CHAPTER 5

ASP.NET 的常用对象　第 5 章

ASP.NET 中有一些常用的内置对象，当 Web 应用程序运行时，这些对象提供了丰富的功能，例如维护 Web 服务器活动状态、网页输入/输出等。另外，通过配置 Global.asax 文件可以实现 Web 应用程序和会话的初始化设置等。

本章学习要点：

- ☑ 理解 Web 应用程序编程的难点。
- ☑ 掌握 Page 对象的使用方法。
- ☑ 掌握 Response、Request、Server 对象的使用方法。
- ☑ 掌握 Application、Session、Cookie 和 ViewState 对象的使用方法。
- ☑ 掌握 Global.asax 文件的配置方法。
- ☑ 掌握 ASP.NET 网页的执行方式和 ASP.NET 状态管理方法。
- ☑ 掌握 ASP.NET 网页生命周期的全过程。
- ☑ 灵活利用 ASP.NET 对象设计较复杂的网页。

5.1　ASP.NET 对象概述

5.1.1　Web 应用程序编程的难点及其应对

相对传统的基于客户端的应用程序编程，Web 应用程序编程有一些特殊的难点，通常包括：

- 实现多样式的 Web 用户界面：使用基本的 HTML 功能来设计和实现用户接口既困难又费事，特别是在网页具有复杂布局且包含大量动态内容和功能齐全的用户交互对象时。
- 客户端与服务器的分离：在 Web 应用程序中，客户端（浏览器）和服务器是不同的程序，它们通常在不同的计算机（甚至不同的操作系统）上运行，因此，共同组成应用程序的这两个部分仅共享很少的信息；它们可以进行通信，但通常只交换很小块的简单信息。

- 无状态执行：当 Web 服务器接收到对某个网页的请求时会找到该网页，对其进行处理，将其发送到浏览器，然后丢弃所有网页信息。如果用户再次请求同一网页，服务器则会重复整个过程，从头开始对该网页进行重新处理。换言之，服务器不会记忆它已处理的网页，即网页是无状态的。因此，如果应用程序需要维护有关某网页的信息，其无状态的性质就成为了一个问题。
- 未知的客户端功能：在许多情况下，Web 应用程序可供许多使用不同浏览器的用户进行访问。浏览器具有不同的功能，因此很难创建将在所有浏览器上都同样正常运行的应用程序。
- 数据访问方面的复杂性：对位于传统 Web 应用程序中的数据源进行读取和写入非常复杂，并且会消耗大量资源。
- 可缩放性方面的复杂性：在许多情况下，由于应用程序的不同组件之间缺乏兼容性，导致用现有方法设计的 Web 应用程序未能实现可伸缩性的目标。对于开发周期较短的应用程序，这往往是一个常见的导致失败的方面。

为了处理这些难点，ASP.NET 网页和 ASP.NET 网页框架通过以下几个方面来处理这些难题。

- 直观、一致的对象模型：ASP.NET 网页框架提供了一种对象模型，它使开发人员能够将窗体当作一个整体，而不是分离的客户端和服务器模块。在此模型中，可以通过比在传统 Web 应用程序中更为直观的方式对网页进行编程，其中包括能够设置网页元素的属性和响应事件。此外，ASP.NET 服务器控件是基于 HTML 网页的物理内容以及浏览器与服务器之间的直接交互的一种抽象模型。通常可以按照在客户端应用程序中使用控件的方式使用服务器控件，而不必考虑如何创建 HTML 来显示和处理控件及其内容。
- 事件驱动的编程模型：ASP.NET 网页为 Web 应用程序带来了一种开发人员熟悉的模型，该模型用于为客户端或服务器上发生的事件编写事件处理程序。ASP.NET 网页框架对此模型进行了抽象，使捕获客户端上的事件、将其传输到服务器并调用适当方法等操作的基础机制都是自动的，并对于开发人员都是不可见的，这样就得到了一个清晰的、易于编写的、支持事件驱动开发的代码结构。
- 直观的状态管理：ASP.NET 网页框架会自动处理网页及其控件的状态维护任务，它使开发人员能够以显式方式维护应用程序特定信息的状态。这种状态管理无须使用大量服务器资源即可实现，而且可以通过向浏览器发送 Cookie 来实现，也可以不通过向浏览器发送 Cookie 来实现。
- 独立于浏览器的应用程序：ASP.NET 网页框架允许开发人员在服务器上创建所有应用程序逻辑，而无须针对浏览器之间的差异进行显式编码。但是，它仍允许开发人员利用浏览器特定的功能，方法是通过编写客户端代码来提供增强的性能和更丰富的客户端体验。
- .NET Framework 公共语言运行库支持：ASP.NET 网页框架是在.NET Framework 的基础上生成的，因此整个框架可用于任何 ASP.NET 应用程序。开发人员的应用程序可以用与运行库兼容的任何语言编写。此外，数据访问通过.NET Framework 提供的数据访问基础结构（包括 ADO.NET）得到了简化。
- .NET Framework 可缩放服务器性能：ASP.NET 网页框架使开发人员能够将 Web

应用程序从一台只装有一个处理器的计算机有效地缩放到多计算机“网络场”，并且无须对应用程序的逻辑进行复杂的更改。

5.1.2　ASP.NET 的内置对象

在具体解决 Web 应用程序编程的难点上，ASP.NET 提供了几个内置对象，如 Response、Request 等，它们是 ASP.NET 技术中最重要的一部分。这些内置对象是由.NET Framework 中封装好的类来实现的。因为这些内置对象是在 ASP.NET 网页初始化请求时自动创建的，是全局变量，不需要声明就可以直接使用，如 Response.Write("Hello World")就是直接使用了 Response 对象。

ASP.NET 中常用的内置对象及其说明如表 5.1 所示。实际上，Response、Request、Server、Application、Session 和 ViewState 都是 Page 类的属性。

表 5.1　ASP.NET 中常用的内置对象

对 象 名	说　明
Page	用于操作整个网页
Response	用于向浏览器输出信息
Request	提供对当前网页请求的访问
Server	提供服务器端的一些属性和方法
Application	提供对所有会话的应用程序范围的方法和事件的访问，还提供对可用于存储信息的应用程序范围的缓存的访问
Session	用于存储特定用户的会话信息
Cookie	用于设置或获取 Cookie 信息
ViewState	获取状态信息的字典，这些信息开发人员可以在同一网页的多个请求间保存和还原服务器控件的视图状态

5.2　Page 对象

Page 类是一个用作 Web 应用程序的用户界面的类，Page 对象其实就是 C# 中 Web 应用程序的.aspx 文件，又称为网页。Page 对象是网页中所有服务器控件的容器。也就是说，每一个 ASP.NET 网页都是一个 Page 对象，Page 对象是由 System.Web.UI 命名空间中的 Page 类来实现的，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，并缓存在服务器内存中。

由于网页编译后所创建的类由 Page 派生而来，因此网页可以直接使用 Page 对象的属性（包括各种 ASP.NET 的内置对象）、方法和事件。

5.2.1　Page 对象的属性

Page 对象的常用属性及其说明如表 5.2 所示，除此之外，Page 对象还包括 Response、Request、Server、Session 和 Application 对象属性，下面介绍其中两个属性的用法。

1. IsPostBack 属性

该属性是一个布尔值，由系统自动设置其值，当为 True 时表示当前网页是为响应客户端回传（PostBack，指网页及操作状态传回服务器）而加载，为 False 时表示首次加载和访问网页。

表 5.2 Page 对象的常用属性及其说明

属　性	说　明
ClientQueryString	获取请求的 URL 的查询字符串部分
ErrorPage	获取或设置错误页，在发生未处理的页异常的事件时请求浏览器将被重定向到该页
Form	获取网页的 HTML 表单
IsAsync	获取一个值，该值指示是否异步处理网页
IsPostBack	获取一个值，该值指示该页是否正为响应客户端回传而加载，或者它是否正被首次加载和访问
IsValid	获取一个值，该值指示页验证是否成功
Master	获取确定页的整体外观的母版页
MasterPageFile	获取或设置母版页的文件名

在 ASP.NET 网页的处理过程中，IsPostBack 属性值的设置如下：

① 用户通过客户端浏览器请求网页，网页第一次运行。

② Web 服务器接受请求，将其代码发给 ASP.NET 引擎，此时 IsPostBack 属性为 False，ASP.NET 引擎执行服务器脚本代码，产生 HTML 文件，交给 Web 服务器，Web 服务器将其发送到客户端。

③ 客户端用户看到显示的页面，输入信息或从可选项中进行选择，或者单击按钮。

④ 此时网页又发送到 Web 服务器(第 2 次或以后运行网页)，Web 服务器接受请求并将其代码发给 ASP.NET 引擎，在 ASP.NET 中称此为“回发”或“回传”，这时 IsPostBack 属性为 True，ASP.NET 引擎执行服务器脚本代码，产生 HTML 文件，交给 Web 服务器，Web 服务器将其发送到客户端。

以上过程循环执行，直到用户退出。

2. IsValid 属性

该属性用于获取一个布尔值，指示网页上的验证控件是否验证成功。若网页验证控件全部验证成功，该值为 True，否则为 False。

IsValid 属性在网页验证中起着重要作用。例如，以下事件过程通过 mylabel 标签输出验证结果：

```
void Button1_Click(Object Sender, EventArgs E)
{    if (Page.IsValid)              //也可写成 if (Page.IsValid == True)
         mylabel.Text = "信息验证成功!";
     else
         mylabel.Text = "信息验证失败";
}
```

5.2.2 Page 对象的方法

Page 对象的常用方法及其说明如表 5.3 所示。

5.2.3 Page 对象的事件

Page 的常用事件及其说明如表 5.4 所示，下面对主要的事件做进一步的介绍。

1. Init 事件

Init 事件对应的事件处理过程为 Page_Init，在初始化网页时触发该事件。Init 事件只触发一次。Init 事件通常用来完成系统所需的初始化，如设置网页、控件属性的初始值。

表 5.3　Page 对象的常用方法及其说明

方　法	说　明
DataBind	将数据源绑定到被调用的服务器控件及其所有子控件
Eval	计算数据绑定表达式
FindControl	在网页中搜索指定的服务器控件
RegisterClientScriptBlock	向网页发出客户端脚本块
MapPath	检索虚拟路径(绝对的或相对的)或应用程序相关的路径映射到的物理路径
Validate	指示网页中的所有验证控件进行验证

