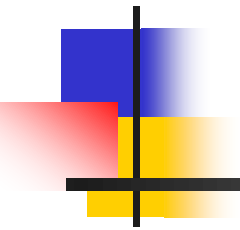
第6章
ASP.NET 的对象

Table of contents

- 6.1 HTTP请求处理
  - 6.1.1 Response对象
  - 6.1.2 Request 对象
  - 6.1.3 Server 对象
- 6.2 状态信息保存
  - 6.2.1 Application对象
  - 6.2.2 Session 对象
  - 6.2.3 Cookie 对象
  - 6.2.4 ViewStates对象
- 6.3 实例

表6-1 ASP.NET内置对象

对象名	功 能	ASP.NET 类
Page	用于设置与网页有关的属性、方法和事件	
Request	读取客户端所提交的数据	HttpResponse
Response	发送信息到客户端	HttpRequest
Application	为所有用户提供共享信息	HttpApplication State
Session	存储特定用户的信息，可以在同一网站的多个页面间共享信息	HttpSessionState
Cookie	在客户端的磁盘上保存用户的数据	HttpCookie
Server	提供服务器端的属性和方法	HttpServerUtility
Context	封装了每个用户的会话、当前HTTP请求和请求页的信息	HttpContext
ViewState	在同一个页面的多次请求间保存状态信息	StateBag
Trace	用于对页面进行跟踪	TraceContext



6.1 HTTP 请求处理

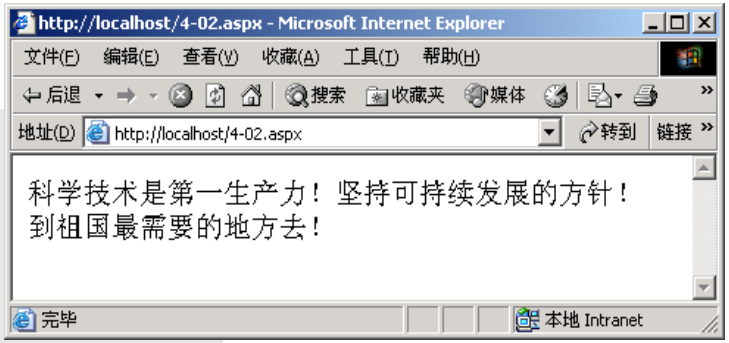
6.1.1 Response对象

Response对象的主要功能是向浏览器输出信息。

方法	说明
AppendCookie	将一个HTTP Cookie添加到内部Cookies集合
AppendToLog	将日志信息添加到IIS日志文件中
BinaryWrite	将一个二进制字符串写入HTTP输出流
Clear	清除服务器缓存中的所有数据
Flush	把服务器缓存中的数据立刻发送到客户端
End	停止处理当前文件，并返回结果
Redirect	用于页面的跳转，使浏览器重定向到另一个URL
Write	直接向客户端输出数据
WriteFile	向客户端输出文本文件的内容

输出数据 Response.Write

利用 Write 方法可以在客户端输出信息，效果和Label标签控件一样。




```
<% @ Page Language="C#" %>
<script language="C#" runat="server">
void Page_Load(Object sender,EventArgs e){
        Response.Write("科学技术是第一生产力！ ");
}
</script>
<%Response.Write("坚持可持续发展的方针！ ");%><br>
<% = "到祖国最需要的地方去！ "%>
```

网页转向 Response.Redirect

- 使用Redirect方法就可以引导至另一个页面。语法如下：
  - Response.Redirect (网址变量或字符串)

- 例如：
  - Response.Redirect("http://www.edu.cn")　　'引导至中国教育网
  - Response.Redirect("other.asp")　'引导至其他网页
  - theURL="http://www.pku.edu.cn"
    Response.Redirect(theURL)　　导至变量表示的网址

Response对象的属性

属性	说明
Buffer	指定页面输出时是否需要缓冲区
Cache	获得网页的缓存策略（过期时间、保密性等）
Charset	设置输出到客户端的HTTP字符集
ContentEncoding	获取或设置输出流的HTTP字符集
ContentType	获取或设置输出流的MIME类型，默认为 text/HTML
Cookies	用于获得 HttpResponse 对象的 Cookie 集合
Expires	设置页面在浏览器中缓存的时限，以分钟为单位
IsClientConnected	表明客户端是否与服务器端连接，其值为True或False
Output	启用到输出 HTTP 响应流的文本输出
Status	服务器返回的状态行的值

例6-1 检查客户端的联机状态

（使用IsClientConnected属性和Redirect方法）

```
protected void Page_Load(object sender, EventArgs e) {
//判断客户端是否与服务器连接
if (Response.IsClientConnected)
    Response.Redirect("other.aspx");    //跳转到当前目录的另一个页面
else
    Response.End();     //停止执行当前页面的代码
}
```

例6-2 判断向客户端的输出

```
protected void Page_Load(object sender, EventArgs e) {
    //设置服务器缓冲为true
Response.Buffer = true;
//获取当前时间的小时数
    int currentHour = DateTime.Now.Hour;
//满足某个特定条件
    if (currentHour == 0) {
       Response.Write("时间到，服务器将停止输出");
       Response.Flush();        //立即发送缓冲区中的数据
       Response.Clear();        //清除缓冲区中的全部数据
       Response.End();          //停止执行代码
  }
    else {
       Response.Write("服务器正常工作中.....");
    }
}
```

使用Response对象设置Cookie

- Response对象的数据集合只有一个，那就是Cookies集合。利用Response.Cookies的Add方法可以创建一个新的会话Cookie。
- 语法：Response.Cookies.Add(Cookies对象名)；


```
//创建一个Cookie
HttpCookie myCookie = new HttpCookie("Username");
myCookie.Value = "张三" ;
//将新Cookie加入Cookies集合中
Response.Cookies.Add(myCookie);

上述代码也可以简化为：
Response.Cookies["Username"].value = "张三" ;
```

