

第 5 章 ASP.NET 对象、状态和配置

ASP.NET 提供了一些列的全局对象来保存和操作 ASP.NET 网站的请求、状态和配置。这些对象包括 Response 对象、Request 对象、Application 对象、Session 对象、ViewState 对象和 Server 对象等。通过这些对象可以处理 HTTP 请求、响应 HTTP 请求，以及处理应用程序、用户和页面级别的数据。本章将讲解使用 ASP.NET 基本对象处理网站状态和配置的方法，主要介绍以下知识点：

- ☐ 使用 Response 对象响应页面的请求；
- ☐ 使用 Request 对象处理页面的请求；
- ☐ 保存网站的状态；
- ☐ 处理应用程序级别的数据；
- ☐ 配置网站。

通过本章的学习，读者可以了解使用 Response、Request、Application、Session、ViewState 和 Server 等对象处理 ASP.NET 网站的请求和状态的方法，以及配置 ASP.NET 网站的方法。

5.1 访问 Web 窗体页的输出流

Response 对象专门用来响应客户端的请求（Request），并将动态生成的响应结果返回给客户端浏览器。最终，将响应结果呈现在客户端的浏览器中。

Response 对象提供对当前 Web 窗体页的输出流（OutputStream）的访问，并处理当前 Web 窗体页的 HTTP 响应信息。Response 对象为用户提供了许多重要功能，如向页面输出文本、向页面输出图像、页面跳转等。

5.1.1 输出文本

Response 对象一个很重要的功能就是能够将请求中的文本信息显示在客户端（如浏览器等）上，该功能通过 Write() 方法实现。该方法能够将字符、对象、字符串、字符数组等信息显示在客户端上。

【示例 5-1】 使用 Write() 方法将一个字符 A 显示在浏览器中。

```
Response.Write('A');
```

【示例 5-2】 使用 Write() 方法将一个对象 o 显示在浏览器中。

```
object o = (object)10;  
Response.Write(o);
```

【示例 5-3】 使用 Write()方法将一个字符串“This is a string.”显示在浏览器中。

```
Response.Write("This is a string.");
```

【示例 5-4】 使用 Write()方法在浏览器中显示一个换行符号。

```
Response.Write("<br />");
```

【示例 5-5】 首先创建一个字符串变量 mystr，它的值为“This is a string.”。然后将 mystr 变量转换为字符数组 buffer。最后调用 Write()方法将字符数组 buffer 显示在浏览器中。

```
string mystr = "This is a string.";
char[] buffer = mystr.ToCharArray();
Response.Write(buffer,0,buffer.Length);
```

5.1.2 输出图像

Response 对象除了能够将请求中的文本信息显示在客户端之外，还可以将图像直接显示在浏览器中，该功能由 BinaryWrite(byte[] buffer)实现。

【实例 5-1】 介绍在浏览器中显示图像的方法，该实例调用 BinaryWrite(byte[] buffer)方法将图像的二进制数据输出到浏览器，并显示图像。具体步骤如下所述。

(1) 打开 Visual Studio 2010 集成开发环境，并创建名称为 Sample_05 的 ASP.NET 空网站。

(2) 右击“解决资源方案管理器”面板中的“D:\...\Sample_05\”节点，弹出“添加新项 - D:\BookCode\ASP.NET4.0Prime\ch05\Sample_05\”对话框。选中“Web 窗体”图标，并在“名称”文本框中输入“ShowPicture.aspx”。

(3) 单击“添加”按钮，可以将 ShowPicture.aspx 页面添加到 Sample_05 网站中。

(4) 打开 ShowPicture.aspx 页面的“源”视图，并设置该页面的标题为“输出图像”。

(5) 使用@Import 指令将 System.IO 命名空间引入到 ShowPicture.aspx 页面中。

(6) 在<script></script>标记元素中添加 Page_Load 事件。该事件实现以下功能：

- ☐ 创建 path 变量保存被显示图像的地址“pic.jpg”（这是一个相对地址，即虚拟地址）。
- ☐ 定义读取文件的流 fs，并通过 path 变量指定到被显示的图像。
- ☐ 创建保存图像二进制数据的 byte 数组 data。
- ☐ 调用 Read()方法读取图像的二进制数据，并保存到 data 数组中。
- ☐ 调用 BinaryWrite(byte[] buffer)方法将图像的二进制数据输出到浏览器。
- ☐ 设置 ShowPicture.aspx 页面的 ContentType 属性的值为 image/pjpeg。此时，该页面将直接显示一张图像。

 **注意：** fs 实例使用 path 指定被显示的图像时，使用 Server.MapPath()方法（在 5.5.1 节中介绍）将相对地址（path 变量的值）转换为绝对地址（即硬盘上的物理地址）。

(7) 综上所述，ShowPicture.aspx 页面的 HTML 设计代码如下：

```
01 <%@ Page Language="C#" %>
02 <%@ Import Namespace="System.IO" %>           <!-- 引入命名空间 -->
```

```

03 <script runat="server">
04 protected void Page_Load(object sender,EventArgs e)
05 {    //设置被显示图像的相对地址
06     string path = "pic.jpg";
07     //定义读取文件的流
08     FileStream fs = new FileStream(Server.MapPath(path),FileMode.Open,
    FileAccess.Read);
09     if(fs == null) return;
10     //定义保存图像数据的二进制数组
11     byte[] data = new byte[(int)fs.Length];
12     //读取图像的二进制数据,并保存在 data 数组中
13     fs.Read(data,0,(int)fs.Length);
14     Response.BinaryWrite(data);           //输出图像的二进制数据
15     Response.ContentType = "image/jpeg"; //设置页面输出的类型
16 }
17 </script>
18 <html xmlns="http://www.w3.org/1999/xhtml" >
19 <head runat="server"><title>输出图像</title></head>
20 <body><form id="form1" runat="server"></form></body>
21 </html>

