第 3 章　ASP.NET 内置对象

Web 服务器与浏览器之间的连接是一种“无状态连接”。Web 应用程序不能像 WinForms 应用程序那样保持客户端状态。通常，在 Web 应用程序中都使用内置对象进行客户端状态保持。本章将学习 ASP.NET 中常见的内置对象，使用 ASP.NET 内置对象为程序员开发 Web 应用程序带来了极大的方便。

- Page 对象
- Response 对象
- Request 对象
- Server 对象
- Application 对象
- Session 对象
- Cookie 对象

3.1　内置对象概述

在 ASP.NET 页面中包含一系列可以直接使用的类，称之为内置对象。在面向对象编码中，对象是一个封装了数据和业务代码的实体。每个对象都有自己的方法和属性，用来完成特定的功能。在使用对象时一般不需要了解对象内部是如何实现的，只需要知道对象的主要功能即可。

ASP.NET 中提供了大量的内置对象，通过调用这些对象的方法和属性可以实现丰富的功能，如页面间的数据传递、页面的跳转、数据流的输出、用户信息的保存等。表 3-1 列出了 ASP.NET 中常见的内置对象及说明。

表 3-1　ASP.NET 中常见的内置对象

名称	说明
Page	用于设置与网页有关的属性、方法和事件
Response	向浏览器输出信息
Request	从客户端或浏览器获取信息
Server	提供服务器端最基本的属性与方法
Application	存储客户端共享信息
Session	存储单个用户信息
Cookie	在客户端保存访问者信息

3.2　Page 对象

Page 页面类继承自 System.Web.UI.Page 类，每一个 ASP.NET 的页面窗体（WebForm）对应一个页面类。Page 对象就是页面类的一个实例，可以通过 Page 对象设置与网页有关的各种属性、方法和事件。

3.2.1　@Page 指令

@Page 指令允许为 ASP.NET 页面（.aspx）指定解析和编译页面时使用的属性和值。

下面是使用@Page 指令的一个示例：

```
<%@ Page Language="C#" AutoEventWireup="true"  CodeFile="Default.aspx.cs"
Inherits="_Default" %>
```

在 Visual Studio 的集成开发环境（IDE）中，每一个新建的页面.aspx 文件都有两种视图：设计视图和源视图。打开页面.aspx 文件源视图，在第一行可以看到上面的示例代码。

示例中各个参数代表的含义：

- Language：指定脚本块和内置显示所使用的语言，如示例中指明使用的是“C#”语言，需要注意的是，这里只支持微软.NET 框架中的语言。
- AutoEventWireup：设置页面是否自动调用网页事件，默认为 true。
- CodeFile：引用与页面相关的后台编码文件。
- Inherits：页面类。

例 3-1　@Page 指令 AutoEventWireup 参数的使用。

操作步骤：

（1）新建一个名称为 ch3-1 网站项目，并将该项目下的默认窗体名称改为 ex3-1.aspx.。

（2）将窗体切换到“源视图”。保证@Page 指令的 AutoEventWireup="true"。设置页面自动调用网页事件。

（3）为 ex3-1.aspx 添加 Page_Load 事件，并设置断点，如图 3-1 所示。

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

图 3-1　设置调试断点

（4）启动调试（按快捷键 F5），程序进入调试模式，如图 3-2 所示。

```
public partial class _Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {

    }
}
```

图 3-2 进入调试模式

（5）将窗体切换到“源视图”。保证@Page 指令的 AutoEventWireup="false"。禁用页面自动调用网页事件。启动调试，程序则不进入调试模式。

3.2.2 Page 对象的常用属性

在实际的程序应用中，Page 对象常用的属性和说明如表 3-2 所示。

表 3-2 Page 对象的常用属性

属性	说明
IsPostBack	只读的 bool 类型的值，用于判断网页在何种情况下加载。为 true 时表示网页是客户端返回数据而“回发”的。为 false 时表示网页是第一次加载
IsValid	只读的 bool 类型的值，用于判断网页上的验证控件是否全部验证成功。为 true 时表示全部验证成功，为 false 时表示至少有一个验证控件验证失败
EnableViewState	可读/写 bool 值，用于获取或设置当前网页结束时，网页是否保持视图状态及其所包含的服务器控件的视图状态，默认值为 true

ASP.NET 的信息处理是传回服务器端进行处理。ASP.NET 不同于静态 HTML 技术的地方是采用了“回发”技术。在 HTML 中，当用户单击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，若要获取当前页内的数据可以使用 Request 的方式获取所有提交过来的内容（详见本章 3.3 节 Request 对象）。但是，在 ASP.NET 中这些不需要这样处理，ASP.NET 已经封装好了，可以直接使用“控件.属性”的方式直接访问控件的相关内容。

因此，有了“回发”（PostBack）的机制，编写 ASP.NET 访问页内数据的页面可以像 WinForms 那样直观快捷。不需要在 HTML 标签中设置 Post 或 Get 方式提交，也不用编写 Request 代码，就可以直接使用控件的属性值。

例 3-2 Page 对象的 IsPostBack 属性的使用。

操作步骤：

（1）新建一个 ch3-2 的网站项目，并将该项目下的默认窗体名称改为：ex3-2.aspx。

（2）界面设计。从 HTML 工具箱中选择一个 table 控件，从“标准”工具箱中选择两个 TextBox 控件、一个 Button 控件。在属性窗口中分别设置控件相应的属性，如表 3-3 所示。

表 3-3 窗体文件 ex3-2.aspx 控件的属性

控件类别	控件	属性名称	属性值	备注
Web 控件	Button	ID	btnSubmit	提交表单
		Text	登 录	
	TextBox	ID	txtUserName	用户名
		Text	空	
	TextBox	ID	txtPwd	密码
		Text	空	
		TextMode	PassWord	
HTML	table	五行两列的表格		

设计最终界面如图 3-3 所示。

图 3-3　用户登录界面

本章节中多个例子采用本例的界面设计。

（3）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-2.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

```
this.txtUserName.Text = "此处输入用户名";
this.txtPwd.Text = "此处输入用户密码";
```

（4）按快捷键 F5 执行程序，并在用户名文本框中输入“张三”，单击“登录”按钮，执行结果如图 3-4 所示。步骤（3）所示的代码本意是指示用户名和密码的输入位置，Page_Load 事件是页面加载事件，当页面加载时该段代码就能执行。也就是说，当用户单击“登录”按钮后，页面回传，这段代码又一次被执行，所以无法获取到用户输入的内容。