6.1.2 Request对象

- Request对象功能是从客户端得到数据，即获得客户填写在 Form 表单中的信息。

数 据 集 合	功 能 说 明
Cookies	读取HTTP请求中的cookies数据
Form	取得客户端用 POST方法传递的数据
QueryString	取得客户端用 GET方法传递的数据
ServerVariables	取得Web服务器端的环境变量

引用数据集合的格式为：

Request.集合名[“变量名”]

1）获取客户端提交的表单数据

- 客户端通过HTML的Form表单向服务器提交数据。
- 通常，提交数据有POST和GET两种方法。
- 当客户端使用GET方法提交，服务器通过QueryString集合获取数据。

  Request.Querystring [“userid”]　//读取URL中userid输入域的值
- 当客户端使用POST方法提交，服务器通过Form集合取出信息。

  Request.Form [“userid”]

注意，两种方式都可以简化为:

Request[“userid”]

2）读取保存的Cookie信息

要从Cookies中读取数据，则要使用Request对象的Cookies集合，其格式为：

Request.Cookies[“Cookies名称”]


```
//获取Cookie对象
HttpCookie MyCookie = Request.Cookies[“Username”];
//读取Cookie值
string myName = MyCookie.Value;
```

3）读取服务器端的环境变量

- 利用Request对象的ServerVariables数据集合可以取得服务器端或客户端的环境变量信息。
- 语法：

Request.ServerVariables[“环境变量名称”]

Request.ServerVariables[“REMOTE_ADDR”]

返回发出请求的客户机的IP地址。

Request.ServerVariables[“LOCAL_ADDR”]

返回接受请求的服务器的地址。

6.1.3 Server 对象

- 通过Server 对象可以访问服务器的方法和属性。比如，得到服务器上某文件的物理路径和设置某文件的执行期限，等等。
- **属性：**
  - ScriptTimeout，MachineName

Server 对象的ScriptTimeout属性规定了脚本文件执行的最长时间（默认90秒），主要用来防止某些可能进入死循环的错误导致服务器过载问题。如果将其设置为-1，则永远不会超时。

```
Server.ScriptTimeout=100      //指定服务器处理脚本的超时期限为100秒
limit=Server.ScriptTimeout    //将设置过的超时期限存放到一个变量中
```

Server对象的方法

方法	说明
ClearError	清除前一个异常
CreateObject	创建COM对象的一个服务器实例
Execute	停止执行当前网页，转到另一个网页执行，执行完后返回原网页
GetLastError	获得前一个异常，可用于访问错误信息
Transfer	停止执行当前网页，转到新的网页执行，执行完后不返回原网页
MapPath	将指定的虚拟路径映射为物理路径
HTMLEncode	对字符串进行HTML编码，并将结果输出
HTMLDecode	对HTML编码的字符串进行解码，并将结果输出
UrlEncode	将字符串按URL编码规则输出
UrlDecode	对在URL中接收的HTML编码字符串进行解码，并将结果输出

1）Server 对象的CreateObject 方法

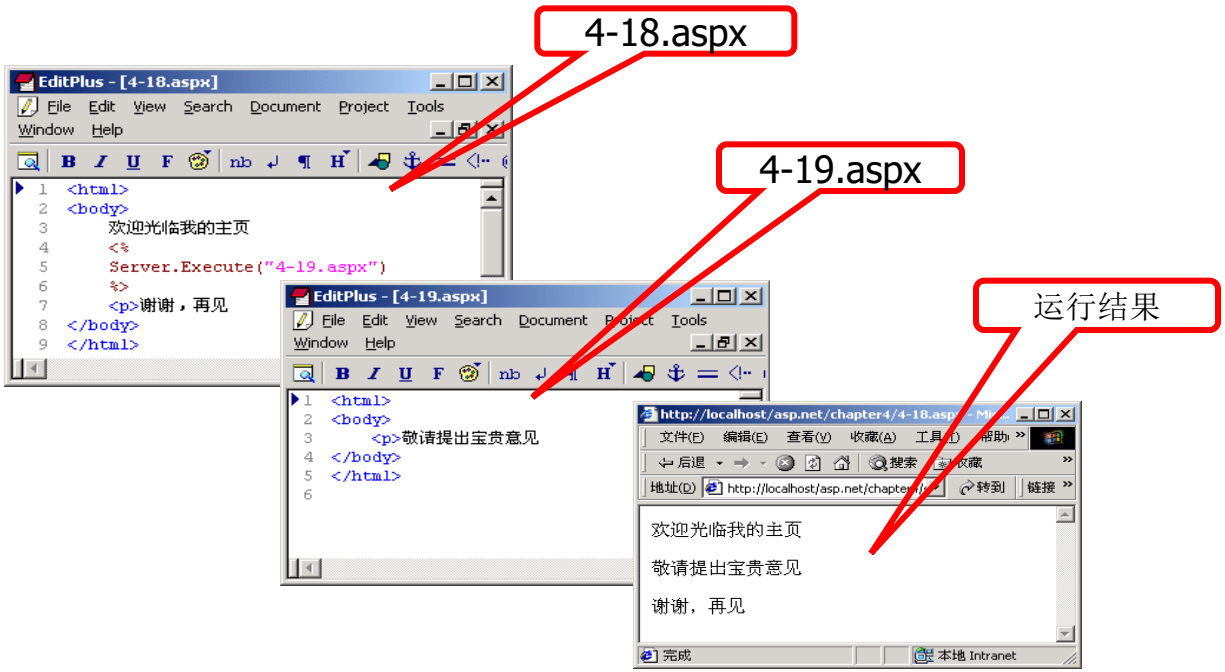
- 该方法用于创建 ActiveX组件、或应用对象的实例。
- 语法如下：
  - Server.CreateObject (ActiveX Server组件)

