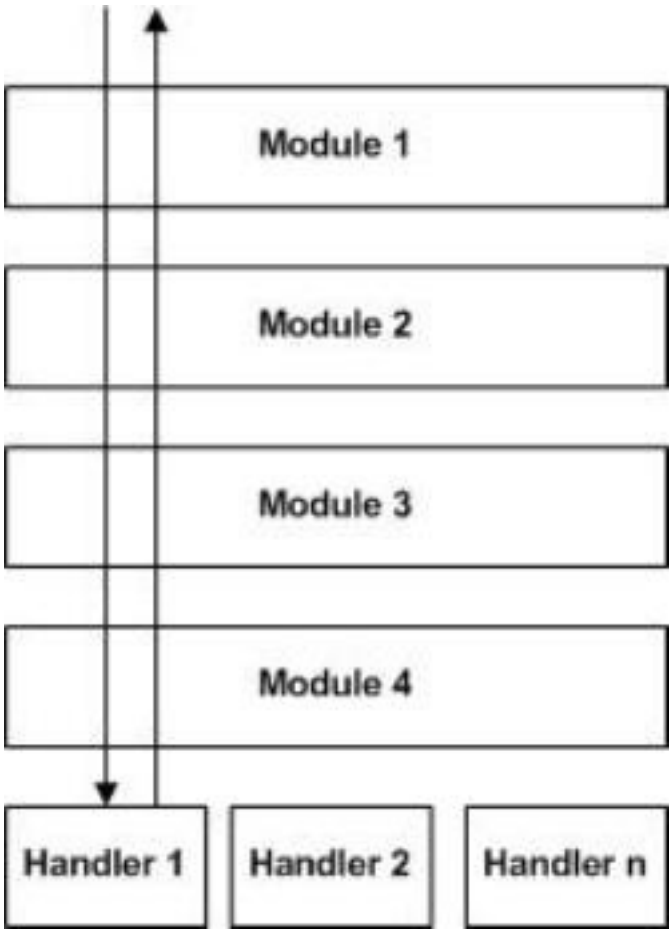
عنوان: **کمی درباره httpmodule**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۰۹/۲۶ ۲۰:۱۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IHttpModule, IhttpHandler

قبل از اینکه به httpmoduleها بپردازیم، اجازه بدید کمی در در مورد httphandler اطلاعات کسب کنیم. httphandler ویژگی است که از asp.net به بعد ایجاد شد و در asp کلاسیک خبری از آن نیست.
یک httphandler کامپوننتی است که از طریق اینترفیس System.Web.IHttpHandler پیاده سازی میشود و به پردازش درخواست‌های رسیده از httprequest رسیدگی می‌کند.
فرض کنید کاربری درخواست صفحه default.aspx را کرده است و سرور هم پاسخ آن را می‌دهد. در واقع پردازش اینکه چه پاسخی باید به کاربر یا کلاینت ارسال شود بر عهده این کامپوننت می‌باشد. برای وب سرویس هم موضوع به همین صورت است؛ هر نوع درخواست HTTP از این طریق انجام می‌شود.
حال به سراغ httpmodule می‌رویم. httpmoduleها اسمبلی یا ماژول‌هایی هستن که بر سر راه هر درخواست کاربر از سرور قرار گرفته و قبل از اینکه درخواست شما به httphandler برسد، اول از فیلتر این‌ها رد می‌شود. در واقع موقعی که شما درخواست صفحه default.aspx را می‌کنید، درخواست شما به موتور asp.net ارسال می‌شود و از میان فیلترهایی رد می‌شود تا به دست httphandler برای پردازش خروجی برسد. برای همین اگر گاهی به جای گفتن asp.net engine عبارت asp.net pipeline هم میگویند همین هست؛ چون درخواست شما از بین بخش‌های زیادی می‌گذرد تا به httphandler برسد که httpmodule یکی از آن بخش هاست. با هر درخواستی که سرور ارسال می‌شود، httpmoduleها صدا زده می‌شوند و به برنامه نویس امکان بررسی اطلاعات درخواستی و پردازش درخواست‌ها را در ورودی و خروجی، می‌دهد و شما میتوانید هر عملی را که نیاز دارید انجام دهید. تعدادی از این ماژول‌های آماده، همان stateها و Authentication می‌باشند.
تصویر زیر نحوه‌ی ارسال و بازگشت یک درخواست را به سمت httphandler نشان می‌دهد