表 5.4　Page 对象的常用事件及其说明

事　件	说　明
PreInit	在网页初始化开始时引发
Init	当服务器控件初始化时引发
InitComplete	在网页初始化完成时引发
PreLoad	在网页 Load 事件之前引发
Load	当服务器控件加载到 Page 对象中时引发
LoadComplete	在网页生命周期的加载阶段结束时引发
PreRender	在加载 Control 对象之后、呈现之前引发
PreRenderComplete	在呈现网页内容之前引发
SaveStateComplete	在网页已完成对页和页上控件的所有视图状态和控件状态信息的保存后引发
Unload	当服务器控件从内存中卸载时引发

2. Load 事件

Load 事件对应的事件处理过程为 Page_Load。当在内存中加载网页时触发该事件,在网页每次加载时都触发,不管是首次加载,还是按用户要求回送信息再次调用网页的回传加载,Page_Load 事件处理过程都会被执行。

如果仅希望在网页第一次加载时执行 Page_Load 事件处理过程,可以使用 Page.IsPostBack 属性。如果 Page.IsPostBack 属性为 False,则网页第一次被载入,如果为 True,则网页传回服务器。例如,在以下 Page_Load 事件处理过程中,通过 Page.IsPostBack 属性可以实现首次加载和回传时执行不同的程序代码:

```
void Page_Load(Object o,EventArgs e)
{    if (!Page.IsPostBack)              //也可写成 if (Page.IsPostBack == False)
     {    //如果网页为首次加载,则进行一些操作
          …
     }
     else
     {    //如果网页为回传,则进行一些操作
          …
     }
}
```

3. Unload 事件

Unload 事件对应的事件处理过程为 Page_Unload,当网页从内存中卸载并将输出结果发送给浏览器时触发该事件。Unload 事件主要用来执行最后的资源清理工作,如关闭文件、关闭数据库连接和释放对象等。由于这个事件是最后事件,网页的所有内容已经传到客户端浏

览器，所以不能使用它来改变控件。这个事件并不是指用户在浏览器端关闭网页，而是从 IIS 角度讲，网页从内存中卸载时发生这个事件。

5.2.4 Page 对象的应用

本节通过一个示例说明 Page 对象的应用。

【例 5.1】 在 D 盘 ASP.NET 目录中建立一个 ch5 的子目录，将其作为网站目录，然后创建一个 WebForm1.aspx 网页，其功能是说明 Page 对象的 IsPostBack 属性的应用。

解：其步骤如下。

① 启动 Visual Studio 2012。

② 选择“文件|新建|网站”命令，出现“新建网站”对话框，选择“ASP.NET 空网站”模板，选择“Web 位置”为“文件系统”，单击“浏览”按钮，然后选择“D:\ASP.NET\ch5”目录，单击“确定”按钮，这样就创建了一个空的网站 ch5。

③ 选择“网站|添加新项”命令，出现“添加新项-ch5”对话框，在中间列表中选择“Web 窗体页”，将文件名称改为 WebForm1.aspx，其他保持默认项，单击“添加”按钮。

④ 进入设计视图，设计该网页界面如图 5.1 所示，其中包含一个文本框 TextBox1、一个按钮 Button1 和一个标签 Label1。在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{    if (Page.IsPostBack == true)
         Label1.Text = TextBox1.Text + ":您好,已经提交了!";
     else
         Label1.Text = "您还没有提交!";
}
```

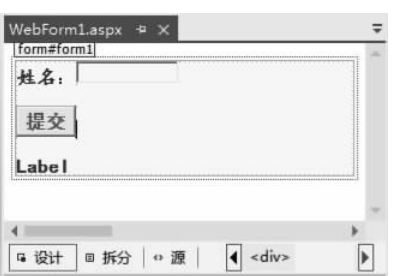

图 5.1 WebForm1 网页设计界面

⑤ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，初始运行界面如图 5.2 所示。此时 Page.IsPostBack 返回 False，在文本框中输入“王华”，单击“提交”按钮，其运行界面如图 5.3 所示，这是因为此时网页是回传状态，所以 Page.IsPostBack 返回 True。

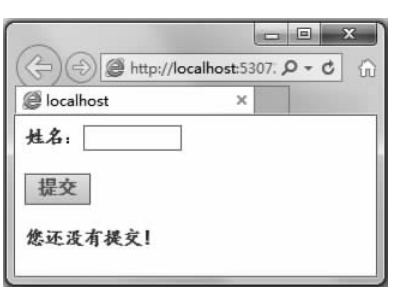

图 5.2 WebForm1 网页运行界面一

图 5.3 WebForm1 网页运行界面二

5.3 Response 对象

Response 对象实际上是与该 Page 对象关联的 HttpResponse 对象，用于控制服务器发送给浏览器的信息，包括直接发送信息给浏览器、重定向浏览器到另一个 URL 或设置 cookie 的

值，特别是可以使用 Response 对象高效地实现在不同页面的转换。

5.3.1　Response 对象的属性

Response 对象的常用属性及其说明如表 5.5 所示。

表 5.5　Response 对象的常用属性及其说明

属　性	说　明
Buffer	获取或设置一个值，该值指示是否缓冲输出，并在完成处理整个响应之后将其发送
BufferOutput	获取或设置一个值，该值指示是否缓冲输出，并在完成处理整个页之后将其发送
Cache	获取网页的缓存策略（过期时间、保密性、变化子句）
Cookies	获取响应 Cookie 集合
Expires	获取或设置在浏览器上缓存的页过期之前的分钟数。如果用户在网页过期之前返回该页，则显示缓存版本。提供 Expires 是为了与以前版本的 ASP 兼容
IsClientConnected	获取一个值，通过该值指示客户端是否仍连接在服务器上

5.3.2　Response 对象的方法

Response 对象的常用方法及其说明如表 5.6 所示，下面介绍几个主要方法的用法。

表 5.6　Response 对象的常用属性及其说明

方　法	说　明
Output	启用到输出 HTTP 响应流的文本输出
OutputStream	启用到输出 HTTP 内容主体的二进制输出
RedirectLocation	获取或设置 HTTP“位置”标头的值
Status	设置返回到客户端的 Status 栏
AppendCookie	将一个 HTTP Cookie 添加到内部 Cookie 集合
AppendToLog	将自定义日志信息添加到 Internet 信息服务（IIS）日志文件
BinaryWrite	将一个二进制字符串写入 HTTP 输出流
Clear	清除缓冲区流中的所有内容输出
ClearContent	清除缓冲区流中的所有内容输出
ClearHeaders	清除缓冲区流中的所有头
Close	关闭到客户端的套接字连接
End	将当前所有缓冲的输出发送到客户端，停止该网页的执行，并引发 EndRequest 事件
Redirect	将客户端重定向到新的 URL
Write	将信息写入 HTTP 响应输出流
WriteFile	将指定的文件直接写入 HTTP 响应输出流

1. Write 方法

Write 方法可以将一个字符串写入 HTTP 响应输出流。例如：

```
Response.Write("现在时间为: " + DateTime.Now.ToString());
```

用于输出当前的时间。

实际上 Write 方法将指定的字符串输出到客户端，由客户端浏览器解释后输出，所以这个输出字符串中可以包含一些 HTML 格式输出标记。

2. End 方法

End 方法用来输出当前缓冲区的内容,并中止当前网页的处理。例如:

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

只输出“欢迎光临”,而不会输出“我的网站!”。End 方法常用来帮助调试程序。

3. Redirect 方法

使用 Redirect 方法可以实现在不同网页之间进行跳转的功能,也就是可以从一个网页地址转到另一个网页地址,可以是本机的网页,也可以是远程的网页地址。其基本使用格式如下:

```
Redirect(url)
```

其中,url 指定目标的位置。例如,执行以下代码显示武汉大学的主页:

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

另一种使用格式如下:

```
Redirect(url,endResponse)
```

其中,endResponse 为 bool 值,指示当前网页的执行是否应终止。例如,以下服务器端代码使用 IsClientConnected 属性来检查请求网页的客户端是否仍与服务器连接。如果 IsClientConnected 为 true,则调用 Redirect 方法,因此客户端可以查看另一网页;如果 IsClientConnected 为 false,将调用 End 方法,并且所有网页处理都将终止:

```
if (Response.IsClientConnected)
{   //如果是连接的,则转向另一个网页并不终止当前网页
    Response.Redirect("anotherpage.aspx", false);
}
else
{   //如果浏览器不是连接的,停止使用的响应处理
    Response.End();
}
```

注意:Redirect 执行客户端重定向时,浏览器请求新资源。此重定向是一个进入系统的新请求,因此需要接受 IIS 和 ASP.NET 安全策略的所有身份验证和授权逻辑的检验。

5.3.3 Response 对象的应用

本节通过一个示例说明 Response 对象的应用。

【例 5.2】 在 ch5 网站中设计一个 WebForm2.aspx 网页,其功能是使用 Response 对象的 Write 方法输出若干文字。

解:其步骤如下。

① 打开 ch5 网站,选择“网站|添加新项”命令,出现“添加新项-ch5”对话框,在中间列表中选择“Web 窗体”,将文件名称改为 WebForm2.aspx,其他保持默认项,单击“添加”按钮。

② 该网页的设计界面中不包含任何内容。进入设计视图,在空白处右击,在出现的快捷菜单中选择“查看代码”命令,进入代码编辑窗口,设计如下事件过程:

```
protected void Page_Load(object sender, EventArgs e)
{     Response.Write("现在时间为: " + DateTime.Now.ToString() + "< br >");
```

```
        Response.Write("<h1>中华人民共和国</h1>");
        Response.Write("<h2>中华人民共和国</h2>");
        Response.Write("<h3>中华人民共和国</h3>");
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，其运行界面如图 5.4 所示。

图 5.4　WebForm2 网页运行界面

5.4　Request 对象

Request 对象实际上是请求的页的 HttpRequest 对象，其主要功能是从客户端获取数据。使用该对象可以访问任何 HTTP 请求传递的信息，包括使用 post 方法或者 get 方法传递的参数、cookie 和用户验证。也就是说，Request 对象使 ASP.NET 能够读取客户端在 Web 请求期间发送的 HTTP 值。

5.4.1　Request 对象的属性

Request 对象的常用属性及其说明如表 5.7 所示。

表 5.7　Request 对象的常用属性及其说明

属　性	说　明
ApplicationPath	获取 ASP.NET 应用的虚拟目录(URL)
PhysicalPath	获得 ASP.NET 应用的物理目录
Browser	获取有关正在请求客户的客户端的浏览器功能的信息
Cookies	获取在请求中发送的 Cookies 集
FilePath	获取当前请求的虚拟路径
Form	获取回传到网页的窗体变量集
Headers	获取 HTTP 头部
ServerVariables	获取服务器变量的名字/值集
QueryString	获取 HTTP 查询字符串变量集合
Url	获取有关当前请求的 URL 的信息
UserHostAddress	获取远程客户端主机的地址
UserHostName	获取远程客户端的 DNS 名称

