单元 3　ASP. NET 内置对象与数据传递

在 ASP. NET 网站中，对 HTTP 的请求、响应及状态管理，都是通过 ASP. NET 内置对象实现的，内置对象通过向用户提供基本的请求、响应和会话等处理功能实现了 ASP. NET 的绝大多数功能。ASP. NET 中的内置对象主要包括 Page、Response、Request、Server、Cookie、Session 和 Application 等，本单元将分别对它们的使用进行介绍。

本单元主要学习目标如下。

- 熟悉 Page 对象，了解 Page 对象的生命周期和常规 Web 页面生命周期阶段。
- 掌握通过 Response 对象向页面输出信息与页面跳转。
- 掌握通过 Request 对象获取客户端信息。
- 掌握用 Session 和 Cookie 对象存储和读取数据。
- 了解 Application 对象读取全局变量。
- 了解 Server 对象字符编码。

3.1　ASP. NET 对象概述及属性方法事件

所谓对象(Object)，可以泛指日常生活中能看到的和看不到的一切事物，在程序设计中可以用一种仿真的方式来表示对象，一般的对象都有一些静态的特征，如对象的外观、大小等，这在面向对象程序(OOP)中就是对象的属性(attribute)，一般的对象如果是有生命、可以动作的，在面向对象程序中就是对象的方法(method)，所以在面向对象程序的概念中，对象有两个重点：一个是“属性”，另一个是“方法”。

一般而言，对象的定义就是每个对象都具有不同的功能与特征，不同对象属于不同的类(Class)，类定义了对象的属性、方法和事件等特征，没有类就没有对象。

(1) 属性代表对象的状态、数据和设置值。属性的设置语法如下：

```
对象名.属性名 = 语句(一般又叫属性值)
```

(2) 方法可以执行动作。方法的调用语法如下：

```
对象名.方法(参数)
```

(3) 事件的概念比较抽象，通常是一个执行的动作，也就是对象所认识的动作，事件的执行由对象触发。

ASP. NET 中的 Page、Response、Request 等对象(见表 3-1)，由. NET Framework 中封

装好的类来实现，并且由于这些对象在 ASP. NET 页面初始化请求时自动创建，所以能在程序中的任何地方直接调用，而无须对类进行实例化操作。

表 3-1　ASP. NET 常用内置对象简要说明

对象	功　能
Page	页面对象，用于整个页面的操作
Response	提供对输出流的控制，如可以向浏览器输出信息、Cookie 等
Request	提供对当前页请求的访问，其中包括请求标题、Cookie、客户端证书、查询字符串等，可以用它来读取浏览器已经发送的内容
Session	为当前用户会话提供信息，还提供对可用于存储信息的会话范围的缓存的访问，以及控制如何管理会话的方法
Application	提供对所有会话的应用程序范围的方法和事件的访问，还提供对可用于存储信息的应用程序范围的缓存的访问
Server	提供用于在页之间传输控件的实用方法，获取有关最新错误的信息，对 HTML 文本进行编码和解码，获取服务器信息等
Cookie	用于保存 Cookie 信息

下面将分别介绍这些对象的常用属性及方法。

3.2　Page 对象

在 ASP. NET 中每个页面都派生自 Page 类，并继承这个类公开的所有方法和属性。Page 类与扩展名为. aspx 的文件相关联，这些文件在运行时被编译为 Page 对象，并被缓存在服务器内存中。

3.2.1　Page 对象的常用属性

1. IsPostBack 属性

Page 对象的 IsPostBack 属性用于获取一个逻辑值，该值指示当前页面是为响应客户端回发而加载还是正在被首次加载和访问，true 表示页面是为响应客户端回发而加载，false 表示页面是首次加载。

2. Title 属性

该属性获取或设置页面的标题，可以根据需要动态更换页面标题。

3. IsValid 属性

该属性获取布尔值，用来判断网页上的验证控件是否全部验证成功，返回 true 表示全部验证成功，返回 false 表示至少有一个验证控件验证失败。

4. IsCrossPagePostBack 属性

该属性判断页面是否使用跨页提交，它是一个布尔值的属性。

Page 类中很多属性是对象的引用，即 Server、Response、Request、Application 和 Session 等都是 Page 对象的属性，这样在页面中可以直接对这些对象进行访问，而无须通过 Page 对象，例如下面两行代码的作用是一样的。

```
Page.Response.Redirect("Default.aspx");
Response.Redirect("Default.aspx");
```


上述代码第一行通过 Page 对象的 Response 属性得到 Response 对象的引用,第二行直接通过 Response 对象名对 Response 对象进行引用。

3.2.2 Page 对象的常用方法

1. DataBind 方法

该方法将数据源绑定到被调用的服务器控件及其所有子控件。

2. FindControl(ID)方法

该方法在页面中搜索带指定标识符的服务器控件。

3. ParseControl(content)方法

该方法将 content 指定的字符串解释成控件,例如以下示例。

```
Control c = ParseControl("< asp:button text = 'Click here!' runat = 'server'/>");
```

4. MapPath(virtualPath)方法

该方法将 virtualPath 指定的虚拟路径转换成物理路径。下面的示例使用 MapPath 方法获得子文件夹的物理路径,然后用此信息来设置 TextBox Web 服务器控件的 Text 属性。

```
string fileNameString = this.MapPath(subFolder.Text);
fileNameString += "\\" + fileNameTextBox.Text;。
```

3.2.3 Page 对象的常用事件

1. Page_Init 事件

当网页初始化时会触发此事件,在 ASP.NET 页面被请求时 Init 是页面第一个被触发的事件。

2. Page_Load 事件

当页面被载入时会触发此事件,即当服务器控件加载到 Page 对象中时发生。

3. Page_Unload 事件

当页面完成处理且信息被写入客户端后会触发此事件。

【示例 3-1】 对 Page_Init 事件和 Page_Load 事件进行比较。

(1) 创建页面文件 Default.aspx,在页面中添加两个列表框控件和一个按钮,代码如下:

```
< body >
    < form id = "form1" runat = "server">
            < div >
                Init 事件的运行效果      Load 事件的运行效果< br />
                  < asp: ListBox ID = "ListBox1" runat = "server" Width = "176px" > </asp:
ListBox >
```

```
                    
        <asp:ListBox ID="ListBox2" runat="server" Width="176px"></asp:ListBox>
                <br />
                <asp:Button ID="Button1" runat="server" Text="引起回发" />
            </div>
    </form>
</body>
```

(2) 在 Default.aspx.cs 中添加 Page_Init 事件和 Page_Load 事件过程代码,如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    ListBox2.Items.Add("页面被加载一次");
}
protected void Page_Init(object sender, EventArgs e)
{
    ListBox1.Items.Add("页面被加载一次");
}
```

说明:Button 按钮只是为了引起服务器回发,不用编写其 Click 事件过程代码。

(3) 按 Ctrl+F5 组合键运行页面,页面首次加载后的运行效果如图 3-1 所示。

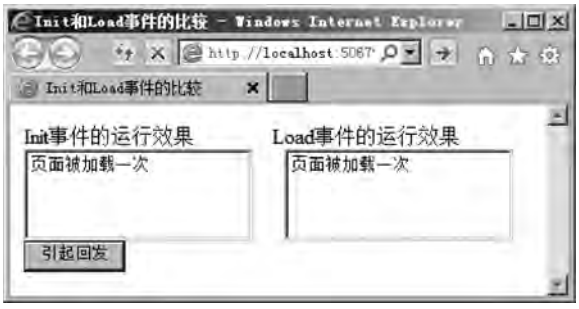

图 3-1 页面首次加载后的运行效果

单击“引起回发”按钮后,由 Page_Init 事件添加的 ListBox1 控件中的内容不会发生变化,而由 Page_Load 事件添加的 ListBox2 控件中的内容发生变化,运行效果如图 3-2 所示。

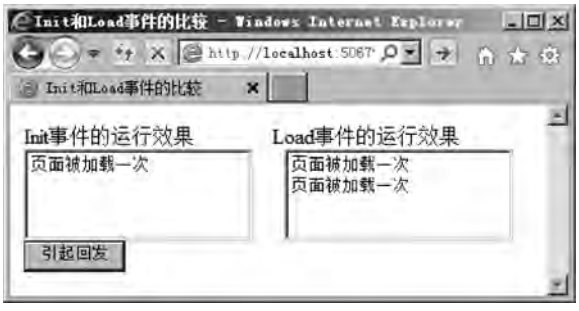

图 3-2 页面回发后的运行效果

从本示例中可以看出,Page 对象的 Init 和 Load 事件均在页面加载过程中发生,但在 Page 对象的生命周期中,Init 事件只在页面初始化时触发一次,Load 事件在初次加载及每次回发时都会触发。Page 对象的事件贯彻于页面执行的整个生命过程,每个阶段,ASP.NET 都触发了可以在代码中处理的事件,对于大多数情况,只需关心 Page_Load 事件,即:Page_Load(object sender, EventArgs e)。该事件的两个参数是由 ASP.NET 定义的,第一个参数定义了产生事件的对象,第二个参数是传递给事件的详细信息。每次触发服务器控件时,页面都会去执行一次 Page_Load 事件,说明页面被加载了一次,这种技术称为回传(又称回送)技术,是 ASP.NET 最为重要的特征之一。

在 ASP.NET 中,当客户端触发了一个事件时,它不是在客户端浏览器上对事件进行处理,而是把该事件的信息传送回服务器进行处理。服务器在接收到这些信息后,会重新加载 Page 对象,然后处理该事件,所以 Page_Load 事件被再次触发。

IsPostBack 属性表示页面是否被首次加载和访问。当 IsPostBack 为 true 时,表示该请求是为响应客户端回发而加载;当 IsPostBack 为 false 时,表示该页是被首次加载和访问。例如:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
       Response.Write("页面首次加载!");
    }
    else
    {
      Response.Write("页面响应客户端回发而加载!");
    }
}
```

任务 3-1　体验页内数据传递

【任务描述】

将用户在文本框中输入的歌手添加到列表控件中显示,观察添加的结果有什么问题。该问题是怎么引起的?如何解决该问题。

【任务实施】

(1) 创建网站项目 rw3-1,并在网站项目下添加页面文件 Default.aspx,进入页面视图,从工具箱分别拖放一个 ListBox、TextBox 和 Button 空间到编辑区,切换至源视图,并编写如下代码。

```
<form id="form1" runat="server">
  <div>
  请选择您喜欢的歌手<br />
<asp:ListBox ID="lbSonger" runat="server" Height="151px" Width="152px"></asp:ListBox>
```

```
    <br />
向列表中添加歌手<br />
<asp:TextBox ID="txtName" runat="server"></asp:TextBox>
<asp:Button ID="btnAddSonger" runat="server" Text="添加" OnClick="Button2_Click" />
</div>
</form>
```

lbSonger 列表控件用于显示歌手的列表，文本框 txtName 用于接收输入信息。

（2）在页面后置代码文件 Default.aspx.cs 中编写 Page_Load 事件过程代码，如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    lbSonger.Items.Add("帕瓦罗蒂");
    lbSonger.Items.Add("多明戈");
    lbSonger.Items.Add("卡雷拉斯");
}
```

（3）编写“添加”按钮在页面后置代码文件 Default.aspx.cs 中对应的 Click 事件过程代码，如下：

