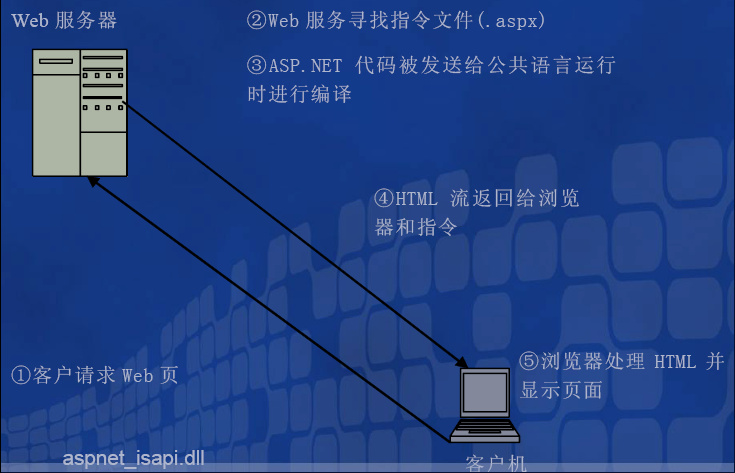
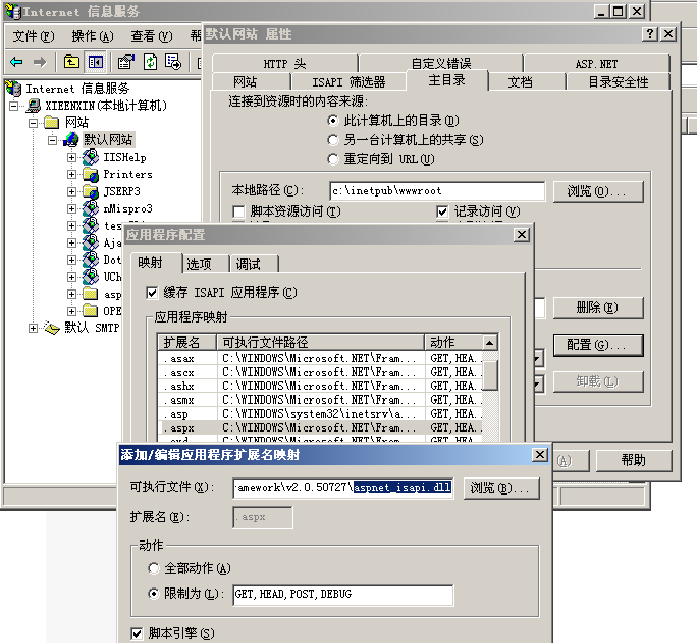
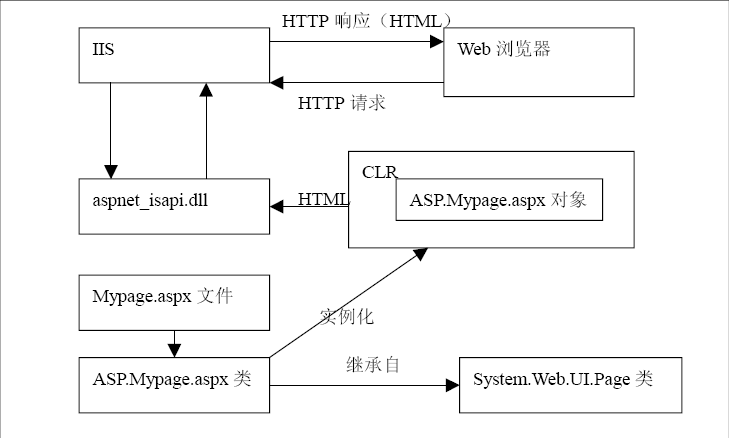
[**ASP.NET架构分析**](http://www.cnblogs.com/xieex/archive/2007/10/29/941300.html)

Posted on 2007-10-29 13:12 [xieex](http://www.cnblogs.com/xieex/) 阅读(5084) 评论(17) [编辑](https://i.cnblogs.com/EditPosts.aspx?postid=941300) [收藏](http://www.cnblogs.com/xieex/archive/2007/10/29/941300.html)

最近听了微软讲师邵志东的讲座“深入解析ASP.NET架构”的视频讲座，我觉得对ASP.NET架构有一定的认识，现在对讲座做一点总结，以及发表我对ASP.NET架构相关知识的自身理解，如有不妥之处，希望各位同仁不吝指出！  
**一、ASP.NET工作原理：**首先通过一个图来了解一下ASP.NET工作原理,  
     
