第3章 ASP.NET内置对象

本章学习目标

- 掌握 ASP. NET 对象的概念及访问方法；
- 掌握 ASP. NET 常用内置对象的属性和方法；
- 掌握 ASP. NET 常用内置对象的应用。

本章介绍 ASP. NET 对象的概念、访问方法以及 ASP. NET 各内置对象的属性、方法和应用，并对 Application 对象、Session 对象和 Cookie 对象进行比较。

3.1 ASP. NET 对象概述

3.1.1 ASP. NET 对象简介

在 ASP. NET 页面中除了要大量使用本书第 2 章中介绍的各种服务器控件以外，还需要使用各种 ASP. NET 对象，这些对象提供了基本的请求、响应、会话等处理功能。ASP. NET 定义了许多内置对象，它们可以直接使用而不必事先创建。ASP. NET 提供的七大内置对象如下。

- Page：用于设置与网页有关的属性、方法和事件。
- Response：服务器端将数据作为请求的结果发送到浏览器端(输出)。
- Request：浏览器端对当前页请求的访问发送到服务器端(输入)。
- Server：定义一个与 Web 服务器相关的类提供对服务器上方法和属性的访问。
- Session：即会话，是指一个用户在一段时间内对某个站点的一次访问。Session 对象用来保存与当前用户会话相关的信息。
- Cookie：保存客户端浏览器请求的服务器页面，存放非敏感用户信息。
- Application：存储所有客户端的共享信息。

除了内置对象以外，ASP. NET 还包含了其他对象，例如文件类和数据库对象，这些对象要先创建才能使用。本书第 5 章将重点介绍数据库对象。

3.1.2 ASP. NET 对象的访问

每个对象有各自的属性、方法和事件。属性用来描述对象的性质，它表示对象的静态特性；方法反映了对象的行为，它表示对象的动态特性；事件指对象在一定条件下产生的信息。

1. 访问对象属性

访问对象属性的语法格式：对象名.属性名

例如，访问 Page 对象的 IsPostBack 属性的语法格式：Page. IsPostBack.

2. 访问对象方法

访问对象方法的语法格式：对象名.方法名(参数表)

例如，访问 Response 对象的 Write 方法的语法格式：Response. Write("你好!")。

3. 对象事件处理的定义

对象事件处理的定义语法格式：对象名_事件名(参数表)或事件名(参数表)

例如：

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

上述代码定义 Page 对象的 Load 处理事件。其中，object sender 是事件处理过程的第一个参数，表示发生该事件的源对象；EventArgs e 是事件处理过程的第二个参数，表示传递过来的该事件的相关信息。

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 则表示全部验证成功，返回 false 则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。

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 对象的几种常用事件

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。

2. 应用举例

【案例 3.1】 Init 和 Load 事件的比较。

本案例对 Page_Init 事件和 Page_Load 事件进行比较。

1）页面设计

在页面中添加两个列表框和一个按钮，代码如下。

```
< html >
< head >< title > Init 和 Load 事件的比较</title ></head >
< 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 ></html >
```

2）后台代码设计

```
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的回发，不用写其 Click 事件代码。

3）运行调试

按 Ctrl＋F5 组合键运行，页面首次加载后的运行效果如图 3.1 所示。

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。

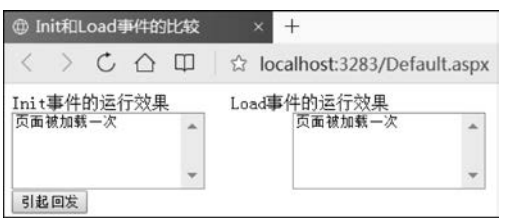

图 3.1　页面首次加载后的状态

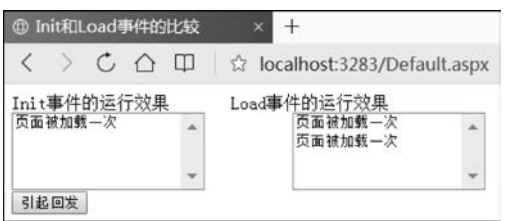

图 3.2　页面回发后的状态

从本案例中可以看出 Page 对象的 Init 和 Load 事件的异同。

- Page 对象的 Init 和 Load 事件均在页面加载过程中发生。
- 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。
- 若希望事件代码只在页面首次加载时被执行，可以将其放入 Init 事件，或放入 Load 事件并利用 Page.IsPostBack 属性判断是否为首次加载。

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期内，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。若此属性的返回值为 true，则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

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，创建 Cookie 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 Cookie 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双撇号括起来；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。

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

2. WriteFile()方法

功能：将指定的文件内容写入 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，会影响程序的可读性。Response.WriteFile()方法用于直接将文件内容输出到客户端，若要输出的文件和执行的网页在同一个目录中，只需直接传入文件名即可；若不在同一目录，则要指定详细的目录名称。

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.baidu.com");

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

或