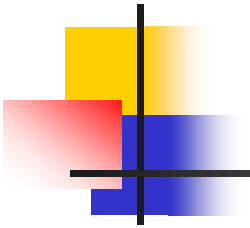
例如：

Server.CreateObject (“ADODB.Connection")

2）Server 对象的Execute 方法

- 该方法用来停止执行当前网页，转到新的网页执行，执行完毕后返回原网页，继续执行 Execute 方法后面的语句。
- 语法：Server.Execute (string URL)





3）Server 对象的Transfer 方法

- Transfer方法的作用是将一个正在执行的脚本文件的控制权转移给另一个文件执行，执行完毕后不返回原文件。
- **语法：** Server.Transfer（string URL）

Server对象的Transfer方法、Execute方法和Response对象的Redirect方法 比较

- 使用Server.Transfer和Server.Execute方法只能转移到本网站的其他网页。而Response.Redirect方法可以转移至任一网站的任一网页。
- Execute方法相当于子程序的调用，执行完后会返回原程序。而Transfer方法、Redirect方法都不再返回原程序执行。
- 使用Transfer方法调用另一个文件，会进行控制权的转移，并且所有内置对象的值都一起“转移”，并保留至新的网页。而Redirect方法则仅仅转移控制权。
- 使用Transfer和Execute方法时，客户端与服务器只进行一次通信。而Redirect方法在重定向过程中，客户端与服务器进行两次来回的通信。第一次通信是对原始页面的请求，得到一个目标已经改变的应答，第二次通信是请求指向新页面，得到重定向后的页面。

4）Server 对象的MapPath 方法

⑩MapPath方法可以将指定的虚拟路径转化为物理路径。

例8-3 Server对象的MapPath方法

```
protected void Page_Load(object sender, EventArgs e)  {
   Response.Write("服务器的根路径：" + Server.MapPath("./"));
   Response.Write("<br>当前文件所在的物理路径：" + Server.MapPath(".") );
   Response.Write("<br>Default.aspx文件的位置：" +
         Server.MapPath("Default.aspx"));
}
```

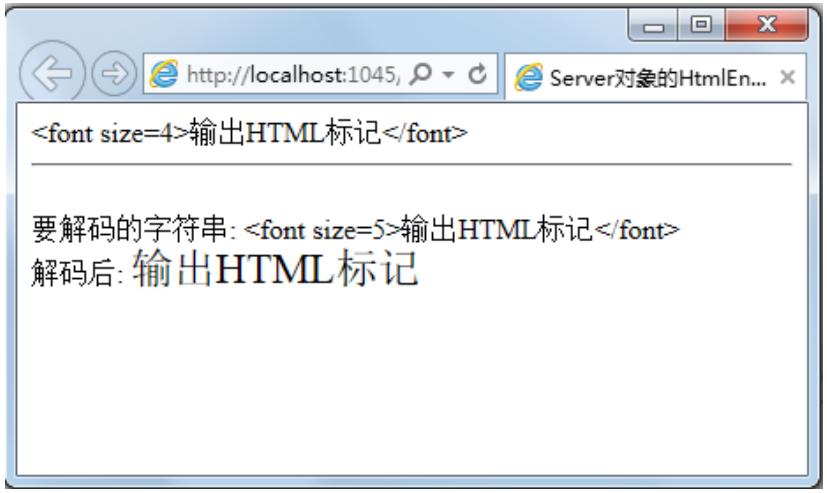
5）对HTML进行编码和解码

- HTMLEncode 方法允许你对HTML系统标记（“<”和“>”）重新编码，即把该标记转换为转义符发送到浏览器，这样就可以在浏览器中看到“<”和“>”标记。
- HtmlDecode方法用于对已经进行HTML编码的字符串进行解码，是HtmlEncode的反操作。

转义符举例：<HTML> --→ <HTML>

例6-4 Server对象的HtmlEncode和HtmlDecode方法

```
protected void Page_Load(object sender, EventArgs e)  {
  string enStr = Server.HtmlEncode("<font size=4>输出HTML标记</font>");
  Response.Write(enStr);
  Response.Write(“<hr>”);
  string deStr = "&lt;font size=5&gt;输出HTML标记&lt;/font&gt;";
  Response.Write("<br>要解码的字符串: " + deStr) ;
  Response.Write("<br>解码后: " + Server.HtmlDecode(deStr));
}
```



6）对URL进行编码和解码

- UrlEncode方法用于编码字符串，以便通过URL从Web服务器到客户端进行可靠的HTTP传输。

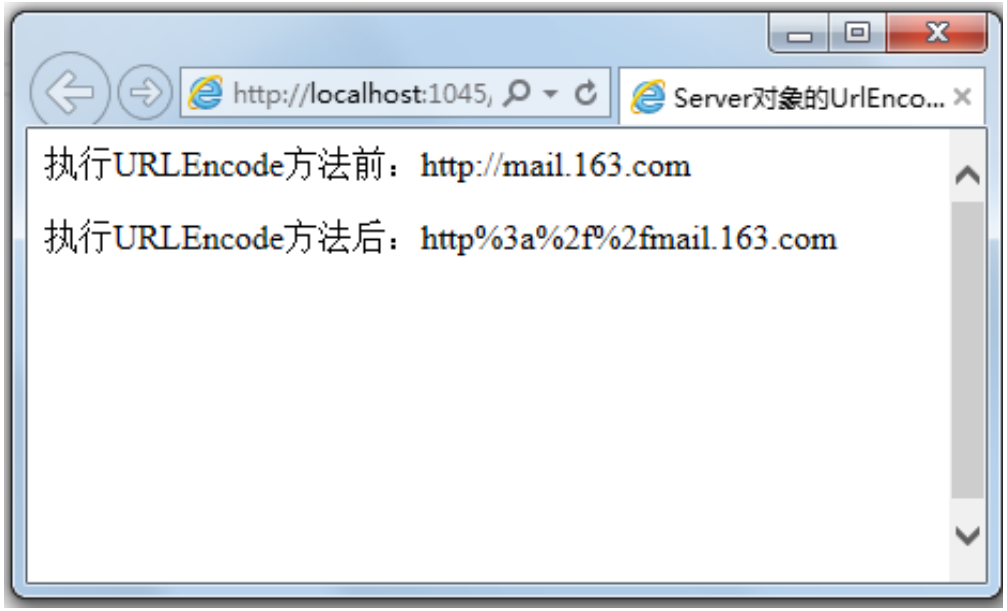
  格式为：Server.UrlEncode (string URL)