                                     图一 ASP.NET工作原理  
通过该图的顺序我们可以看到ASP.NET是如何工作的，可以看到客户端和服务器端是如何交互的。看到客户端是如何请求服务器，服务器接受到客户端请求后，是如何处理请求的，并将处理结果返回给客户端的。  
需要指出的是图中的aspnet\_isapi.dll是用来处理.aspx文件的，其实IIS服务器是只能识别.html文件的，当浏览器对服务器进行http请求时，IIS服务器识别到.aspx文件时，IIS服务器将其交给aspnet\_isapi.dll来处理，aspnet\_isapi.dll将.aspx文件发送给.NET FrameWork中CLR(公共语言运行时)进行编译，编译后将HTML流返回给浏览器。  
这在IIS服务器上是可以来配置的，配置中默认.aspx文件是由aspnet\_isapi.dll来处理的，其实也可以去添加某种特定类型的文件由特定的可执行文件来处理，如：可以添加一个.apx文件来由aspnet\_isapi.dll来处理。  
具体配置如下图：  
  
                                图二 IIS配置

趁此说一下IIS服务的一些知识：  
IIS本事是不支持动态页面的，也就是说它仅仅支持静态html页面的内容，对于如.aspx,.php,.cs等，IIS并不会处理这些标记，它就会把它当作文本，丝毫不做处理发送到客户端。为了解决这个问题。IIS有一种机制，叫做ISAPI的筛选器，它是一个标准组件（COM组件。）  ASP.NET服务在注册到IIS的时候，会把每个扩展可以处理的文件扩展名注册到IIS里面（如：\*.aspx,\*.php等），扩展启动后，就根据定义好的方式来处理IIS所不能处理的文件，然后把控制权跳转到专门处理代码的进程中（如\*.aspx由aspnet\_isapi.dll来处理）。让这个进程开始处理代码，生成标准的HTML代码，生成后把这些代码加入到原有的HTML代码中，最后把完整的HTML返回到IIS，IIS再把内容发送给客户端。

通过下图可以看到更具体的ASP.NET页面的执行过程：

  
                                 图三 ASP.NET页面的执行过程

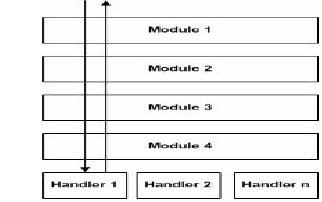
**二、ASP.NET请求的处理过程：**

上面我们已经了解了ASP.NET的工作原理，知道了客户端是如何和服务器端进行交互的。现在我们可以来看看服务器端到底是怎样来处理客户端的请求。  
Web服务器提供了很多处理请求的功能，但是为了满足开发者的需求，开发者需要扩充或扩展Web服务器的功能，就是向Web服务器插入某些组件来增强Web服务器的功能，微软公司提供了ISAPI(Internet服务器API)。就比如说aspnet\_isapi.dll可以来处理.aspx等文件，我们也可以开发组件（如xxx.dll）来处理某种类型的文件或者去做一些其他增强服务器能力的事情。

简单的说如何去增强Web服务器的能力呢，我们就可以去开发一些组件，开发组件需要一种技术叫ISAPI。  
（ISAPI是一种重要的技术，它允许我们增强与ISAPI兼容的Web服务器的能力，其中IIS就是一种与ISAPI兼容的Web服务器。）  
  
增强Web服务器的能力的组件的类型主要有两种：  
1、ISAPI过滤器：客户端每次向服务器发出请求的时候，请求要经过过滤器。客户端不需要在请求中指定过滤器，只需要简单地把请求发送给Web服务器，接着Web服务器把请求传递给相关的过滤器。接下来过滤器可能修改请求，执行某些登录操作等。也就是说，客户端每次请求服务器时，必须要经过“过滤”，如需要检查用户名和密码，符合条件时才让通过请求，“检查用户名和密码”的过程就是ISAPI过滤器所要做的操作。过滤器可以有很多（web.config中指定，后面将讲到），并且请求都必须要经过所有的过滤器进行“过滤”。