```
protected void btnAddSonger_Click(object sender, EventArgs e)
{
    lbSonger.Items.Add(txtName.Text);
}
```

（4）运行页面 Default.aspx，在文本框中输入歌手“韩红”，单击“添加”按钮，发现列表控件中并没有像我们想象的那样只添加了“韩红”，而是重复添加了前面三个歌手列表项，如图 3-3 所示。

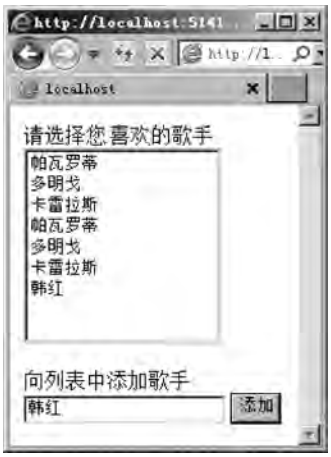

图 3-3　设置 IsPostBack 属性前的运行结果

导致该问题的原因是在页面加载时没有判断当前页面是否是回传页面，导致每次都要执行 Page_Load 方法中的添加前面三个歌手的代码，也就是说，当用户单击“添加”按钮后，

页面回传,这段代码又一次被执行。为了解决这个问题,可以使用 ASP. NET 提供的一个特殊属性 Page. IsPostBack 来进行判断,使用过程可以直接写成 IsPostBack,它是一个布尔值,当该值为真(true)时,则页面为回传,否则就是首次加载。

(5) 清楚了问题产生的原因后,该问题就很好解决了,修改页面 Default. aspx 的 Page_Load 事件过程代码,如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        lbSonger.Items.Add("帕瓦罗蒂");
        lbSonger.Items.Add("多明戈");
        lbSonger.Items.Add("卡雷拉斯");
    }
}
```

(6) 再次运行 Default. aspx 页面,输入歌手名,单击“添加”按钮,就会看到输入的歌手名被添加到列表最后,并且上面的列表项也没有重复添加,如图 3-4 所示。

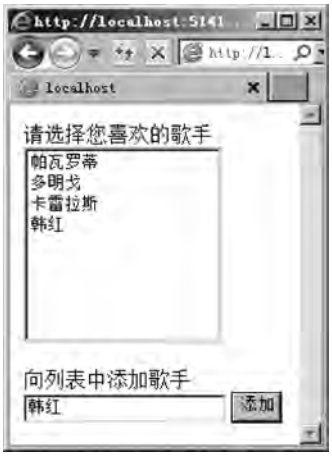

图 3-4　设置 IsPostBack 属性后的运行结果

页内数据传递是最简单的页面数据传递形式,当用户单击页面上的按钮等引起回传的控件时,所有页面上的服务器端控件的值都要回传,而且这些是不需要我们来处理的,ASP. NET 都已经封装好了,所以在 ASP. NET 中可以使用“控件对象名. 属性”的方式直接访问控件的相关内容。

3.3　Response 对象

Response 对象用于响应客户端的请求,将信息发送到客户端浏览器。用户可以使用 Response 对象实现向页面中输出文本或创建 Cookie 信息等,并且可以使用 Response 对象实现页面的跳转。

3.3.1 Response 对象的常用属性

1. BufferOutput 属性

BufferOutput 的默认属性为 true。当页面被加载时，要输出到客户端的数据都暂时存储在服务器的缓冲区内，并等待页面的所有事件程序以及所有的页面对象全部被浏览器解释完毕后，才将所有在缓冲区中的数据发送到客户端浏览器。

2. Charset 属性

Charset 属性设置页面显示中所使用的字符集。此属性设置后，客户端浏览器代码中的 HTML 头信息的 meta 属性增加一个属性值对——Charset＝字符集名。

3. ContentType 属性

ContentType 属性设置客户端 HTTP 文件格式，其格式为类型/子类型。常用的类型/子类型主要有 text/html(默认值)、image/jpeg、application/msword、application/msexcel、application/mspowerpoint 等。

4. IsClientConnected 属性

IsClientConnected 属性为只读属性，表示客户端与服务器端是否连接。如果此属性的返回值为 true，表示客户端与服务器端处于连接状态，否则表示客户端与服务器端已经断开。

5. Cookies 属性

Cookies 是存放在客户端用来记录用户访问网站的一些数据的对象。利用 Response 对象的 Cookies 属性可以在客户端创建一个 Cookies，创建 Cookies 的语法格式如下。

```
Response.Cookies[名称].Value = 值;
Response.Cookies[名称].Expirs = 有效期;
```

例如，创建一个名为 Name、值为“张三”、有效期为一天的 Cookies 信息，代码如下：

```
Response.Cookies["Name"].Value = "张三";
Response.Cookies["Name"].Expirs = DateTime.Now.AddDays(1);
```

3.3.2 Response 对象的常用方法

1. Write 方法

功能：在服务器端将指定数据发送给客户端浏览器。

语法：Response.Write(变量或字符串)；。

说明：在输出字符串常量时，要使用一对引号括起来；当字符串内含有引号时外层使用双引号，内层使用单引号；HTML 标记可以作为特别的字符串进行输出；客户端脚本也可以作为特别的字符串输出。例如：

```
Response.Write("<script>alert('Hello!');</script>");
```

【示例 3-2】 使用 Response.Write 方法输出字符串、HTML 标记及 Script 脚本。

(1) 创建页面文件 Default.aspx，将页面标题设置为“Response.Write 方法使用示例”。

(2) 在 Default.aspx.cs 的 Page_Load 事件中添加如下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("<font size='6' color='red' face='黑体'>欢迎来到新知图书</font><br/>");
    Response.Write("<hr width='75%' color='blue' align='left'/><br/><br/>");
    Response.Write("现在是北京时间: " + DateTime.Now.ToLongTimeString() + "<br/>");
    Response.Write("浏览新闻可以到<a href='http://www.xinhuanet.com/'>新华网</a><br/>");
    Response.Write("<script language='javascript'>alert('使用 Write 方法输出信息');
</script>");
}
```

(3) 按 Ctrl+F5 组合键运行,页面的运行效果如图 3-5 所示。

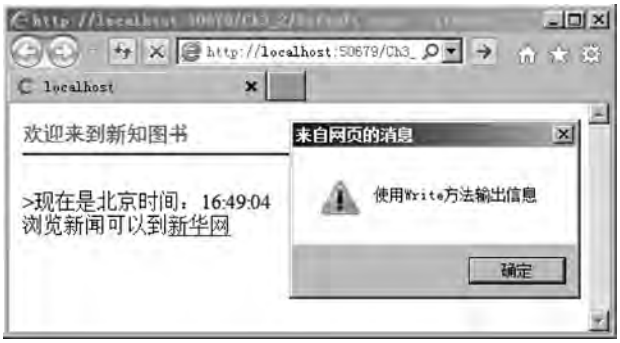

图 3-5 Response.Write 方法使用示例

2. WriteFile 方法

功能:将指定的文件内容写到 HTML 输出流。

语法:Response.WriteFile(filename);。

说明:若有大量数据要发送到浏览器,如果使用 Write 方法,那么其中的参数串会很长,影响程序的可读性。Response.WriteFile 方法用于直接将文件内容输出到客户端,如果要输出的文件和执行的网页在同一个目录中,直接传入文件名即可;如果不在同一个目录中,则要指定详细的目录名称。

举例:

```
Response.WriteFile("OutFile.txt");
```

说明:OutFile.txt 是准备输出的文本文件,存放在网站根目录下。

3. Redirect 方法

功能:使浏览器立即重定向到程序指定的 URL,即实现页面的跳转。

语法:Response.Redirect("网址或网页");。

举例:

```
Response.Redirect("http://www.edu.cn");
Response.Redirect("Default.aspx");
```

或

```
string ThisURL = "http://www.shu.edu.cn";
Response.Redirect(ThisURL);
```

【示例 3-3】 使用 Response 对象的 Redirect 方法跳转到指定页面。

(1) 创建页面文件 Default.aspx,将页面标题设置为“使用 Redirect 方法实现页面跳转”。

(2) 在 Default.aspx.cs 的 Page_Load 事件中添加如下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    int day = int.Parse(DateTime.Now.Day.ToString());
    string url;
    if (day % 2 == 0)
        url = "http://www.sina.com";
    else
        url = "http://www.sohu.com";
    Response.Redirect(url);
}
```

(3) 按 Ctrl+F5 组合键运行,程序先判断当前日期是奇数还是偶数,如果是偶数,跳转到新浪网,否则跳转至搜狐网。

4. End 方法

功能:用来输出当前缓冲区的内容,并中止当前页面的处理。

语法:Response.End();。

举例:

```
Response.Write("欢迎光临");
Response.End();
Response.Write("我的网站!");
```

该程序段只输出“欢迎光临”,而不会输出“我的网站!”。

5. Flush 方法

功能:将页面缓冲区中的数据立即显示。

语法:Response.Flush();。

说明:在编写程序的过程中,某一个请求可能会处理多个任务,可以在处理每个任务之后写一个 Response.Write("这里写一些操作提示信息!"),在后面加上 Response.Flush(),这样就会在每个任务完成之后将提示信息返回到页面。如果没有添加 Response.Flush(),那么所有的提示信息将会在方法执行完毕后才响应到页面。

6. Clear 方法

功能:清除页面缓冲区中的数据。

语法:Response.Clear();。

说明:在使用该方法时缓冲区必须打开,即 Response 的 BufferOutput 属性必须为

true。使用该方法只能清除 HTML 文件的 Body 部分。

7. TransmitFile 方法

功能：将指定文件下载到客户端。

语法：Response.TransmitFile(filename);。

说明：filename 是要下载的文件，如果要下载的文件和执行的网页在同一个目录中，直接传入文件名即可；如果不在同一个目录中，则要指定详细的目录名称。

3.4 Request 对象

当用户发出一个打开 Web 页面的请求时，Web 服务器会收到一个 HTTP 请求，此请求信息包括客户端的基本信息、请求方法、参数名、参数值等，这些信息将被完整地封装，并通过 Request 对象获取它们。Request 对象的主要功能是获取与网页密切相关的数据，包括客户端浏览器信息、用户输入表单中的数据、Cookies 中的数据、服务器端的环境变量等。

3.4.1 Request 对象的常用属性

1. QueryString 属性

Request 对象的 QueryString 属性用于获取客户端附在 URL 地址后的查询字符串中的信息。通过 QueryString 属性能够获取页面传递的参数。在超链接中往往需要从一个页面跳转到另外一个页面，跳转的页面需要获取 HTTP 的值进行相应的操作。例如，如果在地址栏中输入 NewsList.aspx?Id=1，则可以使用 Request.QueryString["id"]获取传递过来的 ID 的值。在使用 QueryString 属性时，表单的提交方式要设置为 Get。

2. Path 属性

Request 对象的 Path 属性用来获取当前请求的虚拟路径。当在应用程序开发中使用 Request.Path.ToString()时，能够获取当前正在被请求的文件的虚拟路径的值，当需要对相应的文件进行操作时，可以使用 Request.Path 的信息进行判断。

3. UserHostAddress 属性

通过使用 UserHostAddress 属性可以获取远程客户端 IP 主机的地址。在客户端主机 IP 的统计和判断中，可以使用 Request.UserHostAddress 进行 IP 的统计和判断。在有些系统中需要对来访的 IP 进行筛选，使用 Request.UserHostAddress 能够轻松地判断用户 IP 并进行筛选操作。