```
string ThisURL = "http://www.baidu.com"; Response.Redirect(ThisURL);
```

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 方法的 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，若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；若不在同一目录，则要指定详细的目录名称。

3.3.3 Response 对象的应用

1. 向浏览器发送信息

【案例 3.2】 Response.Write()方法使用示例。

本案例使用 Response.Write()方法输出字符串、HTML 标记及 Script 脚本。

1) 页面设计

前台只需给页面设个标题。

```
<title>Response.Write 方法使用示例</title>
```

2) 后台代码设计

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("<font size='6' color='red' face='黑体'>欢迎来到我的主页</font><br/>
<br/>");
    Response.Write("<hr width='75%' color='blue' align='left'/><br/><br/>");
    Response.Write("现在的时间是: " + DateTime.Now.ToLongTimeString() + "<br/><br/>");
    Response.Write("浏览新闻可以到<a href='http://www.sohu.com'>搜狐网</a><br/><br/>");
```

```
    Response.Write("<script language='javascript'>alert('使用 Write()方法输出信息');
</script>");
}
```

3）运行调试

按 Ctrl＋F5 组合键运行，运行效果如图 3.3 所示。

图 3.3 Response.Write()方法使用示例

2. 页面跳转

【案例 3.3】 使用 Redirect()方法实现页面跳转。

本案例使用 Response 对象的 Redirect()方法转向指定的页面。

1）页面设计

前台只需给页面设个标题。

```
<title>使用 Redirect 方法实现页面跳转</title>
```

2）后台代码设计

```
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.baidu.com";
    else url = "http://www.sohu.com";
    Response.Redirect(url);
}
```

3）运行调试

按 Ctrl＋F5 组合键运行，程序先判断当前日期是偶数还是奇数，如果是偶数，则跳转至百度网站，否则跳转至搜狐网站。

3. 输出文件内容到客户端

【案例 3.4】 使用 WriteFile()方法输出文本文件。

本案例使用 Response 对象的 WriteFile()方法输出指定的文件。

1）页面设计

前台只需给页面设个标题。

```
<title>使用 WriteFile 方法输出文本文件</title>
```

2）后台代码设计

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

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

说明：OutFile.txt 是准备输出的文本文件，存放在网站根文件夹下。

3）运行调试

按 Ctrl＋F5 组合键运行，运行效果如图 3.4 所示。

图 3.4　使用 WriteFile()方法输出文本文件

如果页面出现乱码，则在 Web.config 的< system.web >与</system.web >之间增加以下代码。

```
<globalization requestEncoding = "gbk" responseEncoding = "gbk" culture = "zh-CN" fileEncoding =
"gbk"/>
```

4. 下载文件

【案例 3.5】 使用 TransmitFile()方法下载文件。

本案例使用 Response 对象的 TransmitFile()方法下载 Excel 文件。

1）页面设计

```
<html>
    <head runat = "server"><title>使用 TransmitFile 方法下载文件</title></head>
    <body>
      <form id = "form1" runat = "server">
        <div>
          <asp:LinkButton ID = "LinkButton1" runat = "server" OnClick = "LinkButton1_Click">
下载成绩册
          </asp:LinkButton>
        </div>
      </form>
    </body>
</html>
```

2）后台代码设计

```
protected void LinkButton1_Click(object sender, EventArgs e)
{
    Response.ContentType = "application/msexcel";
    Response.AddHeader("Content-Disposition", "attachment;filename = 成绩册.xls");
    string filename = Server.MapPath("成绩册.xls");
    Response.HeaderEncoding = System.Text.Encoding.GetEncoding("gb2312");
    Response.TransmitFile(filename);
}
```

说明：Response.AddHeader()方法指定文件下载时的默认名，Response.HeaderEncoding()方法设置当前标头输出流的编码。

3）运行调试

按 Ctrl＋F5 组合键运行，运行效果如图 3.5 所示。

图 3.5 使用 TransmitFile()方法下载文件

单击“下载成绩册”按钮，打开“新建下载任务”对话框，选择保存路径后单击“下载”按钮即可实现文件的下载。

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许多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，若在地址栏中输入 news.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"]。

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 属性值需要为 Post,其语法格式为 Request.From["元素名"]。

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,若文件和执行的网页在同一个目录中,直接传入文件名即可;若不在同一目录,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。