برنامه نویس هم میتواند با استفاده از اینترفیس‌های IHttpModule و IHttpHandler در درخواست‌ها دخالت نماید.
برای شروع یک کلاس ایجاد کنید که اینترفیس IHttpModule را پیاده سازی می‌کند. شما دو متد را باید در این کلاس بنویسید؛ یکی Init و دیگر Dispose. همانطور که مطلع هستید، اولی در ابتدای ایجاد شیء و دیگر موقع از دست رفتن شی صدا زده می‌شود.
متد Init یک آرگومان از نوع httpapplication دارد که مانند رسم نامگذاری متغیرها، بیشتر به اسم context یا app نام گذاری می‌شوند:

```
public void Init(HttpApplication app)
{
   app.BeginRequest += new EventHandler(OnBeginRequest);
}

public void Dispose(){ }
```

همانطور که می‌بینید این شیء یک رویداد دارد که ما این رویداد را به تابعی به نام OnBeginRequest متصل کردیم. سایر رویدادهای موجود در httpapplication به شرح زیر می‌باشند:

BeginRequest	این رویداد اولین رویدادی است که اجرا می‌شود، هر نوع عملی که میخواهید در ابتدای ارسال درخواست انجام دهید، باید در این قسمت قرار بگیرد؛ مثلا قرار دادن یک بنر بالای صفحه
AuthenticateRequest	خود دانت از یک سیستم امنیتی توکار بهره مند است و اگر می‌خواهید در مورد آن خصوصی سازی انجام بدهید، این رویداد می‌تواند کمکتان کند
AuthorizeRequest	بعد از رویداد بالا، این رویداد برای شناسایی انجام می‌شود. مثلا دسترسی ها؛ دسترسی به قسمت هایی خاصی از منابع به او داده شود و قسمت هایی بعضی از منابع از او گرفته شود.
ResolveRequestCache	این رویداد برای کش کردن اطلاعات استفاده می‌شود. خود دانت تمامی این رویدادها را به صورت تو کار فراهم آورده است؛ ولی اگر باز خصوصی سازی خاصی مد نظر شماست می‌توانید در این قسمت‌ها، تغییراتی را اعمال کنید. مثلا ایجاد file caching به جای memory cache و ...
AcquireRequestState	این قسمت برای مدیریت state می‌باشد مثلا مدیریت session ها
PreRequestHandlerExecute	این رویداد قبل از httphandler اجرا می‌شود.
PostRequestHandlerExecute	این رویداد بعد از httphandler اجرا می‌شود.
ReleaseRequestState	این رویداد برای این صدا زده می‌شود که به شما بگوید عملیات درخواست پایان یافته است و باید stateهای ایجاد شده را release یا رها کنید.
UpdateRequestCache	برای خصوصی سازی output cache بکار می‌رود.
EndRequest	عملیات درخواست پایان یافته است. در صورتیکه قصد نوشتن دیباگری در طی تمامی عملیات دارید، میتواند به شما کمک کند.
PreSendRequestHeaders	این رویداد قبل از ارسال طلاعات هدر هست. اگر قصد اضافه کردن اطلاعاتی به هدر دارید، این رویداد را به کار ببرید.
PreSendRequestContent	این رویداد موقعی صدا زده می‌شود که متد response.flush فراخوانی شود.، اگر می‌خواهید به محتوا چیزی اضافه کنید، از اینجا کمک بگیرید.

BeginRequest	این رویداد اولین رویدادی است که اجرا می‌شود، هر نوع عملی که میخواهید در ابتدای ارسال درخواست انجام دهید، باید در این قسمت قرار بگیرد؛ مثلا قرار دادن یک بنر بالای صفحه
Error	این رویداد موقعی رخ می‌دهد که یک استثنای مدیریت نشده رخ بدهد. برای نوشتن سیستم خطایابی خصوصی از این قسمت عمل کنید.
Disposed	این رویداد موقعی صدا زده میشود که درخواست، بنا به هر دلیلی پایان یافته است. برای عملیات پاکسازی و .. می‌شود از آن استفاده کرد. مثلا یک جور rollback برای کارهای انجام گرفته.

کد زیر را در نظر بگیرید:

کد زیر یک رویداد را تعریف کرده و سپس خود httpapplication را به عنوان sender استفاده می‌کند.