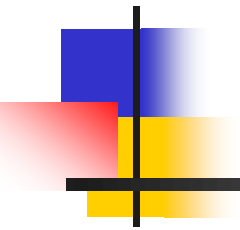
- UrlDecode方法用于对字符串进行解码，可以还原被编码的字符串。

  格式为：Server.UrlDecode (string URL)

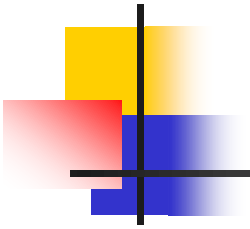
例6-5 Server对象的URLEncode方法应用

```
protected void Page_Load(object sender, EventArgs e)
{
  String str="http://mail.163.com" ;
  Response.Write("<p>执行URLEncode方法前： " + str ) ;
  String strNew = Server.UrlEncode(str);
  Response.Write ("<p>执行URLEncode方法后： " + strNew );
}
```





6.2 状态信息保存



6.2.1 Application对象

- Application对象的用途是**记录整个网站**的信息，也就是说，登陆这个网站的所有用户都可以修改同一个Application对象。比如，聊天室和网站计数器。
- Application对象内保存的信息可以在Web服务整个运行期间保存，并且可以被调用Web服务的所有用户使用

1. Application 对象的键值

Application对象通过使用用户自定义的数据键值来存取信息。

Application(“键名”)= 值

一旦我们分配了 Application 对象的键值，它就会持久地存在，直到关闭 WEB 服务使得 Application 停止。

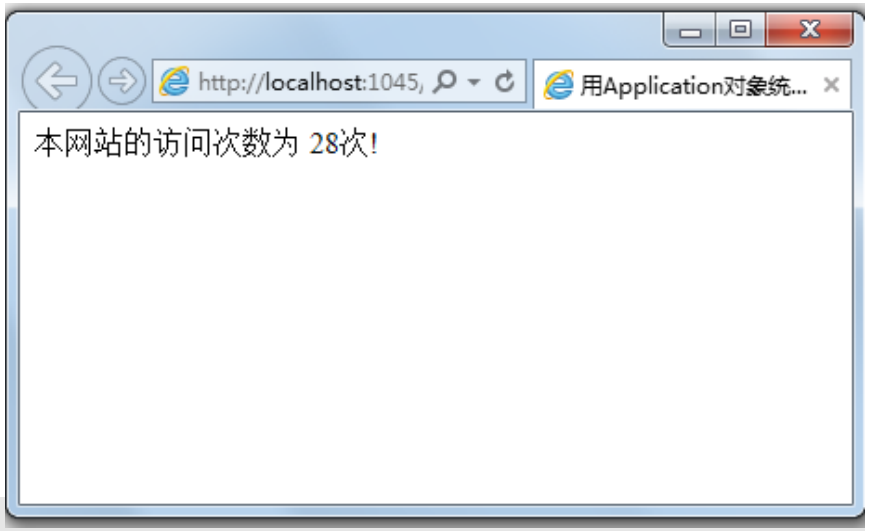

```
//设置Application对象中的数据
        Application[“username"]=“Admin";
//读取Application对象中的数据
        string user;
        user = Application["username“].ToString();
```

注意：Application(“键名")的返回值是一个Object类型的数据，操作时应注意数据类型的转换。

2. Application 对象的方法

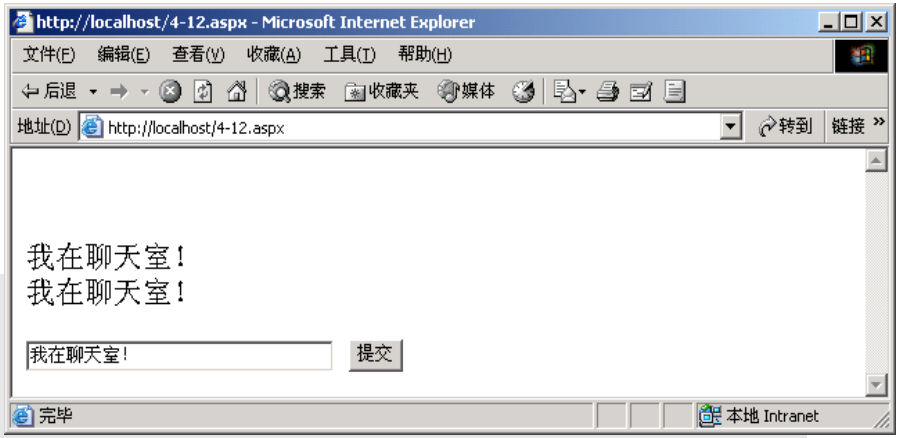
方法	说明
Add()	向 Application 状态添加新对象
Clear()	从 Application 状态中移除所有对象
Remove()	从 Application 集合中按照键名移除项
Lock()	锁定Application对象
UnLock()	解除对Application对象的锁定

