户离开聊天室时，需要刷新聊天室信息，因为一个用户离开了聊天室，那么聊天室信息中的用户数量和用户名称都已经被改变。无论聊天室中用户数量是增加还是减少，都需要进行数据的刷新。

27.3　界面设计

聊天室是用户与用户之间信息沟通的页面，聊天室的界面设计能够加强用户的粘度以便用户的再次回访。提高用户体验能够让用户在聊天应用中感觉到舒适，从而提高网站的口碑让更多的用户参与到在线聊天中。

27.3.1　登陆界面设计

用户能够在登陆界面进行登陆和聊天室的选择，在登陆操作和聊天室选择中可以使用 ASP.NET 3.5 AJAX 进行无刷新验证，登陆界面设计核心代码见光盘中源代码\第 27 章\27-1\27-1\login.aspx。

其中，代码使用了 TextBox 控件用于用户的昵称编写，用户必须填写昵称进行聊天，昵称是用户的一个标识，当用户进入聊天页面进行聊天时昵称就能够显示相应的用户信息，以便聊天中用户身份的区别。上述代码还使用 RadioButtonList 控件进行聊天室的选择，用户在填写昵称后需要选择相应的聊天室进行聊天。如果用户不选择聊天室同样不能够进行聊天，只有选择了相应的聊天室才能够聊天，例如用户可以选择“谈天说地”聊天室进行聊天，在该聊天室中的所有用户都是基于“谈天说地”这个话题进行聊天的。上述代码使用了 AJAX 进行无刷新效果实现，其布局如图 27-3 所示。

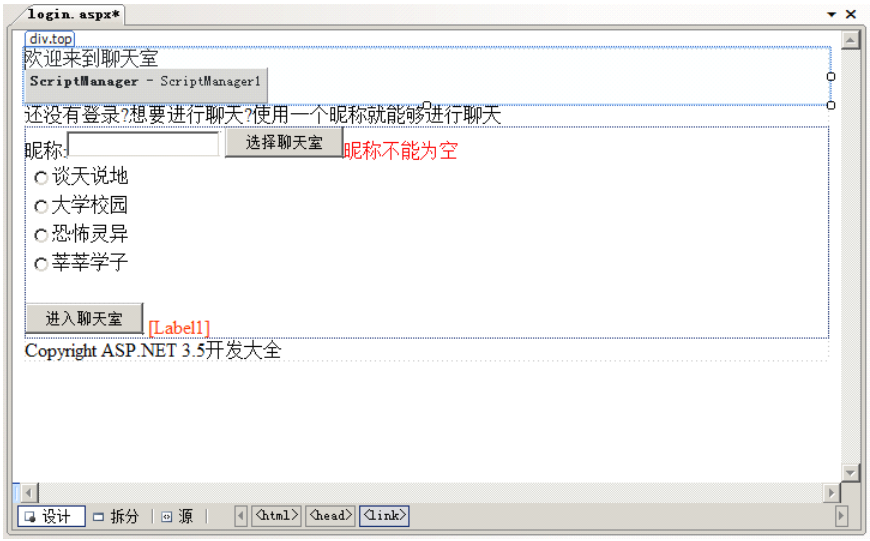


图 27-3　聊天登陆页面基本布局

在页面被初始化时，其中的 RadioButtonList 控件和进入聊天室按钮是不会显示的，当用户填写了昵称并单击选择聊天室才能够显示 RadioButtonList 控件和进入聊天室按钮。如图 27-3 所示，其登陆页面的友好度显然不够，可以使用 CSS 进行样式控制让该页面更加友好和丰富。

27.3.2　登陆界面 CSS

如上一节中的登陆界面可以看出登陆界面并不够友好，友好的登陆界面能够让用户喜欢这个网站并长期进行访问，这样就能够提高网站的访问量和提高用户的粘度。为了让页面的友好度更高可以使用 CSS 进行样式控制，CSS 代码如下所示。

```
body                                                    //控制全局样式
{
    font-size:12px;
    font-family:Geneva, Arial, Helvetica, sans-serif;
    margin:0px 0px 0px 0px;
    background:white url(images/background.gif);
}
.all                                                    //控制登陆框样式
{
    margin:50px auto;
    width:520px;
    background:white;
    border:1px solid #ccc;
}
.top                                                    //控制头部样式
{
    margin:0px auto;
    width:500px;
    padding:10px 10px 10px 10px;
    background:white url(images/bg.png) repeat-x;
}
.center                                                 //控制登陆样式
```

```
{
    margin:0px auto;
    width:500px;
    padding:10px 10px 10px 10px;
}
.end                                                                    //控制底部样式
{
    margin:0px auto;
    width:500px;
    padding:10px 10px 10px 10px;
}
```