```

(8) 把 ShowPicture.aspx 页面设置为 Sample_05 网站的起始页面,并运行该网站。在 IE 浏览器中查看 ShowPicture.aspx 页面,如图 5.1 所示。此时,该页面显示一张图像。

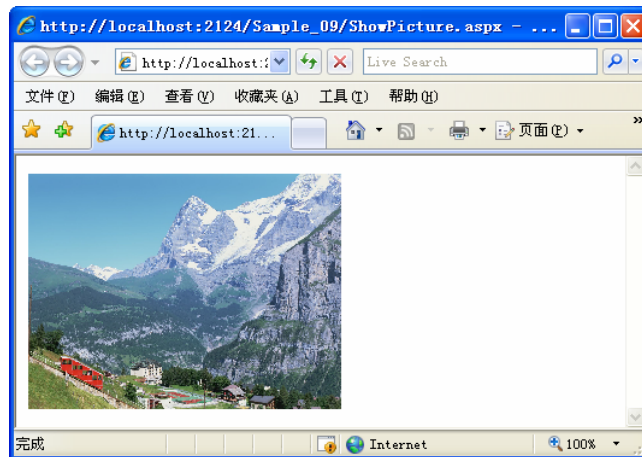


图 5.1 ShowPicture.aspx 页面显示一张图像

5.1.3 页面跳转

Response 对象的 Redirect()方法能够实现从当前页面跳转到指定页面的功能,该方法的原型如下:

```

public void Redirect(string url);
public void Redirect(string url,bool endResponse);

```

url 参数表示新页面的链接地址。endResponse 参数表示是否中止当前页面的响应。如果该参数的值为 true,则中止当前页面的响应;否则不中止。

【示例 5-6】从当前页面 (ResponseInfo.aspx) 跳转到 RequestInfo.aspx 页面。

```
Response.Redirect("~/RequestInfo.aspx");
```

 **注意：**“~”表示当前网站的根目录，它的值等于“/Sample_05”。

【示例 5-7】从当前页面（ResponseInfo.aspx）跳转到当前页面。实际上，该示例把当前页面（ResponseInfo.aspx）刷新了一次。

```
Response.Redirect("~/ResponseInfo.aspx");
```

5.2 获取客户端的 HTTP 请求信息

Request 对象提供对当前 Web 窗体页请求的访问，并能够读取客户端在 HTTP 请求期间发送的 HTTP 信息，如请求标题、查询字符串、Cookie 等。通过 Request 对象，使开发人员能够读取客户端的 HTTP 请求信息，如客户端信息、服务器变量信息、请求地址、请求信息中的参数等。

5.2.1 获取客户端信息

Request 对象的 Form 属性保存了客户端表单的信息。因此，通过该属性可以获取客户端表单的信息。Form 属性的类型为 NameValueCollection，由“名称/值”对组成。

【示例 5-8】使用 foreach 语句显示 Form 属性中所有客户端表单的信息。“Request.Form.AllKeys”获取 Form 属性中所有信息的名称，根据该名称可以获取其相对应的值。

```
foreach(string name in Request.Form.AllKeys)
{
    Response.Write("Form[" + name + "]= " + Request.Form[name].ToString() +
        "<br />");
}
```

5.2.2 获取服务器变量信息

Request 对象的 ServerVariables 属性保存了服务器变量的信息。因此，通过该属性可以获取服务器变量的信息。ServerVariables 属性的类型也为 NameValueCollection，由“名称/值”对组成。

【示例 5-9】使用 foreach 语句显示 ServerVariables 属性中所有服务器变量的信息。“Request.ServerVariables.AllKeys”获取 ServerVariables 属性中所有信息的名称，根据该名称可以获取其相对应的值。

```
foreach(string name in Request.ServerVariables.AllKeys)
{
    Response.Write("ServerVariables[" + name + "]= " + Request.Server
        Variables
        [name].ToString() + "<br />");
}
```

将示例 5-9 放置在 RequestInfo.aspx 页面的 ShowServerInfo()函数中。将 RequestInfo.aspx 页面设置为 Sample_05 网站的起始页面，并运行该网站，RequestInfo.aspx 页面效果如图 5.2 所示。此时，该页面显示 ServerVariables 属性中保存的服务器变量的信息。

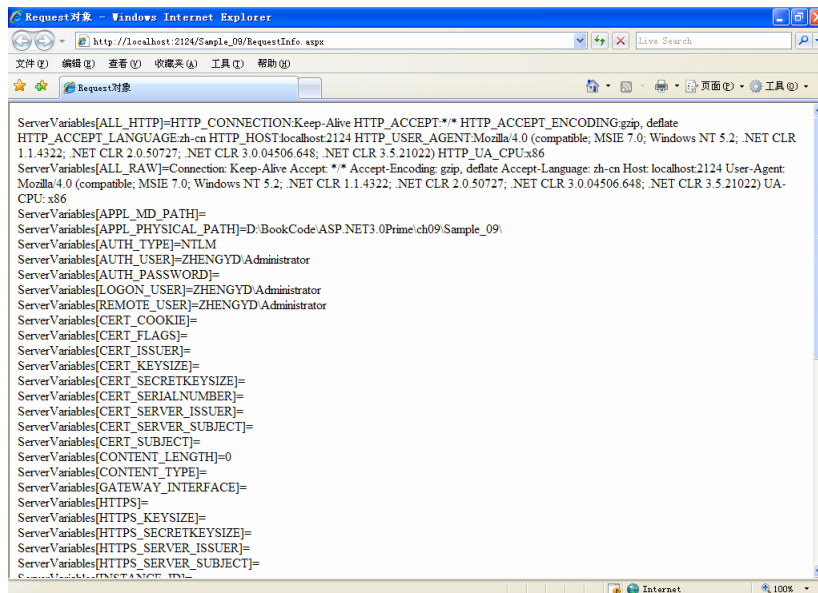


图 5.2 显示 ServerVariables 属性中保存的服务器变量的信息

5.2.3 获取请求的地址

Request 对象的 Url 属性保存了当前请求地址的信息，如当前请求地址的全路径、查询参数和绝对地址等。

【示例 5-10】 将当前请求地址的全路径显示在页面中。

```
Response.Write(Request.Url.ToString() + "<br />");
```