4. Browser 属性

通过使用 Browser 属性可以判断正在浏览网站的客户端的浏览器的版本，以及浏览器的一些信息，其语法格式为 Request.Browser.Type.ToString()。

5. ServerVariables 属性

使用 Request 对象的 ServerVariables 属性可以读取 Web 服务器端的环境变量，其语法格式为 Request.ServerVariables["环境变量名"]。

【示例 3-4】 使用 Request 对象的 ServerVariables 属性获取服务器端环境变量。

(1) 创建页面文件 Default.aspx，将页面标题设置为“获取服务器端环境变量”，并进行页面的样式设置。

(2) 在 Default.aspx.cs 的 Page_Load 事件中添加如下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("获取的服务器端信息: ");
    Response.Write("<hr>");
    Response.Write("当前网页虚拟路径: " + Request.ServerVariables["URL"] + "<br/>");
    Response.Write("当前网页实际路径: " + Request.ServerVariables["PATH_TRANSLATED"] +
"<br/>");
    Response.Write("服务器名: " + Request.ServerVariables["SERVER_NAME"] + "<br/>");
    Response.Write("软件: " + Request.ServerVariables["SERVER_SOFTWARE"] + "<br/>");
    Response.Write("服务器连接端口: " + Request.ServerVariables["SERVER_PORT"] +
"<br/>");
    Response.Write("HTTP 版本: " + Request.ServerVariables["SERVER_PROTOCOL"] + "<br/>");
    Response.Write("客户主机名: " + Request.ServerVariables["REMOTE_HOST"] + "<br/>");
    Response.Write("浏览器: " + Request.ServerVariables["HTTP_USER_AGENT"] + "<br/>");
    Response.Write("<hr>");
}
```

(3) 按 Ctrl+F5 组合键运行,页面效果如图 3-6 所示。

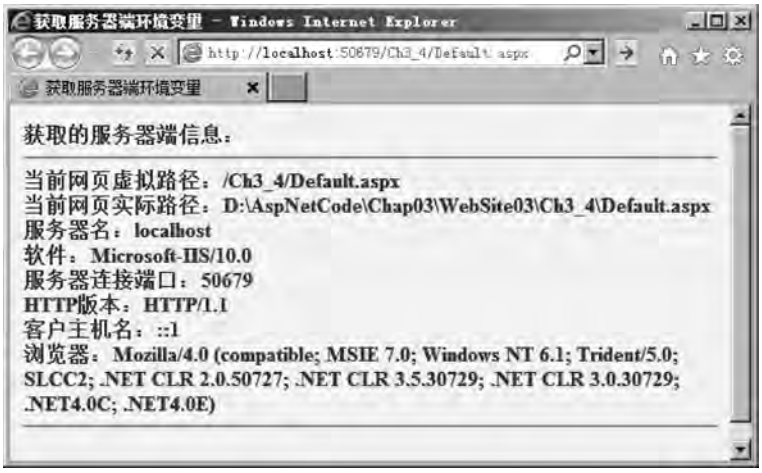

图 3-6 获取服务器端环境变量

6. Form 属性

Form 属性用于获取客户端在 Form 表单中所输入的信息,表单的 method 属性值需设置为 Post,其语法格式为 Request.From["元素名"]。

【示例 3-5】 使用 Request 对象的 Form 属性在两个页面间传递登录的用户名。

1) 设计 Web 页面

(1) 创建登录页面文件 LoginDemo.aspx,从工具箱拖入相应控件,代码如下:

```
<table style="width: 40%;">
    <tr>
        <td style="text-align: right">用户名: </td>
```

```
            <td>
             <asp:TextBox ID = "username" runat = "server"></asp:TextBox>
            </td>
        </tr>
        <tr>
            <td style = "text - align: right">密码: </td>
            <td>
             <asp:TextBox ID = "password" runat = "server" TextMode = "Password"></asp:TextBox>
            </td>
        </tr>
        <tr>
         <td colspan = "2" style = "text - align: center">
          <asp:Button ID = "btnLogin" runat = "server" PostBackUrl = "Welcome.aspx" Text = "登录"/>
         </td>
        </tr>
    </table>
```

(2) 创建登录信息欢迎页面 Welcome.aspx,从工具箱拖入相应控件,代码如下:

```
<form id = "form1" runat = "server">
<div>
    <asp:Label ID = "lblName" runat = "server" Text = "Label" CssClass = "kk"></asp:Label>
    <asp:Label ID = "lblMsg" runat = "server" Text = "Label" CssClass = "kk"></asp:Label>
    <br />
    <asp:Label ID = "Label3" runat = "server" Text = "欢迎您登录新知图书网!" CssClass = "kk">
</asp:Label>
</div>
</form>
```

2) 编写程序代码

在 Welcome.aspx.cs 的 Page_Load 事件中添加如下代码。

```
protected void Page_Load(object sender, EventArgs e)
{
    //获取用户登录名
    lblName.Text = Request["UserName"];
    //将系统时间与数据 13 进行比较,来获取问候语
    int Time = DateTime.Now.Hour.CompareTo(13);
    string str;
    if (Time > 0)
    {
    str = "下午好!";
    }
    else if (Time < 0)
    {
        str = "上午好!";
    }
    else
    {
```

```
            str = "中午好!";
        }
        lblMsg.Text = str;
    }
```

3）调试运行

按 Ctrl＋F5 组合键运行，登录页面效果如图 3-7 所示。

输入用户名、密码，单击“登录”按钮后跳转到欢迎页面，如图 3-8 所示。

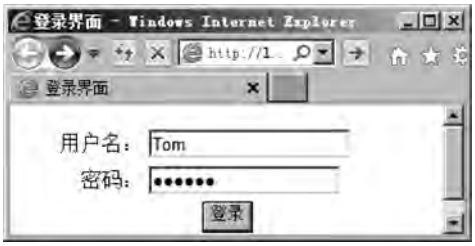

图 3-7　登录页面

图 3-8　登录欢迎页面

7. Cookies 属性

Cookie 是存放在客户端用来记录用户访问网站的一些数据的对象。利用 Response 对象的 Cookies 属性可以在客户端创建一个 Cookie。使用 Request 对象的 Cookies 属性可以读取 Cookie 对象的数据，其语法格式如下。

```
Request.Cookies[名称]
```

例如读取一个名为 Name 的 Cookie 对象的值，示例代码如下：

```
string name = Request.Cookies["Name"].Value;
```

3.4.2　Request 对象的常用方法

1. MapPath 方法

功能：利用 Request 对象的 MapPath 方法获取文件在服务器上的物理路径。

语法：Request. MapPath(filename)。

说明：filename 指文件名，如果文件和执行的网页在同一个目录中，直接传入文件名即可；如果不在同一个目录中，则要指定详细的路径名称。

2. SaveAs 方法

功能：用于将 HTTP 请求的信息存储到磁盘中。

语法：Request. SaveAs(string filename，bool includeHeaders)；。

说明：filename 指文件及其保存的路径，includeHeaders 是一个布尔值，表示是否将 HTTP 头保存到硬盘。

任务 3-2　获取客户端数据与跨页传递数据

【任务描述】

使用 Request 对象和 Response 对象实现一个登录操作,用户登录页面如图 3-9 所示,实现用户名和密码检查,如果用户名和密码都输入正确,则将用户跳转到图 3-10 所示的欢迎页面,在欢迎页面显示用户名、浏览器版本和浏览器语言等信息。

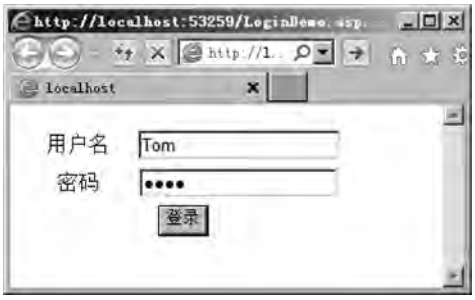

图 3-9　用户登录页面

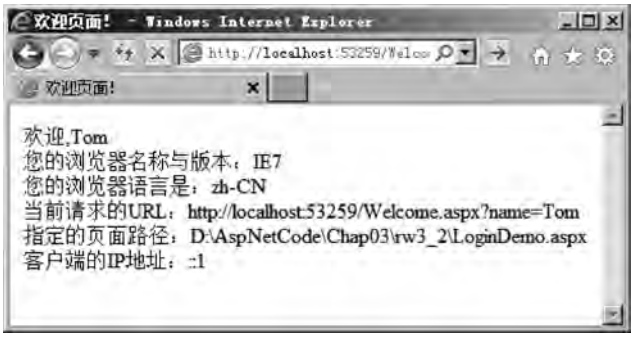

图 3-10　登录成功欢迎页面

【任务实施】

1. 设计 Web 页面

(1) 创建登录页面文件 LoginDemo. aspx,从工具箱拖入相应控件,代码如下:

```
<form id = "form1" runat = "server">
    <div>
        <table style = "vertical - align: middle; text - align: center">
            <tr>
                <td style = "width: 78px; height: 26px">
                    <asp:Label ID = "lblLoginId" runat = "server" Text = "用户名"></asp:Label>
                </td>
                <td style = "width: 160px; height: 26px">
                    <asp:TextBox ID = "txtloginId" runat = "server"></asp:TextBox>
                </td>
            </tr>
```

```
            <tr>
                <td style="width: 78px">
                    <asp:Label ID="lblLoginPwd" runat="server" Text="密码"></asp:Label>
                </td>
                <td style="width: 160px">
                    <asp:TextBox ID="txtLoginPwd" runat="server" TextMode="Password"
Width="146px"></asp:TextBox>
                </td>
            </tr>
            <tr>
                <td colspan="2" style="height: 26px; text-align: center">
                  <asp:Button ID="btnSubmit" runat="server" OnClick="btnSubmit_Click"
Text="登录" />  
                </td>
            </tr>
        </table>
    </div>
    <asp:Label ID="lblMessage" runat="server"></asp:Label>
</form>
```

(2) 创建登录信息欢迎页面 Welcome.aspx。

2. 编写程序代码,实现程序功能

(1) 在登录页后置代码文件 LoginDemo.aspx.cs 中编写单击“登录”按钮事件方法代码,如下:

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
    if (this.txtloginId.Text.Trim().Length == 0)
    {
        this.lblMessage.Text = "请输入用户名!";
        return;
    }
    if (this.txtLoginPwd.Text.Trim().Length == 0)
    {
        this.lblMessage.Text = "请输入密码!";
        return;
    }
    if (this.txtloginId.Text.Trim() == "Tom" && this.txtLoginPwd.Text.Trim() == "123")
    {
        Response.Redirect("Welcome.aspx?name=" + this.txtloginId.Text.Trim());
    }
    else
    {
        this.lblMessage.Text = "用户名/密码错误!";
    }
}
```