3.4.3 Request 对象的应用

1. 获取页面间传送的值

Request.Form 用于表单提交方式为 Post 的情况，而 Request.QueryString 用于表单提交方式为 Get 的情况，如果用错，则获取不到数据。用户可以利用 Request["元素名"]来简化操作。

【案例 3.6】 获取页面间传送的值。

本案例使用 Request 对象的 Form 属性在两个页面间传递登录的用户名。

设计两个页面，在第一个页面 Default.aspx 里登录，在第二个页面 Default2.aspx 里利用 Request["元素名"]来获取用户的登录名。

1）页面设计

第一个页面 Default.aspx：

```
<html>
  <head runat="server"><title>登录界面</title></head>
  <body><form id="form1" runat="server"><div>
    <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="Button1" runat=
"server"PostBackUrl="~/Default2.aspx" Text="登录"/></td></tr>
    </table>
  </div></form></body>
</html>
```

说明：使用 Button1 的 PostBackUrl 属性设置当单击 Button1 时转向 Default2.aspx。

第二个页面 Default2.aspx：

```
<html>
  <head runat="server"><title>欢迎界面</title>
    <style type="text/css">.kk { font-family: 隶书; color: #FF0000; }</style>
  </head>
  <body>
    <form id="form1" runat="server"><div>
      <asp:Label ID="Label1" runat="server" Text="Label" CssClass="kk"></asp:Label>
      <asp:Label ID="Label2" runat="server" Text="Label" CssClass="kk"></asp:Label>
<br />
      <asp:Label ID="Label3" runat="server" Text="欢迎您登录我的网站!" CssClass="kk">
</asp:Label>
    </div></form>
  </body>
</html>
```

2）后台代码设计

第一个页面没有后台代码，第二个页面的后台代码如下。

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

```
{
    Label1.Text = Request["UserName"];          //获取用户登录名
    int Time = DateTime.Now.Hour.CompareTo(13); //将系统时间与数据 13 进行比较来获取问候语
    string str;
    if (Time > 0) { str = "下午好!";}           //系统时间大于 13 显示"下午好!"
    else if (Time < 0) { str = "上午好!";}      //系统时间小于 13 显示"上午好!"
        else{ str = "中午好!";}
    Label2.Text = str;
}
```

3) 运行调试

按 Ctrl+F5 组合键运行,登录界面效果如图 3.6 所示。

输入用户名、密码,单击"登录"按钮后转到欢迎界面,如图 3.7 所示。

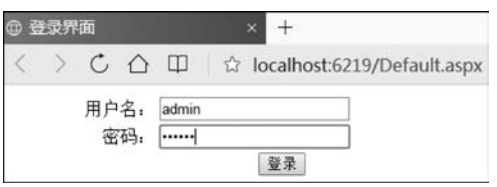

图 3.6 登录界面

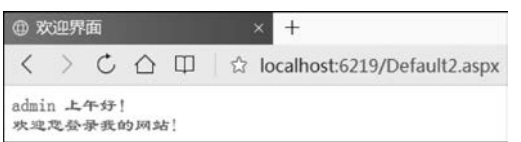

图 3.7 欢迎界面

2. 获取客户端浏览器信息

不同的浏览器或者相同浏览器的不同版本支持的功能不同。在应用程序中用户可能需要知道当前正在使用的是哪种类型的浏览器,并且可能需要知道该浏览器是否支持某些特定的功能。

【案例 3.7】 获取客户端浏览器信息。

本案例主要使用 Request 对象的 Browser 属性获取客户端浏览器信息。

1) 页面设计

在前台页面添加一个页面标题,并给< body >定义了一个样式。

```
< html >
  < head runat = "server">< title >获取客户端浏览器信息</title ></head >
  < body style = "font - weight: bold; color: red; font - family: 隶书; text - align: left;
background - color: #FFFFCC;">
    < form id = "form1" runat = "server">< div ></div >
    </form >
  </body >
</html >
```

2) 后台代码设计

```
protected void Page_Load(object sender, EventArgs e)
{
    Response.Write("当前使用的浏览器信息: "); Response.Write("< hr >");
    Response.Write("浏览器的名称及版本: " + Request.Browser.Type + "< br/>");
    Response.Write("浏览器的类型: " + Request.Browser.Browser + "< br/>");
    Response.Write("浏览器的版本号: " + Request.Browser.Version + "< br/>");
    Response.Write("客户端使用的操作系统: " + Request.Browser.Platform + "< br/>");
    Response.Write("是否支持 HTML 框架: " + Request.Browser.Frames + "< br/>");
    Response.Write("是否支持 JavaScript: " + Request.Browser.JavaScript.ToString() + "< br/>");
    Response.Write("是否支持 Cookies: " + Request.Browser.Cookies + "< br/>");
```

```
    Response.Write("是否支持 ActiveX 控件: " + Request.Browser.ActiveXControls + "<br/>");
    Response.Write("<hr>");
}
```