【示例 5-11】 将当前请求地址的绝对路径显示在页面中。

```
Response.Write(Request.Url.AbsolutePath.ToString() + "<br />");
```

Request 对象的 UrlReferrer 属性保存了上一次请求地址的信息，如上一次请求地址的全路径、查询参数、绝对地址等。

【示例 5-12】 将上一次请求地址的全路径显示在页面中。

```
if(Request.UrlReferrer != null)
{
    Response.Write(Request.UrlReferrer.ToString() + "<br />");
}
```

【示例 5-13】 将上一次请求地址的绝对路径显示在页面中。

```
if(Request.UrlReferrer != null)
{
    Response.Write(Request.UrlReferrer.AbsolutePath.ToString() + "<br />");
}
```

 **注意：**如果当前页面为用户的第一次请求，那么该页面的上一次请求为空；因此，在获取页面的上一次请求地址时，需要首先判断该页面的上一次请求是否为空；否则，有可能引起程序异常。

5.2.4 获取请求信息中的参数

有时，请求地址中可以携带参数，如下请求地址就携带了参数。参数的标识为 CurrentYear，参数的值为 2013。

```
"OtherPage.aspx?CurrentYear=2013"
```

请求地址中的参数信息保存在 Request 对象的 QueryString 属性中。因此，通过该属性可以获取请求地址中的参数信息。

【示例 5-14】获取“OtherPage.aspx?CurrentYear=2013”请求地址中的 CurrentYear 标识的参数值，并显示在页面中。在获取该参数的值之前，首先使用 if 语句判断该参数是否为空。只有当该参数的值不为空时，才显示该参数的值。

```
if(Request.QueryString["CurrentYear"] != null)
{
    Response.Write(Request.QueryString["CurrentYear"].ToString() +
        "<br />");
}
```

请求地址中的参数信息除了保存在 Request 对象的 QueryString 属性中之外，它还保存在 Params 属性中。因此，还可以通过 Params 属性获取请求地址中的参数信息。

【示例 5-15】获取“OtherPage.aspx?CurrentYear=2013”请求地址中的 CurrentYear 标识的参数值，并显示在页面中。该示例通过 Params 属性来获取参数的值。

```
if(Request.Params["CurrentYear"] != null)
{
    Response.Write(Request.Params["CurrentYear"].ToString() + "<br />");
}
```

 **注意：**Request 对象的 Params 属性的值由 Request 对象的 QueryString、Form、ServerVariables 和 Cookies 属性的值组合而成。因此，通过 Params 属性也可以获取 QueryString、Form、ServerVariables 和 Cookies 属性中的值。

5.3 参数的传递

使用 Response 和 Request 对象，可以通过页面地址（URL）把参数从一个页面传递到另外一个页面。

【实例 5-2】介绍将参数从源页面（Src.aspx）传递到目的页面（Dir.aspx）的方法，并把参数显示在目的页面中。具体步骤如下所述。

（1）在 Visual Studio 2010 集成开发环境中，打开名称为 Sample_05 的 ASP.NET 空

网站。

(2) 右击“解决资源方案管理器”面板中的“D:\...\Sample_05\”节点,弹出“添加新项目 - D:\BookCode\ASP.NET4.0Prime\ch05\Sample_05\”对话框。选中“Web 窗体”图标,并在“名称”文本框中输入“Src.aspx”。

(3) 单击“添加”按钮,可以将 Src.aspx 页面添加到 Sample_05 网站中。

(4) 按照步骤(2)和(3)将 Dir.aspx 页面添加到 Sample_05 网站中。

(5) 打开 Src.aspx 页面的“源”视图,并设置该页面的标题为“源页面”。同时,在该页面中添加 Page_Load 事件。该事件首先调用 Thread 类的静态方法 Sleep()方法延时 5 秒钟,然后调用 Redirect()方法从当前页面跳转到 Dir.aspx 页面。

 **注意:** 从当前页面跳转到 Dir.aspx 页面的地址为“~/Dir.aspx?ID=2012”。该地址携带了使用 ID 标识的参数,参数的值为 2012。

综上所述,Src.aspx 页面的 HTML 设计代码如下:

```
01 <%@ Page Language="C#" %>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <script runat="server">
04     protected void Page_Load(object sender,EventArgs e)
05     { //延时 5 秒钟
06         System.Threading.Thread.Sleep(5000);
07         Response.Redirect("~/Dir.aspx?ID=2012");
08     }
09 </script>
10 <html xmlns="http://www.w3.org/1999/xhtml">
11 <head runat="server"><title>源页面</title></head>
12 <body><form id="form1" runat="server"></form></body></html>
```

(6) 打开 Dir.aspx 页面的“源”视图,并设置该页面的标题为“目的页面”。同时,在该页面中添加 Page_Load 事件。该事件判断 Request 对象的 Params 属性中 ID 标识的值是否为空。如果不为空,则调用 Response 对象的 Write()方法将参数的值显示在 Dir.aspx 页面上。

 **注意:** 若需要从页面的地址获取参数,往往在获取参数的操作之前需要判断被获取的参数值是否为空。否则,可能会引起程序异常(当参数的值为空时)。

综上所述,Dir.aspx 页面的 HTML 设计代码如下:

```
01 <%@ Page Language="C#" %>
02 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
    www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
03 <script runat="server">
04     protected void Page_Load(object sender,EventArgs e)
05     { //判断参数是否为空,如果不为空,则显示在网页中
06         if(Request.Params["ID"] != null){Response.Write("ID 参数的值为: " +
07             Request.Params["ID"].ToString());}
08     }
09 </script>
10 <html xmlns="http://www.w3.org/1999/xhtml">
```

```

11 <head runat="server"><title>目的页面</title></head>
12 <body><form id="form1" runat="server"></form></body></html>