2、ISAPI扩展：ISAPI扩展是使用Win32动态链接库（xxx.dll）来实现的，也可以把ISAPI扩展看做一个普通的应用程序。它的处理目标是http请求。也就是说Web服务器可以来处理http请求，但是你可以去扩展服务器，去自定义对http的请求，达到更好的效果。

基于上面的内容，我们可以将ASP.NET请求的处理过程总结如下：

ASP.NET请求处理过程是基于“管道模型”的，客户端向服务器发送http请求时，在模型中ASP.NET把http请求传递给管道中的所有模块（ISAPI过滤器），每个模块都接受http请求并且有完全控制权限，模块可以用任何自认为的方式（通过开发人员来开发，如校验用户名密码等）来处理请求。一旦请求经过了所有Http模块（如用户名、密码符合），就最终被Http处理程序（ISAPI扩展）处理，http处理程序对请求进行一些处理，并且结果将再次经过管道中的http模块。  
  
                图四  管道模型

我自己举例来说吧，我建立了一个.apx文件，要求客户端访问该页面，并在客户端返回"我是apx文件，我是被“ISAPI过滤器”过滤过的，并被"ISAPI扩展"处理过的"。在访问页面时必须需要用户名和密码。  
此时我就需要开发组件来扩充Web服务器，因为Web服务器没有校验指定的用户名和密码的能力也没有处理.apx文件的能力。我就需要开发“ISAPI过滤器”组件（用来校验用户名和密码）和ISAPI扩展组件（处理.apx文件）。

那我们怎么开发“ISAPI过滤器”组件和“ISAPI扩展”组件呢?

**1、开发“ISAPI过滤器”**   HttpModule(Http模块)实现了ISAPI 过滤器的功能，是通过对IhttpModule接口的继承来处理。HttpModule是实现了System.Web.IhttpModule接口的.NET组件，这些组件通过在某些事件中注册自身，把自己插入到ASP.NET请求处理管道中(如上图中的Module1,Module2)。当这些事件发生的时候，ASP.NET调用对请求有兴趣的HTTP模块，这样该模块就能处理请求了。

   HttpModule的实现：  
   1、编写一个类，实现IhttpModule接口，要添加引用System.Web  
   2、实现Init方法，并且要注册需要的方法。如：AuthenticateRequest等  
   3、实现注册的方法  
   4、实现Dispose方法，如果需要手工为类做一些清除工作，可以添加Dispose方法的实现，但这不是必需的，通常可以  
     不为Dispose方法添加任何代码  
   5、在Web.config文件中，注册您编写的类。  
     
   下面通过一个实例来实现HttpModule(具体见附件代码)

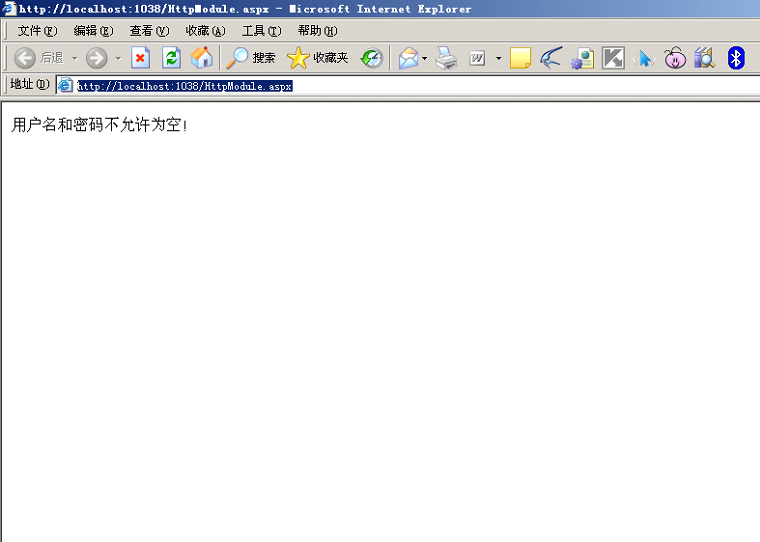
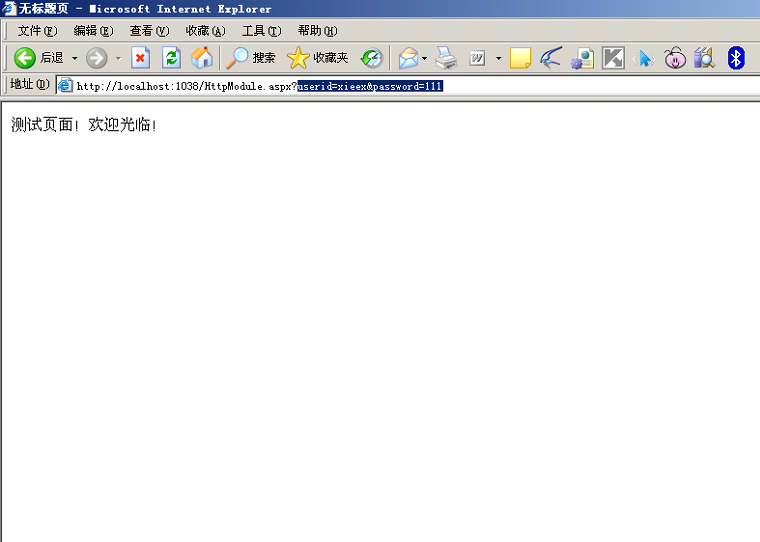
1、我编写了SecurityModules类实现IHttpModule接口  
2、实现了Init方法，并且向Application对象注册事件处理程序（myAuthenticateRequest），当http请求执行到AuthenticateRequest（建立用户标   识时）时，就执行注册的方法myAuthenticateRequest  
3、实现注册的方法myAuthenticateRequest，主要来校验userid和password  
4、实现Dispose方法