در اینجا قصد داریم یکی از صفحات را در خروجی تغییر دهیم. آدرس تایپ شده همان باشد ولی صفحه‌ی درخواست شده، صفحه‌ی دیگری است. این کار موقعی بیشتر کاربردی است که آدرس یک صفحه تغییر کرده و کاربر آدرس قبلی را وارد می‌کند. حالا یا از طریق بوک مارک یا از طریق یک لینک، در یک جای دیگر و شما میخواهید او را به صفحه‌ای جدید انتقال دهید، ولی در نوار آدرس، همان آدرس قبلی باقی بماند. همچنین کار دیگری که قرار است انجام بگیرد محاسبه مدت زمان رسیدگی به درخواست را محاسبه کند ، برای همین در رویداد BeginRequest زمان شروع درخواست را ذخیره و در رویداد EndRequest با به دست آوردن اختلاف زمان فعلی با زمان شروع به مدت زمان مربوطه پی خواهیم برد.

با استفاده از app.Context.Request.RawUrl آدرس اصلی و درخواست شده را یافته و در صورتی که شامل نام صفحه مربوطه بود، با نام صفحه‌ی جدید جابجا می‌کنیم تا اطلاعات به صفحه‌ی جدید پاس شوند ولی در نوار آدرس، هنوز آدرس قبلی یا درخواست شده، قابل مشاهده است.

در خط app.Context.Items["start"] که یک کلاس ارث بری شده از اینترفیس IDictionary است، بر اساس کلید، داده شما را ذخیره و در مواقع لزوم در هر رویداد به شما باز می‌گرداند.

```
public class UrlPath : IHttpModule
    {
        public void Init(HttpApplication app)
        {
            app.BeginRequest+=new EventHandler(_BeginRequest);
            app.EndRequest+=new EventHandler(_EndRequest);
        }

        public void Dispose()
        {

        }

         void _BeginRequest(object sender, EventArgs e)
         {

             HttpApplication app = (HttpApplication) sender;
             app.Context.Items["start"] = DateTime.Now;

             if (app.Context.Request.RawUrl.ToLower().Contains("tours_list.aspx"))
             {

app.Context.RewritePath(app.Context.Request.RawUrl.ToLower().Replace("tours_list.aspx","tours_cat.aspx"
));
             }

         }
         void _EndRequest(object sender, EventArgs e)
         {
             HttpApplication app = (HttpApplication)sender;
             string log = (DateTime.Now -
DateTime.Parse(app.Context.Items["start"].ToString())).ToString();
             Debugger.Log(0,"duration","request took " + log+Environment.NewLine);

         }
```

```
    }
```

حالا باید کلاس نوشته شده را به عنوان یک httpmodule به سیستم معرفی کنیم. به همین منظور وارد web.config شوید و کلاس جدید را معرفی کنید:

```
<httpModules>
     <add name="UrlPath" type="UrlPath"/>
   </httpModules>
```

اگر کلاس شما داخل یک namespace قرار دارد، در قسمت type حتما قبل از نام کلاس، آن را تعریف کنید namspace.ClassName حالا دیگر کلاس UrlPath به عنوان یک httpmodule به سیستم معرفی شده است. تگ httpmodule را بین تگ <system.web> قرار داده ایم.

در ادامه پروژه را start بزنید تا نتیجه کار را ببینید:

اگر IIS شما، هم نسخه‌ی IIS من باشد، نباید تفاوتی مشاهده کنید و می‌بینید که درخواست‌ها هیچ تغییری نکردند؛ چرا که اصلا httpmodule اجرا نشده است. در واقع در نسخه‌های قدیمی IIS یعنی 6 به قبل، این تعریف درست است ولی از نسخه‌ی 7 به بعد IIS، روش دیگری برای تعریف را قبول دارد و باید تگ httpmodule، بین دو تگ <syste.webserver> قرار بگیرد و نام تگ httpmodule به module تغییر پیدا کند.