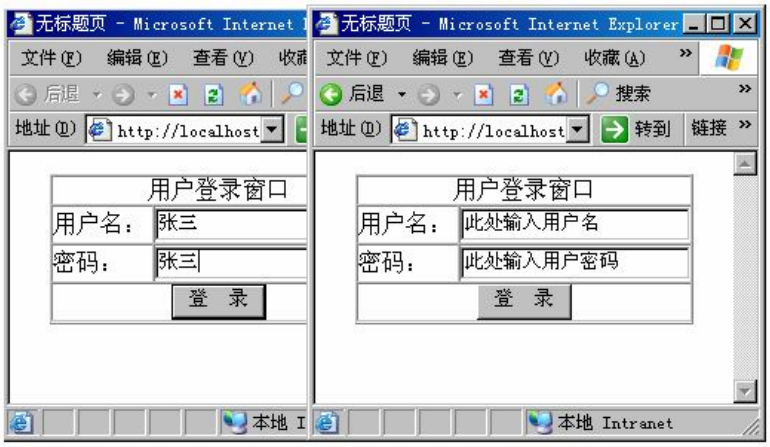

图 3-4　用户登录执行结果 1

（5）修改 Page_Load 事件处理程序的代码，如下所示：

```
if (!Page.IsPostBack)
{
        this.txtUserName.Text = "此处输入用户名";
        this.txtPwd.Text = "此处输入用户密码";
}
```

Page.IsPostBack 用来判断是否是第一次加载，当 IsPostBack 值为 true 时，页面为用户回传，为 false 时是首次加载。在使用过程中 Page.IsPostBack 可以写成 this.IsPostBack 或者 IsPostBack。

（6）按快捷键 F5 执行程序，并在用户名文本框中输入“张三”，单击“登录”按钮，执行结果如图 3-5 所示。

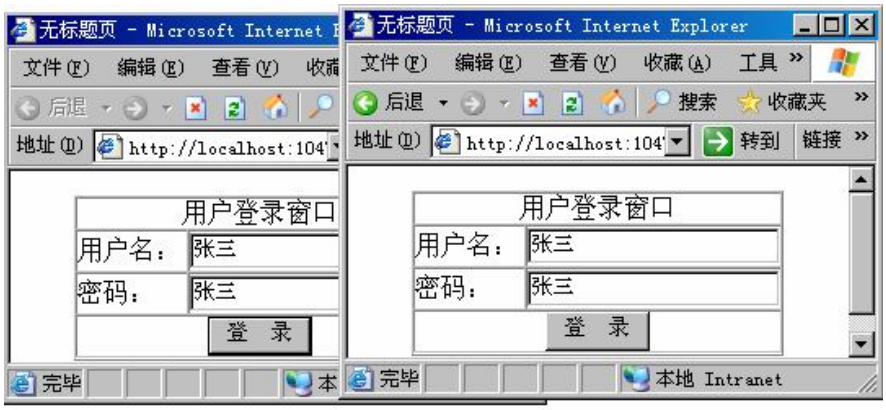

图 3-5　用户登录执行结果 2

3.2.3　Page 对象常用的事件

从 ASP.NET 网页开始载入到最终完全在浏览器中显示的过程中，产生与 Page 对象有关的主要事件有 Init、Load 和 UnLoad 三个事件。ASP.NET 网页执行的流程如图 3-6 所示。

（1）ASP.NET 网页在执行初始化时，首先触发 Init 事件。当 Init 事件发生时，.aspx 源文件中静态声明的所有控件都已被实例化并采用各自的默认值。

（2）当 Web 服务器加载网页时触发 Load 事件。本事件主要生成页面的 HTML 代码。在此事件中所有的控件都已被实例化，所有对象的属性和方法都可以通过代码引用或设置。

（3）在页面上所有 ASP.NET 代码已经执行并完成任务后触发 UnLoad 事件。当用户看到页面时，已经执行了 UnLoad 事件。UnLoad 事件的主要功能是给用户提供进行最后整理的机会，比如在将页面传给用户之前关闭对象。由于 UnLoad 事件是在其他 ASP.NET 代码完成之后执行的，因此不能用控件或 ASP.NET 语句在所关联的 UnLoad 函数中输出任何数据或发送 HTML。

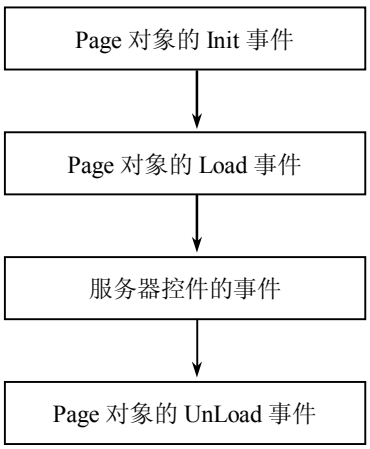

图 3-6　ASP.NET 网页执行流程

例 3-3　Page 对象 Init 和 Load 事件。

操作步骤：

（1）新建一个 ch3-3 的网站项目，并将该项目下的默认窗体名称改为：ex3-3.aspx。

（2）界面设计。从“标准”工具箱中选择一个 Label 控件。在属性窗口中分别设置控件相应的属性，如表 3-4 所示。

表 3-4　窗体文件 ex3-3.aspx 控件的属性

控件类别	控件	属性名称	属性值	备注
Web 控件	Label	ID	LblMsg	显示信息
		Text	空	

（3）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-3.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

```
this.lblMsg.Text += "->触发了 Load 事件";
```

（4）编写 Page_Init 事件处理程序。在代码页添加如下代码：

```
protected void Page_Init(object sender, EventArgs e)
    {
       this.lblMsg.Text+="->触发了 Init 事件";
 }
```

（5）按快捷键 F5 执行程序。执行结果如图 3-7 所示。

图 3-7　Init、Load 事件演示程序

从实验结果看，对于 Label 控件的文字首先初始化为“->触发了 Init 事件”，也就是说先执行了 Page_Init 事件处理程序。

3.3　Response 对象

Response 对象是 HttpResponse 类的一个实例。Response 对象提供了动态响应信息，利用它可以动态响应客户端的请求，并将动态生成的响应结果返回给客户端浏览器。由于 Response 对象映射到 Page 对象的 Response 属性，在 ASP.NET 页面中可以直接使用。

3.3.1　Response 对象常用方法

Response 对象常用方法及说明如表 3-5 所示。

1. Response.Write 方法

- 语法格式：Response.Write(aParam)
- 参数说明：aParam 可以是 C#中任何合法的数据类型（包括脚本字符串）。

表 3-5　Response 对象常用方法

方法	说明
Write	将指定字符串或表达式的结果写到当前的 HTTP 输出
WriteFile	将指定的文件直接写入 HTTP 内容输出流
Redirect	重定向浏览器
Clear	清除缓冲区中 HTTP 输出
End	将当前缓冲区中的所有内容送到客户端然后关闭
AppendToLog	将自定义日志信息添加到 IIS 日志文件