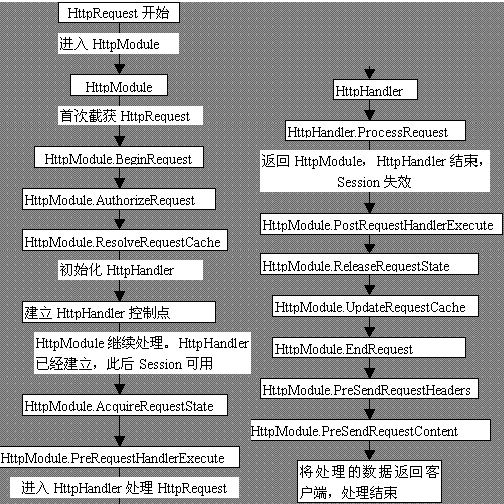
 1http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System;  
 2http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Collections.Generic;  
 3http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Text;  
 4http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Web;  
 5http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Security.Principal;  
 6http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
 7http://www.cnblogs.com/Images/OutliningIndicators/None.gifnamespace MyModule1  
 8http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
 9http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class SecurityModules : IHttpModule  
10http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
11http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Init(HttpApplication objApplication)  
12http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
13http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            // 向Application 对象的事件AuthenticateRequest注册处理程序  
14http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            objApplication.AuthenticateRequest += new EventHandler(this.myAuthenticateRequest);  
15http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
16http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
17http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        private void myAuthenticateRequest(object objSender, EventArgs objEventArgs)  
18http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
19http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            // 鉴别用户的凭证，并找出用户角色  
20http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            HttpApplication objApp = (HttpApplication)objSender;  
21http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            HttpContext objContext = (HttpContext)objApp.Context;  
22http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            if ((objContext.Request["userid"] == null) && (objContext.Request["password"] == null))  
23http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            {  
24http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                objContext.Response.Write("用户名和密码不允许为空！");  
25http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                objContext.Response.End();  
26http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
27http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
28http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            string sUserId = "";  
29http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            sUserId = objContext.Request["userid"].ToString();  
30http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            string sPassword = "";  
31http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            sPassword = objContext.Request["password"].ToString();  
32http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            string[] strRoles;  
33http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            strRoles = AuthenticateAndGetRoles(sUserId,sPassword);  
34http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            if ((strRoles == null) || (strRoles.GetLength(0) == 0))  
35http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            {  
36http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                objContext.Response.Write("用户名和密码错误！");  
37http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                objApp.CompleteRequest();//终止一个http请求  
38http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
39http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            //GenericIdentity类表示具有指定名称和身份验证类型的用户  
40http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            GenericIdentity objIdentity = new GenericIdentity(sUserId, "CustomAuthentication");  
41http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            objContext.User = new GenericPrincipal(objIdentity,strRoles);  
42http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
43http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
44http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        /// <summary>  
45http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// 根据http请求的userid和password来验证和获取角色。  
46http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        /// </summary>  
47http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        private string[] AuthenticateAndGetRoles(string sUserId, string sPassword)  
48http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
49http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            string[] strRoles = null;  
50http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            if ((sUserId.Equals("xieex")) && (sPassword.Equals("111")))  
51http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            {  
52http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                strRoles = new String[1];  
53http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                strRoles[0] = "Administrator";  
54http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
55http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            else if ((sUserId.Equals("zhangsan")) && (sPassword.Equals("222")))  
56http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            {  
57http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                strRoles = new String[1];  
58http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                strRoles[0] = "User";  
59http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
60http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            return strRoles;  
61http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
62http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
63http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void Dispose()  
64http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
65http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif   
66http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
67http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
68http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