(2) 编写欢迎页面 Welcome.aspx 的 Page_Load 事件方法代码,如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        string userName = Request.QueryString["name"];
        Response.Write("欢迎," + userName + "<br/>");
        Response.Write("您的浏览器名称与版本: ");
        Response.Write(Request.Browser.Type);
        Response.Write("<br>您的浏览器语言是: ");
        Response.Write(Request.ServerVariables["HTTP_ACCEPT_LANGUAGE"].ToString());
        Response.Write("<br>当前请求的 URL: ");
        Response.Write(Request.Url);
        Response.Write("<br>指定的页面路径: ");
        Response.Write(Server.MapPath("LoginDemo.aspx"));
        Response.Write("<br>客户端的 IP 地址: ");
        Response.Write(Request.ServerVariables["remote_addr"]);
    }
}
```

3. 调试运行

按 Ctrl+F5 组合键运行,登录页面效果如图 3-9 所示。

输入用户名 Tom、密码 123,单击"登录"按钮后跳转到欢迎页面,如图 3-10 所示。

说明:如果用户名和密码输入正确,LoginDemo.aspx 页面使用 Response.Redirect 将页面跳转到欢迎页面,并将用户姓名添加到 URL 中,Welcome.aspx 页面用 Request.QueryString 从 URL 中获取用户名。本任务中的用户名和密码是固定的,实际项目中往往都是从数据库中读取,后续相关任务中会采用从数据库读取用户名和密码。

3.5 Server 对象

Server 对象提供了服务器端的基本属性与方法,例如将程序的虚拟路径转换为实际路径、执行指定的 ASP.NET 页面、HTML 编码与解码等。Server 对象能够帮助程序判断当前服务器的状态。

3.5.1 Server 对象的常用属性

1. MachineName 属性

该属性获取服务器的计算机名称,是一个只读属性。

2. ScriptTimeout 属性

该属性获取和设置请求超时的时间,单位为秒。

3.5.2 Server 对象的常用方法

1. MapPath 方法

功能:返回与 Web 服务器上的执行虚拟路径相对应的物理文件路径。

语法：Server. MapPath("虚拟路径")；。

2. Execute 方法

功能：使用另一个页面执行当前请求。

语法：Server. Execute("页面文件")；。

3. Transfer 方法

功能：终止当前页面的执行，并为当前请求开始执行新页面。

语法：Server. Transfer("页面文件")；。

4. HtmlEncode 方法

功能：对要在浏览器中显示的字符串进行编码。

语法：Server. HtmlEncode("字符串")；。

5. HtmlDecode

功能：将 HTML 编码字符串按 HTML 语法进行解释。

语法：Server. HtmlDecode("字符串")；。

3.5.3 Server 对象的应用

1. 将虚拟路径转换为实际路径

在程序中给出的文件路径使用的通常是虚拟路径，而有些应用中需要访问服务器的文件、文件夹或数据库文件，此时就需要将虚拟文件路径转换为实际文件路径。使用 Server 对象的 MapPath 方法可以实现这种路径转换，示例如下。

(1) 显示当前目录的实际路径：Server. MapPath("./")；。

(2) 显示父目录的实际路径：Server. MapPath("../")；。

(3) 显示根目录的实际路径：Server. MapPath("/")；。

(4) 显示网页 Server. aspx 的实际路径：Server. MapPath("Server. aspx")；。

2. 用 Execute 方法执行指定页面

Execute 方法类似于高级语言中的过程调用，用于将程序流程转移到指定的页面，该页面执行结束后流程将返回原网页的中断点继续执行。

【示例 3-6】 使用 Server 对象的 Execute 方法执行对另一个页面的请求。

1) 设计 Web 页面

(1) 创建页面文件 ExecuteDemo. aspx，页面文件代码如下：

```
<body>
    <form id="form1" runat="server">
    <div>
      <asp:Button ID="btnExeCute" runat="server" OnClick="Button1_Click" Text="用
Execute 方法执行指定页面" />
    </div>
    </form>
</body>
```

(2) 创建页面 TestPage.aspx,无须设计。

2) 编写程序代码,实现程序功能

(1) 编写 Web 页面 ExecuteDemo.aspx 的 btnExeCute 事件过程代码如下:

```
protected void btnExeCute_Click(object sender, EventArgs e)
{
    Response.Write("<p>调用 Execute 方法之前</p>");
    Server.Execute("TestPage.aspx");
    Response.Write("<p>调用 Execute 方法之后</p>");
}
```

(2) 在 Web 页面 TestPage.aspx 的 Page_Load 事件中添加代码如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("<p>这是一个测试页</p>");
}
```

3) 调试运行

按 Ctrl+F5 运行,页面运行效果如图 3-11 所示。

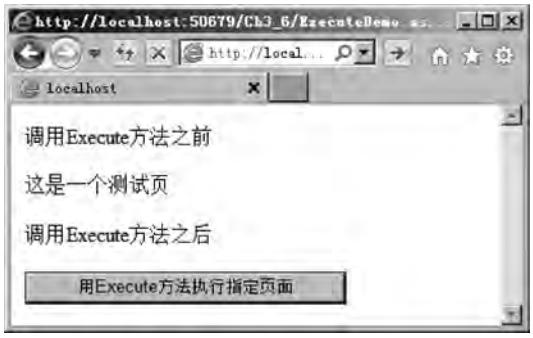

图 3-11 用 Execute 方法执行指定页面

3. 用 Transfer 方法实现网页重定向

用 Transfer 方法可以终止当前网页,执行新的网页,即实现网页重定向。与 Execute 方法不同的是,Transfer 方法转向新网页后不再将控制权返回,而是交给了新的网页。在示例 3-6 中,如果把第一个页面 ExecuteDemo.aspx 的后台代码改成如下形式:

```
Server.Transfer("TestPage.aspx");
Response.Write("<p>调用 Execute 方法之前</p>");
Response.Write("<p>调用 Execute 方法之后</p>");
```

则发现页面转向 TestPage.aspx 后并没有返回到 ExecuteDemo.aspx,因为没有执行第三条语句 Response.Write(" 调用 Execute 方法之后 ";。

Server 对象的 Transfer 方法与 Response 对象的 Redirect 方法都可以实现网页重定向功能,不同的是,Redirect 方法实现网页重定向后地址栏会变成转移后的网页的地址;而用

Transfer 方法实现网页重定向后地址栏不会发生变化，仍是转向前的地址。另外，用 Transfer 方法比用 Redirect 方法执行网页的速度快，因为所有内置对象的值会保留下来而不用重新创建。

4. HTML 编码和解码

在有些情况下希望在网页中显示 HTML 标记，例如< b >，这时不能直接在网页中输出< b >，因为< b >会被浏览器解读为 HTML 语言，即对文本进行加粗，而不会将< b >显示出来。在这种情况下，可以使用 Server 对象的 HtmlEncode 方法对要在网页上显示的 HTML 标记进行编码，然后再输出。同样，可以使用 Server 对象的 HtmlDecode 方法对编码后的字符进行解码，将 HTML 编码字符串按 HTML 语法进行解释。

【示例 3-7】 使用 HtmlEncode 和 HtmlDecode 方法进行编码和解码。

(1) 创建 Web 页面文件 Default.aspx，在该页面设计视图中拖放两个 Button 按钮、两个 Label 和一个 TextBox，代码如下：

```
< form id = "form1" runat = "server">
< div >
请输入 HTML 进行编码: < asp: TextBox ID = "txtHtml" runat = "server" Width = "258px"> </asp:
TextBox >
< asp:Button ID = "btnHtmlEncode" runat = "server" OnClick = "btnHtmlEncode_Click" Text =
"HtmlEncode" />
< br /> < br />
编码后的 HTML 为: < asp: Label ID = "lblHtmlEncode" runat = "server" Text = "Label"> </asp:
Label >
< br /> < br />
对编码后的 HTML 进行解码: < asp:Button ID = "btnHtmlDecode" runat = "server" OnClick =
"btnHtmlDecode_Click" Text = "HtmlDecode" />
  < br /> < br />
   解码后的 HTML 为: < asp:Label ID = "lblDecode" runat = "server" Text = "Label"></asp:Label >
</div >
</form >
```

(2) 编写 Web 页面 Default.aspx 的 btnHtmlEncode 和 btnHtmlDecode 事件过程代码如下：

```
protected void btnHtmlEncode_Click(object sender, EventArgs e)
{
    this.lblHtmlEncode.Text = Server.HtmlEncode(txtHtml.Text);
}
protected void btnHtmlDecode_Click(object sender, EventArgs e)
{
    this.lblDecode.Text = Server.HtmlDecode(lblHtmlEncode.Text);
}
```

(3) 按 Ctrl＋F5 组合键运行，页面运行效果如图 3-12 所示。

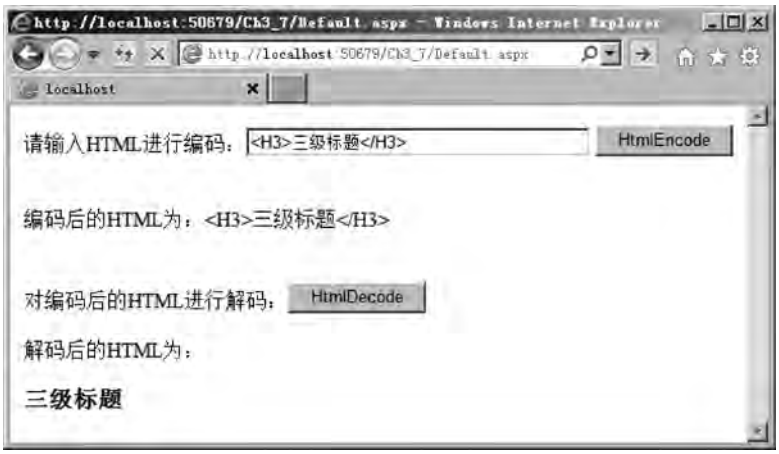

图 3-12　HtmlEncode 和 HtmlDecode 方法示例

3.6　Cookie 对象

Cookie 是一小段存储在客户端的文本信息，当用户请求某页面时，它就伴随着用户的请求在 Web 服务器和浏览器之间来回传递。当用户首次访问某网站时，应用程序不仅发送给用户浏览器一个页面，同时还有一个记录用户信息的 Cookie，用户浏览器将它存储在用户硬盘上的某个文件夹中，Windows 7 系统下通常默认保存在“C:\Users\用户名\AppD\Roaming\Microsoft\Windows\Cookies”的 txt 文件中。当用户再次访问此网站时，Web 服务器会首先查找客户端上是否存在上次访问该网站时留下的 Cookie 信息，如果存在，则会根据具体的信息发送特定的网页给用户。

Cookie 对象将数据保存在客户端，记录了浏览器的信息、何时访问 Web 服务器、访问过哪些页面等信息。使用 Cookie 的主要优势是服务器能依据它快速获得浏览者的信息，而不必将浏览者信息存储在服务器上，可减少服务器端的磁盘占用量。

3.6.1　Cookie 对象的常用属性

1. Name 属性

该属性获取或设置 Cookie 的名称。

2. Value 属性

该属性获取或设置 Cookie 的 Value。

3. Expires 属性

该属性设定 Cookie 变量的有效时间，默认为 1000 分钟，如果设为 0，则可以实时删除 Cookie 变量。

3.6.2　Cookie 对象的常用方法

1. Add 方法

功能：增加 Cookie 变量。

语法：Response. Cookies. Add(Cookie 变量名)；。

2. Clear 方法

功能：清除 Cookie 集合内的变量。

语法：Request. Cookies. Clear()；。

3. Remove 方法

功能：通过 Cookie 变量名称或索引删除 Cookie 对象。

语法：Response. Cookies. Remove(Cookie 变量名)；。

3.6.3 Cookie 对象的应用

1. 创建和读取 Cookie

创建 Cookie 使用的是 Response 对象的 Cookies 属性，例如：

```
Response.Cookies["Name"].Value = "张三";
Response.Cookies["Name"].Expirs = DateTime.Now.AddDays(1);
```