```

(7) 把 Src.aspx 页面设置为 Sample_05 网站的起始页面，并运行该网站。在 IE 浏览器中查看 Src.aspx 页面，该页面将等待 5 秒钟，然后跳转到 Dir.aspx 页面，并显示参数的值，如图 5.3 所示。

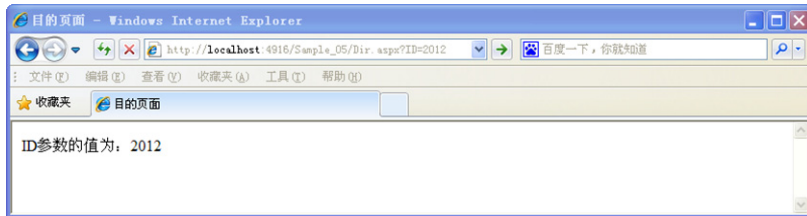


图 5.3 跳转到 Dir.aspx 页面，并显示参数的值

5.4 状态管理

ASP.NET 网站包含 3 个重要的状态（Application、Session 和 ViewState）管理功能。其中 Application 状态被网站所有用户所共享，Session 状态被一个用户所独享，ViewState 状态仅仅对一个页面中的对象所共享。

5.4.1 保存全局信息

应用程序（Application）状态，它可以被网站所有的用户共享。并且，所在 ASP.NET 应用程序的每一个页面都可以使用应用程序状态。换一句话说，ASP.NET 应用程序中的所有类都可以使用该状态。

注意：应用程序状态的数据保存在服务器的内存中，执行速度比较快。因此，应用程序状态往往用来存储信息量比较小的信息。

应用程序状态是 `HttpApplicationState` 类的一个实例，它的信息以“键/值”对的方法存储。因此，通过 `HttpApplicationState` 类提供的方法和属性可以访问和操作应用程序状态。

【示例 5-16】将应用程序状态中 OnlineCount 标识的值设置为 1。

注意：由于应用程序状态被网站的所有用户共享，因此，在访问应用程序状态之前时，首先要调用 `Lock()` 方法对应用程序状态进行加锁。在取消访问应用程序状态之前时，首先要调用 `UnLock()` 方法对应用程序状态进行解锁。

```

Application.Lock();           //Application 对象加锁
if(Application["OnlineCount"] == null)
{
    //设置值为 1
    Application["OnlineCount"] = 1;
}

```

```
Application.Unlock(); //Application 对象解锁
```

5.4.2 保存用户登录信息

当用户第一次访问某一个网站时，该网站将与该用户建立一个会话（Session）状态，并用会话的 ID 值唯一标识该会话。每一个会话仅仅被某一个用户所共享，即不同用户之间不能共享同一个会话。当多个用户共享同一个网站时，该网站将与每一个用户都建立一个唯一的会话。

会话状态是 `HttpSessionState` 类的一个实例，它的信息以“键/值”对的方法存储。因此，通过 `HttpSessionState` 类提供的方法和属性可以访问和操作会话状态。

【示例 5-17】 将用户登录信息（用户的 ID 值为 2012，用户的名称为 admin）保存在会话状态中。其中，用户的 ID 值使用 `UserID` 键进行标识，用户的名称使用 `Username` 键进行标识。

```
Session["UserID"] = 2012;  
Session["Username"] = "admin";
```

5.4.3 会话状态的有效时间

当用户第一次访问某一个网站时，该网站将与该用户建立一个会话（Session）状态。网站在建立该会话时，同时也设置了该会话的有效时间。如果用户在有效时间过后，没有向网站提供任何请求，则网站结束该会话。

 **注意：**ASP.NET 网站默认设置会话的有效时间的长度为 20 分钟。

另外，会话状态提供了 `TimeOut` 属性供开发人员设置会话的有效时间的长度。

【示例 5-18】 将会话的有效时间设置为 120 分钟。

```
Session.Timeout = 120;
```

 **注意：**`TimeOut` 属性值的单位为分钟。

当会话状态过期时，网站会自动清空会话状态保存的信息并停止该会话。

【示例 5-19】 清空会话状态中的所有信息。

```
Session.Clear();
```

【示例 5-20】 停止会话状态。

```
Session.Abandon();
```

5.4.4 页面的状态的保存

视图（`ViewState`）状态是一种非常特殊的状态，它由 Web 窗体页提供，并使用 `ViewState` 属性保存视图状态的信息。`ViewState` 能够使得 Web 窗体页及其控件，在从服务器到客户

端再从客户端返回的往返过程中保持状态信息。

 **注意：**视图状态的信息保存在 Web 窗体页中的隐藏域中。因此，在 Web 窗体页的源代码中 can 查看视图状态的值。

Web 窗体页的 ViewState 属性（视图状态）可以获取类型为 StateBag 的示例。StateBag 类提供了操作视图状态的属性和方法，具体说明如表 5.1 所示。

表 5.1 StateBag 类的属性和方法表

属性或方法	说 明
Count	StateItem 对象的数量
Item	信息的项
Keys	项的键集合
Values	项的值集合
Add()	将新的 StateItem 对象添加到视图状态中
Clear()	清除视图状态中的所有项
Remove()	从视图状态中移除指定的项
SetDirty()	设置所有项的 Dirty 属性的值
SetItemDirty()	设置指定项的 Dirty 属性的值
IsItemDirty()	判断指定项是否被修改

【示例 5-21】将 Web 窗体页的当前页码保存在视图状态中，并由 CurrentPageIndex 键值标识。

```
ViewState["CurrentPageIndex"] = 10;
```