5.4.2　Request 对象的方法

Request 对象的常用方法及其说明如表 5.8 所示，下面介绍几个主要方法的用法。

表 5.8 Request 对象的常用方法及其说明

方　法	说　明
MapPath	返回 URL 的物理路径
SaveAs	将 HTTP 请求保存到文件中
ValidateInput	对通过 Cookies、Form 和 QueryString 属性访问的集合进行验证

1. MapPath 方法

其使用语法格式如下：

```
MapPath(VirtualPath)
```

该方法将当前请求的 URL 中的虚拟路径 VirtualPath 映射到服务器上的物理路径。参数 VirtualPath 用于指定当前请求的虚拟路径(可以是绝对路径,也可以是相对路径),返回值为与 VirtualPath 对应的服务器端物理路径。

例如,以下语句：

```
Response.Write(Request.MapPath("aa"));
```

在浏览器中输出 aa 所在的物理路径。

2. SaveAs 方法

其使用语法格式如下：

```
SaveAs(filename, includeHeaders)
```

该方法将客户端的 HTTP 请求保存到磁盘。参数 filename 用于指定文件在服务器上保存的位置；布尔型参数 includeHearders 用于指定是否同时保存 HTTP 头。

例如：

```
Request.SaveAs("H:\aaa", True);
```

则执行后在 H 盘根目录产生 aaa 文件。

5.4.3 Request 对象的应用

对于网页中的 form 元素,method 属性取值为“get”表示从服务器上获取数据,取值为“post”表示向服务器传送数据。

get 是把参数数据队列加到提交表单的 action 属性所指的 URL 中,值和表单内的各个字段一一对应,在 URL 中可以看到。对于 get 方式,服务器端用 Request.QueryString 获取变量的值。

post 是通过 HTTP 的 post 机制将表单内的各个字段与其内容放置在 HTML HEADER 内一起传送到 action 属性所指的 URL 地址,用户看不到这个过程。对于 post 方式,服务器端用 Request.Form 获取提交的数据。

通常,get 传送的数据量较小,post 传送的数据量较大；get 的安全性非常低,post 的安全性较高,但是 get 的执行效率比 post 高。

1. 获取客户端机器和浏览器的相关信息

通常使用 Request 对象的 Browser、Url、Path 和 PhysicalPath 等属性获取客户端机器和浏览器的相关信息。

【例 5.3】 在 ch5 网站中设计一个 WebForm3.aspx 网页,其功能是使用 Response 对象

的 Write 方法输出若干文字。

解：其步骤如下。

① 打开 ch5 网站，选择“网站|添加新项”命令，出现“添加新项-ch5”对话框，在中间列表中选择“Web 窗体”，将文件名称改为 WebForm3.aspx，其他保持默认项，单击“添加”按钮。

② 其设计界面中不包含任何内容。进入设计视图，在空白处右击，在出现的快捷菜单中选择“查看代码”命令，进入代码编辑窗口，设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{    Response.Write("浏览器名称和主版本号: "
         + Request.Browser.Type + "<br>");
     Response.Write("浏览器名称: " + Request.Browser.Browser + "<br>");
     Response.Write("浏览器平台: " + Request.Browser.Platform + "<br>");
     Response.Write("客户端 IP 地址: " + Request.UserHostAddress + "<br>");
     Response.Write("当前请求的 URL: " + Request.Url + "<br>");
     Response.Write("当前请求的虚拟路径: " + Request.Path + "<br>");
     Response.Write("当前请求的物理路径: " + Request.PhysicalPath + "<br>");
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，其运行界面如图 5.5 所示。

图 5.5　WebForm3 网页运行界面

2. 使用 Request 对象的 QueryString 属性在网页之间传递数据

在上网的过程中，用户经常发现网址后面跟一串字符，这就是通过 URL 后面的字符串在两个网页之间传递参数，它是基于 get 方式的。网页的 QueryString 属性保存这些参数和值，因此可以通过 Request 的 QueryString 在网页之间传递信息。

【例 5.4】　在 ch5 网站中设计 WebForm4 和 WebForm4-1 两个网页，其功能是说明 QueryString 属性的使用方法。

解：其步骤如下。

① 打开 ch5 网站，添加一个网页 WebForm4.aspx。

② 其设计界面如图 5.6 所示，包含有两个文本框(ID 分别为 uname 和 uage)和一个命令按钮 Button1。进入源视图，将<form>元素改为：

```
<form id = "form1" runat = "server" method = "get" action = "WebForm4 - 1.aspx">
```

在该网页中不设计 Button1_Click 事件过程，Button1 命令按钮仅仅起到提交网页的作用。

③ 再添加一个网页 WebForm4-1.aspx，其设计界面如图 5.7 所示，包含有一个标签 Label1。在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{    string uname,uage;
     uname = Request.QueryString["uname"];
     uage = Request.QueryString["uage"];
     Label1.Text = uname + ":您好! 您的年龄为" + uage + "岁";
}
```

图 5.6　WebForm4 网页设计界面

图 5.7　WebForm4-1 网页设计界面

④ 单击工具栏中的 ▶ Internet Explorer 按钮运行 WebForm4 网页，输入姓名为“王华”、年龄为“26”，如图 5.8 所示，然后单击“单击”按钮，出现如图 5.9 所示的界面，可以看到其 URL 地址为“http://localhost:53072/WebForm4-1.aspx? __VIEWSTATE=”后跟一串乱码，这些乱码是表单中 uname 和 uage 的加密数据。WebForm4-1 网页的 Page_Load 事件过程从中提取 QueryString 属性中对应变量的值并输出。

图 5.8　WebForm4 网页运行界面

图 5.9　WebForm4-1 网页运行界面

3. 使用 Request 对象的 Form 属性在网页之间传递数据

使用 Request 的 Form 属性可以获取客户端通过 post 方式传递的表单数据，从而实现网页之间的数据传递。

【例 5.5】 在 ch5 网站中设计 WebForm5 和 WebForm5-1 两个网页，其功能是说明 Form 属性的使用方法。

解：其步骤如下。

① 打开 ch5 网站，添加一个网页 WebForm5.aspx。

② 其设计界面与 WebForm4 相似，包含有两个文本框(ID 分别为 uname 和 uage)和一个命令按钮 Button1。进入源视图，将<form>元素改为：

```
<form id="form1" runat="server" method="post" action="WebForm5-1.aspx">
```

在该网页中不设计 Button1_Click 事件过程，Button1 命令按钮仅仅起到提交网页的作用。

③ 再添加一个网页 WebForm5-1.aspx，其设计界面与 WebForm4-1 相似，包含有一个标签 Label1。在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{     string uname, uage;
      uname = Request.Form["uname"];
      uage = Request.Form["uage"];
      Label1.Text = uname + ":您好! 您的年龄为" + uage + "岁";
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮运行 WebForm5 网页，输入姓名为“王华”、年龄为“26”，类似图 5.8 所示，然后单击“单击”按钮，出现类似图 5.9 所示的界面，WebForm5-1 网页的 Page_Load 事件过程提取 Form 属性中对应变量的值并输出，同时看到其 URL 地址为“http://localhost:53072/WebForm5-1.aspx”，没有看到例 5.4 那样的乱码，这就是 post 和 get 方式的区别。

5.5　Server 对象

Server 对象实际上是 HttpServerUtility 类的实例，提供对服务器的方法和属性的访问，可以获取服务器的信息，对 HTML 文本进行编码和解码等，如文件的物理路径等。

5.5.1　Server 对象的属性

Server 对象的常用属性及其说明如表 5.9 所示。

表 5.9　Server 对象的常用属性及其说明

属　性	说　明
MachineName	作用是获取服务器的名称
ScriptTimeOut	获取和设置请求超时值(以秒计)

5.5.2　Server 对象的方法

Server 对象的常用方法及其说明如表 5.10 所示，下面介绍几个主要方法的用法。

表 5.10　Server 对象的常用方法及其说明

方　法	说　明
Execute	使用另一页执行当前请求
HtmlEncode	对要在浏览器中显示的字符串进行编码
HtmlDecode	对字符串进行 URL 解码并返回已解码的字符串
UrlEncode	对指定字符串以 URL 格式进行编码
UrlPathEncode	对 URL 字符串的路径部分进行 URL 编码并返回编码后的字符串
MapPath	返回与 Web 服务器上的指定虚拟路径相对应的物理文件路径
Transfer	终止当前网页的执行，并开始执行新的请求网页

1. MapPath 方法

使用 MapPath 方法可以获得服务器文件的物理路径。其使用语法格式如下：

```
Server.MapPath(path);
```

其中，path 指定 Web 服务器的虚拟路径。如果 path 为 null，MapPath 方法将返回包含当前应用程序的目录的完整物理路径。

2. Transfer 方法

Transfer 方法对于当前请求，终止当前网页的执行，并使用指定的网页 URL 路径开始执行一个新网页。新网页执行后不再返回原网页。其语法格式如下：

```
Server.Transfer(url);
```

其中，url 表示服务器要执行的新网页的 URL 路径。Transfer 方法转向的网页也应该是.aspx 类型的网页，例如，传输到.asp 或.asmx 等类型的网页是无效的。Transfer 方法保留 QueryString 和 Form 集合。

注意：在使用 Server 对象的 Transfer 方法时，ASP.NET 并不验证当前用户是否有权查看由 Transfer 方法提交的资源。虽然 ASP.NET 授权和身份验证逻辑运行于调用原始资源处理程序之前，但 ASP.NET 仍将直接调用 Transfer 方法指示的处理程序，并且不为新资源重新运行授权和身份验证逻辑，这一点不同于 Response 对象的 Redirect 方法。

3. Execute 方法

有时用户希望在网页运行时执行其他网页的内容后继续执行当前网页的内容，可以使用 Server.Execute 方法。其两种用法的语法格式如下：