例6-6 网站访问计数器




```
protected void Page_Load(object sender, EventArgs e)  {
  if ( Convert.ToInt32(Application["count"]) < 1)  {
      Application[“count"] = 1;
  }
  Application.Lock();
  Application["count"] =Convert.ToInt32(Application["count"]) + 1;
  Application.UnLock();
  Response.Write("本站访问次数为 "+Application[“count"].ToString() +"次!");
}
```

聊天室应用



案例名称：单一文件聊天室

```
<% @ Page Language="C#" %>
<%
        string mywords=Request["mywords"];
        Application.Lock();
        Application["chat_content"] = Application["chat_content"]
                        +"<br>"+mywords;     //保存发言信息
        Response.Write (Application["chat_content"]);
        Application.UnLock();
%>
<FORM ACTION="4-12.aspx" METHOD="post">
<INPUT TYPE="text" SIZE="30" NAME="mywords" VALUE="我在聊天室！ ">
<INPUT TYPE="submit" VALUE="提交">
</FORM>
```

3. Application 对象的事件

- **Application_Start**

Application_Start 事件在创建与服务器的首次会话之前发生。

当服务器启动并允许用户请求时就触发 Application_Start 事件。

- **Application_End**

与Application_Start事件相反，在整个Web应用程序退出时发生，用于回收占用的服务器资源。

- **Application_BeginRequest**

每次页面请求开始时触发（理想情况下是在页面加载或刷新时）

- **Application_EndRequest**

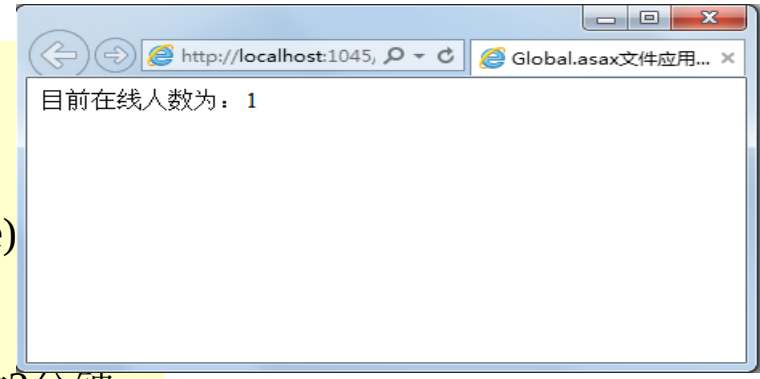
每次页面请求结束时（即每次在浏览器上执行页面时）触发

4. Global.asax 文件

Application事件的代码必须放在**global.asax**文件中：

```
<%@ 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
    }
    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
    }
</script>
```

例6-7 使用Global.asax文件统计在线访问人数。




```
<%@ Application Language="C#" %>
<script runat="server">
//当应用程序启动时，设置全局变量VistorCount为0
protected void Application_Start(object sender,EventArgs e)
    Application[“VisitorCount”]=0;
}
//当会话开始时，在线人数值加1；设定会话超时时限为2分钟
protected void Session_Start(object sender,EventArgs e)  {
Application.Lock( );
Application[“VisitorCount”]=(int)Application[“VisitorCount”]+1;
Application.UnLock( );
Session.Timeout=2;
}
//当会话结束时，在线人数值减1
protected void Session_End(object sender,EventArgs e)  {
Application.Lock( );
Application[“VisitorCount”]=(int)Application[“VisitorCount”]-1;
Application.UnLock( );
}
</script>
```

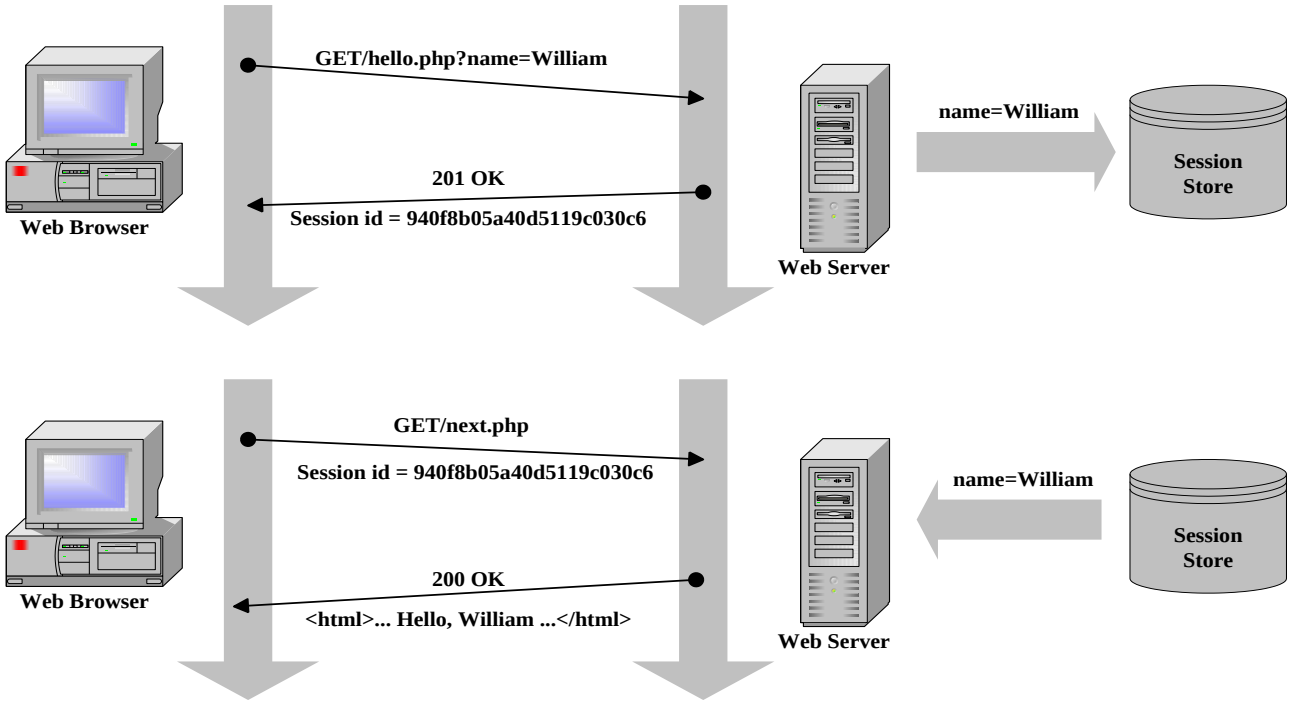
```
protected void Page_Load(object sender, EventArgs e)
  string info=" 目前在线人数为：{0}";
  info= string.Format(info, Application["VisitorCount"]);
  lblInfo.Text = info;
}
```