【示例 5-22】获取保存在视图状态中，由 CurrentPageIndex 键值标识的值。

```
if (ViewState["CurrentPageIndex"] != null)
{
    int pageIndex = Int32.Parse(ViewState["CurrentPageIndex"].
        ToString());
}
```

 **注意：**若要从视图状态中获取值时，往往需要首先判断被获取的值是否存在；否则，可能会引起程序异常。

5.5 Server 对象和 Global.asax 文件

Server 对象提供了用于处理当前请求的助手，并提供了访问服务器对象的方法和属性。如获取最新的错误信息、获取应用程序的物理路径、对文本进行 HTML 编码和解码、对地址进行 URL 编码和解码等。

Global.asax 文件是一个非常特殊的文件，它包含响应 ASP.NET 运行库和注册 HTTP 模块，以及引发应用程序和会话级别事件的代码。

5.5.1 获取对应地址的物理路径

Server 对象的 MapPath()方法可以将指定的虚拟地址转换为其相对应的物理地址。该方法的原型如下:

```
public string MapPath(string path);
```

其中, path 参数表示指定的虚拟地址。该函数的返回值为其相对应的物理地址。

【示例 5-23】使用 MapPath()方法将“~”(当前网站的根目录)转换为其相应的物理地址。diskpath 变量的值为“D:\BookCode\ASP.NET4.0Prime\ch05\Sample_05”。

```
string diskpath = Server.MapPath("~/");
```

 **注意:** “~”表示当前网站的根目录,它的值等于“/Sample_05”。

5.5.2 对文本进行 HTML 编码和解码

Server 对象提供对文本进行 HTML 编码和解码功能。其中, HTML 编码功能由 HtmlEncode()方法实现, HTML 解码功能由 HtmlDecode()方法实现。

当被显示的文本包含 HTML 元素(如<table>、<a>等)时,若要将这些元素作为字符串显示在网页中,则需要对被显示的文本进行 HTML 编码。否则,网页将被显示的文本中的 HTML 元素作为网页内容的一部分。

【示例 5-24】创建包含 HTML 元素的 htmlString 字符串,它的值为“<table><tr><td>这是一个 HTML 字符串。</td></tr></table>”,并调用 HtmlEncode()方法对该字符串进行 HTML 编码,然后将该字符串显示在网页中。

```
string htmlString = "<table><tr><td>这是一个 HTML 字符串。</td></tr></table>";  
Response.Write(Server.HtmlEncode(htmlString));
```

 **注意:** 示例 5-25 将在网页中显示“<table><tr><td>这是一个 HTML 字符串。</td></tr></table>”字符串。如果不对该字符串进行 HTML 编码,则在网页中显示“这是一个 HTML 字符串。”字符串。因为,在显示该字符串时,该字符串中的 HTML 元素(如<table>、<td>等)已经作为网页内容的一部分,而不能被显示。

5.5.3 对地址进行 URL 编码和解码功能

Server 对象提供对地址(URL)进行 URL 编码和解码功能,其中, URL 编码功能由 UriEncode()方法实现, URL 解码功能由 UriDecode()方法实现。

当 URL 包含 Unicode 字符(如中文字符等)时,网页可能会存在潜在的危险。如果使用 URL 编码功能可以消除因 URL 中存在的潜在危险。

【示例 5-25】创建中文字符串 chineseString,它的值为“这是一个中文字符串。”,并

调用 `UrlEncode()` 方法对 `chineseString` 字符串进行 URL 编码，然后构建为一个跳转到 `Dir.aspx` 页面的新地址（URL），保存为 `url` 变量。

```
string chineseString = "这是一个中文字符串。";
string url = "~/Dir.aspx?Value=" + Server.UrlEncode(chineseString);
```

 **注意：**`url` 字符串（已经经过 URL 编码，不再包含中文字符串）的值为“~/Dir.aspx?Value=%e8%bf%99%e6%98%af%e4%b8%80%e4%b8%aa%e4%b8%ad%e6%96%87%e5%ad%97%e7%ac%a6%e4%b8%b2%e3%80%82”。

如果要获取经过 URL 编码之后的 URL 中的编码前的字符串，则需要使用 URL 解码功能，该功能由 `UrlDecode()` 方法实现。

【示例 5-26】调用 `UrlDecode()` 方法对 `url` 字符串进行 URL 解码，并保存为 `urlwithchinese` 变量。此时，该变量的值为“~/Dir.aspx?Value=这是一个中文字符串。”，并包含中文字符串。

```
string urlwithchinese = Server.UrlDecode(url);
```

5.5.4 跳转页面

和 `Response` 对象的 `Redirect()` 方法一样，`Server` 对象的 `Execute()` 和 `Transfer()` 方法都能够实现跳转页面的功能。

【示例 5-27】从当前页面（`ServerInfo.aspx`）跳转到 `ResponseInfo.aspx` 页面。

```
Server.Execute("~/ResponseInfo.aspx");
```

【示例 5-28】从当前页面（`ServerInfo.aspx`）跳转到 `ResponseInfo.aspx` 页面。

```
Server.Transfer("~/ResponseInfo.aspx");
```

5.5.5 添加 Global.asax 文件到网站中

在 ASP.NET 网站中，`Global.asax` 文件是一个非常特殊的文件，它包含响应 ASP.NET 运行库和注册 HTTP 模块，以及引发应用程序和会话级别事件的代码。

`Global.asax` 文件驻留在 ASP.NET 网站的根目录中（位于其他目录中的 `Global.asax` 文件将被忽略），而且一个网站只能包含一个 `Global.asax` 文件。

