# ASP.NET

Using HttpRequest to get user request at server side and send the feedbacks using HttpResponse.

Use WebRequest to request a asp.net webpage and wait for its response using WebResponse.

Then display the web page using HttpWebResponse.ResponseUri.

ASP.NET support URLs like http://forums.asp.net/p/1464618/3375616.aspx?id=12345, where id=12345 is the query string.

There are many ways in passing parameters between pages, as in http://msdn.microsoft.com/en-us/library/6c3yckfw.aspx. Yet the query string may be the easiest one.

* Using a Query String
* Getting Post Information from the Source Page
* Using Session State
* Getting Public Property Values from the Source Page
* Getting Control Information from the Source Page in the Same Application

# ASP.NET 要点

1. 与.NET Framework集成

2. 编译执行而非解释执行

3. 支持多语言

4. 运行在CLR内

5. 面向对象

6. 跨设备和浏览器

       Web服务器空降会根据客户端的情况来自适应的生成HTML。不需要编写额外代码

7. 易于部署和管理

## 设计网页

<div>容器没有内置的外观，它其实是一个浮动的框，下面这个示例中，高度随内容而变

<div style="POSITION:absolute; left: 100px; top: 50px; width: 200px">

VS提供样式构造器来通过css样式格式化静态HTML。如果样式不共享，可以嵌入样式。

两种类型的网站:

**无项目文件网站**：单个页面单独编译, code file, 可选择代码内联或者隐藏。部署时直接拷贝， 运行时编译（也可预编译），webconfig替代项目文件，管理reference等

**项目文件网站**：所有页面编译到一个dll, code behind，代码隐藏，预编译，项目文件

# Web窗体

1. Web 应用程序在服务器上执行, 通过回送

2. Web 应用程序是无状态的， ASP.NET提供工具进行弥补(如视图状态)

HTML中数据发送回服务器端的最简单办法是使用<form>，里面可以放置<input>来表示UI元素. 客户端通过类似querystring的方式传送信息。 而ASP.NET将客户传递的内容转换成相应的控件对象，通过更直观的语法来访问。这种模式基本上取消了对HTML的耦合。

ASP.NET是事件驱动的。服务器端重新构建客户端的状态，调用事件处理，并将转换后的页面按照HTML的方式返回到客户端。

ASP.NET**自动回送**。当Web控件的AutoPostBack为真时，ASP.NET自动在HMTL页面上为该控件添加一个\_doPostBack的JSP方法.

**视图状态:**

ASP.NET在客户端和服务器端之间传递状态的改变（基于初始状态）。每次发送到客户端是，该状态改变序列被传递到客户端(Base64)。发送到服务器时，服务器端利用这些信息将页面恢复到上次发送到客户端的状态，并将利用客户新的改变来更新最新的客户端状态。

HTTPModule和HTTPHandler

ASP.Net处理Http Request时，使用Pipeline（管道）方式，由各个HttpModule对请求进行处理，然后到达 HttpHandler，HttpHandler处理完之后，仍经过Pipeline中各个HttpModule的处理，最后将HTML发送到客户端浏览 器中。

生命周期中涉及到几个非常重要的对 象：HttpHandler,HttpModule,IHttpHandlerFactory，他们的执行(顺序)大致的执行过程是这样的：client 端发送页面请求，被IIS的某个进程截获，它根据申请的页 面后缀(.aspx)不同，调用不同的页面处理程序(.asp->asp.dll; .aspx->ISAPI.dll).而页面处理程序在处理过程中，则要经历HttpModule,HttpHandler的处理：前者HttpModule用于页面处理前和处理后的一些事件的处理，后者HttpHandler进行真正的页面的处理。

如前所说，HttpModule会在页面处理前和后对页面进行处理，所以它不会影响真正的页面请求。通常用在给每个页面的头部或者尾部添加一些信息（如版 权声明）等.曾经见过一些免费的空间，我们的页面上传上去后，浏览的时候发现，在每个页面的头部和尾部多了很多小广告....,如果理解了 HttpModule的原理，要做这个就不是很难了~

IHttpModule与IHttpHandler的区别整理

1.先后次序.先IHttpModule,后IHttpHandler. 注:Module要看你响应了哪个事件，一些事件是在Handler之前运行的，一些是在Handler之后运行的

2.对请求的处理上:

   IHttpModule是属于大小通吃类型,无论客户端请求的是什么文件,都会调用到它;例如aspx,rar,html的请求.