例 3-4　使用 Response.Write 向客户浏览器端输出信息。

操作步骤：

（1）新建一个 ch3-4 的网站项目，并将该项目下的默认窗体名称改为：ex3-4.aspx。

（2）界面设计。界面与例 3-2 相同。

（3）为登陆按钮添加事件处理程序，代码如下：

```
string loginName;
       string loginPwd;
       loginName = this.txtUserName.Text;
       loginPwd = this.txtPwd.Text;
       if (loginName.Trim() == "admin" && loginPwd == "admin")
       {
           Response.Write("<center>"); //输出 center 标记
           Response.Write("<font color=red>"); //输出 font 标记
           Response.Write("Welcome " + loginName); //输出字符串
           Response.Write("</font>");
           Response.Write("</center>");
           Response.Write("<hr>"); //输出水平线
       }
       else
       {
           Response.Write("<script>alert('用户名或密码不正确')</script>"); //输
出 script 脚本
}
```

Response.Write 参数中如果要输出特殊字符要使用转义字符。例如，输出一个双引号，则应使用\"。例如，Response.Write("你说\"Hi!\"");输出结果为：你说"Hi! "。

（4）按快捷键 F5 调试程序，运行结果如图 3-8 和图 3-9 所示。

图 3-8　Response.Write 输出 Script 脚本

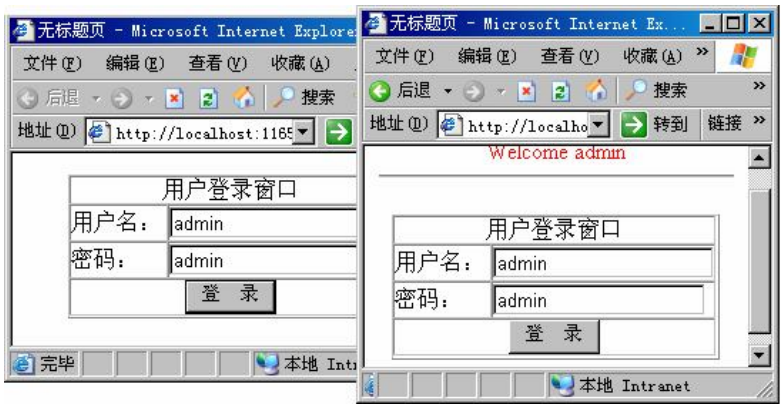

图 3-9　Response.Write 输出字符串和 HTML 标记

程序说明：本示例中 Response.Write 方法输出的内容有 HTML 标记、普通字符串、Script 脚本。

2. Response.Redirect 方法

- 语法格式：Response.Redirect(URL)
- 参数说明：URL 为重定向后的目标地址。在 URL 地址中，可以加入查询字符串。格式为：?变量名=值。如果有多个变量，变量与变量之间用“&”符号分隔。

例如：

```
Response.Redirect("index.htm");//站内网页
Response.Redirect("www.sina.com.cn"); //普通 URL
Response.Redirect("index.htm?loginName=admin&loginPwd=admin");//含变量 URL
```

3. Response.WriteFile 方法

- 语法格式：Response.WriteFile(filePath)
- 参数说明：filePath，输出文件的路径。如果所要输出的文件和网页在同一个目录下，只需要直接传入文件名即可，如果不是在同一个目录下，则要指定详细的目录名称。

例 3-5　Response.WriteFile 的使用。

（1）新建一个 ch3-5 的网站项目，并将该项目下的默认窗体名称改为：ex3-5.aspx。

（2）在网页所在目录新建一个文本文档。文本文档的内容如下：

```
<table align="center" border="1" cellpadding="0" cellspacing="0">
      <tr align="center">
          <td colspan="2">LgoinForm</td>
          <td colspan="1"> </td>
      </tr>
      <tr>
          <td style="width: 67px">UserName:</td>
          <td><input id="txtUserName" name="txtUserName" type="text" /></div>
</td>
          <td></td>
     </tr>
     <tr>
        <td style="width: 67px">Password:</td>
        <td><input id="txtPwd" name="txtPwd" type="text" /></div></td>
        <td></td>
        </tr>
```

```
            <tr align="center">
                <td colspan="2">
                    <input id="Submit1" type="submit" value="submit" /></td>
                <td colspan="1"></td>
            </tr>
</table>
  <br>
     <hr width=80%>
<center>
  <font color=red>
```

说明：本次页面输出的内容包含表格等均来自 WriteFile.txt 文档。

```
</font>
</center>
```

（3）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-2.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

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

（4）按快捷键 F5 运行程序。运行结果如图 3-10 所示。

图 3-10　Response.WriteFile 实验结果

由于浏览器和 ASP.NET 应用程序所使用的编码方式不一致，可能会导致汉字不能够正确显示。ASP.NET 应用程序默认采用的是 UTF-8 的编码方式。解决方法可以为配置文件 Web.config 文件<System.web>节添加<globalization requestEncoding="gb2312" responseEncoding="gb2312"/>。然后保存 Web.config 文件即可。

4. Response.Clear 方法

- 语法格式：Response.Clear()

Response.Clear 方法是将缓冲区中的内容清除。Response.Clear 方法的示例详见 Response 对象的 BufferOutPut 属性。

3.3.2　Response 对象常用属性

Response 对象常用属性及其说明如表 3-6 所示。

表 3-6　Response 对象常用属性

属性	说明
BufferOutPut	设置 HTTP 输出是否启用缓存处理，默认为 true
Charset	设置或获取 HTTP 的输出字符编码
IsClientConnected	返回客户端是否仍然和服务器连接
Status	设置或获取传递服务器 HTTP 的响应状态，默认是 200

1. BufferOutPut 属性

- 语法格式：Response.BufferOutPut=true/false

Response 对象的 BufferOutPut 属性的默认值是 true，要输出到客户端的数据默认暂时都保存在缓冲区中，等到所有的事件程序以及所有的页面对象全部解释完后，才将所有在缓冲区中的数据发送到客户端。

例 3-6　Response.BufferOutPut 的使用。

操作步骤：

（1）新建一个 ch3-6 的网站项目，并将该项目下的默认窗体名称改为：ex3-6.aspx。

（2）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-6.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

```
Response.Write("清除缓冲区前的数据</br>");
    Response.Clear();
Response.Write("清除缓冲区后的数据</br>");
```

（3）按快捷键 F5 运行程序。运行结果如图 3-11 所示。

图 3-11　BufferOutPut 的使用 1

程序结果说明：本例中首先执行 Response.Write("清除缓冲区前的数据</br>");将数据送入缓冲区，接着执行 Response.Clear();将缓冲区中的数据清除，所以清除缓冲区前的数据未能展现在客户端。

（4）在 Response.Write("清除缓冲区前的数据</br>");前加上一句：

```
Response.Write("清除缓冲区前的数据</br>");
```

