知识点

1. 状态管理

必要性：浏览器向服务器请求某个页面，服务器处理完返回该页面对象，之后在服务器端会清除该页面对象的数据，如果用户需要保存页面中的数据，就需要进行状态管理。

1. 视图状态

作用：在单张页面的多次回传请求期间保存数据的方式。

语法：

ViewState 属性是 Control 中定义的用来存储状态数据的属性

保存数据：ViewState[“key”]=value

读取数据：object obj=ViewState[“key”]

原理：

（1） 当页面处理完毕后，视图状态中所有的信息会进行序列化操作，形成 Base64 编码的单个字符串保存在页面中 id 为__VIEWSTATE 的隐藏域控件中，随着页面一起发回到客户端

（2） 当用户操作导致页面回传时此隐藏域中的信息会随着表单一起发回到服务器端，系统将其内容还原为保存前的信息。

保存位置：客户端的隐藏域控件

保存数据的类型：object 类型

注意：

（1） 如果通过视图状态保存的数据是对象类型，要将对象标记为可序列化 Serializable

（2） 通过视图状态保存大量数据时，会占用网络带宽，影响下载速度

2. 应用程序状态管理

作用：在 web 应用程序运行期间保存数据的方式

实现方法：1.使用 Application 实现

语法：

Application 属性是 Page 类中定义的属性，返回类型是 HttpApplicationState 类型的对象，这种类型的对象主要用来进行全局信息保存.

保存数据：Application[string name]=value;

读取数据：object obj = Application[name];

保存位置：服务器端内存中

保存数据类型：object 类型

特点：

（1） 跨用户：任意用户写入的数据，其他的任意用户都能读取到

（2） 跨页面：在任意页面写入的数据，其他的任意页面都能读取到

生命周期：保存数据，直到服务端重启或关闭（整个应用程序运行期间）

实现方法：2.通过 Global 类的静态属性实现

语法：

（1）在项目中添加一个Global.asax文件，有一个全局类Global,继承自HttpApplication
（2）在Global类中定义静态属性，如果需要对属性初始化，则在Application_Start()事件处理中执行
（3）写入数据：Global.静态属性=值
（4）读取数据：Global.静态属性