```
Server.Execute(path);
Execute(path,preserveForm)
```

其中，path 指出要执行的 URL 路径；preserveForm 是一个 bool 值，当为 true 时保留 QueryString 和 Form 集合，为 false 时清除 QueryString 和 Form 集合。

注意：Execute 方法和 Transfer 方法的区别是 Execute 方法执行完新的网页后再返回原网页执行，而 Transfer 方法不再返回原网页执行。它们的相同点是都不同于 Response 对象的 Redirect 方法，执行它们时 ASP.NET 不验证当前用户是否有权查看由 Execute/Transfer 方法提交的资源。

5.5.3 Server 对象的应用

本节通过一个示例说明 Server 对象的应用。

【例 5.6】 在 ch5 网站中设计一个网页 WebForm6，其功能是使用 Server 对象获取服务器端的相关信息。

解：其步骤如下。

① 打开 ch5 网站，添加一个网页 WebForm6.aspx。

② 其设计界面如图 5.10 所示，其中包含两个命令按钮(Button1 和 Button2)和两个标签(Label1 和 Label2)。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{    Label1.Text = "服务器名称:" + Server.MachineName + "<br>" +
        "网页请求超时时间:" + Server.ScriptTimeout.ToString() + "秒";
}
```

```
protected void Button2_Click(object sender, EventArgs e)
{    string mystr1 = "<b>一个字符串</b>";
     string mystr2 = "ab12&@ * % #";
     Label2.Text = "服务器路径:" + Server.MapPath(".") + "<br>" +
          "HtmlEncode:" + Server.HtmlEncode(mystr1) + "<br>" +
          "HtmlDecode:" + Server.HtmlDecode(mystr1) + "<br>" +
          "UrlEncode:" + Server.UrlEncode(mystr2) + "<br>" +
          "UrlDecode:" + Server.UrlDecode(mystr2);
  }
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，然后分别单击其中的两个命令按钮，其结果如图 5.11 所示，从中看到 Server 对象的相关属性值和通过调用 Server 对象的相关方法返回的结果。

图 5.10　WebForm6 网页设计界面

图 5.11　WebForm6 网页运行界面

5.6　Application 对象

Application 对象是为当前 Web 请求获取的 HttpApplicationState 对象，它启用 ASP.NET 应用程序中多个会话和请求之间的全局信息共享。Application 对象是运行在 Web 应用服务器上的虚拟目录及其子目录下所有文件、网页、模块和可执行代码的总和。一旦网站服务器被打开，就创建了 Application 对象，所有的用户共用一个 Application 对象并可以对其进行修改。Application 对象的这一特性使得网站设计者可以方便地创建聊天室和网站计数器等常用的 Web 应用程序。

Application 对象是一个对象集合，可以看作是存储信息的容器，为所有用户共享。

5.6.1　Application 对象的属性

Application 对象的常用属性及其说明如表 5.11 所示。

表 5.11　Application 对象的常用属性及其说明

属　性	说　明
AllKeys	获取 Application 集合中的访问键
Count	返回 Application 集合中的对象个数
Contents	获取对 Application 对象的引用

5.6.2 Application 对象的方法

Application 对象的常用方法及其说明如表 5.12 所示，下面介绍几个主要方法的用法。

表 5.12 Application 对象的常用方法及其说明

方　法	说　明
Add	向 Application 集合中添加新对象
Clear	从 Application 集合中移除所有对象
Remove	从 Application 集合中移除指定名称的对象
RemoveAt	从 Application 集合中移除指定索引的对象
RemoveAll	从 Application 集合中移除所有对象
Lock	禁止其他用户修改 Application 集合中的对象
Unlock	允许其他用户修改 Application 集合中的对象

1. Add 方法

该方法用于将新对象添加到 Application 集合中。其语法格式如下：

```
Application.Add(字符串,对象值)
```

其中，“字符串”指定对象名。例如：

```
string str1 = "mystr";
int int1 = 34;
Application.Add("var1",str1);
Application.Add("var2",int1);
```

这样 Application 集合中新增了 var1 和 var2 两个对象，它们的值分别是“mystr”和 34。用户也可以采用以下方式新增对象：

```
Application["var1"] = str1;
Application["var2"] = int1;
```

可以采用以下方式读取 Application 集合中的信息：

```
int intvar;
string strvar;
object obj1 = Application[0];          //或 obj1 = Application.Contents[0];
objecto bj2 = Application["var2"];     //或 obj2 =  Application.Contents["var2"];
strvar = (string)obj1;                 //强制转换类型
intvar = (int)Application["var2"];     //强制转换类型
```

如果 Application 集合中指定的对象不存在，则访问该对象时返回 null。

2. Remove 和 RemoveAt 方法

它们都用于删除 Application 集合中的指定对象。其使用语法格式如下：

```
Application.Remove(对象名);
Application.RemoveAt(对象索引);
```

例如：

```
Application.Remove("var1")             //删除 var1 对象
Application.RemoveAt(1);               //删除 var2 对象
```

5.6.3　Application 对象的事件

Application 对象的常用事件及其说明如表 5.13 所示，这些事件处理过程应该放在 Global.asax 文件中。

表 5.13　Application 对象的常用事件及其说明

事　件	说　明
Start	在整个 ASP.NET 应用程序第一次执行时引发
End	在整个 ASP.NET 应用程序结束时引发

5.6.4　几种常见功能的实现

应用程序状态是可供 ASP.NET 应用程序中的所有类使用的数据储存库。它存储在服务器的内存中，因此与在数据库中存储和检索信息相比，它的执行速度更快。与特定于单个用户会话的会话状态不同，应用程序状态应用于所有的用户和会话。因此，应用程序状态非常适合存储那些数量少、不随用户的变化而变化的常用数据。

应用程序状态存储在 HttpApplicationState 类中，该类是用户首次访问应用程序中的任何 URL 资源时创建的一个新实例。

1. 用锁定方法将值写入应用程序状态

应用程序状态变量可以同时被多个线程访问。因此，为了防止产生无效数据，在设置值前，必须锁定应用程序状态。

其实现方式是在设置应用程序变量的代码中调用 Application 对象的 Lock 方法，并设置应用程序状态值，然后调用 Application 对象的 UnLock 方法取消锁定应用程序状态，释放应用程序状态以供其他写入请求使用。

例如，以下代码说明了如何锁定和取消锁定应用程序状态。该代码将 PageRequestCount 变量值增加 1，然后取消锁定应用程序状态：

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

2. 从应用程序状态中读取值

确定应用程序变量是否存在，然后在访问该变量时将其转换为相应的类型。

例如，以下代码说明了检索应用程序状态值 AppStartTime，并将其转换为一个 DateTime 类型的名为 appStateTime 的变量：

```
if (Application["AppStartTime"] != null)
{
    DateTime myAppStartTime = (DateTime)Application["AppStartTime"];
}
```

5.6.5　Application 对象的应用

本节通过一个示例说明 Application 对象的应用。

【例 5.7】 在 ch5 网站中设计一个网页 WebForm7，用于实现简单的聊天室功能。

解：其步骤如下。

① 打开 ch5 网站，添加一个网页 WebForm7.aspx。

② 其设计界面如图 5.12 所示，其中包含 3 个文本框，chatBox 用于显示聊天内容（大小为 200px×500px，TextMode 属性置为 MultiLine，ReadOnly 属性置为 True），TextBox1 和 TextBox2 分别用于输入姓名和聊天记录，TextBox2 的 TextMode 属性被设为 MultiLine。另外有两个命令按钮（Button1 用于提交聊天记录，Button2 用于刷新聊天记录）。在该网页上设计如下事件过程：

```
protected void Application_Start(object sender, EventArgs e)
{   Application["chats"] = null;      //聊天记录置空
    Application["chatnum"] = null;    //聊天记录数置空
}
protected void Application_End(object sender, EventArgs e)
{   Application["chats"] = null;      //聊天记录清空
    Application["chatnum"] = null;    //聊天记录数清空
}
protected void Page_Load(object sender, EventArgs e)
{   if (Application["chats"] != null)
        chatBox.Text = Application["chats"].ToString();
}
protected void Button1_Click(object sender, EventArgs e)
{   int num;
    if (TextBox1.Text != "" && TextBox2.Text != "")
    {   Application.Lock();
        if (Application["chatnum"] == null)
            num = 0;
        else
            num = int.Parse(Application["chatnum"].ToString());
        if (num % 5 == 0)             //每 5 条聊天记录添加一个当前时间
            Application["chats"] = TextBox1.Text + "说:" + TextBox2.Text
             + "[" + DateTime.Now.ToString() + "].\n" + Application["chats"];
        else
            Application["chats"] = TextBox1.Text + "说:" + TextBox2.Text + ".\n"
             + Application["chats"];
        num++;
        object obj = num;
        Application["chatnum"] = obj;
        Application.UnLock();
        chatBox.Text = Application["chats"].ToString();
    }
    else
        Response.Write("<script>alert('必须输入姓名和聊天内容')</script>");
}
protected void Button2_Click(object sender, EventArgs e)
{   if (TextBox1.Text != "")
        chatBox.Text = Application["chats"].ToString();
}
```

说明：由于文本框的内容较多时总是定位开头的位置，为了能够看到最新的聊天记录，所以 chatBox 文本框的聊天记录是倒着显示的。

③ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，输入姓名开始聊天。再次启动另一

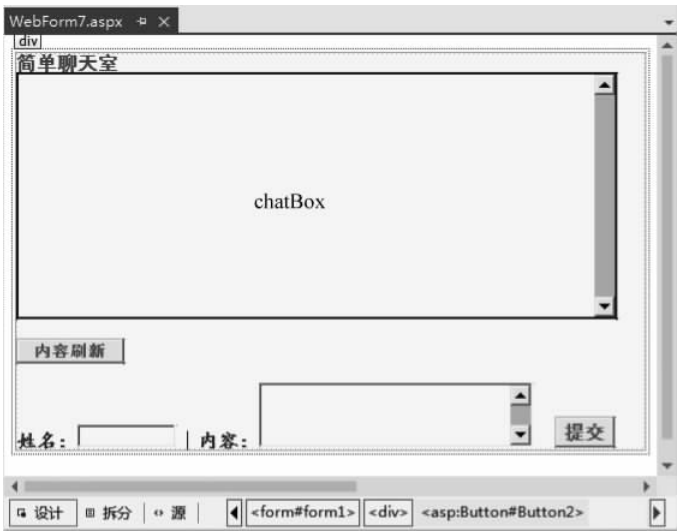

图 5.12　WebForm7 网页设计界面

个浏览器(如百度浏览器),输入地址“http://localhost:53072/WebForm7.aspx”启动本网页,这样两个人就可以相互聊天了,图 5.13 所示是王华的聊天界面(采用 IE 浏览器),图 5.14 所示是李明的聊天界面(采用百度浏览器)。

图 5.13　WebForm7 网页运行界面一

本网页设计的说明如下：

① 程序中 Response.Write("<script>alert('必须输入姓名和聊天内容')</script>")语句是向客户端浏览器发送脚本语句,会在浏览器中输出一个 alert 对话框。

② Application_Start 和 Application_End 两个事件过程用于在应用程序启动和退出时执行 Application["chats"]=null 等将聊天记录清空。

③ 本网页不能即时显示另一个网友的聊天信息,只有在单击“提交”按钮后才显示所有网

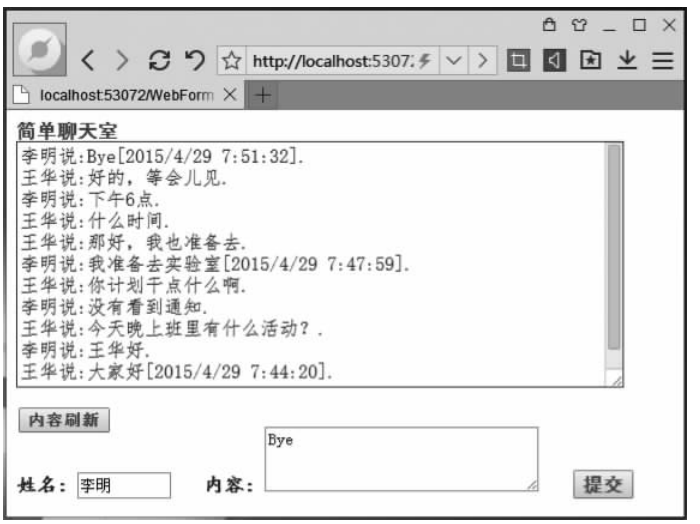

图 5.14　WebForm7 网页运行界面二

友的及时聊天信息，因此增加了一个“内容刷新”按钮，用于实现刷新功能，可以使用 AJAX 控件进行改进，有关 AJAX 控件的内容在第 11 章介绍。

5.7　Session 对象

Session 对象是一个会话对象，是 HttpSessionState 类对象。就 Web 开发而言，一个会话就是客户通过浏览器与服务器之间的一次通话。由于 HTTP 是无状态的，因此无法纪录客户一连串的动作，必须有一种机制使服务器能认得客户，这就引入了会话概念。服务器发给客户一个会话 ID(SessionID)，当客户再访问服务器时就带着这个 ID，服务器就凭着这个唯一的 ID 来识别客户。当用户请求一个 ASP.NET 网页时，系统将自动创建一个 Session 对象，退出应用程序或关闭服务器时该会话撤销。

和 Application 对象一样，Session 对象也是一个对象集合，但 Session 是针对某个特定客户的，客户之间不会产生共享情况。

5.7.1　Session 对象的属性

Session 对象的常用属性及其说明如表 5.14 所示。

表 5.14　Session 对象的常用属性及其说明

属　性	说　明
Contents	获取对当前会话状态对象的引用
Count	获取会话状态集合中的项数
IsCookieless	获取一个值，该值指示会话 ID 是嵌入在 URL 中还是存储在 HTTP Cookie 中
IsNewSession	获取一个值，该值指示会话是不是与当前请求一起创建的
SessionID	用来标识一个 Session 对象
TimeOut	获取并设置会话状态提供程序终止会话之前各请求之间所允许的超时期限

5.7.2　Session 对象的方法

Session 对象的常用方法及其说明如表 5.15 所示，下面介绍几个主要方法的用法。

表 5.15　Session 对象的常用方法及其说明

方　法	说　明
Add	将新的项添加到 Session 集合中
Clear	从 Session 集合中清除所有对象，但不结束会话
Abandon	强行结束用户会话，并清除会话中的所有信息
CopyTo	将 Session 集合复制到一维数组中
Remove	删除会话状态集合中的项
RemoveAll	从会话状态集合中移除所有的键和值
RemoveAt	删除会话状态集合中指定索引处的项

1. Add 方法

该方法用于将新对象添加到 Session 集合中。其语法格式如下：