5、新建一个项目，在Web.config文件中注册该类，当访问该项目的页面时，都要校验userid和password才能访问,并且在该项目的引用中添加MyModule1.dll

在Web.config文件中添加：

http://www.cnblogs.com/Images/OutliningIndicators/None.gif<!--HttpModule通过对HttpApplication对象的一系列事件的处理来对HTTP处理管道施加影响，这些事件在HttpModule的Init方法中进行注册包括BeginRequest等。-->  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif      <httpModules>  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif        <add name="myModule1" type="MyModule1.SecurityModules,MyModule1"/>  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif      </httpModules>  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  <!--add将HttpModule类添加到应用程序-->  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  <!--add格式:<add name="modulename(随便起)" type="命名空间。类名（该类继承IHttpModule接口）, （assemblyname）dll文件名"/>  -->  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  <!--remove从应用程序移除HttpModule类-->  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  <!--remove格式<remove name="modulename">-->  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  <!--clear从应用程序移除所有HttpModule映射-->  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  <!--clear/-->

此时访问该项目中的页面时，就要校验用户名和密码了。  
  
此时输入在URL上增加“?userid=xieex@password=111”时就可以进入页面了。  
  
这样我们就开发完了ISAPI过滤器，注意过滤器可以有多个，只要在web.config中的httpModules节点中Add即可，它们都会起作用的不会覆盖的。  
  
HttpModule通过对HttpAplication对象的一系列事件的处理来对HTTP处理管道施加影响，这些事件在HttpModule的Init方法中进行注册。具体的事件发生顺序如下：

  
                         图五  事件发生顺序  
  
关于事件发生顺序可以参考例子：MultiHttpModule.csproj,这个例子也反应了过滤器可以有多个，只要在web.config中的httpModules节点中Add即可，它们都会起作用的不会覆盖的。

**2、开发“ISAPI扩展”组件**

还是拿下面的例子来说：  
“我自己举例来说吧，我建立了一个.apx文件，要求客户端访问该页面，并在客户端返回"我是apx文件，我是被“ISAPI过滤器”过滤过的，并被"ISAPI扩展"处理过的"。在访问页面时必须需要用户名和密码。  
此时我就需要开发组件来扩充Web服务器，因为Web服务器没有校验指定的用户名和密码的能力也没有处理.apx文件的能力。我就需要开发“ISAPI过滤器”组件（用来校验用户名和密码）和ISAPI扩展组件（处理.apx文件）。”  
上面的ISAPI过滤器可以解决“校验用户名和密码”问题，下面我们利用ISAPI扩展来处理.apx文件。

HttpHandler实现了ISAPI扩展的功能，它处理请求（Request）的信息和发送响应（Response）。HttpHandler功能的实现要继承IHttpHandler接口。  
HTTP处理程序是实现了System.Web.IHttpHandler接口的.NET组件，任何实现了IHttpHandler接口的类都可以用于处理输入的HTTP请求。

HttpHandler的实现：  
1、编写一个实现IHttpHandler接口的类；  
2、在Web.config或machine.config文件中注册这个处理程序；  
3、在Internet服务管理器把文件扩展（你想要的文件扩展名）映射到ASP.NET ISAPI扩展DLL(aspnet\_isapi.dll)上。(该步也可不用做)

同样通过一个实例来实现HttpHandler(具体见附件代码)  
1、编写了HandlerAPX类，该类继承了IHttpHandler接口，该类实现了ProcessRequest方法和IsReusable属性。注意要引用System.Web