上述 CSS 分别为登陆页面进行了样式控制，其中定义了全局样式并定义了页面的字体大小，定义了全局样式后为其他的 DIV 层定义了样式，包括外对齐、宽度和内对齐等。在 CSS 文件中使用了图片让页面看上去更加友好，良好的背景图片和导航的图片的应用能够提升网站的设计效果，CSS 样式控制后的页面效果如图 27-4 所示。

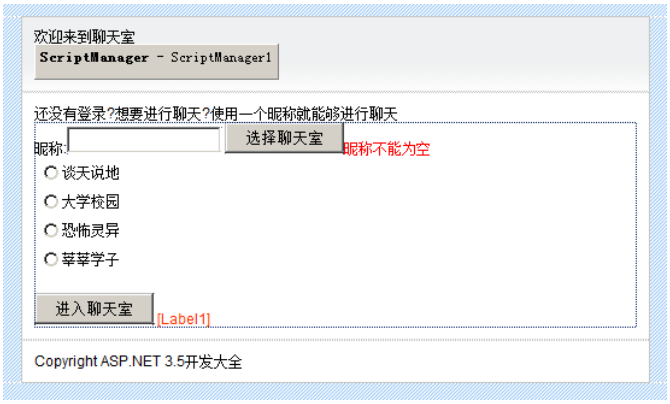


图 27-4　布局后的效果

通过 CSS 进行样式布局后的页面效果明显好很多，当用户访问该页面时会感觉到页面设计的友好度，提高了用户体验，增强了网站对用户的粘度和可信度。

27.3.3　聊天室显示界面

聊天室主窗口包含一些常用的窗口，这些窗口用于呈现相应的文本，包括用户的聊天发布窗口和用户的对话窗口。在聊天室面板中还包含用户列表，这些列表能够为用户提供私聊服务，聊天室显示界面核心代码见光盘中源代码\第 27 章\27-1\27-1\room.aspx。

在页面中的上部分代码实现的是群聊窗口，当用户不指定私聊对象时，其发布的聊天信息将会呈现在群聊窗口。如果用户希望与某个用户进行私聊，就需要使用私聊窗口，示例代码见光盘中源代码\第 27 章\27-1\27-1\room.aspx 中私聊窗口所示。

聊天窗口中为了防止页面的重复刷新，可以使用 AJAX 控件实现无刷新功能。为了能够让页面在指定的时间内进行局部刷新，就需要使用 Timer 控件进行实现，示例代码如下所示。

```
<asp:Timer ID="Timer1" runat="server" Interval="10000" ontick="Timer1_Tick">
</asp:Timer>
```

上述代码实现了聊天页面中的主窗口，其中主窗口包括群聊窗口、私聊窗口、发言窗口和发言人窗口，当用户单击其中的按钮控件进行聊天时，会根据其中的发言窗口和发言人窗口在群聊窗口和私聊窗