```
Session.Add(字符串,对象值)
```

其中，“字符串”指定对象名。例如：

```
string str1 = "mystr";
int int1 = 34;
Session.Add("var1",str1);
Session.Add("var2",int1);
```

这样 Session 集合中新增了 var1 和 var2 两个对象，它们的值分别是“mystr”和 34。用户也可以采用以下方式新增对象：

```
Session["var1"] = str1;
Session["var2"] = int1;
```

可以采用以下方式读取 Session 集合中的信息：

```
int intvar;
string strvar;
object obj1 = Session[0];
objecto bj2 = Session["var2"];
strvar = (string)obj1;                    //强制转换类型
intvar = (int)Session["var2"];            //强制转换类型
```

2. Clear 方法

该方法用于清除 Session 集合中的所有对象。其语法格式如下：

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

5.7.3　Session 对象的事件

Session 对象的常用事件及其说明如表 5.16 所示。

表 5.16　Session 对象的常用事件及其说明

事　件	说　明
Start	建立 Session 对象时引发
End	结束 Session 对象时引发

说明：当用户在客户端直接关闭浏览器退出 Web 应用程序时，并不会触发 End 事件，因为关闭浏览器的行为是一种典型的客户端行为，是不会被通知到服务器端的。End 事件只有在服务器重新启动、用户调用了 Abandon 方法或未执行任何操作达到了 Timeout 设置的值（超时）时才会被触发。

5.7.4 Session 对象的应用

本节通过示例说明 Session 对象的应用。

【例 5.8】 在 ch5 网站中设计两个网页 WebForm8 和 WebForm8-1，用于实现在这两个网页之间传递数据的功能。

解：其步骤如下。

① 打开 ch5 网站，添加一个网页 WebForm8.aspx。

② 其设计界面如图 5.15 所示，其中包含两个文本框（TextBox1 和 TextBox2，分别用于输入用户名和密码，TextBox2 的 TextMode 属性设置为 Password）和一个命令按钮（Button1）。在该网页上设计如下事件过程：

```
protected void Button1_Click(object sender, EventArgs e)
{    Session["uname"] = TextBox1.Text;
     Session["upass"] = TextBox2.Text;
     Server.Transfer("WebForm8 - 1.aspx");
}
```

图 5.15 WebForm8 网页设计界面

③ 再添加一个名称为 WebForm8-1.aspx 的网页，其中只有一个标签 Label1，在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{     string mystr;
      mystr = "用户名:" + Session["uname"].ToString() +
            "<br>密 码:" + Session["upass"].ToString();
      Label1.Text = mystr;
}
```

④ 单击工具栏中的 ▶ Internet Explorer 按钮运行 WebForm8 网页，输入用户名和密码，如图 5.16 所示。单击“确定”按钮，转向 WebForm8-1.aspx 网页，输出结果如图 5.17 所示。

图 5.16 WebForm8 网页运行界面

图 5.17 WebForm8-1 网页运行界面

从本例可以看到，在一次会话中，Session 对象中存储的值都是有效的，这里是在 WebForm8 网页中存储用户名和密码，在 WebForm8-1 网页中使用这些值。

在设计网站时，有时需要禁止用户后退或者刷新以及重复提交数据，解决方法有多种，可

以通过 Session 对象来解决，其思路是设置一个标志，将这个标志放在 Session 中，在提交命令按钮 Button1 中设计相应的判断功能，基本代码如下：

```
protected void Page_Load(object sender,EventArgs e)
{    if (!Page.IsPostBack)
     {    //如果第一次载入网页就将 Updata 设为 False
          Session["Updata"] = False;
     }
}
protected void Button1_Click(object sender,EventArgs e)
{    //如果网页已经提交过(Updata 为 true)则不进行任何操作,直接返回
     if (Session["Updata"] == True) return
     //这里放置表单验证代码,验证正确后将 Updata 设为 True
     Session["Updata"] = True;
     //这里放置转向其他网页的代码
}
```

这样用户只要成功提交之后，在提交过程中刷新网页或提交一次后单击后退按钮都不能再反复提交了。

5.8　Cookie 对象

Response 和 Request 对象都有一个 Cookies 属性，它是存放 Cookie 对象的集合，是 HttpCookieCollection 类对象，提供了操作 Cookie 的方法。而 Cookie 是 HttpCookie 类对象，提供创建和操作各 Cookie 的方法。一个 Cookie 是一段文本信息，能随着用户请求和网页在 Web 服务器和浏览器之间传递。用户每次访问站点时，Web 应用程序都可以读取 Cookie 包含的信息，从而知道用户上次登录的时间等具体信息。Cookie 可以是临时的（具有特定的过期时间和日期），也可以是永久的。

Cookie 对象和 Application、Session 对象一样，都是为了保存信息，它们之间的区别是 Cookie 对象的信息保存在客户端，而 Application 和 Session 对象的信息保存在服务器端。

通常使用 Response 对象的 Cookies 集合属性设置 Cookie 信息，使用 Request 对象的 Cookies 集合属性读取 Cookie 信息。Cookies 集合属性有 Count（返回集合中 Cookie 对象的个数）属性和 Add（向 Cookies 集合中新增一个 Cookie 对象）、Clear（删除 Cookies 集合中的所有对象）及 Remove（删除 Cookies 集合中指定名称的对象）等方法。

5.8.1　Cookie 对象的属性

Cookie 对象的常用属性及其说明如表 5.17 所示，下面介绍其中几个主要的属性。

表 5.17　Cookie 对象的常用属性及其说明

属　性	说　明
Name	获取或设置 Cookie 的名称
Expires	获取或设置 Cookie 的过期日期和时间
Domain	获取或设置 Cookie 关联的域
Path	获取或设置要与 Cookie 一起传输的虚拟路径
Secure	获取或设置一个值，通过该值指示是否安全传输 Cookie
Item	获取 HttpCookie.Values 属性的快捷方式
Value	获取或设置单个 Cookie 值
Values	获取在单个 Cookie 对象中包含的键值对的集合

1. Name 属性

通过 Cookie 的 Name 属性来指定 Cookie 的名称，因为 Cookie 是按名称保存的，如果设置了两个名称相同的 Cookie，后保存的那一个将覆盖前一个，所以在创建多个 Cookie 时，每个 Cookie 都必须具有唯一的名称，以便日后读取时识别。

例如，mycookie 是一个 Cookie 对象，则 mycookie.Name 返回该 Cookie 对象的名称。

2. Value 属性

Cookie 的 Value 属性用来指定 Cookie 中保存的值，因为 Cookie 中的值都是以字符串的形式保存的，所以当为 Value 指定值时，如果不是字符串类型的要进行类型转换。

例如，mycookie 是一个 Cookie 对象，则 mycookie.Value 返回该 Cookie 对象的值。

5.8.2 Cookie 对象的方法

Cookie 对象的常用方法及其说明如表 5.18 所示。

表 5.18 Cookie 对象的常用方法及其说明

方　法	说　明
Equals	判断指定的 Cookie 对象是否等于当前的 Cookie 对象
ToString	返回此 Cookie 对象的一个字符串表示形式

5.8.3 Cookie 对象的应用

1. 创建 Cookie 对象

Cookie 对象是由 HttpCookie 类来实现的，创建一个 Cookie 对象就是建立 HttpCookie 类的一个实例。HttpCookie 类具有以下构造函数：

```
public HttpCookie (string name)
public HttpCookie(string name,string value)
```