 **注意：**`Global.asax` 文件是一个可选文件，且网站外部用户不能直接下载或查看 `Global.asax` 文件的内容。

`Global.asax` 文件包括 4 个部分，分别是应用程序指令、包含文件、代码声明块和静态属性（或全局变量）。它们的具体介绍如下所述。

1. 应用程序指令

`Global.asax` 文件包含 3 种应用程序指令，即 `@Applicaion`、`@Import` 和 `@Assembly` 指令。

具体说明如下所述。

- ❑ **@Applicaion** 指令：只能应用在 Global.asax 文件中，它包含 6 个属性，分别是 CodeFile、CodeBehind、CompilerOptions、Inherits、Language 和 Description。
- ❑ **@Import** 指令：可以用来引入命名空间到 Global.asax 文件中。
- ❑ **@Assembly** 指令：可以指定应用程序链接到的程序集。它包含 Name 和 Src 两个属性。其中，Name 属性指定被链接程序集的名称，Src 属性指定被链接程序集的源文件的所在路径。

【示例 5-29】显示 Global.asax 文件中的 @Application 指令。该指令使用 Language 属性设置 Global.asax 文件的编程语言为“C#”。

```
<%@ Application Language="C#" %>
```

2. 包含文件

Global.asax 文件可以使用“#include”标记将指定文件的内容（如类、用户控件等）引入到 Global.asax 文件，从而使得 Global.asax 文件能够使用被引入文件的内容。语法如下：

```
<!-- #include file | virtual = "filename" -->
```

file 和 virtual 属性都指定被引入的文件。它们是两个互斥的属性，即不能在同一个“#include”标记中同时出现。

【示例 5-30】将 myText.txt 文件引入到 Global.asax 文件中。

```
<!-- #include file = "myText.txt" -->
```

3. 代码声明块

如果要在 Global.asax 文件中添加逻辑代码，则必须把这些逻辑代码放置在代码声明块中。代码声明块包含在<script></script>标记元素中，且必须包含“runat="server"”属性。

【示例 5-31】在代码声明块中包含了一个名称为 Application_Start 的事件。

```
<script runat="server">
void Application_Start(object sender, EventArgs e)
{
    //当应用程序开始时将执行这些代码
}
</script>
```

4. 静态属性

在 Global.asax 文件中可以声明一种特殊的静态变量。这种静态变量又称为静态属性，它能够被整个应用程序所共享，即整个应用程序都可以访问该变量。

【示例 5-32】在 Global.asax 文件中声明一个名称为 CurrentVisitCount 的静态属性。该属性保存当前网站在线访问人数，它的初始化为 0。

```
// <summary>
// 保存网站在线访问人数
// </summary>
public static int CurrentVisitCount = 0;
```

【示例 5-33】 VisitCount.aspx 页面访问示例 5-33 中声明的静态属性 CurrentVisitCount, 并把它值显示在页面中。

```
Response.Write(ASP.global_asax.CurrentVisitCount.ToString());
```

5.5.6 使用 Global.asax 文件的事件处理全局信息

Global.asax 文件有一个很重要的功能就是, 在该文件中可以定义 ASP.NET 网站的全局事件, 这些事件将处理 ASP.NET 网站的全局信息和操作, 如页面的请求、用户身份的验证、检查用户的权限、更新缓存信息等。在 Global.asax 文件中可以定义的常用事件如下所述。

- ❑ Application_Start 事件: 当应用程序开始时, 引发该事件。
- ❑ Application_End 事件: 当应用程序结束之前时, 引发该事件。
- ❑ Application_Error 事件: 当应用程序发生错误时, 引发该事件。
- ❑ Session_Start 事件: 当会话状态开始时, 引发该事件。
- ❑ Session_End 事件: 当会话状态结束之前时, 引发该事件。

【示例 5-34】在 Application_Start(object sender, EventArgs e)事件中初始化 CurrentVisitCount 变量的值。如果该值大于 0, 则设置为 0。

```
void Application_Start(object sender, EventArgs e)
{
    //初始化 CurrentVisitCount 变量的值
    if(CurrentVisitCount > 0){CurrentVisitCount = 0;}
}
```

5.6 统计网站在线人数

【实例 5-3】使用 Application 对象, 以及 Application_Start、Session_Start 和 Session_End 事件, 共同实现统计网站在线人数的功能。原理如下:

(1) 当应用程序第一次启动时, 将引发 Application_Start 事件。此时, 可以在该事件中将计数器的值设置为 1。

 **注意:** 计数器由 Application 对象实现, 计数器的值由 Application["OnlineCount"]表达式保存。

(2) 当用户访问该网站时, 该网站将建立与用户之间的会话 (Session), 并引发 Session_Start 事件。此时, 可以在该事件中将计数器的值增 1。

(3) 当用户离开该网站时, 该网站将取消与用户之间的会话 (Session), 并引发 Session_End 事件。此时, 可以在该事件中将计数器的值减 1。

实现统计网站在线人数功能的具体步骤如下所述。

(1) 在 Visual Studio 2010 集成开发环境中, 打开名称为 Sample_05 的 ASP.NET 网站。

(2) 右击“解决资源方案管理器”面板中的“D:\...\Sample_05\”节点, 弹出“添加新项 - D:\BookCode\ASP.NET4.0Prime\ch05\Sample_05\”对话框。选中“全局应用程序类”