中显示相应的数据，如图 27-5 所示。

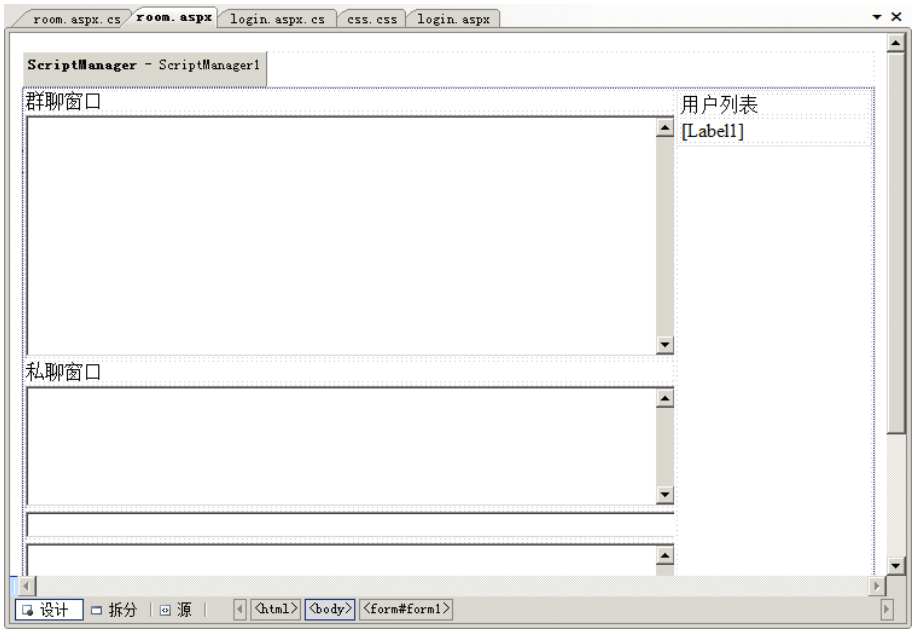


图 27-5　聊天页面窗口

在聊天页面的右侧有一个用户列表，当加载该页面时会初始化页面信息并载入用户列表，当用多个用户时用户列表就会呈现为多个用户，用户可以单击用户列表与相应的用户进行聊天。

27.3.4　聊天室界面 CSS

同样该聊天室窗口不太人性化也没有任何的用户体验，使用 CSS 能够提高用户体验，不同的页面的 CSS 都可以放置在同一个 CSS 文件中，这些 CSS 文件能够被单个或多个页面使用，减少了冗余代码，CSS 代码如下所示。

```
.room
{
    margin:10px auto;
    width:800px;
    background:white;
    border:1px solid #ccc;
}
.banner
{
    width: 536px;
    background:white url(images/bg.png) repeat-x;
}
```

由于两个页面使用的是同一个 CSS 文件所以很多样式控制可以无需再次编写，只需要对该页面中需要使用的样式进行样式编写，聊天室界面需要控制其主聊天窗口的长度和宽度，为了提高用户友好度，也可以使用图片进行样式控制，如图 27-6 所示。

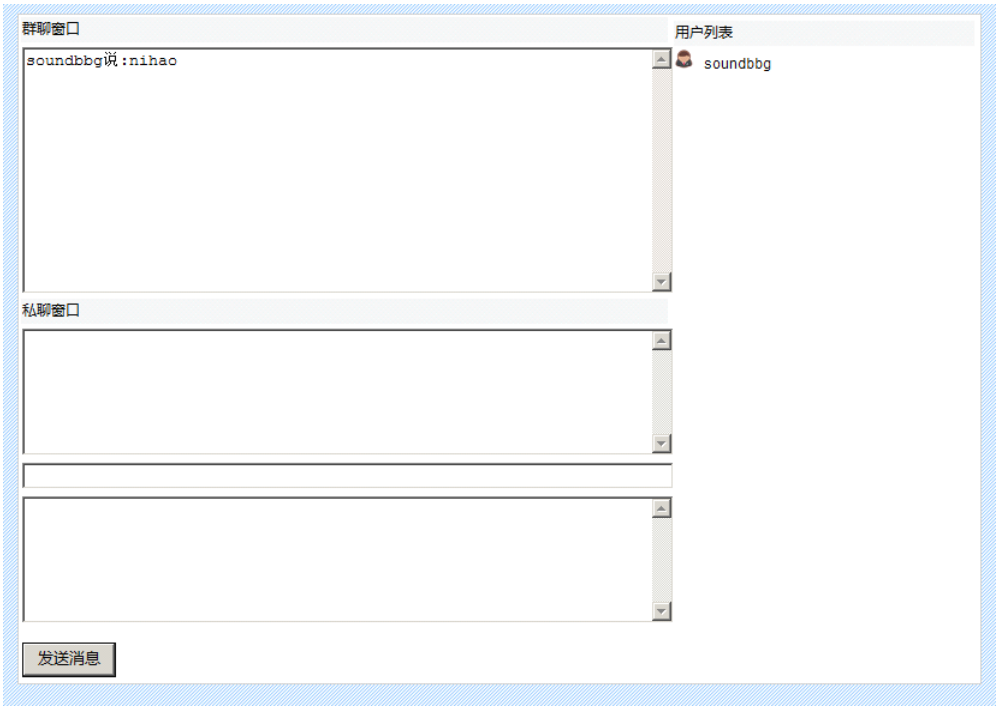


图 27-6　布局后的聊天窗口

正如图 27-6 所示，在使用 CSS 进行样式控制后的页面具有更好的友好度，在用户列表区域，当有多个用户时会呈现多个不同的用户列表。在用户列表中用户可以选择相应的用户并与用户发送私密消息，当用户发送私密消息时，其消息不会呈现在群聊窗口，而是呈现在私聊窗口。