其中，name 表示 Cookie 对象的名称（对应 Name 属性），value 表示 Cookie 对象的值（对应 Value 属性）。例如：

```
HttpCookie cookie1 = new HttpCookie("mycookie1");
                                    //新建名称为 mycookie1 的 Cookie 对象
cookie1.Value = "mystring";         //其值设为"mystring"
Response.Cookies.Add(cookie1);      //添加 cookie1 对象
HttpCookie cookie2 = new HttpCookie("mycookie2","good");
                                    //新建名称为 mycookie2 的 Cookie 对象,其值为"good"
Response.Cookies.Add(cookie2);      //添加 cookie2 对象
```

2. 设置多值 Cookie

一个 Cookie 对象可以有多个值，通过子键区分。

例如，当一个名称为 mycookie 的 Cookie 对象已添加到 Response 对象中后，可以通过以下语句设置两个子键的值：

```
Response.Cookies["mycookie"]["uname"] = "Smith";
Response.Cookies["mycookie"]["uage"] = 23.ToString();
```

或者在创建 Cookie 对象的同时设置多个值：

```
HttpCookie cookie = new HttpCookie("mycookie");
```

```
cookie.Values["uname"] = "Smith";
cookie.Values["uage"] = 23.ToString();
Response.Cookies.Add(cookie);
```

3. 读取 Cookie 对象

对于单值 Cookie 对象，直接用 Request.Cookies[Cookie 的 Name 属性值]来读取其 Cookie 值。

对于多值 Cookie 对象，还需加上子键名称。例如，以下语句将 Name 为 mycookie1 的 Cookie 对象的两个子键值分别在两个文本框中输出：

```
TextBox1.Text = Request.Cookies["mycookie1"]["uname"];
TextBox2.Text = Request.Cookies["mycookie1"]["uage"];
```

4. Cookie 的有效期

Cookie 的 Expires 属性是 DateTime 类型的，用来指定 Cookie 的过期日期和时间，即 Cookie 的有效期。浏览器在适当的时候删除已经过期的 Cookie。如果不给 Cookie 指定过期日期和时间，则为会话 Cookie，不会存入用户的硬盘，在浏览器关闭后就会被删除。

应根据应用程序的需要来设置 Cookie 的有效期，如果用来保存用户的首选项，则可以把其设置为永远有效（例如 100 年）；如果用来统计用户访问次数，则可以把有效期设置为半年。即使设置长期有效，用户也可以自行决定将其全部删除。

5. 修改和删除 Cookie

修改某个 Cookie 实际上是指用新的值创建新的 Cookie，并把该 Cookie 发送到浏览器，覆盖客户机上旧的 Cookie。

删除 Cookie 是修改 Cookie 的一种形式。由于 Cookie 位于用户的计算机中，所以无法直接将其删除。但是，用户可以修改 Cookie 将其有效期设置为过去的某个日期，从而让浏览器删除这个已过期的 Cookie。

【例 5.9】 在 ch5 网站中设计一个网页 WebForm9，其功能是说明 Cookie 对象的使用方法。

解：其步骤如下。

① 打开 ch5 网站，添加一个网页 WebForm9.aspx。

② 其设计界面如图 5.18 所示，其中包含两个文本框（TextBox1 和 TextBox2）和两个命令按钮（Button1 和 Button2）。在该网页上设计如下事件过程：

图 5.18　WebForm9 网页设计界面

```
protected void Button1_Click(object sender, EventArgs e)
{   //写入 Cookie 事件过程
    HttpCookie mycookie = new HttpCookie("cookiemc");
    mycookie.Value = TextBox1.Text;
    mycookie.Expires = DateTime.Now.AddMinutes(1); //保存 1 分钟
    Response.Cookies.Add(mycookie);                //添加到 Response 对象中
}
protected void Button2_Click(object sender, EventArgs e)
{   //读取 Cookie 事件过程
    if (Request.Cookies["cookiemc"] != null)
       TextBox2.Text = Request.Cookies["cookiemc"].Value;
    else
       Response.Write("<script>alert('Cookie 已过期')</script>");
}
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，在 TextBox1 文本框中输入“China”字符串，然后单击“写入 Cookie”命令按钮，再单击“读取 Cookie”命令按钮，则在 TextBox2 文本框中输出 Cookie 的值，如图 5.19 所示。由于 Request.Cookies["cookiemc"]的保留时间被设置为 1 分钟，当到期后，单击“读取 Cookie”命令按钮会出现如图 5.20 所示的消息提示框，表示该 Cookie 不能再使用。

图 5.19　WebForm9 网页运行界面

图 5.20　消息提示框

说明：Cookies 是 Cookie 的集合，Page 对象并没有 Cookies 属性，所以要保存 Cookie 信息，应将其添加到 Response 对象中，由 Request 对象读出来使用。

5.9　ViewState 对象

ViewState 对象是视图状态对象，它是 Page 对象的一个属性，是 StateBag 类对象。视图状态是 ASP.NET 网页框架默认情况下用于保存往返过程之间的网页和控件值的方法。视图状态用于在同一网页的多个请求间保存和还原服务器控件的信息。当呈现网页的 HTML 形式时，需要在回发过程中保留的网页的当前状态和值将被序列化为 Base64 编码的字符串，并输出到视图状态的隐藏字段中。在第 1 章的例 1.1 中看到的服务器运行 WebForm1 网页后发送给客户端的代码中，__VIEWSTATE、__VIEWSTATEGENERATOR 和__EVENTVALIDATION 就是这样的隐藏字段，它们的值是一串乱码，实际上是序列化的结果。

用户可以通过使用网页的 ViewState 属性将往返过程中的数据保存到 Web 服务器来利用自己的代码访问视图状态。ViewState 属性是一个包含密钥/值对（其中包含视图状态数据）的字典。可以存储在视图状态中的数据类型有字符串、整数、布尔值、Array 对象、ArrayList 对象和哈希表等。

5.9.1　ViewState 对象的属性

ViewState 对象的常用属性及其说明如表 5.19 所示。

表 5.19　ViewState 对象的常用属性及其说明

属　性	说　明
Count	获取视图状态对象中的状态项个数
Item	获取或设置在视图状态对象中存储的项的值
Keys	获取表示视图状态对象中的项的键集合
Values	获取存储在视图状态对象中的视图状态值的集合

5.9.2　ViewState 对象的方法

ViewState 对象的常用方法及其说明如表 5.20 所示。

表 5.20　ViewState 对象的常用属性及其说明

方　法	说　明
Add	将新的状态项对象添加到 StateBag 对象。如果该项已经存在于视图状态对象中，则此方法会更新该项的值
Clear	从当前视图状态对象中移除所有项
Remove	将指定的密钥/值对从视图状态对象中移除

例如：

```
ArrayList myarr = new ArrayList(4);              //定义名为 myarr 的 ArrayList 对象
myarr.Add("item 1");                             //向 myarr 中添加 4 个元素
myarr.Add("item 2");
myarr.Add("item 3");
myarr.Add("item 4");
ViewState.Add("mystate",myarr);                  //将 myarr 存储到键名为 mystate 的视图项中
ArrayList parr;                                  //声明 parr
parr = (ArrayList)ViewState["mystate"];          //读取键名为 mystate 的视图项到 parr 中
```

说明：视图状态信息是使用 Base64 编码存储的，并在呈现期间包括在网页中，这会增加网页大小。在回发网页时，视图状态的内容作为网页回发信息的一部分发送。由于这会大大增加网络通信量和降低连接速度，因此建议不要在视图状态中存储大量信息。

5.9.3　ViewState 对象的应用

ViewState 对象的应用与 Application、Session 十分相似，只是将相关的键名和值存储在隐藏字段__VIEWSTATE 中。

【例 5.10】 在 ch5 网站中设计一个网页 WebForm10，其功能是说明 Cookie 对象的使用方法。

解：其步骤如下。

① 打开 ch5 网站，添加一个网页 WebForm10.aspx。

② 其设计界面如图 5.21 所示，其中包含两个文本框(TextBox1 和 TextBox2)、两个命令按钮(Button1 和 Button2)和一个标签 Label1。在该网页上设计如下事件过程：

```
protected void Page_Load(object sender, EventArgs e)
{
    Button2.Enabled = False;
}
protected void Button1_Click(object sender, EventArgs e)
{    if (TextBox1.Text != "" && TextBox2.Text != "")
     {    ViewState["name"] = TextBox1.Text;
          ViewState["age"] = TextBox2.Text;
          Button2.Enabled = True;
     }
}
protected void Button2_Click(object sender, EventArgs e)
{   Label1.Text = "ViewState 信息如下：<br>";
```

图 5.21　WebForm10 网页设计界面