Application对象和Session对象的事件

事件	说明
Application_Start	调用当前应用程序目录（或其子目录）下的第一个 ASP.NET 页面时触发。
Application_End	应用程序的最后一个会话结束时触发。
Application_BeginRequest	每次页面请求开始时触发（理想情况下是在页面加载或刷新时）
Application_EndRequest	每次页面请求结束时（即每次在浏览器上执行页面时）触发
Session_Start	每次新的会话开始时触发
Session_End	会话结束时触发。

6.2.2 Session 对象

- 使用Session对象可以记载特定客户的信息。
- 当用户第一次访问一个网站时，服务器就给该用户建立了一个Session对象，并分配一个唯一的SessionID。Session对象所创建的变量如同全局变量一样，在该用户访问的每个Web页面程序中都可以直接读取。
- 通过向客户端发送Cookie可以管理服务器上的该客户的Session对象。



1. Session对象的属性

属性	说明
SessionID	获取会话的唯一标识符（只读，长整型）
Timeout	获取和设置会话时间的超时时限，默认值为 20 分钟
Count	获取会话状态集合中的项数
Item	获取或设置会话值的名称
IsNewSession	若该会话是由当前请求创建的，该属性将返回值 true

第一次给Session变量赋值即自动创建Session对象：

Session [“键名”] = 值

设置有效期和使Session失效

--- Timeout 属性

- Session对象有它的有效期，默认为20分钟。客户端每次新打开一个浏览器窗口，就会创建一个Session对象，如果超出20分钟没有刷新或请求网页，或者关闭了浏览器，则该Session对象就会自动结束。
- 修改有效期语法如下：
  - Session.Timeout=整数（分钟）

例：Session.Timeout=90　　'将有效期改为90分钟

2. Session对象的方法

方法	说明
Abandon	清除用户的Session对象，释放系统资源。调用改该方法后会出发session_OnEnd事件
Add	添加一个新项到会话状态中
Clear	清除当前会话状态的所有值
CopyTo	将当前会话状态值的集合复制到一个一维数组中
Remove	删除会话状态集合中的项
RemoveAt	删除会话状态集合中指定索引处的项
RemoveAll	清除会话状态集合中的所有的键和值

删除Session对象

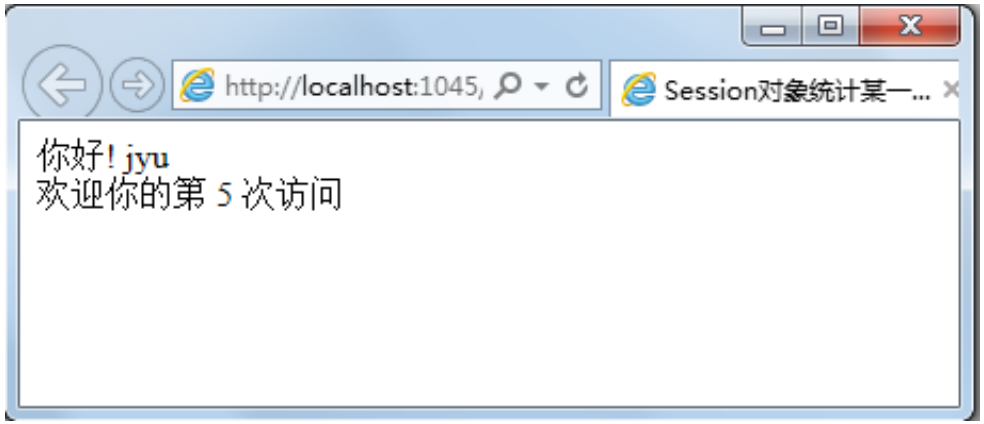
--- Abandon 方法

- Abandon 方法删除存储在Session对象中的所有对象和变量，释放系统资源。
- Session对象到期后会自动清除，但到期前可以用Abandon方法强行清除。
- 语法：Session.Abandon
- 例如： Session("user_name")= "萌萌"

  Session.Abandon

例6-8 使用Session对象统计某一用户的访问次数

```
protected void Page_Load(object sender, EventArgs e)  {
  //创建Session变量username
  Session["username"] = "jyu" ;
  //创建Session变量visits
  Session["visits"] = Convert.ToInt32(Session["visits"]) + 1;
  String StrName = Session["username"].ToString();
  Response.Write ("你好! " + StrName) ;
  Response.Write ("<br> 欢迎你的第 " + Session["visits"] + " 次访问");
}
```

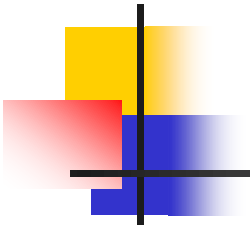


3. Session对象的事件

- **Session_OnStart事件：**在用户与服务器创建一个新的会话时触发，服务器在执行请求的页面之前先处理该脚本。可以在该事件中定义所有在页面中需要使用的Session变量。