一个完整的 Cookie 对象包含三个参数，即名称、值和有效期。上述语句中创建的 Cookie 对象的名称为“Name”，值为“张三”，有效期为一天。即 Cookie 对象的生命周期是由开发者来设定的，如果在创建 Cookie 对象时没有设置其有效期，那么此 Cookie 对象会随着浏览器的关闭而失效；如果希望设置一个永不过期的 Cookie，可以设置一个比较长的时间，例如 50 年。

读取 Cookie 使用的是 Request 对象的 Cookies 属性，例如：

```
string name = Request.Cookies["Name"].Value;
```

2. 修改 Cookie

由于 Cookie 是存储在客户端硬盘上的，由客户端浏览器进行管理，因此无法从服务器端直接进行修改。修改 Cookie 其实相当于创建一个与要修改的 Cookie 同名的新的 Cookie，设置其值为要修改的值，然后发送到客户端覆盖客户端上的旧版本 Cookie。

例如，要将名称为“Name”的 Cookie 的值由“张三”改为“zhangsan”，代码如下：

```
Response.Cookies["Name"].Value = "zhangsan";
```

3. 删除 Cookie

和服务器无法修改 Cookie 一样，服务器端也无法对 Cookie 直接进行删除，但是可以利用浏览器自动删除到期 Cookie 的功能来删除 Cookie。具体做法是创建一个与要删除的 Cookie 同名的新的 Cookie，并将该 Cookie 的有效期设置为当前日期的前一天，当浏览器检查 Cookie 的有效期时就会删除这个已过期的 Cookie。例如，要删除前面创建的 Cookie 对象 Name，执行如下代码即可。

```
Response.Cookies["Name"].Value = "zhangsan";
Response.Cookies["Name"].Expirs = DateTime.Now.AddDays( - 1);
```

【示例 3-8】 使用 Cookie 对象在登录时记住密码。

(1) 创建 Web 页面文件 CookieDemo.aspx,在该页面设计视图中拖放控件,完成页面设计,代码如下:

```
<form id="form1" runat="server">
    <div>
      <asp:Label ID="Label1" runat="server" Text="用户名" Width="60px"></asp:Label>
      <asp:TextBox ID="UserName" runat="server"></asp:TextBox>
      <br />
      <asp:Label ID="Label2" runat="server" Text="密码" Width="60px"></asp:Label>
      <asp:TextBox ID="UserPassword" runat="server" TextMode="Password"></asp:TextBox>
      <br />
      <asp:CheckBox ID="PwdChecked" runat="server" Text="记住密码" TextAlign="Left" />
      <br /><br />
      <asp:Button ID="btnLogin" runat="server" OnClick="btnLogin_Click" Text="登录" />
      <asp:Button ID="btnRest" runat="server" Text="重置" Width="40px" />
    </div>
</form>
```

(2) 编写 Web 页面 CookieDemo.aspx 的 Page_Load 和 btnLogin_Click 事件过程代码如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request.Cookies["password"] != null)
    {
        if (DateTime.Now.CompareTo(Request.Cookies["password"].Expires) > 0)
        {
            UserPassword.Text = Request.Cookies["password"].Value;
        }
    }
}
protected void btnLogin_Click(object sender, EventArgs e)
{
    if (PwdChecked.Checked)
    {
        Response.Cookies["password"].Value = UserPassword.Text;
        Response.Cookies["password"].Expires = DateTime.Now.AddSeconds(10);
    }
}
```

说明:为了让运行效果明显,此处特意把 Cookie 变量的有效期设为 10 秒。

(3) 按 Ctrl+F5 组合键运行,首次访问时两个文本框均为空,用户输入用户名和密码后选择“记住密码”复选框,单击“登录”按钮,页面运行效果如图 3-13 所示。关闭浏览器,在 10 秒内重新登录该页面,可以看到密码框内已经记住了密码。

选择记住密码的 CheckBox,就创建了一个 Cookie 用于记录密码的内容,同时设置有效期。在下次加载时判断有没有这个密码 Cookie,如果有再判断这个 Cookie 是否过期,如果

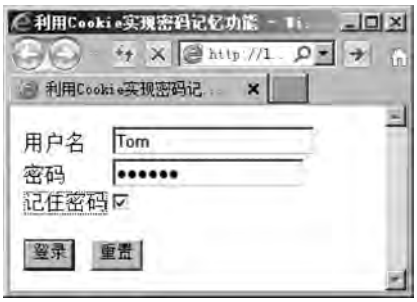

图 3-13　利用 Cookie 实现密码记忆功能

未过期，就将这个 Cookie 里存的值取出来，放到对应的文本框中。

把有效期设置为 10 秒，这样可以使看到的效果明显一些。在 10 秒之前，密码部分还一直有值，过了 10 秒就自动清空了，因为 Cookie 到期了。

3.7　Session 对象

Session 对象一般用于保存用户从登录网页到离开网页这段时间内的相关信息，如用户名、密码、IP 地址、访问时间等，Session 对象把用户的这些私密信息保存在服务器端。

当用户请求一个 ASP. NET 页面时，系统会自动创建一个 Session 对象，并为每一次会话分配一个唯一的 SessionID，以此来唯一标识一个用户。Session 对象的生命周期始于用户第一次连接到网页，在以下情况之一发生时结束。

(1) 关闭浏览器窗口。

(2) 断开与服务器的连接。

(3) 浏览者在有效时间内未与服务器联系。

3.7.1　Session 对象的常用属性

1. IsNewSession 属性

如果用户访问页面时是创建新会话，则此属性将返回 true，否则返回 false。

2. TimeOut 属性

该属性传回或设置 Session 对象变量的有效时间，如果在有效时间内没有任何客户端动作，则会自动注销。如果不设置 TimeOut 属性，则系统默认的超时时间为 20 分钟。

3. SessionID 属性

一个用户对应一个 Session，用户首次与 Web 服务器建立连接时，服务器会给用户分发一个 SessionID 作为标识。

SessionID 是一个由 24 个字符组成的随机字符串。用户每次提交页面时，浏览器都会把这个 SessionID 包含在 HTTP 头中提交给 Web 服务器，这样 Web 服务器就能区分当前请求页面的是哪一个客户端。在客户端，浏览器会将本次会话的 SessionID 值存入本地的 Cookie 中，当再次向服务器提出页面请求后，该 SessionID 值将作为 Cookie 信息传送给服务器，服务器就可以根据该值找到此次会话以前在服务器上存储的信息。

3.7.2 Session 对象的常用方法

1. Add 方法

功能：创建一个 Session 对象。

语法：Session.Add("对象名称",对象的值);。

2. Abandon 方法

功能：该方法用来结束当前会话并清除对话中的所有信息,如果用户重新访问页面,则可以创建新会话。

语法：Session.Abandon();。

3. Clear 方法

功能：此方法将清除全部的 Session 对象变量,但不结束会话。

语法：Session.Clear();。

4. Remove 方法

功能：清除某一个 Session 变量。

语法：Session.Remove("Session 变量名");。

3.7.3 Session 对象的事件

对应于 Session 的生命周期,Session 对象也拥有自己的事件,即 Session_Start 与 Session_End,它们存放在 Global.asax 文件中。

1. Session_Start 事件

该事件在某个用户第一次访问网站的某个网页时发生。

当客户端浏览器第一次请求 Web 应用程序的某个页面时触发 Session_Start 事件。此事件是设置会话期间变量的最佳时机,所有的内置对象(Response、Request、Server、Application、Session)都可以在此事件中使用。

2. Session_End 事件

该事件当某个用户 Session 超时或关闭时发生。

当一个会话超时或 Web 服务器被关闭时触发 Session_End 事件。在此事件中只有 Server、Application、Session 对象是可用的。

3.7.4 Session 对象的应用

1. 将数据存入 Session 对象

通常有两种方法将数据存入 Session 对象。

(1) Session["对象名称"]=对象的值;。

(2) Session.Add("对象名称",对象的值);。

2. 读取 Session 对象的值

读取 Session 对象的值的语法格式如下：

```
变量 = Session["对象名称"];
```

【示例 3-9】 使用 Session 对象在两个页面之间传送密码的值。

1）设计 Web 页面

（1）创建 Web 页面文件 SessionDemo.aspx，在该页面设计视图中拖放控件，完成页面设计，代码如下：

```
<form id="form1" runat="server">
    <div>
      用户名：<asp:TextBox ID="txtUser" runat="server"></asp:TextBox>
      <br />
      密码：<asp:TextBox ID="txtPwd" runat="server" TextMode="Password"></asp:TextBox>
      <br /><br />
      <asp:Button ID="btnLogin" runat="server" Text="登录" OnClick="btnLogin_Click" />
      <asp:Button ID="btnReset" runat="server" Text="重置" />
    </div>
</form>
```

（2）创建 Web 页面文件 Welcome.aspx，该页面只设置了页面标题。

2）编写程序代码，实现程序功能

（1）编写 Web 页面 SessionDemo.aspx 的 btnLogin_Click 事件过程代码，如下：

```
protected void btnLogin_Click(object sender, EventArgs e)
{
  if (txtUser.Text != "" || txtPwd.Text != "")
  {
       Session["username"] = txtUser.Text;
       Session["password"] = txtPwd.Text;
       Response.Redirect("Welcome.aspx");
  }
  else
     Response.Write("<script language='javascript'>alert('用户名或密码不能为空!');
</script>");
}
```

（2）编写 Web 页面 Welcome.aspx 的 Page_Load 事件过程代码，如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if(Session["username"]!= null && Session["password"]!= null )
    {
        string name = Session["username"].ToString();
        string pwd = Session["password"].ToString();
        Response.Write("欢迎" + name + "光临本站，请记住你的密码：" + pwd);
    }
}
```

说明：为了让运行效果明显，此处特意把 Cookie 变量的有效期设为 10 秒。

3）调试运行

按 Ctrl＋F5 组合键运行，单击“登录”按钮后程序首先判断用户名和密码是否为空，只

要有一个为空,就会弹出一个提示对话框,提示用户“用户名或密码不能为空!”;如果都不为空,如图 3-14 所示,则把用户名和密码框里的值分别存到两个 Session 变量里,然后跳转到欢迎页面 Welcome.aspx。

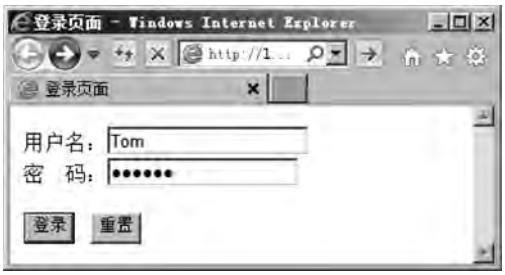

图 3-14 登录页面

在 Welcome.aspx 中获取并显示前一个页面用 Session 变量保存的用户名和密码,如图 3-15 所示。

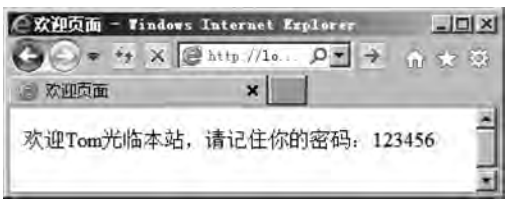

图 3-15 欢迎页面

任务 3-3 实现防非法访问的登录功能

【任务描述】

在任务 3-2 的基础上,结合 Session 和 Cookie 对象实现新知书店管理后台防非法访问的登录功能,符合以下需求。

(1) 如果用户试图直接在浏览器地址栏输入后台管理首页 URL:http://xxx/Admin/Default.aspx,则直接跳转到登录页面。

(2) 登录页面加载时,给出用户名的输入提示,如果客户端保存了用户名,则显示用户名,如图 3-16 所示。

(3) 实现用户名和密码的非空验证,如果都不为空,进行用户名和密码的数据验证(为简化操作,本任务的用户名和密码仍然固定),否则给出“请输入用户名和密码”的提示信息。