然后再运行程序，则得到如图 3-12 所示的结果。

程序结果说明：这次并没有将使用 Clear 方法前的数据清除，由此可见，数据是直接输出到客户端并没有放入缓冲区。

2. Status 属性

- 语法格式：Response.Status=StatusExpression

图 3-12　BufferOutPut 的使用 2

StatusExpression 表示状态描述字符串，该字符串包含表示状态的 3 位数字和简短的说明字符。Status 属性所表示的状态字符串如表 3-7 所示。

表 3-7　Status 属性状态字符串

状态值	状态描述	状态值	状态描述
200	OK	401	UNAUTHORIZED
201	CREATED	403	FORBIDDEN
202	ACCEPTED	404	PAGE NOT FOUND
204	NO CONTENT	500	INTERANT SEVER ERROR
301	MOVED PERMANENTLY	501	NOT IMPLEMENTED
302	MOVED TEMPORARILY	502	BAD GATE WAY
304	NOT MODIFIED	503	SERVICE UNAVAILABLE
400	BAD REQUET		

3.4　Request 对象

Request 对象主要是让服务器取得客户端浏览器的一些数据，可以访问 HTML 表单的数据和通过 URL 发送的参数列表的数据以及来自用户的 Cookie 的信息。Request 属性提供对 HttpRequest 类的属性和方法的编程访问。

Request 对象的属性和方法比较多，表 3-8 列出了常用的属性。

表 3-8　Request 对象常用的属性

属性	说明
ApplicationPath	返回当前正在执行程序的服务器的虚拟目录名称
Broswer	返回客户端浏览器的功能信息
ClientCertificate	返回客户端安全认证的信息
ConnectionID	返回当前客户端所发出的网页浏览请求的 ID
ConnectEncoding	返回客户端所支持的字符设定
ContentType	返回目前请求的 MIME 内容类型
Cookies	返回一个 HttpCookieCollection 对象集合
FilePath	返回当前执行网页的相对路径

续表

属性	说明
Files	返回客户端上传的文件集合
Form	返回有关表单变量的集合
Params	返回 QueryString、Form、ServerVariables 以及 Cookies 全部的集合
PhysicalPath	返回当前请求网页在服务器端的真实路径
QueryString	返回附在 URL 查询字符串中参数的内容
Url	返回有关当前请求的 Url 信息
UserAgent	返回客户端浏览器的版本信息
UserHostAddress	返回远程客户端机器的主机 IP 地址
UserHostName	返回远程客户端机器的主机名称
UserLanguage	返回客户端机器使用的语言

3.4.1　Request 对象获取客户端数据

1. 使用 QueryString 集合获取客户端表单数据

当表单的 Method 方法设置为“Get”时，表单数据将以查询字符串的形式附加在 URL 之后，此时应使用 Request 对象的 QueryString 集合来获取表单数据。

- 语法格式：Request.QueryString["表单元素名"]

例 3-7　利用 QueryString 集合获取客户端数据。

操作步骤：

（1）新建一个 ch3-7 的网站项目，并将该项目下的默认窗体名称改为：ex3-7.aspx。

（2）向项目添加一个 HTML 文件（右键单击解决方案中项目的名称→选择“添加新项”→在项目模板中选择“HTML 文件”），将 HTML 文件命名为 ex3-7.html。切换到 ex3-7.html 的“源”视图，在 HTML 文件中输入如下代码：

```
<form name="frmEx37" method="get" action="ex3-7.aspx"></form>
```

（3）界面设计。将例 3-6 中 WriteFile.txt 文件中的整个 table 标记复制到

```
<form name="frmEx37" method="get" action="ex3-7.aspx">table 标记复制到此处
</form>
```

表单里。

（4）为 ex3-7.aspx 文件的 Page_Load 中添加事件处理程序。

```
string loginName = Request.QueryString["txtPwd"].ToString();//获取传过来的用户名
       string loginPwd = Request.QueryString["txtUserName"].ToString();//获
取传过来的用户密码
       if (loginName == "admin" && loginPwd == "admin")
       {
           Response.Write("<center>");
           Response.Write("欢迎您<font color=red>" + loginName+"</font>来到个
人空间！");
           Response.Write("</center>");
   }
```

（5）设置 ex3-7.html 为起始页。按快捷键 F5 运行程序。程序运行结果如图 3-13 所示。

图 3-13　程序运行结果

2. 使用 Form 集合获取客户端表单数据

当表单的 Method 方法设置为“POST”时，此时应使用 Request 对象的 Form 集合来获取表单数据。

- 语法格式：Request.Form["表单元素名"]

例如：将例 3-7 中 ex3-7.htm 文件中<form>表单的 Method 方法改为“Post”。将 ex3-7.aspx.cs 文件中所有 Request.QueryString 改为 Request.Form 即可。

3.4.2　Request 对象获取 URL 查询字符串的数据

在上一节 Response.Redirect 方法中已经了解了 URL 传递查询字符串的格式。

Request 对象获取 URL 查询字符串的格式：

- 语法格式：Request["变量名"]

例 3-8　Request 对象获取 URL 查询字符串的数据。

操作步骤：

（1）新建一个 ch3-8 的网站项目，并将该项目下的默认窗体名称改为：ex3-8.aspx。

（2）ex3-8.aspx 页面进行界面设计。从标准工具箱中添加一个 Button 控件、一个 Label 控件、一个 DropDownList 控件到 ex3-8.aspx 窗体上。

窗体控件属性设置如表 3-9 所示。

表 3-9　窗体控件属性设置

控件类别	控件	属性名称	属性值	备注
Web 控件	Button	ID	btnSubmit	提交表单
		Text	提交	
	DropDownList	ID	drpCp	用户名
	Label	ID	lblCp	提示信息
		Text	选择您的省份：	

（3）为 ex3-8.aspx 文件的 Page_Load 和 btnSubmit_Click 中添加事件处理程序。

Page_Load 事件处理程序代码如下：

```
if (!IsPostBack)
        {
```

```
            this.drpCp.Items.Add("请选择");
            this.drpCp.Items.Add("安徽");
            this.drpCp.Items.Add("北京");
            this.drpCp.Items.Add("上海");
            this.drpCp.Items.Add("四川");
    }
```

btnSubmit_Click 事件处理程序代码如下：

```
string cp=this.drpCp.SelectedItem.Text;
    string index=this.drpCp.SelectedIndex.ToString();
Response.Redirect("welCome.aspx?cp="+cp+"&index="+index);
```