- **Session_OnEnd事件：**在Session结束时被调用。当程序调用了Session对象的Abandon方法或发生超时情况时，触发Session_OnEnd事件。Session_OnEnd事件一般用于清理系统对象或变量的值，释放系统资源。

这两个事件的代码也存储在Global.asax文件中。



6.2.3 Cookie对象

- 利用 Cookie 是一种可以在客户端保存信息的方法。
- Session 对象将客户的所有信息保存在服务器上，Cookie 对象将客户的信息保存在客户端的浏览器上。
- Session对象并不能够持久的保存用户信息，当用户在限定时间内没有任何操作时，用户的Session对象将被注销和清除。使用Cookie对象能够持久化的保存用户信息。

1. 什么是Cookie

- Cookie是一个很小的文本文件，由网站服务器在用户第一次访问时生成，发送到客户端的硬盘里，用来存储用户的特定信息。当用户再次访问该站点时，浏览器就会在本地硬盘上查找与该网站相关联的Cookie。如果存在，就将它与页面请求一起发送到网站服务器，服务器上的Web应用程序就可以读取Cookie中包含的信息。
- Cookie 以文本存储于在客户端保存。客户端是可以禁用Cookie的。
- Cookie在每次请求或发送时都会被加载，影响传输。Cookie易被攻破，所有不适合存储安全信息。

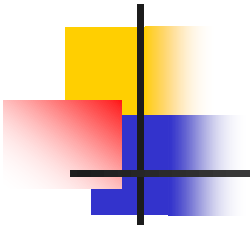
2. Cookie对象的主要属性

属性	说明
Domain	获取或设置将此Cookie与其关联的域
Expires	获取或设置Cookie的过期日期和时间
Name	获取或设置Cookie的名称
Path	获取或设置要与当前Cookie一起传输的虚拟路径
Secure	指定是否通过SSL（即仅通过HTTPS）传输Cookie
Value	获取或设置Cookie的Value
Values	获取在Cookie对象中包含的键值对的集合

3. Cookie对象的主要方法

方法	说明
Add	增加Cookie
Clear	清除Cookie集合内的变量
Get	通过键名或索引得到Cookie的值
Remove	通过Cookie的键名或索引删除Cookie对象

注：由于Cookie存储在客户端，所以不能在服务器端编程直接修改Cookie。如果确实需要修改的话，就创建一个同名的Cookie，然后发送到客户端覆盖原Cookie。



4. 访问Cookie对象

利用Response对象设置Cookie

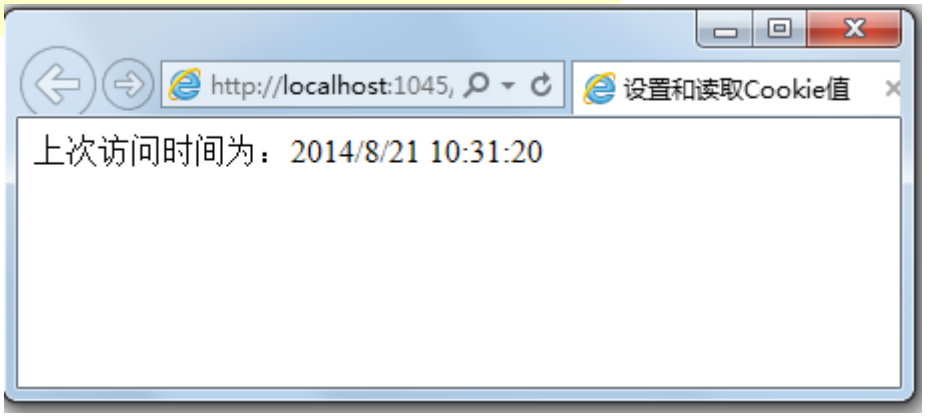
```
Response.Cookies["UserId"].Value=“Admin”;
```

利用Request对象的Cookies集合读取Cookie

```
mycook =
Request.Cookies["user"].Value
```

例6-9 设置和读取Cookie值

```
protected void Page_Load(object sender, EventArgs e) {
    DateTime now = DateTime.Now;
    //创建了一个Cookie对象，名为LastVistTime
    HttpCookie MyCookie = new HttpCookie("LastVistTime");
    //设置Cookie值为当前时间
    MyCookie.Value = now.ToString();
    //设置Cookie超时时间为2个小时
    MyCookie.Expires = now.AddHours(2);
    //将新Cookie添加到Response对象的Cookies集合中
    Response.Cookies.Add(MyCookie);
    //从Request对象的Cookie集合中读取名为LastVistTime的Cookie值
    string myVistTime = Request.Cookies["LastVistTime"].Value;
    Response.Write("上次访问时间为：" + myVistTime);
}
```



6.2.4 ViewState对象

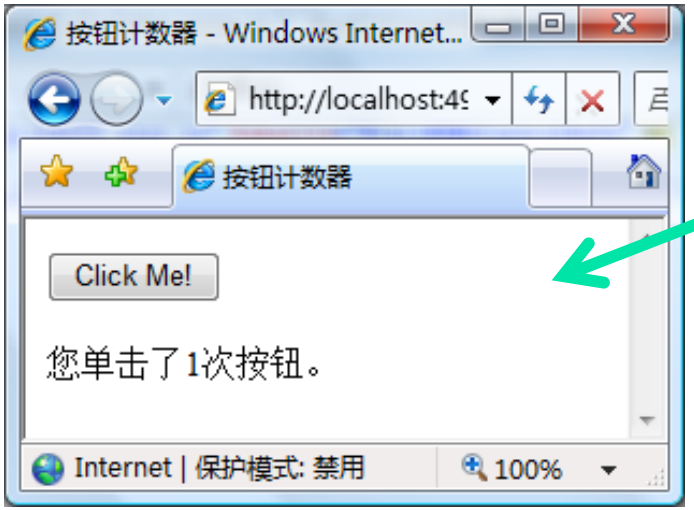
- 在针对同一页面的多次请求间，可以使用ViewState对象保存服务器控件的状态信息。
- 与Session对象相比较，Session对象保存在服务器内存，大量使用Session会使得服务器负担加重。而ViewState对象将数据存入到页面隐藏控件里，不占用服务器资源。
- Session的默认超时时间是20分钟，而ViewState则永远不会超时。