```
        Label1.Text += "姓名: " + ViewState["name"] + "<br>";
        Label1.Text += "年龄: " + ViewState["age"];
    }
```

③ 单击工具栏中的 ▶ Internet Explorer 按钮运行本网页，在 TextBox1 文本框中输入“王华”，在 TextBox2 文本框中输入“30”，如图 5.22 所示，然后单击“存储视图状态”命令按钮，再单击“读取视图状态”命令按钮，则在 Label1 标签中输出存储的视图状态的值，如图 5.23 所示。

图 5.22 WebForm10 网页运行界面一

图 5.23 WebForm10 网页运行界面二

5.10 配置 Global.asax 文件

每个 ASP.NET 网页中都会存在许多的事件，如 Page_Load 等，用户可以在网页中通过编程来处理这些事件。作为一个 ASP.NET 应用程序也存在这样的事件，如应用程序开始时要执行什么操作，一个新 Session 被创建的时候要进行什么操作等。那么对这些事件的处理要写在什么地方呢？通常情况下这些事件处理过程应放在 Global.asax 和 Web.config 这两个文件中。有关 Web.config 文件的内容将在以后介绍，这里只讨论 Global.asax 文件。

在 ASP.NET 中都不会自动创建 Global.asax 文件，如果要创建该文件，选择“网站|添加新项”命令，在打开的“添加新项”对话框中选择“全局应用程序类”选项，如图 5.24 所示，单击“添加”按钮即可创建一个 Global.asax 文件。该文件位于 ASP.NET 应用程序的根目录下，其作用就是用来处理与应用程序相关的一些事件。

常用的应用程序相关事件及事件被触发时间如表 5.21 所示。

表 5.21 应用程序相关事件

事件名称	事件被触发的时间	事件名称	事件被触发的时间
Application_Start	应用程序启动时	Application_Error	发生错误时
Session_Start	会话启动时	Session_End	会话结束时
Application_BeginRequest	每个请求开始时	Application_End	应用程序结束时

下面通过一个示例说明 Global.asax 文件的作用。

【例 5.11】 在 ch5 网站中设计一个网页 WebForm11，其功能是统计访问网页的在线人数。

解：其步骤如下。

① 打开 ch5 网站。

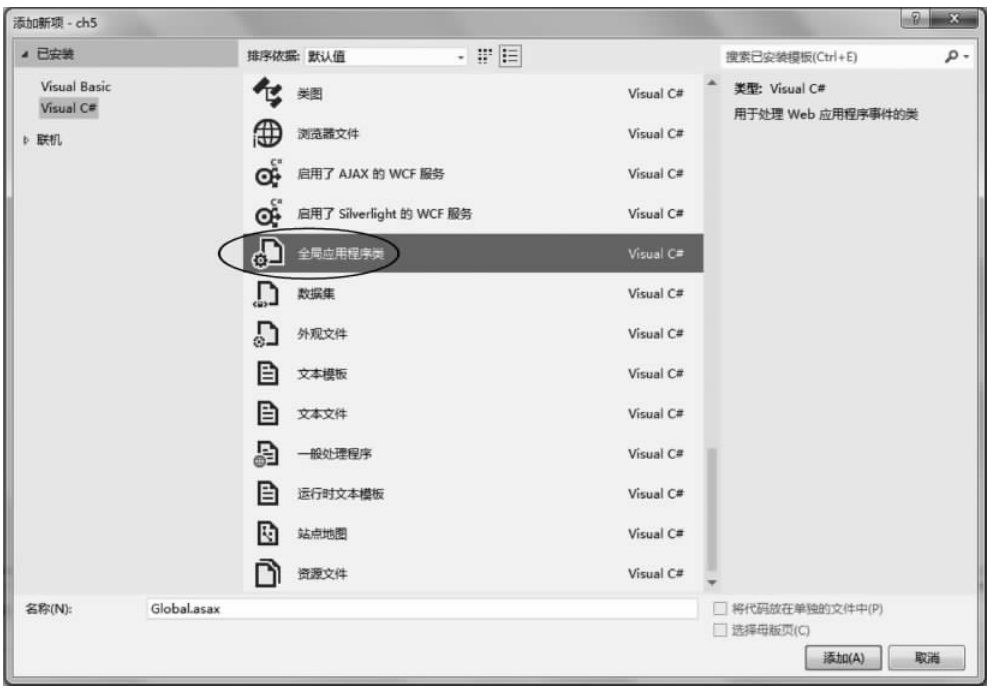

图 5.24　添加 Global.asax 文件

② 若本网站中没有 Global.asax 文件，则添加一个，否则打开该文件，修改其内容如下（只添加粗体代码部分，其他不变）：

```
<%@ Application Language = "C#" %>
<script runat = "server">
void Application_Start(object sender, EventArgs e)
{   //在应用程序启动时运行的代码
    Application.Lock();
    Application["counter"] = 0;
    Application.UnLock();
}
void Application_End(object sender, EventArgs e)
{
    //在应用程序关闭时运行的代码
}
void Application_Error(object sender, EventArgs e)
{
    //在出现未处理的错误时运行的代码
}
void Session_Start(object sender, EventArgs e)
{   //在新会话启动时运行的代码
    Application.Lock();
    Application["counter"] = (int)Application["counter"] + 1;
    Application.UnLock();
}
void Session_End(object sender, EventArgs e)
{
    //在会话结束时运行的代码
```

```
        //注意: 只有在 Web.config 文件中的 sessionstate 模式设置为
        //InProc 时才会引发 Session_End 事件。如果会话模式设置为 StateServer
        //或 SQLServer,则不会引发该事件
        Application.Lock();
        Application["counter"] = (int)Application["counter"] - 1;
        Application.UnLock();
    }
    </script>
```

③ 添加一个 WebForm11 的网页,在其中拖放一个标签 Label1。在该网页上设计如下事件过程:

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

④ 单击工具栏中的 Internet Explorer 按钮运行本网页,看到在线人数为 1,如图 5.25 所示。不关闭 IE,再次启动百度浏览器,并输入地址“http://localhost:53072/WebForm11.aspx”,此时的网页界面如图 5.26 所示,在线人数变为 2。这是因为在第一次启动本网页时,执行一次 Global.asax 文件中的 Application_Start 事件过程,将 Application["counter"]置为 0,则以后每次出现一次会话,便执行一次 Global.asax 文件中的 Session_Start 事件过程,使 Application["counter"]增 1,当退出会话时,执行一次 Global.asax 文件中的 Session_End 事件过程,使 Application["counter"]减 1,所以 Application["counter"]的值就是在线的人数。

图 5.25 WebForm11 网页运行界面一

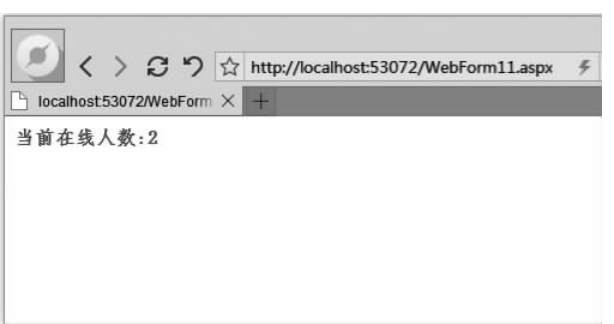

图 5.26 WebForm11 网页运行界面二

5.11 ASP.NET 网页框架

在前面的各种示例中已经介绍了动态网页设计的基本方法,本节归纳网页执行和传统桌面应用程序执行的不同点,以及网页的生命周期和网页的生命周期中的事件。

5.11.1 网页的执行方式和 ASP.NET 状态管理

当用户在浏览器中输入一个 URL 的时候,浏览器显示出带有文本框、图像、命令按钮或其他网页元素的 Web 网页。当用户输入文本框内容,然后单击某个提交按钮时,网页上就显示出新的数据,对用户的操作进行响应。所谓提交按钮,是指其 HTML 元素的 type 属性为“submit”或者导致执行表单 submit()方法的网页元素。那么这一执行过程是怎样

的呢？

对于传统的桌面应用程序，有一个用户坐下来开始运行这个应用程序，并且是在一段时间内不停地运行它，然后关闭这个应用程序。用户与这个应用程序进行交互的这段时间称为会话。如果用户在某个地方输入他的姓名，应用程序会在一定时期内记住这个姓名（这段时期称为该变量的作用域），以备随后使用。用户的姓名以及用户所做的其他更改就是所谓应用程序的状态（state）。应用程序至少需要在同一会话期间内保持这个状态，有时候需要在不同会话之间保持这个状态。

桌面应用程序总是需要保持应用程序的状态，而 Web 网页不需要。事实上，Web 应用程序模型被特意而明确地设计为“无状态”的。而 Web 用户期待 Web 应用程序能够像桌面应用程序一样具有相同的行为，所以 Web 应用程序也需要以某种方式来保持状态。

其实现方式是每次从服务器请求网页时都会创建网页类的一个新实例。这通常意味着在每次往返过程中将会丢失所有与该页面及其控件关联的信息。例如，如果用户将信息输入到 HTML 网页上的文本框中，此信息将发送到服务器，但是不会返回到客户端。为了克服 Web 编程的这一固有局限性，ASP.NET 网页框架包含几种状态管理功能，可以将往返过程之间的网页和控件值保存到 Web 服务器。ASP.NET 状态管理有以下两种方式。

1. 基于客户端的状态管理方法

基于客户端的状态管理涉及在网页中或客户端计算机上存储信息，在各往返行程间不会在服务器上维护任何信息。其常见技术有视图状态、控件状态和 Cookie 等。

（1）视图状态

视图状态管理是通过使用 ViewState 对象实现的，在前面已介绍。

（2）控件状态

用户可以使用 ViewState 属性来保存控件状态数据，也可以使用 ControlState 属性保持特定于某个控件的属性信息。

（3）隐藏域

开发人员可以在 ASP.NET 网页的表单中自己创建 HiddenField 控件（自定义隐藏域）来存储任何特定于网页的信息。隐藏域在浏览器中不以可见的形式呈现，但可以像对待标准控件一样设置其属性。当向服务器提交网页时，隐藏域的内容将在 HTTP 窗体集合中随其他控件的值一起发送。

（4）Cookie

Cookie 管理是通过使用 Cookie 对象实现的，在前面已介绍。

（5）查询字符串

查询字符串是在网页 URL 的结尾附加的信息。例如：