图标,如图 5.4 所示。此时,“名称”文本框中自动显示“Global.asax”。

(3) 单击“添加”按钮,可以将 Global.asax 文件添加到 Sample_05 网站中。

(4) 打开 Global.asax 文件,并设计 Application_Start、Session_Start 和 Session_End 事件的程序代码。

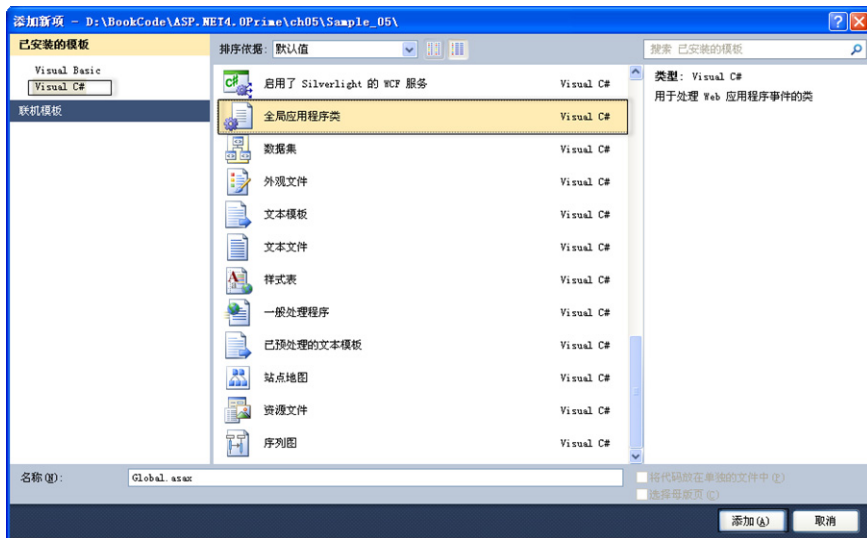


图 5.4 “添加新项 - D:\BookCode\ASP.NET4.0Prime\ch05\Sample_05\”对话框

1. 设计Application_Start事件, 初始化计数器

在 Application_Start 事件中将初始化计数器的值,即把计数器的值设置为 1。程序代码如下:

```
01 void Application_Start(object sender, EventArgs e)
02 {
03     Application.Lock(); //Application 对象加锁
04     if(Application["OnlineCount"] == null)
05     { //初始化在线人数为 1
06         Application["OnlineCount"] = 1;
07     }
08     Application.Unlock(); //Application 对象解锁
09 }
```

注意: 由于 Application 对象被网站所有用户共享,因此,在访问该对象时需要对该对象进行加锁和解锁操作。

2. 设计Session_Start事件, 计数器增1

当新用户访问该网站时,该网站将建立与该用户之间的会话,并引发 Session_Start 事件。此时,网站在线人数将增 1。因此,在 Session_Start 事件中将计数器的值增 1。程序代码如下:

```
01 void Session_Start(object sender, EventArgs e)
```

```

02 {
03     Application.Lock(); //Application 对象加锁
04     if(Application["OnlineCount"] != null)
05     { //获取当前在线人数
06         int count = Int32.Parse(Application["OnlineCount"].ToString());
07         //设置当前在线人数, 计数器增 1
08         Application["OnlineCount"] = count + 1;
09     }
10     else
11     { //计数器初始化为 1
12         Application["OnlineCount"] = 1;
13     }
14     Application.Unlock(); //Application 对象解锁
15 }

```

 **注意：**如果计数器为空，则表示当前用户为网站的第一个在线用户，因此将计数器的值设置为 1。

3. 设计 Session_End 事件，计数器减 1

当用户离开该网站时，该网站将取消与该用户之间的会话，并引发 Session_End 事件。此时，网站在线人数将减 1。因此，在 Session_End 事件中将计数器的值减 1。程序代码如下：

```

01 void Session_End(object sender, EventArgs e)
02 {
03     Application.Lock(); //Application 对象加锁
04     if(Application["OnlineCount"] != null)
05     { //获取当前在线人数
06         int count = Int32.Parse(Application["OnlineCount"].ToString());
07         //设置当前在线人数, 计数器减 1
08         Application["OnlineCount"] = count - 1;
09     }
10     else
11     { //计数器初始化为 1
12         Application["OnlineCount"] = 0;
13     }
14     Application.Unlock(); //Application 对象解锁
15 }

```

 **注意：**如果计数器为空，则表示当前用户为网站的最后一个在线用户，因此将计数器的值设置为 0。

5.7 配置网站

ASP.NET 网站也是一个应用程序，因此，ASP.NET 也为 ASP.NET 网站提供了配置方法。ASP.NET 网站通过一个名称为 Web.config 的文件来实现配置功能。

5.7.1 什么是 Web.config 文件

ASP.NET 网站和普通的应用程序一样，也可以包含配置文件，名称为 Web.config。一个 ASP.NET 网站可以包含一个或多个 Web.config 配置文件，或者不包含配置文件。

开发人员创建 ASP.NET 网站之后，系统将自动添加一个 Web.config 配置文件到网站中，其内容如下：

```

01 <?xml version="1.0"?>
02 <!--
03 有关如何配置 ASP.NET 应用程序的详细信息请访问
04  http://go.microsoft.com/fwlink/?LinkId=169433
05  -->
06 <configuration>
07   <connectionStrings>
08     <add name="ApplicationServices"
09         connectionString="data source=.\SQLEXPRESS;Integrated Security=
10         SSPI;AttachDBFilename=|DataDirectory|\aspnetdb.mdf;User
11         Instance=true"
12         providerName="System.Data.SqlClient" />
13   </connectionStrings>
14   <system.web>
15     <compilation debug="false" targetFramework="4.0" />
16     <authentication mode="Forms">
17       <forms loginUrl="~/Account/Login.aspx" timeout="2880" />
18     </authentication>
19     <membership>
20       <providers>
21         <clear/>
22         <add name="AspNetSqlMembershipProvider" type="System.Web.
23             Security.SqlMembershipProvider" connectionStringName="
24             ApplicationServices"
25             enablePasswordRetrieval="false" enablePasswordReset="true"
26             requiresQuestionAndAnswer="false" requiresUniqueEmail="false"
27             maxInvalidPasswordAttempts="5" minRequiredPasswordLength="6"
28             minRequiredNonalphanumericCharacters="0" passwordAttempt
29             Window="10"
30             applicationName="/" />
31       </providers>
32     </membership>
33     <profile>
34       <providers>
35         <clear/>
36         <add name="AspNetSqlProfileProvider" type="System.Web.Profile.
37             SqlProfileProvider" connectionStringName="ApplicationServices"
38             applicationName="/" />
39       </providers>
40     </profile>
41     <roleManager enabled="false">
42       <providers>
43         <clear/>
44         <add name="AspNetSqlRoleProvider" type="System.Web.
45             Security.SqlRoleProvider" connectionStringName="Application
46             Services" applicationName="/" />
47         <add name="AspNetWindowsTokenRoleProvider" type="System.Web.
48             Security.WindowsTokenRoleProvider" applicationName="/" />
49       </providers>
50     </roleManager>