ViewState对象保存数据采用“Key-Value”对的形式。格式如下：

ViewState["键名"] = 数据值;

```
ViewState[“Var”]=Count;          //给ViewState对象添加值
string Test=ViewState[“Val”];    //获取ViewState对象中的值
```

■ 引例：按钮计数器（ClickCounter1.aspx）



不管单击多少次按钮，
始终只显示1次

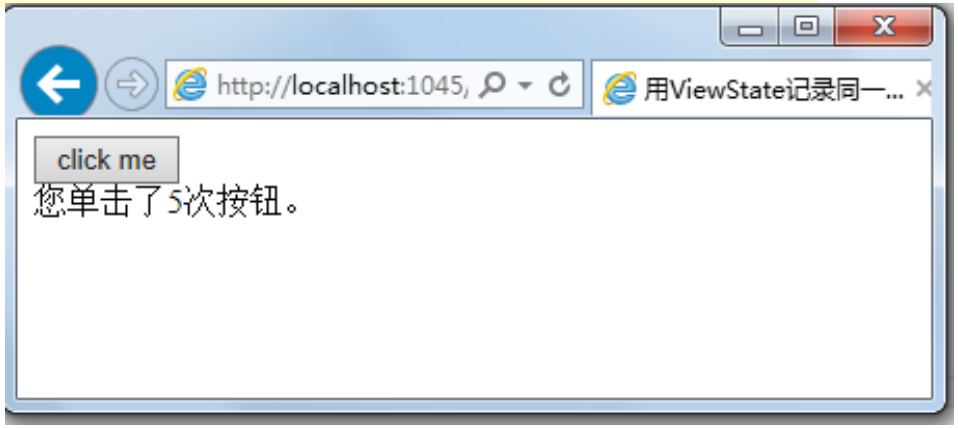
```
protected void Page_Load(object sender, EventArgs e)  {
}
private int counter=0;  //计数器
protected void btnClickMe_Click(object sender, EventArgs e)
{
    counter++;  //累加计数器
    lblInfo.Text = "您单击了" + counter.ToString() + "次按钮。";
 }
```

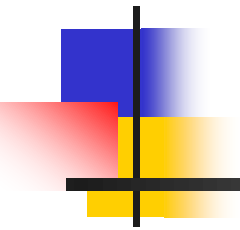
HTTP无状态特性

- 在向Web服务器提交的多次HTTP请求之间，Web服务器不会保留前一次访问的状态信息，不管同一个客户端连续多少次访问，Web服务器都将其看成是一个“陌生人”的“拜访”。
- 所谓“状态”，其实就是与每次HTTP请求相关联的一些信息，例如用户在某个页面上输入的一些数据。
- 在实际开发中，往往需要在多次HTTP请求之间保存状态。因此，任何一个Web开发技术都必须面对这样一个问题并给出一个解决方案。

例6-10 用ViewState记录同一个页面按钮被单击的次数

```
protected void btnClick_Click(object sender, EventArgs e)  {
    int counter;   //计数器
    if (ViewState["Counter"] == null) {
        //如果是第一次单击按钮
        counter = 1;
    }
    else {
        //从ViewState对象中取出上次保存的counter变量值，并累加1
        counter = (int)ViewState["Counter"] + 1;
    }
    ViewState["Counter"] = counter;   //将counter变量值保存到ViewState对象中
    lblInfo.Text = "您单击了" + counter.ToString() + "次按钮。";
}
```





6.3 实例：一个简单的购物车

6.3.1 设计说明

本实例包括两个页面。第一个页面为图书浏览页面Default.aspx，显示出客户正在浏览的图书信息（书名、ISBN、单价、作者、出版社、出版时间等）；第二个页面为查看购物车页面ShopCart.aspx，显示当前购物车的信息（图书名称，单价，数量，小计等）。

需要说明的是，原则上全部图书的信息应该存储在数据库中，由于本章还未讲到数据库，所以暂时将它们存在一个ArrayList对象中。

6.3.1 设计说明

具体的设计思路：

（1）创建一个图书类Book，其属性包括：图书ID、书名、单价、作者、出版社、出版时间等。

（2）为了保存全部的图书信息，建立一个ArrayList来存储全部图书的实例对象列表。

（3）为了保存购物车的信息，建立一个HashTable，分别用key和value来保存购物车中的图书ID和购买数量。

（4）利用Session对象保存ArrayList的实例与HashTable的实例。

（5）图书浏览页面：用户可以查看所有图书的详细信息，当用户选中一本图书，并点击“加入购物车”按钮时，则选中图书的ID通过Session传递至购物车ShopCart.aspx页面中。

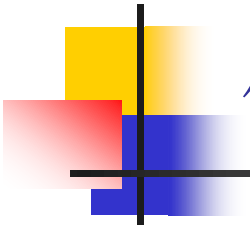
（6）购物车页面：根据获取的图书ID，在ArrayList图书集合中查询对应的图书，最终显示加入购物车中的图书信息。

6.3.2 程序实现

图书浏览界面

6.3.2 程序实现

查看购物车界面



作业4

1、Application对象和Session对象有什么区别？

2、如何用Cookie对象保存用户的登录信息？

3、从一个HTML表单中输入学生的学号和姓名，利用Request对象让其在另一网页中显示，简述其实现过程。