3）运行调试

按 Ctrl＋F5 组合键运行，运行效果如图 3.8 所示。

图 3.8 获取客户端浏览器信息

3. 获取服务器端环境变量

用户一般习惯通过域名来访问服务器，但服务器之间是靠 IP 地址识别的，域名和 IP 地址是一一对应的，由此可见确定对方的 IP 地址很重要，而且在实际应用中还可以通过 IP 地址搜索到其所在地。

【案例 3.8】 获取服务器端环境变量。

本案例主要使用 Request 对象的 ServerVariables 属性获取服务器端环境变量。

1）页面设计

前台页面只是加了一个页面标题，并给<body>定义了一个样式。

```
<html>
  <head runat="server"><title>获取服务器端环境变量</title></head>
  <body style="font-weight: bold; color: blue; font-family: 隶书; text-align: left;
background-color: #FFFFCC;">
    <form id="form1" runat="server"><div></div>
    </form>
  </body>
</html>
```

2）后台代码设计

```
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）运行调试

按 Ctrl＋F5 组合键运行，运行效果如图 3.9 所示。

图 3.9　获取服务器端环境变量

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

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()方法

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。

显示当前目录的实际路径：Server.MapPath("./");。

显示父目录的实际路径：Server.MapPath("../");。

显示根目录的实际路径：Server.MapPath("/");。

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。

【案例 3.9】 用 Execute()方法执行指定页面。

本案例主要使用 Server 对象的 Execute()方法执行对另一个页面的请求。

1）页面设计

本案例共有两个页面，第一个页面是 Default.aspx。

```
<html>
  <head runat="server"><title>用 Execute()方法执行指定页面</title></head>
  <body>
    <form id="form1" runat="server"><div>
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click"
Text="用 Execute()方法执行指定页面"/></div>
    </form>
  </body>
</html>
```

第二个页面是 TestPage.aspx 的前台界面，无须设计。

2）后台代码设计

第一个页面 Default.aspx 的后台代码如下。

```
protected void Button1_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>");
}
```

图 3.10　用 Execute()方法执行指定页面

第二个页面 TestPage.aspx 的后台代码如下。

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

3）运行调试

按 Ctrl＋F5 组合键运行，运行效果如图 3.10 所示。

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。在案例 3.9 中，如果把第一个页面 Default.aspx 的后台代码改成如下形式：

```
Response.Write("<p>调用 Execute()方法之前</p>");
Server.Transfer ("TestPage.aspx");
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 Default.aspx，因为没有执行第 3 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，这时不能直接在网页中输出，因为会被浏览器解读为 HTML 语言，即对文本进行加粗，而不会将显示出来。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。

【案例 3.10】 HTML 编码和解码。

本案例主要使用 Server 对象的 HtmlEncode()方法进行编码，使用 Server 对象的 HtmlDecode()方法进行解码。

1）页面设计

前台页面只添加了一个页面标题。

```
<html>
  <head runat="server"><title>HTML 编码和解码</title></head>
  <body>
    <form id="form1" runat="server"><div></div></form>
  </body>
</html>
```

2）后台代码设计

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

```
{
    Response.Write(Server.HtmlEncode("<h3>三级标题</h3>"));
    Response.Write("<hr/>");
    Response.Write(Server.HtmlDecode(Server.HtmlEncode("<h3>三级标题</h3>")));
}
```

3）运行调试

按 Ctrl＋F5 组合键运行，运行效果如图 3.11 所示。

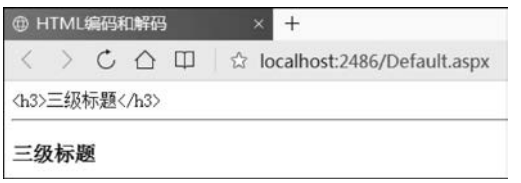

图 3.11 HTML 编码和解码

查看当前网页的源代码，可以看到编码后的字符串。

```
&lt;h3&gt;三级标题 &lt;/h3&gt;<hr/><h3>三级标题</h3>
```

从中可以看出，使用 HtmlEncode 方法进行编码实际上就是将 HTML 标记中的一些特殊符号用特定的标记表示。例如本案例中的“<”用“<”表示，“>”用“>”表示，经过这样的处理后包含 HTML 标记的字符串就可以在浏览器中原样输出。在使用 HtmlDecode 方法进行解码时相应的字符会被转换回来，并呈现在客户端浏览器中。