 1http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System;  
 2http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Collections.Generic;  
 3http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Text;  
 4http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Web;  
 5http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
 6http://www.cnblogs.com/Images/OutliningIndicators/None.gifnamespace MyHandler  
 7http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
 8http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class HandlerAPX : IHttpHandler  
 9http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
10http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        #region Implementation of IHttpHandler  
11http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        /// <summary>  
12http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// http处理程序的核心。我们调用这个方法来处理http请求。  
13http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// </summary>  
14http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        /// <param name="context"></param>  
15http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void ProcessRequest(HttpContext context)  
16http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
17http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            HttpResponse objResponse = context.Response;  
18http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
19http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            objResponse.Write("<html><body><h1>你好！我是apx文件！");  
20http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            objResponse.Write("</body></html>");  
21http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
22http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
23http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        /// <summary>  
24http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// 我们调用这个属性来决定http处理程序的实例是否可以用于处理相同其它类型的请求。  
25http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// HTTP处理程序可以返回true或false来表明它们是否可以重复使用。  
26http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        /// </summary>  
27http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public bool IsReusable  
28http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
29http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            get  
30http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            {  
31http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                return true;  
32http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
33http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
34http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        #endregion  
35http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
36http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

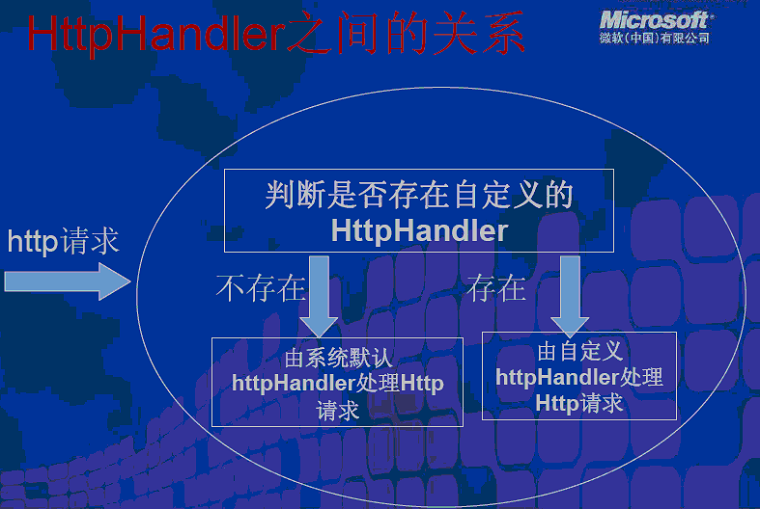
2、新建一个项目，在Web.config文件中注册该类,并且在该项目的引用中添加MyHandler.dll

在Web.config文件中添加：

http://www.cnblogs.com/Images/OutliningIndicators/None.gif      <httpHandlers>  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif        <add verb="\*" path="\*.apx"   
http://www.cnblogs.com/Images/OutliningIndicators/None.gif               type="MyHandler.HandlerAPX,MyHandler" />  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif      </httpHandlers>  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif      <!--add格式:<add verb="\*" path="要处理文件"  type="命名空间.类名（该类继承IHttpHandler接口）, （assemblyname）dll文件名"/>          -->

说明当IIS识别到.apx文件时，就调用HandlerAPX类来处理。

http请求通过httpHandler来处理，本来系统就对其有默认处理，那和自定义的处理是如何协调的呢？看下图就可以明白，其实两者是选择其一的：

  
                                     图六  HttpHandler之间的关系

在“图五 事件发生顺序”中，我们看到HttpHandler建立后，此后Session就可以用了，下面来看看在HttpHandler中如何访问Session.  
1、不能直接通过HttpContext访问  
2、必须实现IRequiresSessionState接口  
3、IRequriresSessionState接口指定目标HTTP处理程序接口具有对会话状态值的读写访问权限，这是一个标记接口，没  
   有任何方法。

同样通过一个实例来实现在HttpHandler访问Session(具体见附件代码)  
1、编写类HandlerSession,该类实现了接口IHttpHandler,IRequiresSessionState