（4）向项目中添加一个 welCome.aspx 页面窗体。并为该页面添加 Page_Load 事件处理程序。

```
//获取省份索引变量
        int index = Convert.ToInt32(Request.QueryString["index"]);
        //获取省份变量
        string cp = Request.QueryString["cp"].ToString();
        if (index == 0)
        {
            //如果用户没有选择，提示用户选择
            Response.Write("您没有选择省份，请选择");
            Response.Write("<br><br>");
            Response.Write("<a href='ex3-8.aspx'>返回</a>");
        }
        else
        {
            Response.Write("欢迎来自"+cp+"地区的朋友!");
    }
```

（5）将 ex3-8.aspx 设置为起始页。按快捷键 F5 运行程序。运行结果如图 3-14 所示。

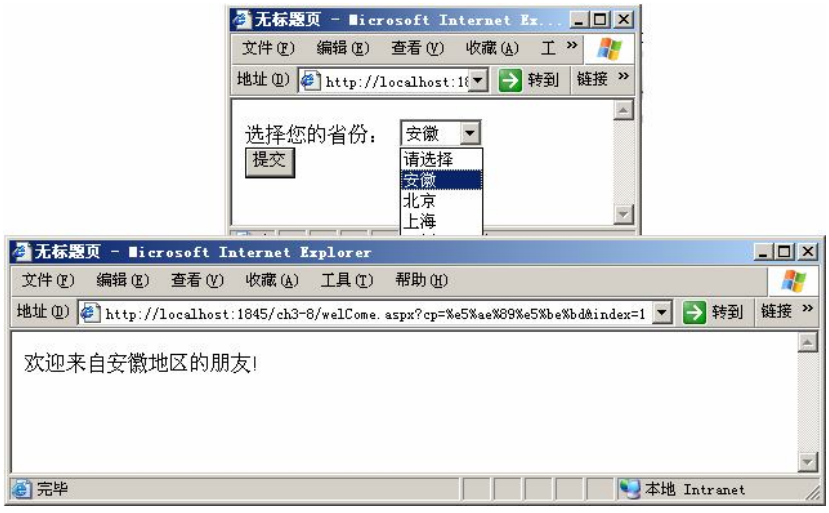

图 3-14　Request 对象获取 URL 数据

3.4.3　Request 对象获取客户端浏览器信息

Request 对象可以使用 Browser 属性取得目前和服务器连接的浏览器信息，该属性是一个

集合。

例 3-9 Request 对象获取客户端浏览器信息。

操作步骤：

（1）新建一个 ch3-9 的网站项目，并将该项目下的默认窗体名称改为：ex3-9.aspx。

（2）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-9.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

```
Response.Write("浏览器类型：" + Request.Browser.Browser + "<br>");
        Response.Write("浏览器版本：" + Request.Browser.Version + "<br>");
        Response.Write("<HR color=red width=50% align=left>");
        Response.Write("浏览器其他属性：<br>");
        Response.Write("是否支持框架：" + Request.Browser.Frames + "<br>");
        Response.Write("是否支持 Cookie：" + Request.Browser.Cookies + "<br>");
        Response.Write("是否支持表格：" + Request.Browser.Tables + "<br>");
    Response.Write("是否支持背景音乐：" + Request.Browser.BackgroundSounds +
"<br>");
```

（3）按快捷键 F5 调试运行程序，程序运行结果如图 3-15 所示。

图 3-15 Request 获取客户端浏览器信息

3.5 Server 对象

Server 对象是 HttpServerUtility 的一个实例。Server 对象提供对服务器上的方法和属性的访问，获取有关服务器的信息。Server 对象常用属性和方法如表 3-10 所示。

表 3-10 Server 对象常用属性和方法

属性与方法	说明
MachineName 属性	返回服务器端机器名称
ScriptTimeOut 属性	返回请求超时时间
HtmlEnCode 方法	该方法有一个字符串类型的参数，其功能是对其参数进行编码，然后在浏览器端显示
HtmlDeCode 方法	该方法与 HtmlEnCode 功能相反，它提取用 HTML 编码的字符串

续表

属性与方法	说明
MapPath 方法	该方法返回与 Web 服务器上的虚拟目录对应的物理路径
UrlEncode 方法	该方法对 URL 中字符进行编码
UrlDecode 方法	该方法可以通过 URL 传递参数，并将它们转换成普通字符

3.5.1　Server.HtmlEncode 方法

- 语法格式：Server.HtmlEncode(string)

如果要在网页上显示 HTML 标记时，若在网页中直接输出则会被浏览器识别为 HTML 的内容，应使用 Server.HtmlEncode 方法编码后输出；若要将编码后的内容恢复成原来的内容，应使用 Server.HtmlDecode 方法进行反编码操作。

例 3-10　HtmlEncode 与 HtmlDecode 方法的使用。

操作步骤：

（1）新建一个 ch3-10 的网站项目，并将该项目下的默认窗体名称改为：ex3-10.aspx。

（2）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-10.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

```
Response.Write("<center><B>未使用 HtmlEnCode 的演示</B></center>");
        Response.Write("<HR color=red >");
        Response.Write(Server.HtmlEncode("<center><B> 使 用  HtmlEnCode  的 演 示
</B></center>"));
        Response.Write("<HR color=red >");
    Response.Write(Server.HtmlDecode("<center><B>  使  用  HtmlDeCode  的  演  示
</B></center>"));
```

（3）按快捷键 F5 运行程序，输出结果如图 3-16 所示。

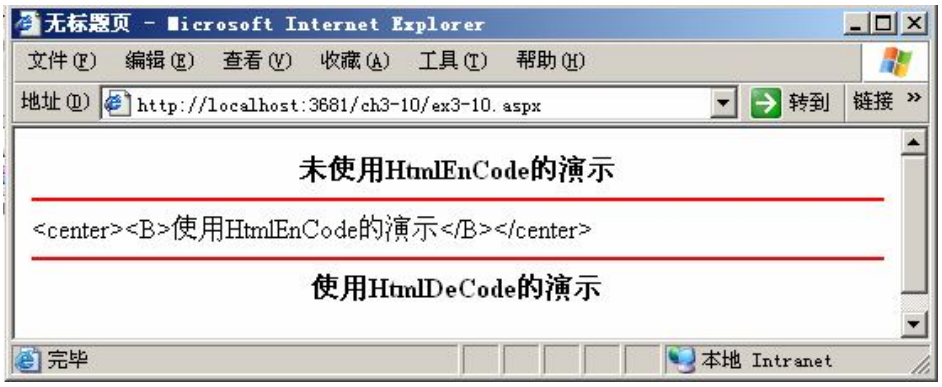

图 3-16　HtmlEncode 的使用

程序说明：

第一行代码由于字符串未采用 HtmlEncode 进行编码，因此字符串中的 <B></B> 被浏览器识别为 HTML 标记。第二行代码由于字符串采用了 HtmlEncode 进行编码，因此字符串被原样输出。第三行代码中的字符串采用了 HtmlDeCode 进行反编码，因此字符串