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\AppData\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，若有，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 的主要优点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，若设为 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 3 个参数，即名称、值和有效期。上面语句中创建的 Cookie 对象的名称为“Name”，值为“张三”，有效期为 1 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，那么可以设置一个比较长的时间，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，若要删除前面创建的 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);
```

4. 利用 Cookie 实现密码记忆功能

【案例 3.11】 利用 Cookie 实现密码记忆功能。

本案例主要使用 Cookie 对象在登录时记住密码。

1）页面设计

```
< html >
  < head runat = "server">< title >利用 Cookie 实现密码记忆功能</title ></head >
  < body >
    < form id = "form1" runat = "server">< div >
      < asp:Label ID = "Label1" runat = "server" Text = "用户名" width = "60px"></asp:Label >
      < asp:TextBox ID = "TextBox1" runat = "server"></asp:TextBox >< br />
      < asp:Label ID = "Label2" runat = "server" Text = "密码" width = "60px"></asp:Label >
      < asp:TextBox ID = "TextBox2" runat = "server" TextMode = "Password"></asp:TextBox >< br />
      < asp:CheckBox ID = "CheckBox1" runat = "server" Text = "记住密码" TextAlign = "Left"/>< br />
      < asp:Button ID = "Button1" runat = "server" OnClick = "Button1_Click" Text = "登录"/>
      < asp:Button ID = "Button2" runat = "server" Text = "重置"/></div >
    </form >
  </body >
</html >
```

2）后台代码设计

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Request. Cookies["password"] != null)
    {
        if (DateTime. Now. CompareTo(Request. Cookies["password"]. Expires) > 0)
        {
            TextBox2. Text = Request. Cookies["password"]. Value;
        }
    }
}
protected void Button1_Click(object sender, EventArgs e)
{
    if (CheckBox1. Checked)
    {
        Response. Cookies["password"]. Value = TextBox2. Text;
        Response. Cookies["password"]. Expires = DateTime. Now. AddSeconds(10);
    }
}
```

说明：为了让案例效果明显，这里特意把 Cookie 变量的有效期设为 10s。

3）运行调试

按 Ctrl＋F5 组合键运行，首次访问时两个文本框均为空，输入用户名和密码后选中“记住密码”复选框，单击“登录”按钮，运行效果如图 3.12 所示。关闭浏览器，在 10s 内重新登录该页面，可以看到密码框内已经记住了密码。

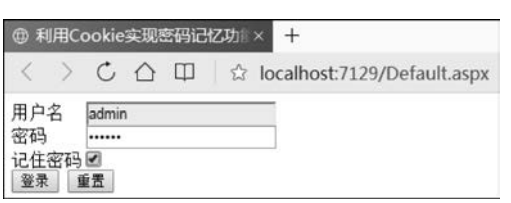

图 3.12 利用 Cookie 实现密码记忆功能

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。当下次加载的时候判断有没有这个密码 Cookie,如果有再判断这个 Cookie 是否过期,若未过期,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 10s,这样可以使看到的效果明显一些。在 10s 之前,密码部分还一直有值,过了 10s 就自动清空了,因为 Cookie 到期了。

5. 利用 Cookie 控制投票次数

【案例 3.12】 利用 Cookie 控制一天内投一次票。

本案例主要使用 Cookie 对象控制投票次数。

1) 页面设计

```
<html>
  <head runat="server"><title>利用 Cookie 控制一天内投一次票</title></head>
  <body>
    <form id="form1" runat="server"><div>
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="我要投票"/>
</div>
    </form>
  </body>
</html>
```

2) 后台代码设计

```
protected void Button1_Click(object sender, EventArgs e)
{
    string UserIP = Request.UserHostAddress.ToString();
    HttpCookie oldCookie = Request.Cookies["userIP"];
    if (oldCookie == null)
    {
        Response.Write("<script>alert('投票成功,谢谢您的参与!')</script>");
        HttpCookie newCookie = new HttpCookie("userIP");      //定义新的 Cookie 对象
        newCookie.Expires = DateTime.Now.AddDays(1);
        newCookie.Values.Add("IPaddress", UserIP);
                                                  //添加新的 Cookie 变量 IPaddress,值为 UserIP
        Response.AppendCookie(newCookie);                     //将变量写入 Cookie 文件中
        return;
    }
    else
    {
        string userIP = oldCookie.Values["IPaddress"];
        if (UserIP.Trim() == userIP.Trim())
        {
            Response.Write("<script>alert('一个 IP 地址一天内只能投一次票,谢谢您的参与!');
history.go(-1);</script>");
            return;
        }
        else
        {
            HttpCookie newCookie = new HttpCookie("userIP");
            newCookie.Values.Add("IPaddress", UserIP);
            newCookie.Expires = DateTime.Now.AddDays(1);
            Response.AppendCookie(newCookie);
```

```
            Response.Write("<script>alert('投票成功,谢谢您的参与!')</script>");
            return;
        }
    }
}
```