27.4　代码实现

聊天模块不需要进行数据存储和读取，在聊天模块中，当用户发送一个信息到页面中需要进行聊天时，在相应的窗口就应该显示用户发布的信息。对于不同的用户而言，不同的用户所看到的页面是不同的，这里必须使用 Application 对象进行跨页存储。

27.4.1　登陆代码实现

当用户进行页面访问时，页面中的一些控件初始化是无法看见的，只有当用户填写了用户名昵称并选择了相应的聊天室才能够进行登陆操作，在用户选择聊天室之前，必须填写用户名。用户填写用户名之后就能够单击【选择聊天室】按钮进入聊天室。示例代码如下所示。

```
protected void Button1_Click(object sender, EventArgs e)
{
    RadioButtonList1.Visible = true;                    //显示控件
    Button2.Visible = true;                             //显示控件
    TextBox1.ReadOnly = true;                           //锁定用户名
}
```

当用户填写了用户名之后并单击选择聊天室按钮进入聊天室选择，在选择聊天室时就不能够再修改用户名，因为一旦用户确定了用户名并进行聊天室选择时就无法再修改自己的用户名。在这里其实也可

以让用户能够修改用户名，只是这样做会造成网站应用的不安全。当用户进行聊天室选择后就可以进入聊天室，进入聊天室按钮代码实现如下所示。

```
protected void Button2_Click(object sender, EventArgs e)
{
    if (RadioButtonList1.SelectedIndex == -1)                                    //判断单选列表
    {
        Label1.Text = "请选择一个聊天室";                                          //提示选择
    }
    else
    {
        Session["roomid"] = RadioButtonList1.SelectedItem.Value;                 //赋予 Session 值
        Session["username"] = TextBox1.Text;
        Response.Redirect("room.aspx?id="+ RadioButtonList1.SelectedItem.Value +""); //跳转
    }
}
```

上述代码当用户单击选择聊天室按钮后就能够进行聊天室的选择，用户必须选择一个自己感兴趣的聊天室进行聊天，否则系统会提示“请选择一个聊天室”。如果用户选择了聊天室并进行登陆，系统会为用户赋予两个 Session 值，这两个 Session 值分别为 roomid 和 username，其中 roomid 为聊天室的 ID 号，而 username 用于存储进行聊天的用户名。当聊天页面被载入时，用户的 Session 对象的 roomid 值会与传递的参数进行判断，如果是相应的聊天室就允许用户进行聊天，如果不是相应的聊天室则不允许用户聊天。

27.4.2 多人聊天代码实现

多人聊天相对比较简单，用户可以在页面中直接进行信息输入发布聊天信息。多人聊天时，用户发布的信息能够被所有人看见，这也就是说用户的信息能够呈现在多人聊天窗口。当多人聊天时，不同的用户所打开的页面是不相同的，这样就造成可能信息呈现的时间不一致，为了保证信息的一致性，这里使用了 AJAX 的 Time 控件进行刷新。

在聊天页面加载时，首先需要判断用户是否包含 Session 值或者聊天室是否为用户选择的聊天室，否则会跳回登陆页面，不仅如此，当页面被初次加载时还需要进行一些初始化工作，示例代码如下所示 。

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["roomid"] == null || Session["username"] == null)                //判断是否登陆
    {
        Response.Redirect("login.aspx");                                         //页面跳转
    }
    if (Request.QueryString["id"] != Session["roomid"].ToString())               //判断是否匹配
    {
        Response.Redirect("login.aspx");
    }
    Label1.Text = "";                                                            //清空控件的值
    for(int i=0;i<Session.Count/2;i++)                                           //添加用户列表
    {
        if (Session[i * 2] == Session["roomid"])                                 //配置 Session
        {
            Label1.Text += "<img src=\"images/p.png\">   "+(Session[i * 2 + 1] + "<br/>");
```