(4) 如果用户名和密码输入正确,则跳转到新知书店的后台页面,并提示“欢迎,**** ”,否则给出“用户名或密码错误”的提示信息,如图 3-17 所示。

【任务实施】

1. 思路分析

(1) 使用 Cookie 判断客户端是否保存了用户名,如果 Cookie 为空,则提示输入用户名,否则使用 Request.Cookies[Cookie 的名称].Value 读取用户名并显示。

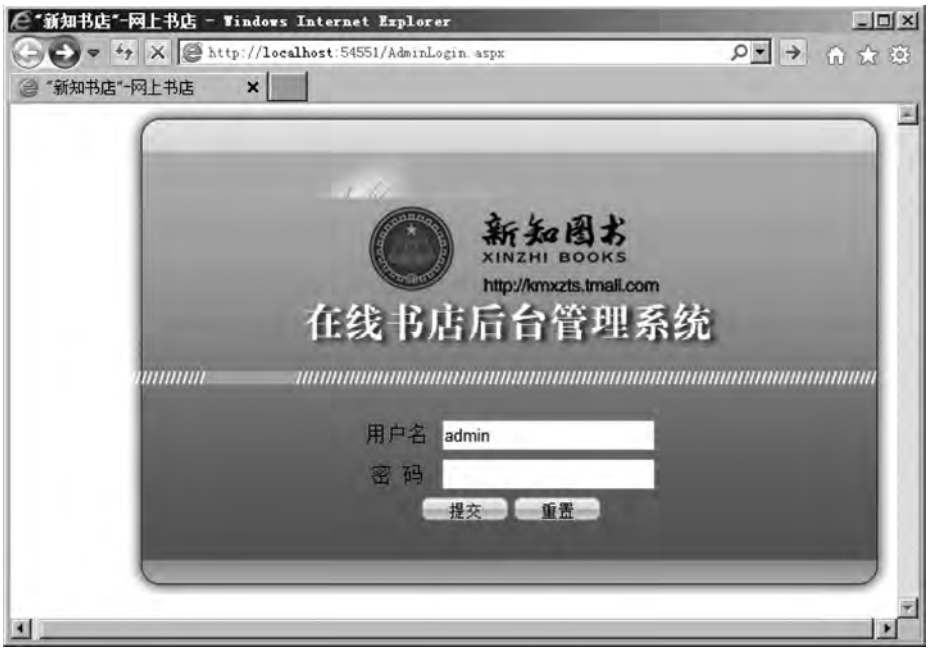

图 3-16　登录页面效果

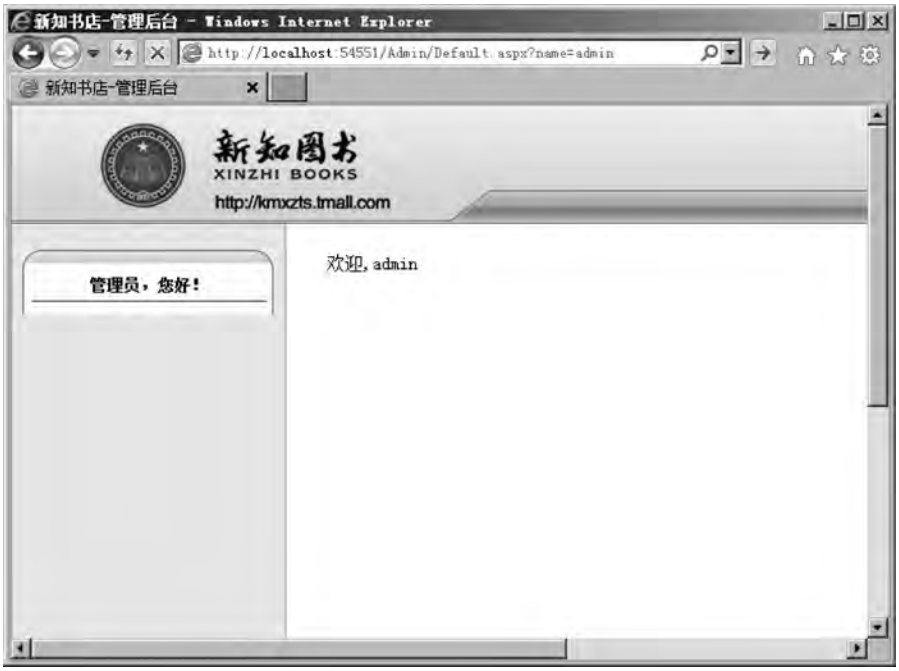

图 3-17　新知书店后台登录成功效果

(2) 使用 Session 保存用户名和密码。

(3) 在加载后台首页时判断 Session 是否为空，如果为空，使用 Response.Redirect()将用户重定向到登录页面。

2. Web 页面设计及编码

(1) 创建网站项目 rw3-3,右击网站项目新建文件夹 Images 并复制登录页面图片素材至目录下,添加页面 AdminLogin. aspx,在页面< head ></head >标签对中编写样式代码,如下:

```
<head>
  <style type = "text/css">
    .login
    {
        position: absolute;
        top: 50 % ;
        left: 50 % ;
        margin: - 250px 0 0 - 250px;
    }
    .login_t
    {
        margin: 0 auto;
        width: 598px;
        height: 78px;
        background: url(images/login_03.gif) no - repeat;
    }
    .login_m
    {
        margin: 0 auto;
        width: 598px;
        height: 142px;
        background: url(images/login_05.gif) no - repeat;
    }
    .login_b
    {
        margin: 0 auto;
        width: 598px;
        height: 150px;
        padding - top: 24px;
        background: url(images/login_06.gif) no - repeat;
        text - align: center;
    }
    .login_b p
    {
        margin: 12px 0;
    }
    .login_input
    {
        width: 160px;
        height: 20px;
        margin - left: 6px;
        line - height: 20px;
        border: 1px solid # 999;
```

```
    }
    .login_sub
    {
        width: 67px;
        height: 19px;
        background: url(images/login_sub.gif) no-repeat;
        text-align: center;
        border: none;
        line-height: 22px;
    }
  </style>
</head>
```

(2) 从工具箱拖入两个 TextBox 控件、两个 Button 控件至页面 AdminLogin.aspx，并添加代码如下：

```
<form id="Login" name="Login" method="post" runat="server">
  <table align="center" >
    <tr>
      <td>用户名</td>
      <td><asp:TextBox runat="server" ID="txtLoginId" CssClass="login_input"></asp:
TextBox></td>
    </tr>
    <tr>
        <td>密   码</td>
        <td><asp:TextBox runat="server" ID="txtLoginPwd" TextMode="Password"
                CssClass="login_input"></asp:TextBox></td>
    </tr>
    <tr>
        <td colspan="2">
           <asp:Button runat="server" ID="btnConfirm" OnClick="btnConfirm_Click"
                    Text="提交" CssClass="login_sub" />
           <asp:Button runat="server" ID="btnCancel" OnClick="btnCancel_Click"
                    Text="重置" CssClass="login_sub" />
        </td>
    </tr>
  </table>
</form>
```

(3) 在登录页后置代码文件 AdminLogin.aspx.cs 中编写单击“提交”“重置”按钮的事件方法及页面 Page_Load 事件方法代码，如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
        if (Request.Cookies["UserName"] == null)
        {
```

```
                this.txtLoginId.Text = "请在此输入用户名";
            }
            else
            {
                this.txtLoginId.Text = Request.Cookies["UserName"].Value;
            }
            this.txtLoginPwd.Text = "请在此输入密码";
        }
    }
    protected void btnConfirm_Click(object sender, EventArgs e)
    {
        string strMsg = string.Empty;
        if (this.txtLoginId.Text.Trim() == "admin" && this.txtLoginPwd.Text.Trim() ==
    "123456")
        {
            Response.Cookies["UserName"].Value = this.txtLoginId.Text.Trim();
            Response.Cookies["UserName"].Expires = DateTime.Now.AddDays(10);
            //以下注释的三行代码为实现 Cookie 的另外一种方式
            //HttpCookie hcCookie = new HttpCookie("UserName", this.txtLoginId.Text.Trim());
            //hcCookie.Expires = DateTime.Now.AddDays(1);
            //Response.Cookies.Add(hcCookie);
            UserInfo userInfo = new UserInfo();
            userInfo.UserName = this.txtLoginId.Text.Trim();
            userInfo.UserPwd = this.txtLoginPwd.Text.Trim();
            Session["user"] = userInfo;
            Response.Redirect("Admin/Default.aspx?name=" + this.txtLoginId.Text.Trim());
        }
        else
        {
            Response.Write(strMsg);
        }
    }
    protected void btnCancel_Click(object sender, EventArgs e)
    {
        this.txtLoginId.Text = String.Empty;
        this.txtLoginPwd.Text = String.Empty;
    }
```

(4) 右击网站项目 rw3-3,新建文件夹 Admin,并在文件夹 Admin 下新建两个子文件夹 Css、Images,用于存放后台首页的样式文件 admin.css 和图片资源,右击 Admin 文件夹添加新知书店后台首页 Default.aspx,拖入一个 Label 标签,修改 ID 为 blCome,并编写代码如下:

```
<head runat="server">
    <title>新知书店 - 管理后台</title>
    <link href="CSS/admin.css" rel="stylesheet" type="text/css" /><% -- 从 Css 文件夹导
入样式文件 -- %>
</head>
```

```
<body>
    <form id="Form1" runat="server">
    <div id="header">
        <img src="images/admin_top.gif" alt="" /></div>
    <div id="main">
        <div id="opt_list">
            <h1>管理员,您好!</h1>
        </div>
        <div id="opt_area"><br />
            <asp:Label ID="lblCome" runat="server" Text="Label"></asp:Label>
        </div>
    </div>
    </form>
</body>
```

(5) 编写后台首页页面后置代码文件 Default.aspx.cs 中的代码,如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["user"] == null)
    {
        Response.Redirect("~/AdminLogin.aspx");
    }
    if (!IsPostBack)
    {
        UserInfo user = Session["user"] as UserInfo; //使用 Session 实现
        lblCome.Text = "欢迎," + user.UserName;
    }
}
```

3.8 Application 对象

Application 对象的用途是在服务器端记录整个网站的信息,它可以使在同一个应用内的多个用户共享信息,并在服务器运行期间持久地保存数据。Application 对象可以记录不同浏览器端共享的变量,无论有几个浏览者访问网页,都只会产生一个 Application 对象,即只要是正在使用这个网页的浏览器端都可以存取这个变量。Application 对象变量的生命周期始于 Web 服务器开始执行时,止于 Web 服务器关机或重新启动时。