http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System;  
http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Collections.Generic;  
http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Text;  
http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Web;  
http://www.cnblogs.com/Images/OutliningIndicators/None.gifusing System.Web.SessionState;  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gifnamespace MyHandlerSession  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gif{  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif    public class HandlerSession : IHttpHandler,IRequiresSessionState  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif    {  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        #region Implementation of IHttpHandler  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        /// <summary>  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// http处理程序的核心。我们调用这个方法来处理http请求。  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// </summary>  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        /// <param name="context"></param>  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public void ProcessRequest(HttpContext context)  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            HttpResponse objResponse = context.Response;  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            HttpRequest objRequest = context.Request;  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            HttpSessionState objSession = context.Session;  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif             
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            objResponse.Write("<html><body><h1>欢迎使用自定义的HttpHandler!<br>");  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            objSession["test"] = "Session测试<br>";  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            objResponse.Write("Session的值为:" + objSession["Test"].ToString());  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        /// <summary>  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// 我们调用这个属性来决定http处理程序的实例是否可以用于处理相同其它类型的请求。  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        /// HTTP处理程序可以返回true或false来表明它们是否可以重复使用。  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        /// </summary>  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif        public bool IsReusable  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif        {  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            get  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockStart.gif            {  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif                return true;  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif            }  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        }  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif        #endregion  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedSubBlockEnd.gif    }  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockEnd.gif}

2、新建一个项目，在Web.config文件中注册该类,并且在该项目的引用中添加MyHandlerSession.dll

在Web.config文件中添加：

http://www.cnblogs.com/Images/OutliningIndicators/None.gif      <httpHandlers>  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif        <add verb="\*" path="\*.apx"  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif               type="MyHandler.HandlerAPX,MyHandler" />  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif        <add verb="\*" path="HttpModule.aspx"  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif       type="MyHandlerSession.HandlerSession,MyHandlerSession" />  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif      </httpHandlers>

说明当IIS识别到HttpModule.aspx文件时，就调用HandlerSession类来处理。但是如果是.apx文件时，就会调用HandlerAPX类来处理。

也就是说注册多个HttpHandler会覆盖的，这与HttpModule是不一样的。如果是对同一类文件注册了不同的HttpHandler，会执行最后一个。

即：如果Web.config配置为

http://www.cnblogs.com/Images/OutliningIndicators/None.gif      <httpHandlers>  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif        <add verb="\*" path="\*.apx"  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif               type="MyHandler.HandlerAPX,MyHandler" />  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif        <add verb="\*" path="\*.apx"  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif       type="MyHandlerSession.HandlerSession,MyHandlerSession" />  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif      </httpHandlers>

说明当IIS识别到\*.apx文件时，就调用HandlerSession类来处理，而不是调用HandlerAPX类来处理。

**三、ASP.NET事件模型机制**1、ASP.NET之所以对于以前的ASP是一个革命性的巨变，在很大程度上是由于ASP.NET技术是一种基于事件驱动的全新技术。  
2、在ASP.NET中时间的触发和处理是在客户端和服务端进行的。  
3、ASP.NET中，如果频繁和服务器进行事件信息传递，会大大降低服务器的处理效率和性能，因而有些事件如OnMouseOver没有提供；  
4、但提供了Change事件，为了提高效率它们被缓存在客户端，等到再一次事件信息被发送到服务器端时一同发送回去。  
如文本框的change事件，下拉框的change事件，  
如两个控件的change事件中：

http://www.cnblogs.com/Images/OutliningIndicators/None.gif        protected void DropDownList1\_SelectedIndexChanged(object sender, EventArgs e)  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gif        {  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Response.Write("DropDownList控件选择改变!<br>");  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockEnd.gif        }  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif  
http://www.cnblogs.com/Images/OutliningIndicators/None.gif        protected void TextBox1\_TextChanged(object sender, EventArgs e)  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gif        {  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif            Response.Write("TextBox文本改变!<br>");  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockEnd.gif        }

如果控件本身的AutoPostBack设置为false（默认是false）时，文本框和下拉框发生改变时，不会执行change事件的，而是将事件信息缓存在客户端，当在页面上点击一个服务器端控件Button,

http://www.cnblogs.com/Images/OutliningIndicators/None.gif        protected void Button1\_Click(object sender, System.EventArgs e)  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockStart.gif        {  
http://www.cnblogs.com/Images/OutliningIndicators/InBlock.gif           Response.Write("点击了Button按钮!<br>");  
http://www.cnblogs.com/Images/OutliningIndicators/ExpandedBlockEnd.gif        }

此时将客户端中的事件信息发送到服务器端，执行所有的事件，返回到客户端的信息为：  
TextBox文本改变!  
DropDownList控件选择改变!  
点击了Button按钮!

具体例子代码见附件。  
[HttpModuelandHttpHandler代码](http://files.cnblogs.com/xieex/HttpModuelandHttpHandler.rar)