```
            }
        }
        if (Application["char"] != null)                                    //初始化聊天信息
        {
            TextBox3.Text = Application["char"].ToString();                 //获取 Application
        }
    }
```

上述代码在页面加载时被执行，当用户访问页面时，首先会对用户的身份进行判断，判断用户是否已经登陆，如果用户没有登陆则跳转到登陆页面进行登陆。如果用户已经登陆，还需要判断用户登陆是否所属对应的房间，如果不是对应的房间则会被认为是非法进入房间，需要重新进行登陆操作。当用户身份验证通过后，就需要进行初始化操作清除相应的控件的值并循环添加用户列表项，在添加用户列表时，需要遍历 Session 对象的值进行列表的初始化，示例代码如下所示。

```
        for(int i=0;i<Session.Count/2;i++)
        {
            if (Session[i * 2] == Session["roomid"])                        //判断 Session
            {
                Label1.Text += "<img src=\"images/p.png\">   "+(Session[i * 2 + 1] + "<br/>");
            }
        }
```

在用户登陆后，会给用户分配两个 Session 对象，一个用户记录聊天室的 ID，另一个用于记录用户名。这也就是说当遍历 Session 时，会有多个 Session 对象，即一个用户有 2 个 Session 对象，当遍历用户列表时需要筛选这些 Session 值，去掉聊天室 ID 的 Session 的值而呈现用户的 Session 值。在用户登陆完成后，在用户之前已经有很多人进行聊天了，在用户进入聊天室后，需要加载这些聊天信息，示例代码如下所示。

```
        if (Application["char"] != null)
        {
            TextBox3.Text = Application["char"].ToString();                 //加载聊天信息
        }
```

上述代码使用了 Application 对象进行跨页的数值存储，Application 对象是页面中的公共对象，就算是不同的用户之间也能够共享Application对象，在页面进行聊天信息发布时，可以将值添加到Application对象中被其他用户读取。当用户单击按钮控件进行消息发布时，需要在页面中的相应位置进行呈现，在多人聊天代码实现中，可以直接将信息内容增加到文本框中，示例代码如下所示。

```
            TextBox3.Text += Session["username"] + "说:" + TextBox2.Text + "\n";
            TextBox2.Text = "";                                             //清空窗口
            Application["char"] = TextBox3.Text;                            //添加公共对象
```

当用户进行消息发布时，其中多人聊天窗口的信息要增加刚才用户发布的信息，如上述代码所示，其中多人聊天窗口 TextBox3 的信息增加了“XX 说：…”的字符串。由于页面使用了 AJAX 控件进行无刷新的实现，用户基本看不出来页面被刷新。当用户单击按钮控件时就能够进行局部刷新，实现多人聊天。

单个用户进行聊天信息的发布并不能被其他用户阅读，当两个人打开一个页面，其中一个人进行的操作不会呈现给另一个人。在这里不仅要使用 Application 对象还需要进行页面刷新，这里使用了 AJAX 控件中的 Timer 控件实现，当 Timer 控件执行更新时，页面中的对话框的数据也会进行更新，示例代码如下所示。

```
    protected void Timer1_Tick(object sender, EventArgs e)
    {
```

```
        TextBox3.Text = Application["char"].ToString();                    //更新数据
    }
```

当 AJAX 中的 Timer 控件执行刷新操作后，其多人聊天窗口的文本值就会等于 Application 对象的相应值，这样在其他页面中就能够查看现有的数据。

27.4.3 单人聊天代码实现

当用户需要进行单人聊天时，其聊天的内容不能够被多人聊天窗口捕获和呈现，单人聊天的信息必须呈现在私人聊天窗口中。在聊天页面中，与多人聊天不同的是，多人聊天时无论信息是怎样发布的其信息都会呈现在多人聊天窗口中，这样也没有谁发送给谁之说。而在单人聊天时，当用户 soundbbg 发送信息给 guojing 时，就需要判断是否有这个用户并且提示用户接收信息。

多人聊天时无需考虑到用户是否需要接受，这就和广播一样，广播台无需关心用户是否在接听广播，广播台只需要负责发送，用户可以选择接收或不接收。而单人聊天时需要判断是否有这个用户，如果有这个用户，不仅在发送者的聊天窗口中需要添加字符串“你给 XX 发送了 XX 信息”，同样在接收者中需要添加“XX 给你发送了 XX 信息”。在显示信息时，需要遍历 Application 对象进行判断，实现代码如下所示。