3.8.1 Application 对象的常用方法

1. Add 方法

功能:新增一个 Application 对象变量。

语法:Application.Add("对象名称",对象的值);。

2. Clear 方法

功能:清除全部的 Application 对象变量。

语法：Application.Clear();。

3. Remove 方法

功能：使用变量名移除一个 Application 对象变量。

语法：Application.Remove("Application 变量名");。

4. Set 方法

功能：使用变量名更新一个 Application 对象变量的内容。

语法：Application.Set("对象名称",对象的值);。

5. Lock 方法

功能：锁定全部的 Application 对象变量,防止其他客户端更改 Application 对象变量的值。

语法：Application.Lock();。

6. UnLock 方法

功能：解除锁定 Application 对象变量,允许其他客户端更改 Application 对象变量的值。

语法：Application.UnLock();。

3.8.2 Application 对象的事件

1. Application_Start 事件

该事件在应用程序启动时被触发。它在应用程序的整个生命周期中仅发生一次,此后除非 Web 服务器重新启动才会再次触发该事件。

2. Application_End 事件

该事件在应用程序结束时被触发,即 Web 服务器关闭时被触发。在该事件中常放置用于释放应用程序所占资源的代码段。

3.8.3 Application 对象的应用

【示例 3-10】 通过 Application 对象和 Session 对象统计当前在线用户数量。

(1) 创建页面文件 Default.aspx,在页面中添加一个 Label 标签控件,用来显示当前在线人数,ID 采用默认名称。

(2) 右击网站 WebSite03 根目录,新建全局应用程序类文件 Global.asax,当应用程序启动时初始化计数器,代码如下：

```
void Application_Start(object sender, EventArgs e)
{
   // 在应用程序启动时运行的代码
   Application["counter"] = 0;
}
```

当新会话启动时计数器加 1,代码如下：

```
void Session_Start(object sender, EventArgs e)
{
    // 在新会话启动时运行的代码
```

```
    Application.Lock();
    Application["counter"] = (int)Application["counter"] + 1;
    Application.UnLock();
}
```

当会话结束时计数器减 1,代码如下:

```
void Session_End(object sender, EventArgs e)
{
    // 在会话结束时运行的代码.
    Application.Lock();
    Application["counter"] = (int)Application["counter"] - 1;
    Application.UnLock();
}
```

注意:只有在 Web.config 文件中的 sessionstate 模式设置为 InProc 时,才会引发 Session_End 事件。如果会话模式设置为 StateServer 或 SQLServer,则不引发该事件。

(3) 在 Web 页面 Default.aspx 的 Page_Load 事件中添加代码如下:

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)Label1.Text = "当前在线人数:" + Application["counter"].ToString();
}
```

说明:通过 IsPostBack 属性实现回传不计入新增人数。

(4) 按 Ctrl+F5 组合键运行,页面运行效果如图 3-18 所示。

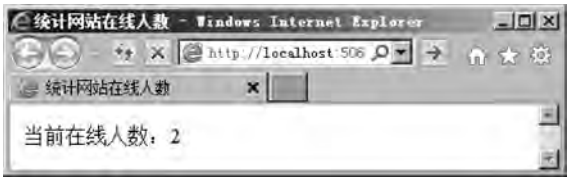

图 3-18 统计网站在线人数

【示例 3-11】 通过 Application 对象和对文件的读写操作来统计网站的访问总量(选做)。

(1) 关键技术:在实现统计网站访问总量功能时用到了两个关键技术。

① 对文件的读/写操作。StreamReader 对象以一种特定的编码从字节流中读取字符,其方法 ReadLine 从当前中读取一行字符并将数据作为字符串返回。StreamWriter 对象以一种特定的编码向字节流中写入字符,其方法 WriteLine 写入重载参数指定的某些数据,后跟行结束符。

② 应用 Application 对象。创建一个文本文件 counter.txt,将网站访问总量保留到其中。当应用程序启动时,将从文件 counter.txt 中读取的数据保存在 Application 对象中,新会话启动时需要获取 Application 对象中的数据。

(2) 创建页面文件 Default.aspx,用来显示网站统计的访问总量,代码如下:

```
<div>
   <table style="width: 100%;">
        <tr>
           <td style="text-align: center">网站访问总量为:<%=Application["counter"]%
></td>
        </tr>
    </table>
</div>
```

(3) 右击网站 Ch3_11 根目录,新建全局应用程序类文件 Global.asax,当应用程序启动时读取文件中的数据,将其值赋给 Application 对象,代码如下:

```
void Application_Start(object sender, EventArgs e)
{
   // 在应用程序启动时运行的代码
   int count = 0;
   StreamReader srd;
   string file_path = Server.MapPath("counter.txt");        //取得文件的实际路径
   srd = File.OpenText(file_path);                          //打开文件进行读取
   while (srd.Peek() != -1)
   {
       string str = srd.ReadLine();
       count = int.Parse(str);
   }
   srd.Close();
   object obj = count;
   Application["counter"] = obj; //将从文件中读取的网站访问量存放在 Application 对象中
}
```

当新会话启动时,需要获取 Application 对象中的数据信息并使访问总量加 1,代码如下:

```
void Application_End(object sender, EventArgs e)
{
    // 在应用程序关闭时运行的代码
    int Stat = 0;
    Stat = (int)Application["counter"];
    string file_path = Server.MapPath("counter.txt");
    StreamWriter srw = new StreamWriter(file_path, false);
    srw.WriteLine(Stat);
    srw.Close();
}
```

当应用程序结束时,将已更改的访问总量存放在 counter.txt 文件中,代码如下:

```
void Session_Start(object sender, EventArgs e)
{
    // 在新会话启动时运行的代码
```

```
        Application.Lock();
        int Stat = 0;                                   //数据累加器
        Stat = (int)Application["counter"];    //获取 Application 对象中保存的网站访问总量
        Stat += 1;
        object obj = Stat;
        Application["counter"] = obj;
        string file_path = Server.MapPath("counter.txt");
        StreamWriter srw = new StreamWriter(file_path, false);      //将数据记录写入文件
        srw.WriteLine(Stat);
        srw.Close();
        Application.UnLock();
    }
```

(4) 按 Ctrl+F5 组合键运行,页面运行效果如图 3-19 所示。

图 3-19　统计网站访问总量

3.8.4　Application、Session、Cookie 对象的区别

Application 对象和 Session 对象都是用来记录浏览器端的变量,都将信息保存在服务器端。二者不同的是,Application 对象记录的是所有浏览器端共享的变量,而 Session 对象变量只记录单个浏览器端专用的变量,即每个连接的用户有各自的 Session 对象变量,但共享同一个 Application 对象。

Cookie 对象与 Application 对象和 Session 对象类似,也是用于保存数据的。Cookie 对象与它们最大的不同是,Cookie 对象将数据保存在客户端,而 Application 对象和 Session 对象将数据保存在服务器端。

Application 对象的生命周期始于 Web 服务器开始执行时,止于 Web 服务器关机或重新启动时。Session 对象的生命周期是间隔的,这里以默认的 20 分钟为例,从创建开始计时,如果在 20 分钟内没有访问 Session,那么 Session 的生命周期被销毁;但是,如果在 20 分钟内的任一时间访问过 Session,那么将重新计算 Session 的生命周期。Cookie 对象的生命周期是累计的,从创建开始计时,到达设定的时间后 Cookie 对象的生命周期就结束。

Application、Session、Cookie 对象的区别如表 3-2 所示。

表 3-2　Application、Session、Cookie 对象的区别

对象	信息量	保存时间	应用范围	保存位置
Application	任意大小	整个应用程序生命周期	所有用户	服务器端
Session	小量、简单的数据	默认 20 分钟,可以修改	单个用户	服务器端
Cookie	小量、简单的数据	可以根据需要设定	单个用户	客户端

任务 3-4　制作简易在线聊天室

【任务描述】

应用 Application 对象、Session 对象和全局应用程序类(即 Global.asax 文件)设计一个简易聊天室。

(1) 该聊天室包括两个 Web 页面,一个是登录页面,另一个是聊天页面。用户登录聊天室时,应用 Application 对象存储登录用户名和在线用户数量,应用 Session 对象记录登录用户名。聊天页面中显示当前在线人数、聊天内容列表,并能输入聊天内容,且添加到聊天内容列表中。

(2) 在 Global.asax 文件中对 Application 对象进行初始化,包括聊天内容列表和当前在线聊天人数。

【任务实施】

1. 设计 Web 页面

(1) 创建网站项目 rw3-4,创建登录页面文件 Login.aspx,从工具箱拖入相应控件,页面主体部分代码如下:

```
<body>
    <form id="form1" runat="server">
        <div>
        <asp:Label ID="lblName" runat="server" Text="用户名" Width="60px"></asp:
Label>
        <asp:TextBox ID="txtName" runat="server"></asp:TextBox>
         <br />
         <asp:Label ID="lblPwd" runat="server" Text="密码" Width="60px"></asp:Label>
         <asp:TextBox ID="txtPwd" runat="server" TextMode="Password"></asp:TextBox>
         <br /><br />
         <asp:Button ID="btnLogin" runat="server" Text="登录" OnClick="btnLogin_Click" />
         <asp:Button ID="btnReset" runat="server" Text="重置" />
        </div>
    </form>
</body>
```

(2) 创建在线聊天页面 ChatWeb.aspx,页面主体部分代码如下:

```
<body>
  <form id="form1" runat="server">
     <div style="font-family: 仿宋; font-size: xx-large; color: #FF0000; background-
color: #C0C0C0; text-align: center; height: 60px; line-height: 60px; font-weight:
bolder;">在线聊天室</div>
     <div style="background-color: #FFCCCC; line-height: 40px; height: 40px">
          <asp:Label ID="lblPersonCount" runat="server" Text="Label"></asp:Label>
     </div>
     <div>
```

```
            <asp:TextBox ID = "txtChatList" runat = "server" BackColor = "#FFCCFF" ForeColor
= "#0033CC" Height = "300px" TextMode = "MultiLine" Width = "100%"></asp:TextBox>
        </div>
        <div style = "background - color: #C0C0C0; line - height: 40px; width: 80%; height:
40px; float: left;">
            <asp:Label ID = "Label2" runat = "server" Text = "lblName"></asp:Label>  
 
            <asp:TextBox ID = "txtChatContext" runat = "server" Width = "570px"></asp:TextBox>
        </div>
        <div style = "line - height: 40px; height: 40px; width: 20%; float: left; clear: right;
background - color: #808080; text - align: center;">
            <asp:Button ID = "btnSubmit" runat = "server" Text = "提交" OnClick = "Button1_Click" />
        </div>
    </form>
</body>
```

2. 新建 Global.asax 文件

右击网站 rw3_4 根目录，新建全局应用程序类文件 Global.asax，当应用程序启动时初始化计数器，代码如下：

```
void Application_Start(object sender, EventArgs e)
{
    // 在应用程序启动时运行的代码
    Application["online"] = 0;      //在线人数初始值为 0
    Application["chat"] = "";       // 聊天内容初始值为空
}
```

当新会话启动时计数器加 1，代码如下：

```
void Session_Start(object sender, EventArgs e)
{
    // 在新会话启动时运行的代码
    Application.Lock();
    Application["online"] = (int)Application["online"] + 1;
    Application.UnLock();
}
```

当会话结束时计数器减 1，代码如下：

```
void Session_End(object sender, EventArgs e)
{
    // 在会话结束时运行的代码.
    Application.Lock();
    Application["online"] = (int)Application["online"] - 1;
    Application.UnLock();
}
```

3. 编写程序代码，实现程序功能

(1) 在登录页后置代码文件 Login.aspx.cs 中编写单击“登录”按钮的 Click 事件过程

代码,如下:

```
protected void btnLogin_Click(object sender, EventArgs e)
{
   if (txtName.Text != "" || txtPwd.Text != "")
   {
      Session["name"] = txtName.Text;
      Response.Redirect("ChatWeb.aspx");
    }
   else
      Response.Write("<script language='javascript'>alert('用户名或密码不能为空!');
</script>");
}
```

在上述代码中首先判断两个文本框是否为空,如果用户名或密码为空,就会弹出提示框,提示用户"用户名或密码不能为空!";如果不为空,则把用户名赋给一个 Session 对象,通过 Session 对象将用户名传递到在线聊天页面,然后通过 Response 对象的 Redirect 方法跳转到在线聊天页面 ChatWeb.aspx。

(2) 编写在线聊天页面 ChatWeb.aspx 的 Page_Load 事件过程代码,如下:

```
protected void Page_Load(object sender, EventArgs e)
{
   if (Session["name"] != null)
   {
       lblPersonCount.Text = "当前在线人数为:" + Application["online"].ToString();
       txtChatList.Text = Application["chat"].ToString();
       lblName.Text = Session["name"].ToString();
       Response.AddHeader("refresh", "30");
   }
   else
       Response.Redirect("Login.aspx");
}
```

在上述代码中首先判断 Session["name"] 是否为空,如果不为空,说明用户在登录页面登录成功后跳转至当前页面,则在标签上控件 lblPersonCount 中显示当前在线人数,在多行文本框中显示所有的聊天内容,在标签控件 lblName 中显示用户的名字,并且设置页面自动刷新时间为 30 秒;如果 Session["name"] 为空,说明用户没有登录,则要求用户返回登录页面重新登录。

(3) 编写在线聊天页面 ChatWeb.aspx 的 btnSubmit_Click 事件过程代码,如下:

```
protected void btnSubmit_Click(object sender, EventArgs e)
{
   string newmessage = Session["name"] + ":" + DateTime.Now.ToString() + "\r" +
txtChatContext.Text + "\r" + Application["chat"];
    if (newmessage.Length > 800)
        newmessage = newmessage.Substring(0,799);
```

```
        Application.Lock();
        Application["chat"] = newmessage;
        Application.UnLock();
        lblName.Text = "";
        txtChatList.Text = Application["chat"].ToString();
    }
```

上述代码主要实现将用户发布的聊天内容添加到聊天室中，而且设置聊天室的聊天内容只能保留最新的 800 个字符。

4. 调试运行

按 Ctrl＋F5 组合键运行，进入登录页面。输入用户名和密码，登录成功后跳转至在线聊天页面，页面载入时会显示当前在线人数和当前聊天内容，如图 3-20 所示。

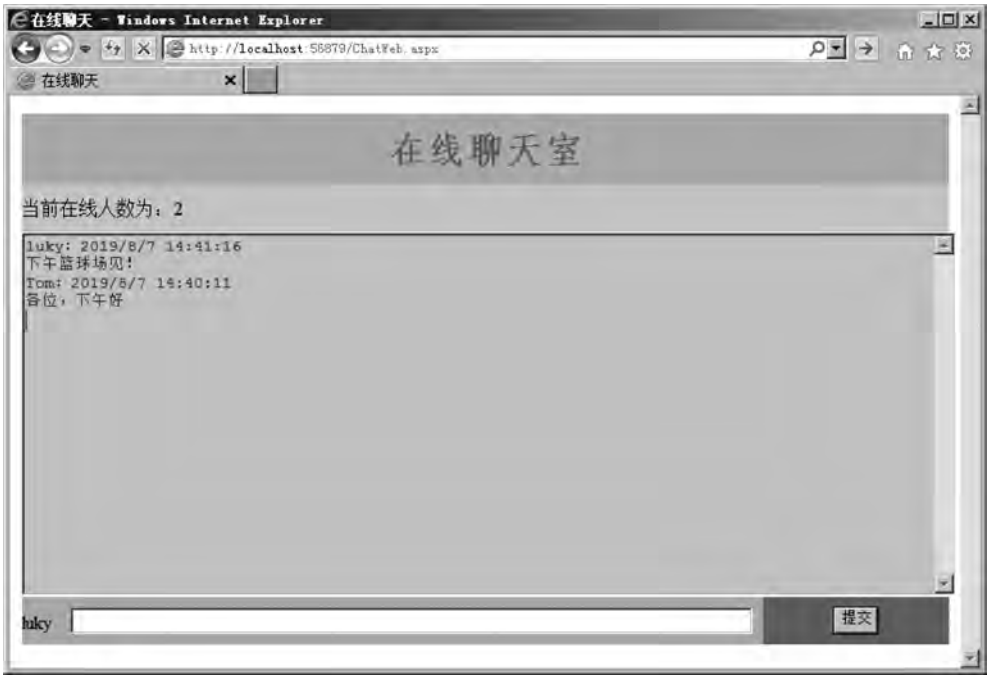

图 3-20　在线聊天室界面

单 元 小 结

本单元讲解了页面生命周期、页面传值及 ASP. NET 内置对象，包括如何创建 ASP. NET 内置对象和使用 ASP. NET 内置对象。本单元重点介绍了以下六个对象。

(1) Response 对象：可以向客户端输出

(2) Request 对象：用来获取客户端信息。

(3) Server 对象：专为处理服务器上的特定任务。

(4) Cookie 对象：一种可以在客户端保存信息的方法。

(5) Session 对象：记载特定客户的信息。

(6) Application 对象：存储 ASP.NET 应用程序多个会话和请求之间的全局共享信息。

Web 应用程序从本质上来讲是无状态的，为了维护客户端的状态，还重点阐述了如何使用 Session、Cookie、Application 等 ASP.NET 内置对象进行客户端状态的维护。

单元练习题

一、选择题

1. (　　) 文件主要定义应用开始和结束、会话开始和结束、请求开始和结束等事件发生时，要做的事情。

A. web.config　　B. Global.inc　　C. Config.asax　　D. Global.asax

2. 一个 ASP.NET 应用程序中一般只有(　　) 个 Global.asax 文件有效。

A. 0　　B. 1　　C. 若干　　D. 以上都不对

3. 在一个 ASP.NET 应用程序中，希望在每一次新的会话开始时，进行一些初始化任务，应该在(　　) 事件中编写代码。

A. Application_Start　　B. Application_BeginRequest

C. Session_Start　　D. Session_End

4. 下列选项中，只有(　　) 不是 Page 指令的属性。

A. CodePage　　B. Debug　　C. namespace　　D. Language

5. 在一个名为 Login 的 Web 网页中，先需要在其 Page_Load 事件中判断该页面是否回发，需要使用(　　) 属性。

A. Page.IsCallback　　B. Page.IsAsync

C. Page.IsPostBack　　D. Login.IsPostBack

6. (　　) 事件在页面被加载时，自动调用该事件。

A. Page_Load　　B. Page_UnLoad　　C. Page_OnLoad　　D. Page_Submit

7. 下面程序段执行完毕后，页面显示的内容是(　　)。

```
Response.Write("Hello");
Response.End();
Response.Write("World");
```

A. HelloWorld　　B. World　　C. Hello　　D. 出错

8. (　　) 方法用于将客户浏览器重新定向到一个新的 URL 地址。

A. Redirect　　B. BinaryRead　　C. UrlPathEncode　　D. UrlDecode

9. 使用(　　) 对象的 SaveAs 方法可以将 HTTP 请求保存到磁盘上。

A. Request　　B. Response　　C. Session　　D. Application

10. 一家在线测试中心 TestKing 公司创建一个 ASP.NET 应用程序。在用户结束测试后，这个应用程序需要在用户不知道的情况下，提交答案给 ProcessTestAnswers.aspx 页。ProcessTestAnswers.aspx 页面处理这些答案，但不提供任何显示消息给用户。当处理完成时，PassFailStatus.aspx 页面显示结果给用户。在 PassFailStatus.aspx 页面中添加(　　) 代码，来执行 ProcessTestAnswers.aspx 页面中的功能。

A. Server. Execute("ProcessTestAnswers. aspx")

B. Response. Redirect("ProcessTestAnswers. aspx")

C. Response. WriteFile("ProcessTestAnswers. aspx")

D. Server. Transfer("ProcessTestAnswers. aspx",True)

11. 一个应用程序中一般有(　　)个 web. config 文件有效。

A. 0　　B. 1　　C. 若干　　D. 以上都不对

12. 在名为 Login 的页面的 Page_Error 事件中捕获了一个未处理的异常,现需要清除刚产生的异常,需要使用下列(　　)语句。

A. HttpServerUtility. ClearError()　　B. Page. ClearError()

C. Login. ClearError()　　D. Server. ClearError()

13. 在一个 ASP. NET 的网站中,如果需要在应用程序级捕获未处理的异常,应该使用(　　)事件。

A. Response_Error　　B. Server_Error

C. Application_Error　　D. Page_Error

14. Request 对象中获取 Get 方式提交的数据的方法是(　　)。

A. Cookies　　B. ServerVariables

C. QueryString　　D. Form

15. 创建一个显示金融信息的 Web 用户控件。如果希望该 Web 用户控件中的信息能在网页的请求之间一直被保持,应该采取(　　)方法。

A. 设置该 Web 用户控件的 PersistState 属性为真

B. 设置该 Web 用户控件的 EnableViewState 属性为真

C. 设置该 Web 用户控件的 PersistState 属性为假

D. 设置该 Web 用户控件的 EnableViewState 属性为假

16. Session 对象的默认有效期为(　　)分钟。

A. 10　　B. 15　　C. 20　　D. 30

17. 下面程序段执行完毕,页面显示的内容是(　　)。

```
string strName;
strName = "user_name";
Session["strName"] = "Mary";
Session[strName] = "John";
Response.Write(Session["user_name"]);
```

A. Mary　　B. John

C. user_name　　D. 语法有错,无法正常运行

18. 下列(　　)对象经常用来制作网页计数器。

A. Response　　B. Application　　C. Request　　D. Session

19. 在同一个应用程序的页面 1 中执行 Session. Timeout＝30,那么在页面 2 中执行 Response. Write(Session. Timeout),则输出值为(　　)。

A. 15　　B. 20　　C. 30　　D. 25

20. Application 对象的默认有效期为(　　)。

A. 10 天　　B. 15 天

C. 20 天　　D. 从网站启动到终止

21. 下面代码实现一个站点访问量计数器,空白处的代码为(　　)。

```
void _____(object sender, EventArgs e)
{
Application.Lock();
Application["AccessCount"] = (int)Application["AccessCount"] + 1;
Application.UnLock();
}
```

A. Application_Start　　B. Application_Error

C. Session_Start　　D. Session_End

二、填空题

1. 列举 ASP.NET 中的 7 个内置对象:_____、_____、_____、_____、_____、_____、_____。

2. 应用程序开始时,调用_____事件;应用程序结束时,调用_____事件。

3. 一次新的会话开始时,调用_____事件;会话结束时,调用_____事件。

4. _____方法可获得网站根目录的物理路径。

三、问答题

1. 简述 Global.asax 文件的结构。Web 应用程序可以在哪些目录中放置此文件?

2. ASP.NET 页面包含哪些内置对象?

3. 简述 ASP.NET 网页文件由哪几部分组成。

4. 试说明什么是 Application 和 Session 对象,其差异是什么?如果存储用户专用信息,应该使用哪个对象变量来存储?

5. 什么是 Cookie?如何创建和读取 Cookie 对象?

6. Application 对象的 Lock 方法和 UnLock 方法各具有什么作用?