保存位置：服务器端内存中

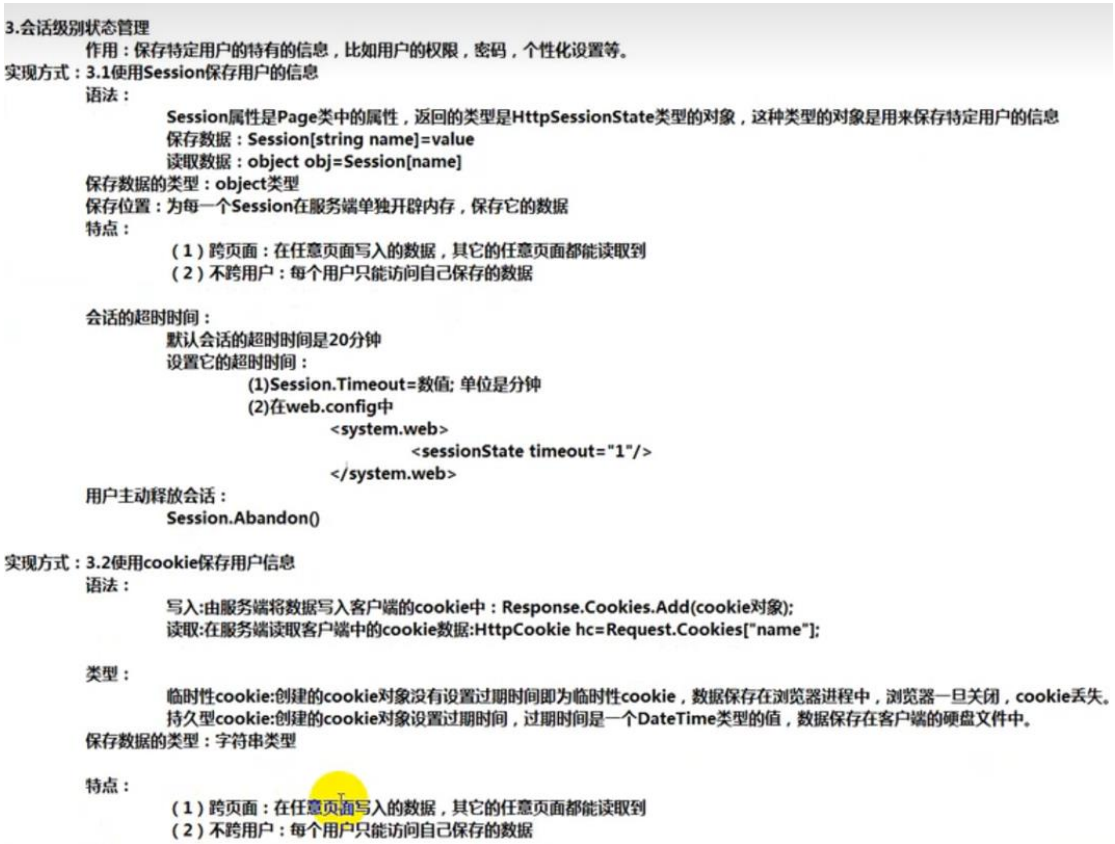
特点：

（1） 跨用户：任意用户写入的数据，其他的任意用户都能读取到

（2） 跨页面：在任意页面写入的数据，其他的任意页面都能读取到

生命周期：保存数据，直到服务端重启或关闭（整个应用程序运行期间）

3. 会话状态管理



4. 隐藏域：

隐藏域（Hidden Field）是一种 HTML 表单元素，它允许在表单中存储一些数据，但这些数据不会被显示在页面上。它通常被用于在表单提交时传递一些额外的参数，这些参数可以是用户无法修改的，也可以是一些需要保存在页面上但又不希望用户看到的信息。

在 HTML 页面中，隐藏域的实现是通过在表单中添加一个 input 元素，并设置其 type 属性为"hidden"。例如：

```
<input type="hidden" name="param1" value="value1">
```

在 ASP.NET 中，隐藏域可以通过 HtmlInputHidden 控件或者在 aspx 页面中使用<input type="hidden">元素来创建和使用。在页面中，可以通过代码设置隐藏域的值，也可以通过 JavaScript 动态修改隐藏域的值。当表单提交时，隐藏域中的数据会被一并提交到后台服务器，开发者可以在后台代码中通过 Request 对象来获取这些值。

需要注意的是，由于隐藏域中的数据可以通过查看页面源代码来查看，因此不能将敏感信息存储在隐藏域中。

5. **解析过程：**

   IIS 服务器发现浏览器请求的是 aspx 界面，IIS 服务器无法自己处理，而是将其交给.net 应用程序拓展，.net 先解析 aspx 界面然后解析 aspx.cs 文件生成 App_Web_xxxx.dll,该文件执行先生成.cs 文件的输出流，再生成 aspx 文件的输出流，交给 IIS 服务器放在 Http 响应头后面发给浏览器

代码：

1. 在线人数统计

```
void Application_Start(object sender, EventArgs e)
{
    // 在应用程序启动时运行的代码
    Application["ApplicationName"] = "畅想网络学院";
    Application["PageSize"] = "18";
    Application["OnlinePersonTotal"] = 0;
    Application["OnlinePersons"] = "";

}

void Session_Start(object sender, EventArgs e)
{
    // 在新会话启动时运行的代码
    Session["REMOTE_ADDR"]= equest.ServerVariables["REMOTE_ADDR"].ToString();
    Application["OnlinePersonTotal"] =(int)Application["OnlinePersonTotal"]+ 1;
}
void Session_End(object sender, EventArgs e)
{

   Application["OnlinePersonTotal"] = (int)Application["OnlinePersonTotal"] - 1;
    // 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。
```