3）运行调试

按 Ctrl＋F5 组合键运行，单击“我要投票”按钮后，程序首先判断用户是否在本日已投过票，如果用户是第一次投票，则利用 Cookie 对象保存用户的 IP 地址，并弹出对话框提示用户投票成功；如果用户已投过票，则弹出对话框提示用户已投过，如图 3.13 所示。

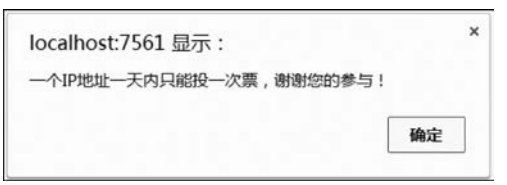

图 3.13 提示重复投票的对话框

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：

- 关闭浏览器窗口；
- 断开与服务器的连接；
- 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 20min。

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

该事件当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,所有的内建对象(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 及 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"];
```

3. 使用 Session 对象进行页面间传值

【案例 3.13】 使用 Session 对象进行页面间传值。

本案例主要使用 Session 对象在两个页面之间传送密码的值。

1）页面设计

登录页面 Default.aspx：

```
<html>
  <head runat="server"><title>登录页面</title></head>
  <body>
    <form id="form1" runat="server"><div>
      <asp:Label ID="Label1" runat="server" Text="用户名" width="60px"></asp:Label>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
      <asp:Label ID="Label2" runat="server" Text="密码" width="60px"></asp:Label>
      <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox><br />
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="登录"/>
      <asp:Button ID="Button2" runat="server" Text="重置"/></div>
    </form>
  </body>
</html>
```

欢迎页面 Welcome.aspx 只设置了一个页面标题：

```
<title>欢迎页面</title>
```

2）后台代码设计

登录页面 Default.aspx 的后台代码：

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text != "" && TextBox2.Text != "")
    {
        Session["username"] = TextBox1.Text; Session["password"] = TextBox2.Text;
        Response.Redirect("Welcome.aspx");} else
        Response.Write("<script language='javascript'>alert('用户名或密码不能为空!');
</script>");
    }
}
```

欢迎页面 Welcome.aspx 的后台代码：

```
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);
    }
}
```

3）运行调试

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,然后转向欢迎页面 Welcome.aspx。

在 Welcome.aspx 中获取并显示在前一个页面用 Session 变量保存的用户名和密码,如图 3.15 所示。

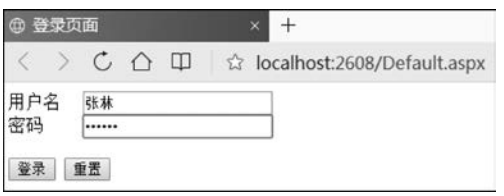

图 3.14 登录页面

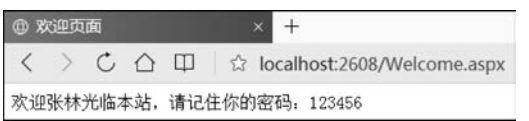

图 3.15 欢迎页面

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 变量,防止其他客户端更改 Application 变量的值。

语法:`Application.Lock();`。

6. UnLock()方法

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

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

1. 统计网站在线人数

【案例 3.14】 统计网站在线人数。

本案例通过 Application 对象和 Session 对象统计当前在线用户数量。

1）页面设计

在 Default.aspx 页面中添加一个 Label 控件，用来显示当前在线人数，ID 采用默认名称。

2）新建全局应用程序类文件 Global.asax

右击网站根文件夹，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)
{
    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)
{
    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)
{
    Application.Lock();Application["counter"] = (int)Application["counter"] - 1;
    Application.UnLock();
}
```

3）后台代码设计

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

```
    if (!IsPostBack) Label1.Text = "当前在线人数: " + Application["counter"].ToString();
}
```

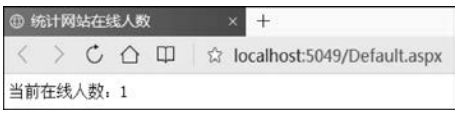

图 3.16　统计网站在线人数

4) 运行调试

按 Ctrl+F5 组合键运行,运行效果如图 3.16 所示。

2. 统计网站总访问量

【案例 3.15】 统计网站总访问量。

本案例通过 Application 对象和对文件的读/写操作来统计网站的总访问量。

1) 关键技术

在实现统计网站的总访问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()从当前流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,将网站总访问量保留到其中。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) 页面设计