```
        TextBox3.Text += "你对" + TextBox4.Text + "说:" + TextBox2.Text + "\n";    //输出聊天记录
        TextBox1.Text += "你对" + TextBox4.Text + "说:" + TextBox2.Text + "\n";    //输出聊天记录
        Application[Session["username"].ToString()] = TextBox1.Text;               //增加 Application 对象
        for (int i = 0; i < Application.Count; i++)                                //遍历 Application 对象
        {
            if (Application[TextBox4.Text] != null)                                //判断 Application
            {
                Application[TextBox4.Text] += TextBox4.Text + "对你" + TextBox4.Text + "说:"
                + TextBox2.Text + "\n";                                            //增加相应聊天记录
            }
        }
        TextBox2.Text = "";                                                        //清空窗口
```

上述代码在用户发送相应的信息后，系统会遍历 Application 对象查找是否有这个用户，如果没有这个用户则不在相应的用户信息框中呈现信息，如果存在这个用户，则在该用户的 Application 对象中添加字符串“XX 对你说 XX 信息”。

但是上述代码有一定的缺陷，就是当用户和用户之间发送私密信息时，同样还是会将信息发送到群聊文本框中，这就需要进行判断。如果没有填写相应的用户名，则说明这个信息是一个群发信息，进行广播，在群聊文本框中就会显示该信息，如果填写了用户名，则说明这个信息是一个私聊信息，就不会进行广播，示例代码如下所示。

```
    protected void Button1_Click(object sender, EventArgs e)
    {
        if (String.IsNullOrEmpty(TextBox4.Text))                                   //判断是是否群发
        {
            TextBox3.Text += Session["username"] + "说:" + TextBox2.Text + "\n";    //群发窗口添加记录
            TextBox2.Text = "";                                                    //清空窗口
            Application["char"] = TextBox3.Text;                                   //更新 Application 对象
        }
        else
        {
            //TextBox3.Text += "你对" + TextBox4.Text + "说:" + TextBox2.Text + "\n";
```

```
            TextBox1.Text += "你对" + TextBox4.Text + "说:" + TextBox2.Text + "\n";
            Application[Session["username"].ToString()] = TextBox1.Text;          //获取 Application 对象
            for (int i = 0; i < Application.Count; i++)                           //遍历 Application 对象
            {
                if (Application[TextBox4.Text] != null)                           //查找 Application 对象
                {
                    Application[TextBox4.Text] += TextBox4.Text + "对你" +
                            TextBox4.Text + "说:" + TextBox2.Text + "\n";          //修改相应用户记录
                }
            }
            TextBox2.Text = "";                                                   //清空窗口
        }
    }
```

上述代码在发送信息前，首先会判断是否是群发，如果是群发则不进行其他的任何操作，直接进行文本框中的数据的呈现，并将相应的值添加到公共对象 Application["char"]中去。如果用户进行的是私聊，那么就需要遍历 Application 对象进行信息的发布，发布信息的同时不仅要修改发布者的 Application 对象，同样需要修改接收者的 Application 对象，当 AJAX 的 Timer 控件进行刷新时，用户就能够看到发送者的信息。

单人聊天实现的过程比多人聊天的过程更加复杂，在多人聊天时只需要将数据添加到公用对象中，而无需考虑发送者和接受者，当进行单人聊天时，不仅需要修改发送者的 Application 对象，还需要遍历 Application 对象进行接受者的 Application 对象的修改。

27.4.4 聊天记录保存实现

当用户希望保存聊天记录时，可以单击相应的文本框旁边的保存记录进行聊天记录保存。聊天记录可以保存为 txt 文本文档到用户的目录中，当用户希望查阅聊天记录时，可以打开相应的文件进行查阅，示例代码如下所示。

```
    protected void LinkButton1_Click(object sender, EventArgs e)
    {
        if (!Directory.Exists("C:\\chat\\group"))                                 //保存群聊记录
        {
            Directory.CreateDirectory("C:\\chat\\group");                         //创建路径
            //开始通过编写文本保存路径创建文件
            StreamWriter sw = File.CreateText("C:\\chat\\group\\" + DateTime.Now.Year +
                                              DateTime.Now.Month + DateTime.Now.Day +
                                              DateTime.Now.Hour + DateTime.Now.Second + ".txt");
            sw.Write(TextBox3.Text);                                              //编写文本
            sw.Close();                                                           //关闭对象
        }
        else
        {
            StreamWriter sw = File.CreateText("C:\\chat\\group\\" + DateTime.Now.Year +
                                              DateTime.Now.Month + DateTime.Now.Day +
                                              DateTime.Now.Hour + DateTime.Now.Second + ".txt");
            sw.Write(TextBox3.Text);                                              //编写文本
            sw.Close();                                                           //关闭对象
        }
```