中的 <B></B> 被浏览器识别为 HTML 标记。

3.5.2 Server.MapPath 方法

- 语法格式：Server.MapPath(path)

参数 path 的值若为空字符串（""）或“.”则返回当前文件的物理路径；若 path 以“/”或以“\”开头时，MapPath 方法返回网站根目录下文件的位置，如 Server.MapPath(“/test.txt”)，返回文件 test.txt 文件在网站根目录下的位置。

3.5.3 Server.UrlEncode 方法

- 语法格式：Server.UrlEncode(string)

在通过 URL 传递参数时，参数是附在 URL 之后，如果参数中包含一些特殊字符“&”、“#”等，则会获取不到这些特殊字符后的参数。因此在需要传递含有特殊字符的参数时，应先对其进行编码然后传递。

例 3-11 Server.UrlEncode 的使用。

操作步骤：

（1）新建一个 ch3-11 的网站项目，并将该项目下的默认窗体名称改为：ex3-11.aspx。

（2）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-11.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

```
Response.Write("<a href=read.aspx?cp=" + Server.UrlEncode("1&0") +
 "&index=0>使用 Server.UrlEncode 进行编码</a>");
   Response.Write("<br>");
Response.Write("<a href=read.aspx?cp=1&0&index=0>未使用 Server.UrlEncode 进行
编码</a>");
```

（3）为 ch3-11 项目添加一个 read.aspx 页面，并为 read.aspx 页面添加 Page_Load 事件处理程序：

```
Response.Write(Request["cp"]);
      Response.Write("<br>");
  Response.Write("<a href=ex3-11.aspx>返回</a>");
```

（4）设置 ex3-11.aspx 页面为起始页。按快捷键 F5 运行调试程序，程序运行结果如图 3-17 和图 3-18 所示。

图 3-17 UrlEncode 已编码结果

图 3-18　未使用 UrlEncode 编码结果

3.6　Application 对象

Application 对象是一个应用程序级的对象，使用 Application 对象可以产生一个整个 Web 应用程序都可以访问的变量，用来存放所有用户间共享信息，并可以在 Web 应用程序运行期间持久的保持数据。当应用程序第一次启动时，Application 对象被创建。创建成功后，在整个应用程序中都可以访问该对象的值，直到应用程序结束或使用 Clear 方法清除。

表 3-11 列出了 Application 对象常用的属性。

表 3-11　Application 对象常用属性

属性	说明
All	返回全部的 Application 对象变量到一个对象数组
AllKeys	返回全部的 Application 对象变量名到一个字符串数组
Count	取得 Application 对象变量的数量
Item	允许使用索引或 Application 变量名称传回内容值

表 3-12 列出了 Application 对象常用的方法。

表 3-12　Application 对象常用方法

方法	说明
Add	新增一个 Application 对象
Clear	清除全部的 Application 对象变量
Get	使用索引值或变量名称传回变量值
Set	使用变量名称更新一个 Application 对象变量的内容
Lock	锁定全部的 Application 变量
Unlock	解除锁定 Application 变量
Remove	使用变量名称移除一个 Application 变量
RemoveAll	移除全部 Application 对象变量

3.6.1　Application 对象的使用

1. 设置 Application 对象的变量

- 语法格式：Application["变量名"]=值
  或 Application.Add("变量名",值)

例如：

```
Application["num1"]=0;
Application.Add("num2",0);
```

2. 获取 Application 对象变量的值

- 语法格式：值=Application["变量名"]
  或值=Application[index]

说明：index 表示从 0 开始的索引下标，最大值为 Application 对象变量总数减 1（Application.Count-1）。

例如：

```
string Name=Application["Name"].ToString();
string pwd=Application[2].ToString();
```

3. 获得 Application 对象中的变量名

- 语法格式：Application.GetKey(index)

说明：index 表示从 0 开始的索引下标，最大值为 Application 对象变量总数减 1。

4. 获取 Application 对象中变量总数

- 语法格式：Application.Count

5. Application 对象变量的锁定与解锁

（1）Application 对象变量的锁定。

- 语法格式：Application.Lock()

（2）Application 对象变量的解锁。

- 语法格式：Application.UnLock()

说明：Application 对象允许页面共享变量的同时也引发了一个问题，那就是各个页面都可以修改共享变量的值。为了避免多个用户同时修改一个 Application 对象变量引发的冲突，可以使用 Application 对象的 Lock 方法。当执行了 Application.Lock()后的 Application 对象变量的值只能由一个用户修改，这样就避免了多个用户同时修改一个变量值的情况。但是，当用户修改活动结束后，应该允许其他用户修改该变量，这就需要解除对 Application 对象变量的锁定，可以使用 Application.UnLock()方法解除对 Application 对象变量的锁定。

例 3-12 利用 Application 对象实现简单的网站计数器。

操作步骤：

（1）新建一个 ch3-12 的网站项目，并将该项目下的默认窗体名称改为：ex3-12.aspx。

（2）编写 Page_Load 事件处理程序。在设计视图下双击 ex3-12.aspx 空白处，为 Page_Load 事件处理程序添加如下代码：

```
Application.Lock();
       Application["Count"] = Convert.ToInt32(Application["Count"]) + 1;
       Application.UnLock();
       Response.Write("您使用的计数器变量名称是：" + Application.GetKey(0));
       Response.Write("<br><HR>");
       Response.Write("<center>");
       Response.Write("您是第" + Application["Count"] + "个访客");
  Response.Write("</center>");
```

（3）按快捷键 F5 调试程序，程序运行结果如图 3-19 所示。

图 3-19　Application 对象的使用

3.6.2　Application 对象的事件

1. Application_Start 事件

Application_Start 事件只在应用程序第一次启动时触发，即 Web 应用程序的第一个用户的第一次请求时触发。该事件在整个应用程序的生命周期内只触发一次，在此事件内可以对整个程序的变量进行初始化。只有 Application 和 Server 内置对象可在该事件内使用。在 Application_Start 事件内引用 Request、Response 等内置对象将导致错误发生。

2. Application_End 事件

Application_End 事件与 Application_Start 事件相反，它是在应用程序关闭时触发。该事件在整个应用程序的生命周期内也只触发一次。Application_End 事件中只有 Application 内置对象可以使用。

3. Global.asax 文件

Global.asax 文件是 Web 应用程序的系统文件，属于可选文件。当需要使用 Application 和 Session 对象的事件处理程序时，需要创建此文件。Global.asax 文件是 ANSI 文本文件，使用记事本或 Visual Studio 都可以编辑该文件的内容。