```
<html>
  <head runat="server"><title>统计网站总访问量</title></head>
  <body>
  <form id="form1" runat="server"><div>
    <table style="width:100%;">
        <tr><td style="text-align: center">统计网站总访问量</td></tr>
        <tr><td style="text-align: center">您是第<%=Application["counter"]%>位访问
者</td></tr>
      </table></div>
    </form>
  </body>
</html>
```

3) 新建全局应用程序类文件 Global.asax

右击网站根文件夹,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)
{
    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到 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总访问量加 1，代码如下。

```
void Application_End(object sender, EventArgs e)
{
    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总访问量存放到文件中，代码如下。

```
void Session_Start(object sender, EventArgs e)
{
    Application.Lock();int Stat = 0;
    Stat = (int)Application["counter"]; Stat += 1;
    object obj = Stat; Application["counter"] = obj;
    string file_path = Server.MapPath("counter.txt");
    StreamWriter srw = new StreamWriter(file_path, false);
    srw.WriteLine(Stat); srw.Close();Application.UnLock();
}
```

4）运行调试

按 Ctrl＋F5 组合键运行，Default.asp 页面加载时通过代码块<%＝Application["counter"]%>将总访问量显示在页面上，运行效果如图 3.17 所示。

图 3.17 统计网站总访问量

3. 利用 Application 对象实现聊天功能

【案例 3.16】 一个简单的在线聊天室。

1）页面设计

设计两个页面，分别是登录页面 Login.aspx 和在线聊天页面 Default.aspx。

登录页面 Login.aspx 的前台代码：

```
<html>
  <head runat="server"><title>用户登录</title></head>
  <body>
    <form id="form1" runat="server"><div>
      <asp:Label ID="Label1" runat="server" Text="用户名" width="60px"></asp:Label>
      <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><br />
      <asp:Label ID="Label2" runat="server" Text="密码" width="60px"></asp:Label>
      <asp:TextBox ID="TextBox2" runat="server" TextMode="Password"></asp:TextBox><br />
      <asp:Button ID="Button1" runat="server" OnClick="Button1_Click" Text="登录"/>
      <asp:Button ID="Button2" runat="server" Text="重置"/></div>
```

```
    </form>
  </body>
</html>
```

在线聊天页面 Default.aspx 的前台代码：

```
<html>
<head runat="server"><title>用户登录</title></head>
<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="Label1" runat="server" Text="Label"></asp:Label></div>
    <div><asp:TextBox ID="TextBox1" 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="Label"></asp:Label>  
  
      <asp:TextBox ID="TextBox2" 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="Button1" runat="server" Text="提交" OnClick="Button1_Click"/>
</div>
    </form>
  </body>
</html>
```

2）新建全局应用程序类文件 Global.asax

右击网站根文件夹，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)
{
    Application["online"] = 0;        //在线人数初始值为 0
    Application["chat"] = "";         //聊天内容初始值为空
}
```

当新会话启动时在线人数加 1，代码如下。

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

当会话结束时在线人数减 1，代码如下。

```
void Session_End(object sender, EventArgs e)
{  Application.Lock();Application["online"] = (int)Application["online"] - 1;
   Application.UnLock();}
```

3）后台代码设计

（1）登录页面 Login.aspx 的后台代码：

```
protected void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text != "" || TextBox2.Text != "")
    {
        Session["name"] = TextBox1.Text; Response.Redirect("Default.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，如果用户名或密码没填，就弹出提示框，提示用户“用户名或密码不能为空!”；如果都不为空，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 Default.aspx。

（2）在线聊天页面 Default.aspx 的后台代码：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (Session["name"] != null)
    {
        Label1.Text = "当前在线人数为: " + Application["online"].ToString();
        TextBox1.Text = Application["chat"].ToString();
        Label2.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，则在标签控件 Label1 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 Label2 中显示用户的名字，并且设置页面自动刷新时间为 30s。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。

```
protected void Button1_Click(object sender, EventArgs e)
{
    string newmessage = Session["name"] + ": " + DateTime.Now.ToString()
      + "\r" + TextBox2.Text + "\r" + Application["chat"];
    if (newmessage.Length > 500)
      newmessage = newmessage.Substring(0,499);
    Application.Lock();
    Application["chat"] = newmessage;
    Application.UnLock();
    Label2.Text = "";
    TextBox1.Text = Application["chat"].ToString();
}
```