```
http://www.mysite.com/default.aspx?name = smith&age = 30
```

在上面的 URL 路径中，查询字符串以问号（?）开始，并包含两个属性/值对，一个名为 name，另一个名为 age。

查询字符串提供了一种维护状态信息的方法，这种方法很简单，但有使用上的限制。大多数浏览器和客户端设备会将 URL 的最大长度限制为 2083 个字符。

2. 基于服务器的状态管理方法

ASP.NET 提供了多种方法用于维护服务器上的状态信息，而不是保持客户端上的信息。

通过基于服务器的状态管理，为了保留状态，可以减少发送给客户端的信息量，但它可能会使用服务器上高成本的资源。其常见技术有应用程序状态、会话状态及配置文件属性。

应用程序状态管理是采用 Application 对象实现的，会话状态管理是采用 Session 对象实现的。有关配置文件属性的内容将在本章后面介绍。

回到前面的问题，用户在浏览器中输入一个 URL，即发出一个网页请求，Web 服务器找到相应的 ASP.NET 网页，将其交给 ASP.NET 引擎，ASP.NET 引擎执行相关的程序代码，生成 HTML 网页文件，由 Web 服务器发给用户浏览器进行显示（并保存这个网页的相关信息，以便识别是第一次加载还是回传），这是第一次网页显示，其详细过程如图 5.27(a)所示。

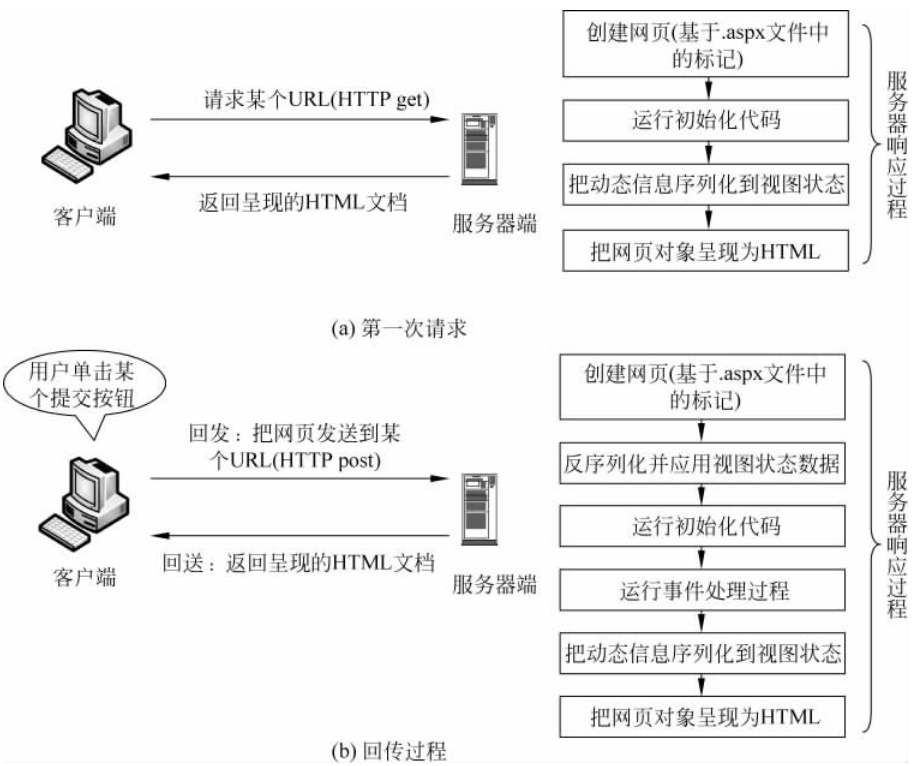

图 5.27　ASP.NET 网页请求

当用户输入文本框内容，然后单击命令按钮，这称为提交（并非每次用户操作都要提交，如果有客户端代码，如 JavaScript 代码，这些代码的执行是在客户端进行的，不需要提交），网页再次传给 Web 服务器，Web 服务器再次交给 ASP.NET 引擎，该引擎执行相关的事件处理程序，又生成 HTML 网页文件，由 Web 服务器发给用户浏览器进行显示，这称为第二次显示，这一过程称为回传过程，其详细过程如图 5.27(b)所示。这种交互过程可以一直进行下去，直到用户终止执行。

5.11.2　网页的生命周期

当请求 Web 服务器上的一个 ASP.NET 网页时，这个网页就会被加载到 Web 服务器的内存中，经过处理后，发送给用户，即从内存中卸载出去。这个过程称为网页的生命周期，它的目标就是为发送网页请求的浏览器呈现适当的 HTML 网页。

网页生命周期有两种稍微不同的顺序：一种是首次加载网页时，另一种是在回传过程中再次加载网页时，图 5.28 说明了网页生命周期的各个阶段的顺序。

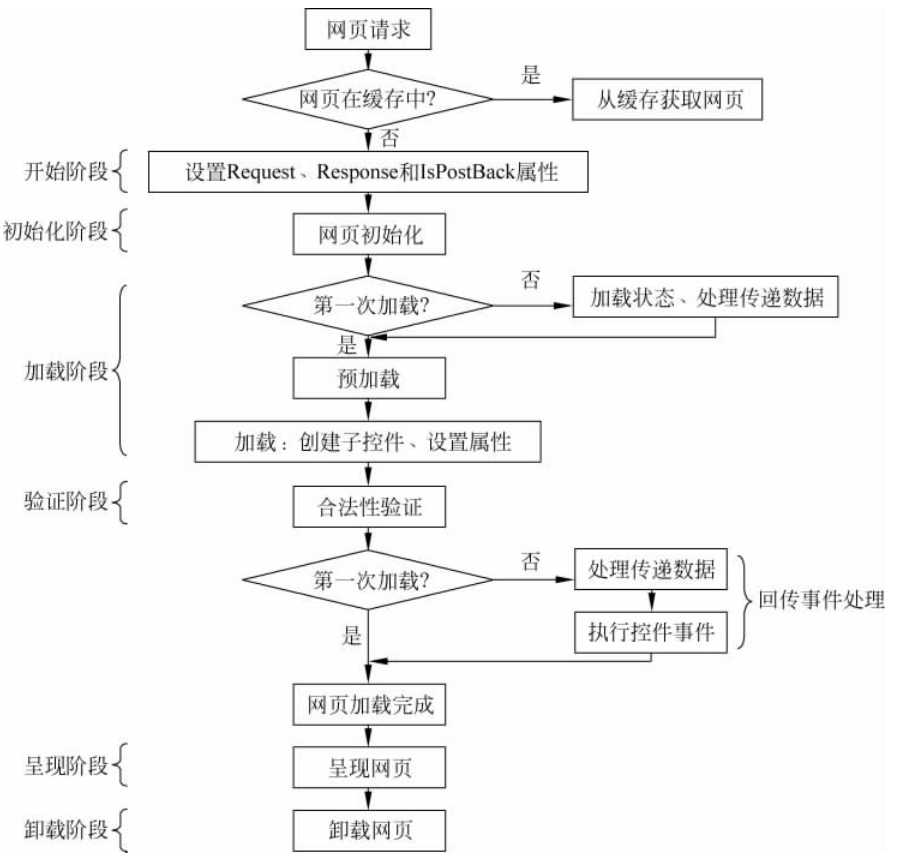

图 5.28　ASP.NET 网页的生命周期

在第一次加载网页时，生命周期由以下几个部分组成：

① 浏览器将对网页的请求发送到 Web 服务器上，ASP.NET 引擎首先要确定这个网页是否已经存在于缓存（专门用于保存最近使用过的对象的一部分内存空间）中。如果缓存中存在这个网页，Web 服务器就会取得这个网页并返回到浏览器上，本次任务就完成了；如果缓存中不存在这个网页，这个时候才真正开始网页的生命周期。

② 在开始阶段，回传模式就已经确定了。如果这个网页是另一个网页发出的请求，那么它不是一个回传（IsPostBack 置为 False）；如果这个网页是返回到 Web 服务器进行处理和重新显示的，那么它才是一个回传（IsPostBack 置为 True）。对 IsPostBack 和 PreviousPage 属性进行相应的设置，同时也要设置网页的 Request 和 Response 属性。

③ 在网页初始化阶段包含了两个通常要处理的事件，即 PreInit 和 Init 事件。如果没有编写相应的事件处理程序代码，ASP.NET 会执行事件的默认行为。在 PreInit 事件中，先在网页初始化之前确定目标设备的类型、设置母版页、创建控件树以及给控件分配唯一的 ID，这些 ID 都已经在代码中规定好了。在这个阶段，还会在网页中加载和应用用户个性化设置和主题等。

PreInit 是生命周期中的第一个能被拦截和处理的事件，这就意味着它是第一个可以编码

的事件，可以通过编写代码来更改初始化网页的默认行为。在 Init 事件中，将会读取或初始化控件的属性。如果这是一个回传，那么存储在视图状态（视图状态指网页及网页上所有控件的状态，而状态是指当前用户的当前会话中所有的控件和变量的当前值）中的任何值现在还没有恢复到控件中。

④ 在加载事件中，所有的控件属性都要进行设置。用户可以使用视图状态信息，也可以访问网页的控件层次系统中的所有控件。通常在 Page_Load 方法中对加载阶段的处理例程进行更改。

⑤ 在验证阶段会调用网页中所有的验证控件的 Validate 方法，并设置相应控件和网页的 IsValid 属性。

⑥ 在呈现（或绘制）阶段，用户个性化设置、控件及视图状态都会保存下来。依次调用网页中的控件以绘制在浏览器上，也就是将各个控件组合到 HTML 标记中，标记内容可以通过网页的 Response 属性进行访问。通常使用 Page_PreRender 方法来处理 PreRender 事件。在 PreRender 事件中，HTML 实际上已经生成好，并已经发送到了原来请求的网页中，除非有自定义的用户控件。

⑦ 卸载阶段是生命周期中的最后一个阶段，Unload 事件可以处理任何最后的扫尾工作，如关闭文件、释放数据库连接等。

在回传过程中，生命周期与第一次加载网页的过程相似，但有以下几点不同：

① 在加载阶段，在完成网页初始化之后会根据需要加载和应用视图状态。

② 在验证阶段完成后，回传的数据已经经过处理，这个时候才执行控件的事件处理程序。这是很重要的，只有在网页初始化完成，Load 事件已经处理完毕之后才会调用控件的事件处理程序（如命令按钮的 Click 事件）。之所以重要是因为各种事件处理程序的执行顺序对程序处理逻辑来说是十分关键的。

5.11.3 网页生命周期中的事件

在生命周期的每个阶段都提供了可以使用的方法和事件，供程序开发人员来重写 ASP.NET 引擎的默认处理行为，或增加自己的程序处理逻辑。

网页生命周期中的事件如图 5.29 所示。从中看到，尽管 Web 应用程序模型和传统的桌面应用程序模型有较大的不同，但它通过各种事件和方法提供了极大的编程灵活性，掌握网页生命周期以及事件，就可以编写类似于桌面应用程序那样具有很强交互性的 Web 应用程序。总之，深入理解 ASP.NET 网页框架理论对于开发复杂的 Web 应用程序是十分必要的。

图 5.29 网页生命周期中的事件

练习题 5

1. 简述 ASP.NET 网页的处理过程。

2. 简述 Page 对象的 IsPostBack 属性和 IsValid 属性的含义，分别说明 Page 对象的 Init 事件、Load 事件和 Unload 事件何时发生。

3. 简述 Response 对象的作用。

4. 简述 Request 对象的作用。

5. 简述使用 Response.Redirect 方法、Server.Transfer 方法和 Server.Execute 方法实现网页转向上的差异。

6. 简述使用 Application 对象和 Session 对象保存用户信息上的差异。

7. 简述 Cookie 对象的作用。

8. 简述 ViewState 对象的作用。

9. 简述 Global.asax 文件的作用。

10. 简述网页的生命周期。

11. 简述网页生命周期中的主要事件。

上机实验题 5

在 ch5 网站中添加一个名称为 Experment5 的网页，用于输入学生的学号、姓名、性别和班号，均采用文本框接受用户输入。当用户单击“确定”命令按钮时在另一个网页 Experment5-1 中显示该学生信息，如图 5.30 所示。

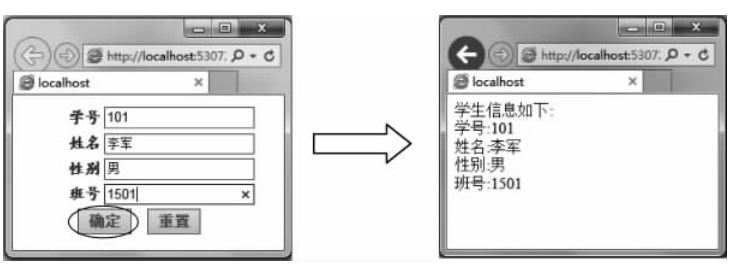

图 5.30　上机实验题 5 网页的运行界面