在.NET 1.0 版本中通过 Visual Studio 创建一个 ASP.NET 应用程序时会自动添加 Global.asax 文件。在.NET 2.0 版本中需要手动的添加该文件。添加方法：选择“文件”→“新建”→“文件”命令，然后在弹出的“添加新项”对话框中选择“全局应用程序类”即可。

3.7　Session 对象

Session 对象和 Application 对象一样，用来存放跨网页的变量或对象。与 Application 对象不同的是 Session 对象为某一用户所私有，而 Application 对象是整个应用程序所共有，即所有的用户共用一个 Application 对象，每个用户只能访问自己的 Session 对象。使用 Session 对象可以存储特定的用户会话所需的信息，该对象是 HttpSession 类的一个实例。

Session 对象存储的数据和 Application 对象一样都是存储在服务器端。针对每一个连接，系统会自动分配一个 ID 标识每一个不同的用户。Session 对象常用的属性和方法如表 3-13 和表 3-14 所示。

表 3-13　Session 对象常用的属性

属性	说明
Contents[name/index]	name/index 分别表示 Contents 集合中 Session 对象的名称和索引下标
Count	Contents 集合中的变量个数
IsReadOnly	获取 Session 对象是否为只读，默认为 false
IsNewSession	获取是否创建了新的会话，若创建新会话，此属性返回 true
TimeOut	获取或设置 Session 对象的有效时间，默认为 20 分钟

表 3-14　Session 对象常用方法

方法	说明
Add(name,value)	向会话状态集合中添加一个名为 name，值为 value 的变量
Abandon	结束当前会话，并清除会话中所有信息
Clear	清除会话状态集合中所有的变量
CopyTo(Array,index)	复制 Session 对象的会话状态集合到 Array 指定的数组中
Remove	从会话状态集合中删除会话变量
RemoveAt	按索引删除会话状态集合中会话变量
RemoveAll	删除会话状态集合中所有会话变量

3.7.1　Session 对象的使用

1. 设置 Session 对象的变量

- 语法格式：Session["变量名"]=值
  或 Session.Add("变量名",值)

例如：

```
Session ["num1"]=0;
Session.Add("num2",0);
```

2. 获取 Session 对象变量的值

- 语法格式：值= Session ["变量名"]
  或值= Session [index]

说明：index 表示从 0 开始的索引下标，最大值为 Session 对象变量总数减 1（Session.Count-1）。

例如：

```
string Name= Session ["Name"].ToString();
string pwd= Session [2].ToString();
```

3. 获取 Session 对象的 SessionID

- 语法格式：Session.SessionID

说明：当一个 Session 创建后，系统会自动创建一个 SessionID。SessionID 是 Session 对象的唯一性标识，它由服务器随机分配一个数据。

4. 获取或设置 Session 对象的有效时间

- 语法格式：Session.TimeOut=time

说明：系统默认的 Session 超时时间为 20 分钟。如果用户在这个期限内没有对站点内的

任意一个页面提出请求或者刷新，那么服务器就认为该用户已经离开了站点，从而结束为该用户创建的 Session 会话。除了可以通过 TimeOut 属性来设置 Session 对象的有效时间外，还可以通过 IIS 来更改该设置。通过 IIS 设置 Session 对象的有效时间方法：打开 IIS 窗口→右键选中要设置 IIS 的虚拟目录→在弹出的快捷菜单中选择“属性”选项→在打开的属性窗口中，选择“虚拟目录”选项卡（如图 3-20 所示），单击“配置”按钮，在弹出的“应用程序配置”对话框中选择“选项”选项卡（如图 3-21 所示）。

图 3-20　“虚拟目录”选项卡

图 3-21　Session 会话超时设置

5. Session 对象的终止

Session 对象的终止有 3 种方式：

- 在 Session 对象有效时限内，用户主动离开网站。
- 用户在有效期限内没有对站点内的任意一个页面提出请求或者刷新。
- 调用 Session 对象的 Session.Abandon()方法。

例 3-13　利用 Session 对象实现用户登录。

操作步骤：

（1）新建一个 ch3-13 的网站项目，并将该项目下的默认窗体名称改为：login.aspx。

（2）界面设计（参见例 3-2）。

（3）为 login.aspx 页面的“登录”按钮添加 btnSubmit_Click 事件处理程序，代码如下：

```
Session["UserName"] = this.txtUserName.Text.Trim();
        Session["Pwd"] = this.txtPwd.Text.Trim();
        if (Session["UserName"].ToString() == "" && Session["Pwd"].ToString()
== "")
        {
            this.lblMsg.Text = "用户名或密码不能为空";
        }
        else
        {
            Response.Redirect("welCome.aspx");
    }
```

（4）为项目添加 welCome.aspx 页面。并为 welCome.aspx 添加 Page_Load 事件处理程序，代码如下：

```
Response.Write("<center>");
        Response.Write("欢迎："+Session["UserName"].ToString()+"的到来!");
        Response.Write("</center><hr>");
        Response.Write("<center>");
        Response.Write("该用户的会话 ID 为："+Session.SessionID.ToString());
   Response.Write("</center>");
```

（5）设置 login.aspx 页面为起始页面，然后按快捷键 F5 调试程序，运行结果如图 3-22 所示。

图 3-22　Session 对象的使用

3.7.2　Session 对象的事件

Session 对象的事件和 Application 对象的事件一样，都在 Globl.asax 文件当中。

1. Session_Start 事件

Session_Start 事件是在服务器创建新会话时发生的。Session_Start 事件是设置会话期间变量的最佳时机。所有内置对象（Application、Server、Request、Response、Session）都可以在此事件中使用。

2. Session_End 事件

Session_End 事件是在会话结束时触发该事件。

Application_Start、Application_End、Session_Start、Session_End 四个事件发生的先后顺序如图 3-23 所示。

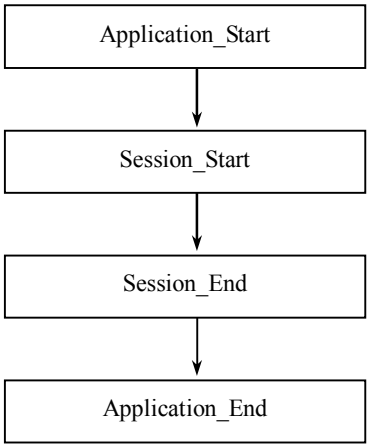

图 3-23　Application、Session 的事件发生顺序

例 3-14　Session 对象和 Application 对象实现用户在线统计。

操作步骤：

（1）新建一个 ch3-14 的网站项目，并将该项目下的默认窗体名称改为：ex3-14.aspx。

（2）为 ex3-14.aspx 添加 Page_Load 事件处理程序，代码如下：

```
Response.Write("<cemter>");
Response.Write("当前在线人数："+Application["OnlineNum"]);
```