```

```

35     </system.web>
36     <system.webServer>
37         <modules runAllManagedModulesForAllRequests="true"/>
38     </system.webServer>
39 </configuration>

```

系统添加的默认配置文件 Web.config 具体说明如下。

- ❑ <?xml version="1.0"?>声明该文件的为 XML 文件，且为 1.0 版本。
- ❑ <connectionStrings>元素专门用来配置数据库的连接字符串。
- ❑ <system.web>元素配置与 ASP.NET 网站相关的属性。
- ❑ <compilation>元素配置编译或调试应用程序相关的属性。
- ❑ <authentication>元素配置应用程序的验证方式。

【示例 5-35】把 ASP.NET 网站配置可以被调试的网站（即把 debug 属性的值设置为 true）。在开发 ASP.NET 网站时，往往需要把 debug 属性值配置为 true。而在发布 ASP.NET 网站时，需要把 debug 属性值配置为 false。

```
<compilation debug="true">
```

5.7.2 配置数据库的连接字符串

Web.config 配置文件提供一个名称 connectionStrings 的元素专门用来配置数据库的连接字符串。该元素的配置语法如下：

```

<connectionStrings >
    <add />
    <clear />
    <remove />
</connectionStrings>

```

<add>子元素可以向连接字符串集合添加“名称/值”对形式的连接字符串。<clear>元素将移除继承过来的所有连接字符串，并仅仅运行使用当前的<add>元素添加的连接字符串。<remove>元素将移除指定的继承过来的连接字符串。

【示例 5-36】在<connectionStrings>元素中添加连接字符串，并使用 SQLCONNECTIONSTRING 标识。该连接字符串的值为“data source=localhost;user id=sa;pwd=123456;database=SqlServerDB”具体说明如下。

- ❑ data source=localhost：表示数据库为本机。
- ❑ user id=sa：表示登录数据库的用户名称为 sa。
- ❑ pwd=123456：表示登录数据库的登录密码为 123456。
- ❑ database=SqlServerDB：表示被连接数据库的名称为 SqlServerDB。

```

<connectionStrings>
    <add name="SQLCONNECTIONSTRING"
        connectionString="data source=localhost;user id=sa;pwd=123456;
        database=SqlServerDB"
        providerName="System.Data.SqlClient"/>
</connectionStrings>

```

 **注意：**示例 5-36 配置的连接字符串还使用 providerName 属性，指定提供程序所在的程序集为 System.Data.SqlClient。

5.7.3 获取数据库的连接字符串

在 Web.config 配置文件的<connectionStrings>元素配置好数据库的连接字符串之后,如果要使用该连接字符串,则首先从 Web.config 配置文件中获取连接字符串。

ASP.NET 提供了专门用来获取连接字符串的类 ConfigurationManager。该类的 ConnectionStrings 属性专门用来获取连接字符串的 connectionString 属性的值,即连接字符串。

【示例 5-37】从 Web.config 配置文件的<connectionStrings>元素中获取连接字符串。该连接字符串由 SQLCONNECTIONSTRING 标识。获取的连接字符串保存在 constring 变量中。

```
string constring = ConfigurationManager.ConnectionStrings["SQLCONNECTIONSTRING"].ConnectionString;
```

5.8 小 结

本章主要介绍了使用 ASP.NET 基本对象(包括 Response、Request、Application、Session、ViewState、Server 等对象)处理网站的请求和状态的方法,以及配置 ASP.NET 网站的方法。其中,读者要着重掌握 Response、Request、Session 和 Server 对象处理网站数据的方法,以及使用 Web.Config 配置文件配置网站的方法。只有这样,才能为后续发布网站提供保障。第 6 章将要介绍数据库操作知识。

5.9 习 题

1. 填空题

(1) _____对象专门用来响应客户端的请求(Request),并将动态生成的响应结果返回给客户端浏览器。最终,将响应结果呈现在客户端的浏览器中。

(2) _____对象提供对当前 Web 窗体页请求的访问,并能够读取客户端在 HTTP 请求期间发送的 HTTP 信息,如请求标题、查询字符串、_____等。

2. 选择题

Server 对象的哪个方法可以将指定的虚拟地址转换为其相对应的物理地址_____。

- | | |
|------------------|------------------|
| A. MapPath 方法 | B. HtmlEncode 方法 |
| C. HtmlDecode 方法 | D. Execute 方法 |

3. 上机实践

在 Visual Studio 2010 集成开发环境中调试下列代码判断是否能够正常运行。如果能够

正常运行，请写出运行结果；如果不能正常运行，请指出错误代码，并改正。

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender,EventArgs e)
{    //设置被显示图像的相对地址
    string path = "pic.jpg";
    //定义读取文件的流
    FileStream fs = new FileStream(path,FileMode.Open,FileAccess.Read);
    if(fs == null) return;
    //定义保存图像数据的二进制数组
    byte[] data = new byte[(int)fs.Length];
    //读取图像的二进制数据，并保存在 data 数组中
    fs.Read(data,0,(int)fs.Length);
    Response.BinaryWrite(data);           //输出图像的二进制数据
    Response.ContentType = "image/jpeg";  //设置页面输出的类型
}
</script>
<html xmlns="http://www.w3.org/1999/xhtml" >
    <head runat="server">
        <title>习题 5.1</title>
    </head>
    <body>
        <form id="form1" runat="server">
        </form>
    </body>
</html>
```