}

2. 防止绕过登录

```
protected void Page_Load(object sender, EventArgs e)
        {
            if (Session["S_uname"] == null)//防止直接进入 myhome 窗体
            {
                Response.Redirect("login.aspx");
            }
            Label2.Text = "当前用户为：" + Session["S_uname"].ToString() + "密码为：" +
Session["S_upwd"].ToString();
            Label3.Text = "当前时间为：" + System.DateTime.Now.ToString();

            ListBMyClass.Items.Clear();
            ListBMyClass.Items.Add("我所需选修的课程");//在下拉列表中新增一项
        }
```

3. 请列举你所知道的常见的跨页面变量传递方法，至少 3 种。

方法一：利用 QueryString。1.在源页面的代码中用需要传递的名称和值构造 URL 地址。在源页面的代码用 Response.Redirect(URL);重定向到上面的 URL 地址中。在目的页面的代码使用 Request. QueryString ["name"]; 取出 URL 地址中传递的值。

例如：string s_url;

```
s_url = "b.aspx?name=" + Label1.Text;
Response.Redirect(s_url);
//另一个页面
Label2.Text = Request.QueryString["name"];
```

方法二：利用 session，在源页面的代码中创建你需要传递的名称和值构造 Session 变量，在目的页面的代码使用 Session 变量取出传递的值。

例如： Session["name"] = Label.Text;

```
//另一个页面
string name;     name = Session["name"].ToString();
```

方法三：利用 cookie，在源页面的代码中创建你需要传递的名称和值构造 Cookie 对象，在目的页面的代码使用 Cookie 对象取出传递的值。

例如：HttpCookie objCookie = new HttpCookie("myCookie","Hello,Cookie!");

```
Response.Cookies.Add(objCookie);
//另一个页面
string myName1Value;
myName1Value = Request.Cookies[ "myCookie" ].Value;
```

方法四：利用 application，.在源页面的代码中创建你需要传递的名称和值构造 Application 变量，在目的页面的代码使用 Application 变量取出传递的值。

例如： Application["name"] = Label1.Text;

```
//另一个页面
string name;
Application.Lock();
name = Application["name"].ToString();
Application.UnLock();
```

4. 表单提交后，刷新页面与在浏览器地址栏回车访问该页面，分别代表什么意思？

“刷新”是在你现有页面的基础上，检查网页是否有更新的内容。在检查时，会保留之前的一些变量的值，因此有可能会造成刷新后网页出现错误，或者打不开的情况；

“转到”和在地址栏回车，则相当于你重新输入网页的 URL 访问，这种情况下，浏览器会尽量使用已经存在于本机中的缓存。

“刷新” 是取网页的新内容来更新本机缓存，在更新的同时保留之前的一些变量；“转到”则是一种全新的访问，它会尽量使用本机缓存中的文件，但不会保留之前的变量，另外，按着 Ctrl，还可以进行强制刷新，跟转到的作用差不多。

5.重定向：

重定向是一种通过 HTTP 响应状态码来告诉客户端浏览器将请求重定向到其他 URL 的机制。在进行重定向时，服务器发送一个 HTTP 响应，其中包括一个状态码（例如 301 或 302），一个响应头（包括重定向 URL），以及一个可选的响应体。客户端浏览器在接收到这个响应后，会根据响应头中的 Location 字段重新发送一个新的请求，以获取重定向后的页面内容。

重定向操作是在服务器端完成的，如果在重定向之前已经向客户端浏览器发送了响应数据，例如 HTML 文档、Cookie 等，那么这些数据可能已经被客户端浏览器开始解析和显示了。此时再发送重定向响应将无法生效，因为客户端浏览器已经在解析先前发送的数据了。

因此，为了确保重定向操作能够成功，应该在重定向之前避免向客户端浏览器发送任何响应数据。这通常可以通过在服务器端进行适当的逻辑处理来实现。