```
        LinkButton1.Text = "已经保存";                                    //提示保存信息
    }
```

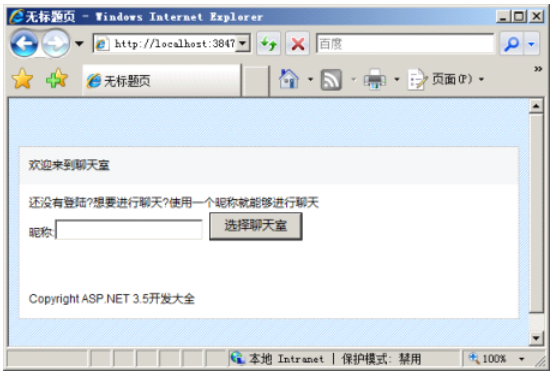
上述代码首先在用户的计算机的 C 盘中进行文件夹判断，如果存在这个目录，就直接进行 txt 文件的创建，如果不存在则首先创建相应的目录然后再进行文件的存储。用户私聊记录同样可以保存在用户计算机中，其代码实现基本相同，示例代码如下所示。

```
    protected void LinkButton2_Click(object sender, EventArgs e)
    {
        if (!Directory.Exists("C:\\chat\\pri"))                           //保存私聊记录
        {
            Directory.CreateDirectory("C:\\chat\\pri");                   //创建文件
            StreamWriter sw = File.CreateText("C:\\chat\\pri\\" + DateTime.Now.Year +
                                              DateTime.Now.Month + DateTime.Now.Day +
                                              DateTime.Now.Hour + DateTime.Now.Second + ".txt");
            sw.Write(TextBox2.Text);                                      //编写内容
            sw.Close();                                                   //关闭对象
        }
        else
        {
            StreamWriter sw = File.CreateText("C:\\chat\\pri\\" + DateTime.Now.Year +
                                              DateTime.Now.Month + DateTime.Now.Day +
                                              DateTime.Now.Hour + DateTime.Now.Second + ".txt");
            sw.Write(TextBox2.Text);                                      //编写内容
            sw.Close();                                                   //关闭对象
        }
        LinkButton1.Text = "已经保存";