（3）为 ch3-14 项目添加 Global.asax 文件（添加方法参见 3.6.2 节）。

（4）为 Global.asax 文件添加：Application_Start、Session_Start、Session_End 事件处理程序，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.Lock();
        Application["OnlineNum"] = 0;
        Application.UnLock();
}
void Session_Start(object sender, EventArgs e)
    {
        // 在新会话启动时运行的代码
        Application.Lock();
        Application["OnlineNum"]=Convert.ToInt32(Application["OnlineNum"])+
1;
        Application.UnLock();
```

```
    }
    void Session_End(object sender, EventArgs e)
    {
        // 在会话结束时运行的代码
        Application.Lock();
        Application["OnlineNum"] = Convert.ToInt32(Application["OnlineNum"]) -
1;
        Application.UnLock();
 }
```

程序说明：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，则不会引发该事件。

（5）程序运行结果如图 3-24 所示。

图 3-24　在线统计

3.8　Cookie 对象

Cookie 对象提供为 Web 应用程序保存用户的相关信息。当用户访问一个网站时可以利用 Cookie 保存用户访问的特定信息，如上次访问的位置、花费的时间或用户的首选项等。

Cookie 是个存储在浏览器目录中的文本文件，当浏览该 Cookie 对应的站点时，Cookie 作为 HTTP 头部文件的一部分在浏览器和服务器之间互相传递。Cookie 是与 Web 站点而不是具体的某个页面相关联，所以无论用户请求浏览站点中的哪个页面，浏览器和服务器都会交换 Cookie 信息。

目前，Cookie 技术的争议较多。许多人认为，Cookie 技术给 Web 应用程序开发人员增加了系统保存、传递变量的能力，另一方面却侵犯了用户的个人隐私。IE、NetScape 等浏览器都提供了屏蔽 Cookie 的技术。

3.8.1　Cookie 对象的读/写

Cookie 对象并不属于 Page 对象。Cookie 对象分别属于 Request 和 Response 对象。每一个 Cookie 对象都属于 Cookies 集合并被其管理。所有访问 Cookie 的方式可以使用索引器的方法来访问。

1. Response 对象写入 Cookie

- 语法格式：Response.Cookies[Cookie 的名称].Value=变量值
- 语法格式：Response.Cookies.Add(Cookie 的名称,Value)

在创建 Cookie 时，若指定的 Cookie 不存在，则创建它；若已存在，则以新值覆盖旧值。

2. Request 对象获取 Cookie

- 语法格式：Request.Cookies[Cookie 的名称].Value

在读取 Cookie 时，若指定的 Cookie 不存在，就会产生一个“未将对象引用到实例”的异常。

例 3-15　利用 Cookie 记录用户的喜好。

操作步骤：

（1）新建一个 ch3-15 的网站项目，并将该项目下的默认窗体名称改为：readCookie.aspx

（2）为 readCookie.aspx 添加 Page_Load 事件处理程序，代码如下：

```
if (Request.Cookies["MyLike"] == null)
       {
          Response.Write("您第一次登陆，还没有选择！请选择……<p>");
          Response.Write("<a href=writeCookie.aspx target=_blank>选择</a>");
       }
       else
       {
          string MyLike = Request.Cookies["MyLike"].Value;
          Response.Write("您喜欢的栏目是：" + MyLike);
   }
```

（3）为 ch3-15 的网站项目添加一个 writeCookie.aspx 页面。

（4）设计 writeCookie.aspx 页面。从标准工具箱中拖入一个 Label 控件、一个 DropDownList 控件、一个 Button 控件。控件的属性设置如表 3-15 所示。

表 3-15　控件属性设置

控件类别	控件	属性名称	属性值	备注
Web 控件	Button	ID	btnSubmit	提交表单
		Text	选 择	
	DropDownList	ID	drpLike	栏目
	Label	ID	lblMsg	提示信息
		Text	请选择您喜欢的栏目：	

（5）为 writeCookie.aspx 添加 Page_Load 事件处理程序，代码如下：

```
if (!IsPostBack)
       {
          this.drpLike.Items.Add("请选择……");
          this.drpLike.Items.Add("青青草原");
          this.drpLike.Items.Add("心情驿站");
   }
```

（6）为 writeCookie.aspx 的“选择”按钮添加 btnSubmit_Click 事件处理程序，代码如下：

```
if (this.drpLike.SelectedIndex > 0)
       {
          string MyLike = this.drpLike.SelectedItem.Text;
          Response.Cookies["MyLike"].Value = MyLike;
          Response.Write("您喜欢的栏目是：" + MyLike);
   }
```

（7）将 readCookie.aspx 设置为起始页，按快捷键 F5 调试程序，运行结果如图 3-25（最下面的网页表示是第一运行程序的 readCookie.aspx 页面，上面的图片是选择了一个栏目后的 writeCookie.aspx 页面）和图 3-26（writeCookie.aspx 页面选择一个栏目后，重新刷新 readCookie.aspx 的页面）所示。

图 3-25　示例页面 1

图 3-26　示例页面 2

3.8.2　Cookie 的生存周期

Cookie 主要分为两类：临时 Cookie 和永久 Cookie。如果没有设置 Cookie 的失效日期，则为临时 Cookie，它们仅保存到关闭浏览器程序为止，如例 3-15 中，如果关闭所有浏览器，则所保存的 Cookie 信息自动删除。如果利用 Cookie 对象的 Expires 属性设置 Cookie 的有效时间，就属于永久 Cookie。永久 Cookie 还可以根据需要设置一定的时限，比如有效期为“一天”、“一年”、“永不失效”等。

Cookie 有效时间的设置：

- 语法格式：Response.Cookies[cookie 名称].Expires=DateTime;

例 3-16　设置例 3-15 中的 Cookie 有效至某个时间。

```
//设置 Cookie 有效至 2012 年 1 月 1 日
Response.Cookies["MyLike"].Expires=Convert.ToDateTime("1/1/2012")
```

例 3-17　设置例 3-15 中的 Cookie 从当前时间起有效 1 个月。

```
Response.Cookies["MyLike"].Expires=DateTime.Now.AddMonths(1)
```

例 3-18　设置例 3-15 中的 Cookie 永不失效。

```
Response.Cookies["MyLike"].Expires=DateTime.MaxValue
```

一、问答题

1．简述 ASP.NET 内置对象的主要功能。
2．简述 Page 对象的 IsPostBack 属性的作用。
3．简述利用 Request 对象获取客户端数据的方法。
4．简述 Application 对象、Session 对象、Cookie 对象的用法与区别。
5．简述如何创建 Global.asax 文件。
6．简述 Application 对象的 Lock 和 UnLock 的作用。

二、操作题

制作“深化版站点访问计数器”。要求：
（1）解决重复刷新重复计算的问题。
（2）解决同一 IP 地址反复登录问题。