上述代码主要实现将用户发表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存最新的 500 个字符。

4) 按 Ctrl+F5 组合键运行,进入登录页面

输入用户名和密码,登录成功后页面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的聊天内容,如图 3.18 所示。

图 3.18　在线聊天界面

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。两者不同的是,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 20min 内没有访问 Session,那么 Session 的生命周期被销毁;但是,如果在 20min 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 对象、Session 对象和 Cookie 对象的区别如表 3.1 所示。

表 3.1　Application 对象、Session 对象和 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	整个应用程序的生命周期	所有用户	服务器端
Session	小量,简单的数据	默认 20min,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

习题 3

1. 填空题

(1) Request 对象的主要功能是从_____接收信息。

(2) 下面是设置和取出 Session 对象的代码。

设置 Session 的代码：Session["greeting"]="hello world !";

取出该 Session 对象的代码：string Mystr=_____;

(3) 下面是网页中的指令，目的是在网页中显示"新网页的 Url"字符串：

Response._____("新网页的 Url");

(4) 可以用 Session 对象的_____属性区分不同的用户会话。

(5) 设置 Session 对象有效时间的属性是_____。

(6) _____对象可以用来存储 ASP.NET 应用程序中的全局共享信息。

(7) 下面是使用 Application 对象时防止竞争的代码。

```
Application._____;                    //锁定 Application 对象
Application["counter"] = (int) Application["counter"] + 1;
Application._____;                    //解除对 Application 对象的锁定
```

(8) 如果要将虚拟文件路径转换为实际文件路径，可以使用 Server 对象的_____方法。

(9) Request 对象的_____属性可以获取标识在 URL 后面的变量及其值。

(10) Session 对象启动时会触发_____事件。

2. 单项选择题

(1) 通过 Request 对象的_____属性可以取得客户端的浏览器版本。

A. Browser　　B. ServerVariables　　C. QueryString　　D. Form

(2) 下面代码可以在客户端浏览器输出信息的是_____。

A. Request.QueryString["user_name"]　　B. Response.Write("春秋")

C. Response.Redirect("index.aspx")　　D. Request.Form["user_name"]

(3) Cookies 是保存在客户硬盘上的_____文件。

A. 图片　　B. 视频　　C. 文本　　D. 以上都不是

(4) Session 对象的默认有效期为_____分钟。

A. 10　　B. 15

C. 20　　D. 应用程序从启动到结束

(5) Global.asax 文件中的 Session_Start 事件_____。

A. 在每个请求开始时激发　　B. 尝试对使用进行身份验证时激发

C. 启动新会话时激发　　D. 在应用程序启动时激发

(6) Session 与 Cookie 之间最大的区别在于_____。

A. 存储的位置不同　　B. 类型不同

C. 生命周期不同　　D. 容量不同

(7) 下面的程序段执行完毕后页面上显示的内容是_____。

```
Response.Write ("春秋");
Response.End();
Response.Write ("战国");
```

A. 春秋　　B. 战国

C. 春秋战国　　D. 春秋(换行)战国

(8) 假定当前工作路径是"D:/aspnet/ch03",使用 Server.MapPath("../ex/kk.mdb")获取的数据库物理路径是_____。

A. D:\aspnet\ex\kk.mdb　　B. D:\ch03\ex\kk.mdb

C. D:\ex\kk.mdb　　D. D:\aspnet\ch03\ex\kk.mdb

(9) 设置 Cookie 有效期使用的属性是_____。

A. Timeout　　B. Expires　　C. Value　　D. Count

(10) 下列关于 ASP.NET 内置对象的说法中不正确的是_____。

A. Application 和 Session 信息都保存在服务器端

B. Cookie 信息保存在客户端

C. Session 对象具有 TimeOut 属性

D. Cookie、Application 和 Session 信息都保存在客户端

3. 上机操作题

(1) 请设计一个页面,显示来访者的 IP 地址,如果 IP 地址以 219.139 开头,显示欢迎信息;否则显示为非法用户,并终止程序。

(2) 请设计一个页面,当客户第一次访问时需要在线注册姓名、性别等信息,然后把信息保存到 Cookies 中。如果下一次该客户再访问,则显示"某某,您好! 您是第几次访问本站"的欢迎信息。

(3) 请设计两个页面,在第一个页面中用户输入姓名,然后保存到 Session 中;在第二个页面中读取该 Session 信息,并显示欢迎信息。如果用户没有在第一页登录就直接访问第二页,则将用户重定向到第一页。