    }
```

上述代码将用户的私聊记录进行保存，同样会判断目录的存在性，如果不存在目录则会创建相应的目录进行文件保存。

27.5　实例演示

聊天模块能够在网站开发过程中进行用户和用户的信息的交互，也能够进行用户和客户服务之间的交互，也能够实时的进行问题的反应和反馈，在用户进行聊天之前需要选择用户名并选择相应的聊天，如图 27-7 和图 27-8 所示。



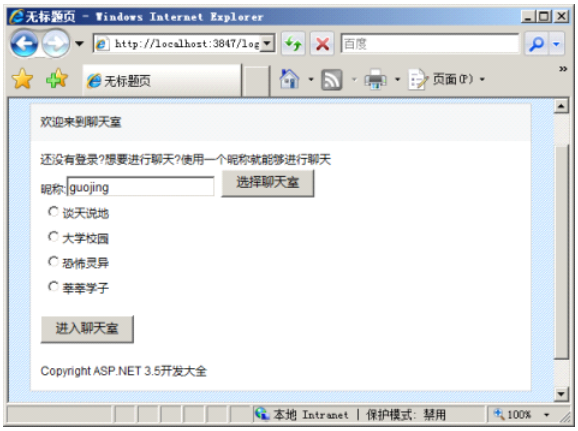


图 27-7　填写用户名　　图 27-8　选择聊天室

用户在选择聊天室之前必须输入用户名，如果不输入用户名就不能够进行聊天室选择和登陆，当用户选择了相应的用户名之后就能够选择聊天室进行聊天，当单击【进入聊天室】按钮后就能够跳转到相应的聊天室页面，如图 27-9 所示。

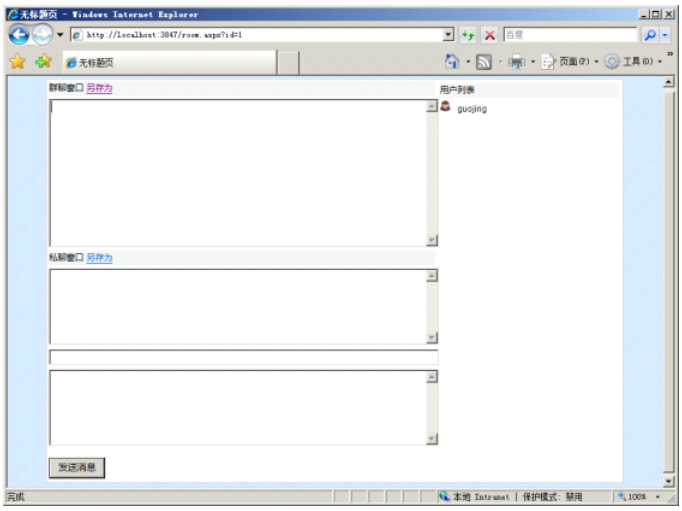


图 27-9　聊天室页面

在聊天室页面能够进行聊天，用户可以在最下面的对话框中输入信息，输入的信息将作为群发信息发送到群消息中，如果用户在用户名窗口中输入用户名则消息会发送到私聊窗口而不会发送到群发窗口，如图 27-10 和图 27-11 所示。

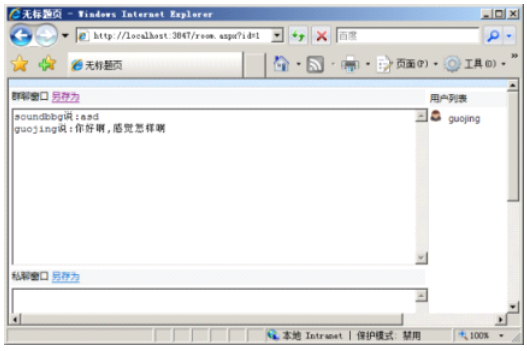


图 27-10　发送群发消息

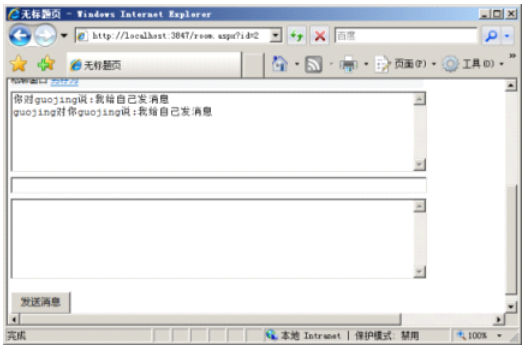


图 27-11　发送私聊信息

用户可以选择相应的用户名进行聊天，用户可以在聊天窗口中进行聊天信息的发送和接收，当用户聊天之后希望将聊天记录保存时，可以在相应的位置选择“另存为”就能够将记录保存在相应的目录下。群体记录保存在 C:\chat\group 目录下而私聊记录保存在 C:\chat\pri 目录下。这些记录都会按照时间进行保存，如图 27-12 所示。

图 27-12　生成目录

正如图 27-12 所示，用户能够将聊天记录保存在计算机的目录中也可以随时将记录删除，当用户需要使用到聊天记录时可以调出系统中的相应日期的 txt 文件就能够进行聊天记录的查看。

27.6　小结

本章通过聊天模块的开发对 ASP.NET 内置对象进行了详细的讲解。在聊天模块中，虽然没有对数据库进行存储和读取，但是需要更多的编程技巧进行 Web 应用中用户状态的保存和编程。聊天模块是网站开发中一个基本模块，但是聊天模块并不是常用的模块，虽然聊天模块并不是常用模块，但是通过聊天模块可以更加深入的了解 ASP.NET 内置对象。本章还巩固了：

- Web 窗体基本控件。
- Web 窗体数据控件。
- ASP.NET 内置对象。
- 生成静态的概念
- 自定义控件和用户控件。
- ASP.NET 3.5 与 AJAX

ASP.NET 内置对象在 ASP.NET 应用开发过程中是非常重要的，也是维持 Web 状态的一个重要的方法，虽然聊天模块在现在的网站开发中并不常用，但是也是学习 ASP.NET 内置对象的最佳方法。在进行聊天模块的开发中，本章讲解的开发方法并不是最好的方法，如果要能够异步进行跨页信息发送和接受，可以开发服务器端和客户端分别尽心信息发送和接收，由于篇幅限制和难度限制，本书不对服务器端/客户端的聊天开发方法进行详细的介绍。