# 1. 页面管理

## 1.1. 单一文件模式

`ASP.NET`的页面包含两个部分

- 一部分是可视化元素
- 另一部分是页面的程序逻辑

### 1.1.1. 单一文件模式

此种模式下页面的标签和代码在同一个文件下，程序代码在`script`代码块中实现

- 在运行时，单一页面被视为继承`Page`类

### 1.1.2. 后台代码模式

后台代码将可视化元素和程序代码分别放置在不同的文件中

- 界面代码在`.aspx`文件
- 程序代码在`.aspx.cs`文件中

## 1.2. 页面的往返与处理机制

`ASP.NET`网页是作为代码在服务器上运行，每次页面提交后，都会在服务器端运行代码

- `ASP.NET`的处理机制如下
- (1) 用户通过客户端浏览器向服务器请求页面，页面第一次运行，执行初步处理，进行程序员定义的初始化
- (2) 执行的结果以标记的形式返回呈现给浏览器，浏览器对标记进行解释并显示
- (3) 用户进行输入或选择，如果是按钮就进一步处理当前页面，超链接就连接到另一个页面，当前的页面不会被进一步处理
- (4) 当前的页面发送到`Web`服务器，即`回发`，即当前页面发送其自身
- (5) 在服务器上，再次运行当前页面，使用用户输入或者选择的信息
- (6) 执行后将页面以`HTML`标记的形式发送给客户端浏览器

## 1.3. 页面的生命周期

### 1.3.1. 生命周期阶段

**Page Request → Start → Page Initialization → Load → Validation → Postback event handling → Rendering → Unload**

- (1) 页请求 (Page Request )
    - 确定是否需要分析或编译页面，或是否可以在不运行页的情况下发送页面的缓存版本以响应

- **(2) 启动 (Start)**
  - 设置页属性，判断请求是回发请求还是新请求，设置`IsPostBack`、`UICulture`属性
- **(3) 初始化 (Page Initialization)**
  - 使用页中的控件，设置每个控件的`UniqueID`，应用主题或母版页，如果是回发则回发的数据尚未加载
- **(4) 加载 (Load)**
  - 如果是回发加载信息
- **(5) 验证 (Validation)**
- **(6) 回发事件处理 (Postback event handling)**
  - 如果是回发，那么调用所有验证程序控件的验证方法
- **(7) 呈现\渲染 (Rendering)**
  - 对该页和所有控件保存视图状态，针对每个控件使用`Render`方法
- **(8) 卸载 (Unload)**
  - 引发`UnLoad`事件，卸载页属性

### 1.3.2. 页面生命周期事件

- **(1) Start**
  - `Request`、`Response`、`IsPostBack`、`UICulture`被创建
  - 如果是回传(Postback)的请求，所有控件的值还没从`view state`中还原，可能被重写或覆盖
- **(2) Initialize**
  - `theme`被应用，所有控件被设置了唯一的`ID`，可以调用`Init`、`InitComplete`、`PreLoad`方法
  - `Init`：这个时间发生在所有控件被初始化且皮肤设置被应用后，来读取控件的初始化值
  - `InitComplete`：处理所有需要初始化完成后才能做的事情
  - `PreLoad`：页面或空间在进入`Load`前有什么处理的都在这个阶段处理，加载`viewstate`并处理所有`Request`中的`postback`数据
- **(3) Load**
  - 在这个阶段所有控件都被`viewstate`中信息填充并加载
  - `OnLoad`事件被触发，为页面上所有的服务端控件设置属性，得到`query string`，建立数据库连接
- **(4) Validation**
  - 若控件要求验证，验证会在这个阶段发生，检查控件的`IsValid`属性判断是否有验证控件
- **(5) Event Handling**
  - 所有服务器端控件的时间处理发生在此阶段
  - **LoadComplete**
  - **PreRender**
  - **SaveStateComplete**
- **(6) Render**
  - 调用每一个控件的`Render`方法从而顺序地输出控件的`HTML`代码。
- **(7) Unload**
  - 被各个控件一一触发，最后被页面触发。
  - 清理工作，如关闭数据库连接和打开的文件或者登记时间记录等其他任务

# 2. 常用内置对象

# 2.1. Response 对象

- **Request 类似于输入，Response类似于输出**

## 2.1.1. 属性

- `Expires`：**获取和设置浏览器缓存超时时间**
- `ContentType`：**获取和设置输出流的类型，默认为`text/html`**

## 2.1.2. 方法

- `Write`：**输出信息**

# 2.2. Request 对象

## 2.1.1. 属性

- `Form`：**获取`Post`提交的数据**
- `QueryString`：**获取`Get`方法提交的数据**

## 2.1.2. 方法

- `SaveAs`：**保存`HTTP`请求到硬盘**
- `Get`：**通过`Request.QueryString`获取传递的数据，数据量限制在2kB左右**
- `Post`：**通过`Request.Form`获取传递的数据，数据大小无限制，地址栏不会显示相关的查询字符串，适合用来传递保密信息。**

