# 一般处理程序(HttpHandler)

- **一般处理程序(HttpHandler)：**是一个实现

  System.Web.IHttpHandler 接口的特殊类。

  任何一个实现了 IHttpHandler 接口的类 ,是作为一个外部请求的目标程序的前提。( 凡是没有实现此接口的类，就不能被浏览器请求。)

- 它由支持 ASP.NET 的服务器调用和启动运行。一个 HttpHandler 程序负责处理它所对应的一个或一组 URL 地址的访问请求 , 并接收客户端发出的访问请求信息 ( 请求报文 ) 和产生响应内容 ( 响应报文 )。

- 咱可以通过创建一个我们自己的 HttpHandler 程序来生成浏览器代码发送回客户端浏览器。

- HttpHandler 程序可以完成普通类程序所能完成的大多数任务：

  1.获取客户端通过 HTML 的 Form 表单提交的数据和 URL 参数

  2.创建对客户端的响应消息内容

  3.访问服务器端的文件系统

  4.连接数据库并开发基于数据库的应用

  5.调用其他类

## 新建网站

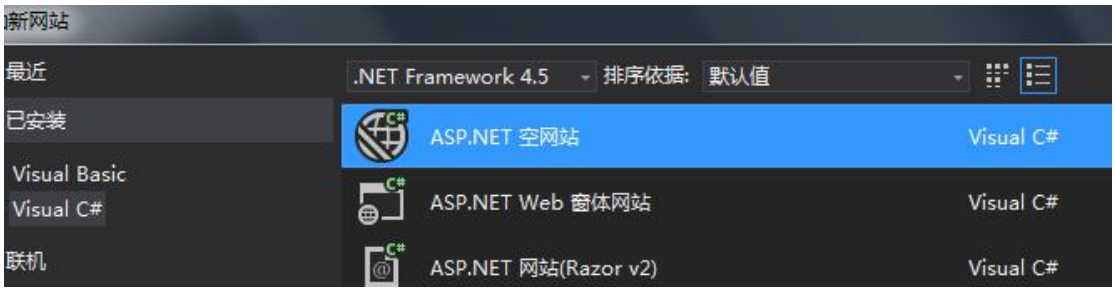
第一步：


文件(F)
编辑(E)
视图(V)
调试(D)
团队(M)
工具(T)
测试(S)
体系结构(C)
分析(N)
窗口(W)
新建(N)
打开(O)
关闭(C)
关闭解决方案(T)
项目(P)...
Ctrl+Shift+N
网站(W)...
Shift+Alt+N
团队项目(T)...
文件(F)...
Ctrl+N

第二步：


管理解决方案的 NuGet 程序包
启用 NuGet 程序包还原(G)
新建 解决方案资源管理器 视图
计算代码度量值(C)
添加(D)
将解决方案添加到源代码管理(
粘贴(P)
重命名(M)
在文件资源管理器中打开文件夹
属性(R)
新建项目(N)...
现有项目(E)...
新建网站(W)...
现有网站(B)...
新建项(W)...
Ctrl+Shift+A
现有项(G)...
Shift+Alt+A
新建解决方案文件夹(D)

第三步：


新网站
最近
已安装
Visual Basic
Visual C#
联机
.NET Framework 4.5
排序依据:
默认值
ASP.NET 空网站
Visual C#
ASP.NET Web 窗体网站
Visual C#
ASP.NET 网站(Razor v2)
Visual C#

第四部：


## 使用一般处理程序创建一个表格

```
using System;
using System.Web;
using System.Text;
public class First : IHttpHandler {

    public void ProcessRequest (HttpContext context) {
        context.Response.ContentType = "text/html";
        StringBuilder sb = new StringBuilder();
        sb.Append("<html><head><title>用户列表</title></head>");
        sb.Append("<body>");
        sb.Append("<table><tr><td>用户名</td><td>张三</td></tr><tr><td>密码
</td><td>11111</td></tr></table></body></html>");
        context.Response.Write(sb.ToString());
    }

    public bool IsReusable {
        get {
            return false;
        }
    }

}
```

以上的方法通过拼接字符串创建表格，但是如果 HTML 内容比较复杂，那么创建起来就非

常困难

## 读取模板创建一般处理程序

```
<table>
        <tr><td>用户名</td><td>$userName</td></tr>
        <tr><td>密码</td><td>$pwd</td></tr>
    </table>
```

```
using System;
using System.Web;
using System.IO;
public class UserInfoList : IHttpHandler {

    public void ProcessRequest (HttpContext context) {
      context.Response.ContentType = "text/html";
      //获取该文件的物理路径.(在 Web 开发中对文件或文件夹进行操作时，必须先获取物理路径)
      string filePath = context.Request.MapPath("UserList.html");
      //读取文件内容
      string strHtml=File.ReadAllText(filePath);
      strHtml = strHtml.Replace("$userName", "张三").Replace("$pwd","123");
      context.Response.Write(strHtml);
    }

    public bool IsReusable {
        get {
            return false;
        }
    }

}
```

# Get 请求与 Post 请求

表单是以 Post 方式提交过来的，接收时必须用 Request.Form 来接收

表单是以 get 请求那么，接收的时候必须使用 QueryString 来接收

```
<form method="get" action="Accept.ashx">
```

```
        用户名:<input type="text" name="txtName" /><br />
        密码:<input type="password" name="txtPwd" /><br />
        <input type="submit" value="提交" />
        <span name="span1">sss</span>
    </form>
```

```
public void ProcessRequest (HttpContext context) {
        context.Response.ContentType = "text/plain";
      // string userName= context.Request.Form["txtName"];
      // string userPwd = context.Request.Form["txtPwd"];


        string userName = context.Request.QueryString["txtName"];
        string userPwd = context.Request.QueryString["txtPwd"];
      context.Response.Write("用户名是:"+userName+"密码是:"+userPwd);
  }
```

表单元素必须有 name 属性，而 Form 指定的键的名称就是 name 属性的值.

1.如果是 post 方式进行请求，那么表单中的数据会放在请求报文体中，发送到服务端，如果以 get 方式进行请求，那么表单中的数据会放在 URL 地址栏中发送到服务端。（注意,表单元素必须有 name 属性）

2.在服务端接收方式 不一样，如果是 post 请求用 request.Form,get 请求用 request.QueryString.

3.Post 请求比 get 请求安全.以后像注册，登录等表单都要用 post 提交.

4.Post 请求发送的数据要比 get 请求大。

注意:只能将表单元素的 value 值提交到服务端（span,div 等不熟悉表单元素，所以无法提交），并且表单元素必须要加 name 属性.

# 自增运算

```
<form method="post" action="SelfAdd.ashx">
```

```
        <input type="text" name="txtCount" value="$value" />
        <input type="submit" value="自增" />
    </form>
```

```
using System;
using System.Web;
using System.IO;
public class SelfAdd : IHttpHandler {
    //protected static int count = 0;
    public void ProcessRequest (HttpContext context) {
        context.Response.ContentType = "text/html";
        string filePath = context.Request.MapPath("SelfAdd.html");
        string fileContent = File.ReadAllText(filePath);
        int count =Convert.ToInt32(context.Request.Form["txtCount"]);
        count++;
       string strHtml= fileContent.Replace("$value",count.ToString());
       context.Response.Write(strHtml);
    }

    public bool IsReusable {
        get {
            return false;
        }
    }

}
```

注意：并不是所有的 html 元素的内容都能提交到服务端的，只能将表单元素的内容提交到服务端。

```

```

```

```