   IHttpHandler则属于挑食类型,只有ASP.net注册过的文件类型(例如aspx,asmx等等)才会轮到调用它.

3.IHttpHandler按照你的请求 生成响应的内容，IHttpModule对请求进行预处理，如验证、修改、过滤等等，同时也可以对响应进行处理

 1.使用QueryString变量

QueryString是一种非常简单的传值方式，他可以将传送的值显示在浏览器的地址栏中。如果是传递一个或多个安全性要求不高或是结构简单的数值时，可以使用这个方法。但是对于传递数组或对象的话，就不能用这个方法了。下面是一个例子：

PassGetParamForm.aspx的C#代码

private void btnSubmit\_Click(object sender, System.EventArgs e)

{

string strUrl = "ReceiveParamForm.aspx?UserName=" +txtName.Text.Trim();

Response.Redirect(strUrl);

}

ReceiveParamForm.aspx中C#代码

private void Page\_Load(object sender, EventArgs e)

{

  lblRecValue.Text = Request.QueryString["UserName"];

}

2.使用Application 对象变量

Application对象的作用范围是整个全局，也就是说对所有用户都有效。其常用的方法用Lock和UnLock。

ApplicationParamForm.aspx中的C#代码

private void btnSubmit\_Click(object sender, System.EventArgs e)

{

Application.Lock();

Application["UserName"] = txtName.Text.Trim();

Application.UnLock();

Server.Transfer("ReceiveParamForm.aspx");

}

ReceiveParamForm.aspx中C#代码

private void Page\_Load(object sender, EventArgs e)

{

Application.Lock();

this.lblRecValue.Text = Application["UserName"].ToString();

Application.UnLock();

}

3.  使用Session变量

这个是使用中最常见的用法了，其操作与Application类似，作用于用户个人，所以，过量的存储会导致服务器内存资源的耗尽。

SessionParamForm.aspx中的C#代码

private void btnSubmit\_Click(object sender, System.EventArgs e)

{

//创建Session变量

Session["UserName"] =txtName.Text.Trim();

Response.Redirect("ReceiveParamForm.aspx");

}

ReceiveParamForm.aspx中C#代码

private void Page\_Load(object sender, EventArgs e)

{

  //接收Session对象变量传递的值

lblRecValue.Text =Session["UserName"].ToString();

}

4. 使用Cookie对象变量

这个也是常使用的方法，与Session一样，其是什对每一个用户而言的，但是有个本质的区别，即Cookie是存放在客户端的，而session是存放在服务器端的。而且Cookie的使用要配合ASP.NET内置对象Request来使用。

CookieParamForm.aspx的C#代码

private void btnSubmit\_Click(object sender, System.EventArgs e)

{

HttpCookie cookie\_UserName = new HttpCookie("MyUserName");

cookie\_UserName.Value = this.txtName.Text.Trim();

Response.AppendCookie(cookie\_UserName);

Response.Redirect("ReceiveParamForm.aspx"); }

ReceiveParamForm.aspx中C#代码

private void Page\_Load(object sender, EventArgs e)

{

    //接收Cookie对象变量传递的值

lblRecValue.Text= Request.Cookies["MyUserName"].Value.ToString();

}

5.使用Server.Transfer方法

这个才可以说是面象对象开发所使用的方法，其使用Server.Transfer方法把流程从当前页面引导到另一个页面中，新的页面使用前一个页面的应答流，所以这个方法是完全面象对象的，简洁有效。

ServerParamForm.aspx的C#代码

private void btnSubmit\_Click(object sender, System.EventArgs e)

{

Server.Transfer("ReceiveParamForm.aspx");

}

public string UserName

{

get

{

return txtName.Text.Trim();

}

}

ReceiveParamForm.aspx中C#代码

private void Page\_Load(object sender, EventArgs e)

{

  //接收Server.Transfer方法传递的值

ServerParamForm newWeb; //实例ServerParamForm窗体

newWeb = (ServerParamForm)Context.Handler;

lblRecValue.Text = newWeb.UserName;

}

6.PostBackURL

按钮的属性。直接跳转。将当前页面发送到目标页面。

http://www.glish.com/css

http://www.csszenggarden.com