# 2.3. Application 对象

- **即全局变量，利用Application 保存要进行传递的变量，**

## 2.3.1. 属性

- `AllKeys`： **获得集合的所有键**
- `Contents`：**获得对象的引用**
- `Count`：**获得集合的数量**
- `Item`：**通过名称和索引访问集合**

## 2.3.2. 方法

- `Add`**方法**
- **向`Stat`集合中加入一个名为string1 的值为test 的字符串**

```
Application.Add("string1","test")
Application("string1")="test"
Application.Item("string1")="test"
```

### 2.3.3. 统计网站访问人数

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class example3_3 : System.Web.UI.Page{
  // 页面加载事件，当网页加载完成后的发生的事件
    protected void Page_Load(object sender, EventArgs e){
        Application.Lock(); // 加锁 所有网站共用，只能独占性打开。
        Application["usercount"] = (Convert.ToInt32(Application["usercount"]) +
1).ToString();
        // 转换为数字后执行+1，然后转换为string
        Application.UnLock(); // 解锁 所有网站共用
        labContent.Text = Application["usercount"].ToString();
        // 输出当前用户数字符串
    }

}
```

## 2.4. Session 对象

- 即局部变量，只能自己使用
- `Session`的生命周期有限，默认是20分钟，可以通过`Timeout`属性设置

### 2.4.1. 传递用户信息

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class example3_4_1 : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e){
        if (TextBox1.Text!="") {
          Session["username"] = TextBox1.Text;
          Response.Redirect("example3-4-2.aspx");
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class example3_4_2 : System.Web.UI.Page{
    protected void Page_Load(object sender, EventArgs e){
        if (Session["username"] != null){
            Label1.Text = Session["username"].ToString();
        }
        else{
            //请输入用户名
            Response.Redirect("example3-4-1.aspx");
        }
    }
}
```

## 2.5. ViewState

- 类似数组

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class example3_5 : System.Web.UI.Page{
    string VSString ="basketball";
    protected void Page_Load(object sender, EventArgs e){
        if (!Page.IsPostBack){ // 未回传时
            ViewState.Add("favorite", VSString);
        }
    }
    protected void Button1_Click(object sender, EventArgs e){
        if (TextBox1.Text!= ""){
            VSString = this.TextBox1.Text;
            ViewState["favorite"] = VSString;
        }
    }
    protected void Button2_Click(object sender, EventArgs e){
        if (ViewState["favorite"] != null){
            this.Label1.Text = ViewState["favorite"].ToString();
        }
        else{
            this.Label1.Text = "查看的ViewState值不存在";
```

```
        }
    }
}
```

## 2.6. Cookie 对象

- 存储少量浏览者的信息，如用户名、Email等，当其再次登录网站时，不必再填写这些信息
- 多数浏览器支持最大4096字节的Cookie

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public partial class example3_6 : System.Web.UI.Page{
    protected void Button1_Click(object sender, EventArgs e){
        Response.Cookies["userName"].Value = "BUAA University";
        Response.Cookies["userName"].Expires = DateTime.Now.AddDays(1);
        this.Label1.Text = "写入完毕！";
    }
    protected void Button2_Click(object sender, EventArgs e){
        if (Request.Cookies["userName"] != null){
            Label1.Text = Server.HtmlEncode(Request.Cookies["userName"].Value);
        }
    }
}
```

# 3. 课后习题

- 常用的代码模式有几种

**ASP.NET 的页面包含两个部分:一部分是可视化元素，包括标签、服务器控 件以及一些静态文本等;另一部分是页面的程序逻辑，包括事件处理句柄和其他程序代码。 ASP.NET 提供两种模式来组织页面元素和代码:一种是单一文件模式，另一种是后台代码 模式。两种模式功能是一样的，可以在两种模式中使用同样的控件和代码，但使用的方式不同。**

**(1) 单一文件模式**

**在单一文件模式下，页面的标签和代码在同一个.aspx 文件中，程序代码包含在`<script runat="server"></script>`的服务器程序脚本代码块中间，并且代码中间可以实现对一些方法和属性以及其他代码的定义，只要在类文件中可以使用的都可以在此处进行定义。运行时， 单一页面被视为继承 Page 类。**

**(2) 后台代码模式**

**后台代码页面模式将可视化元素和程序代码分别放置在不同的文件中，如果使用 C#， 则可视化页面元素为.aspx 文件，程序代码为.cs 文件，根据使用语言的不同，代码后缀也不同，这种模式也被称为代码隐藏模型。**

- `Web.config`和`Global.asax`