پس کد فوق به این صورت تغییر می‌کند:

```
<system.webServer>

    <modules>
      <add name="UrlRewrite" type="UrlRewrite"/>
    </modules>
</system.webServer>
```

حالا اگر قصدا دارید که پروژه‌ی شما در هر دو IIS مورد حمایت قرار گیرد، باید این ماژول را در هر دو جا معرفی کرده و در تگ system.webserver نیاز است تگ زیر تعریف شود که به طوری پیش فرض در webconfig می‌باشد:

```
<validation validateIntegratedModeConfiguration="false"/>
```

در غیر این صورت خطای زیر را دریافت می‌کنید:

HTTP Error 500.22 - Internal Server Error در کل استفاده از این ماژول به شما کمک می‌کند تمامی اطلاعات ارسالی به سیستم را قبل از رسیدن به قسمت پردازش بررسی نمایید و هر نوع تغییری را که میخواهید اعمال کنید و لازم نیست این تغییرات را روی هر بخش، جداگانه انجام دهید یا یک کلاس بنویسید که هر بار در یک جا صدا بزنید و خیلی از موارد دیگر

Global.asax و HttpModule

اگر با global.asax کار کرده باشید حتما می‌پرسید که الان چه تفاوتی با httpmodule دارد؟ در فایل global هم همین‌ها را دارید و دقیقا همین مزایا مهیاست؛ در واقع global.asax یک پیاده سازی از httpapplication هست.

کلاس‌های httpmodule نام دیگری هم دارند به اسم Portable global .asax به معنی یک فایل global.asax قابل حمل یا پرتابل. دلیل این نام گذاری این هست که شما موقعی که یک کد را در فایل global می‌نویسید، برای همیشه آن کد متعلق به همان پروژه هست و قابل انتقال به یک پروژه دیگر نیست ولی شما میتوانید httpmoduleها را در قالب یک پروژه به هر پروژه ای که دوست دارید رفرنس کنید و کد شما قابلیت استفاده مجدد و Reuse پیدا می‌کند و هم اینکه در صورت نیاز می‌توانید آن‌ها را در قالب یک dll منتشر کنید.

نظرات خوانندگان

نویسنده: عباس حجتی
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۰:۲۴

با سلام؛ ممنون بابت مطلب خوبی که ارسال کردید.
من می‌خوام از httpMuodule برای کنترل دسترسی کاربران (Authorization) استفاده کنم و به سشن کاربر نیاز دارم. مشکلی که هست موقع استفاده به این شکل HttpContext.Current.Session پیغام زیر رو میده:

Object not reference to instance of an object

آیا برای استفاده از سشن راه خاصی هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۴:۲۲

بله. در جدولی که تهیه کردند این مورد دقیقا ذکر شده:
« **AcquireRequestState** : این قسمت برای مدیریت state می‌باشد مثلا مدیریت **session** ها»
به این معنا که سشن در تمام رویدادگردان‌های آن مهیا نیست. فقط تعدادی از آن‌ها دسترسی به سشن دارند. برای مثال:

```
public class SimpleModule : IHttpModule
    {
        void IHttpModule.Init(HttpApplication application)
        {
            application.BeginRequest += new System.EventHandler(BeginRequest);
            application.AcquireRequestState += new EventHandler(application_AcquireRequestState);
        }

        public void BeginRequest(object sender, EventArgs e)
        {
            // no session here
        }

        void application_AcquireRequestState(object sender, EventArgs e)
        {
            HttpApplication app = sender as HttpApplication;
            app.Session.Add("Message", "hello module");
        }

        public void Dispose()
        {
        }
    }
```

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۴:۵۰

از رویداد AcquireRequestState استفاده کنید ، همانطور که گفتیم این رویداد برای مدیریت stateهاست ، عموما در رویدادهای ابتدایی sessionها هنوز ایجاد نشدند
کد زیر نمونه ای از دسترسی به session هاست :

```
public void Init(HttpApplication app)
        {
            app.AcquireRequestState+=new EventHandler(_AUTH);
        }

    private void _AUTH(object sender, EventArgs e)
        {
            HttpApplication app = (HttpApplication) sender;
            HttpContext context = app.Context;
            HttpSessionState session = context.Session;

                if(session!=null)
            {
                string text = "session is not exist";
```

```
                if (session["userid"] != null)
                {
                    text = "session is exist";
                }
                    context.Response.Write(text);

            }

        }
```

توجه داشته باشید که خط if(session!=null) بسیار مهم هست و در صورت نبودن خط شرطی بعدی دچار خطایی که شما گرفتید می‌شود

نویسنده: عباس حجتی
تاریخ: ۱۳۹۳/۰۹/۲۹ ۱۷:۱۰

ممنون، درست شد.
یک مشکل دیگه اینکه درخواست‌های AJAX دیگه سمت سرور نمیره، می‌تونه به HttpModule ارتباطی داشته باشه؟

ممنون، این مشکل هم با فیلتر کردن درخواست‌های AJAX برطرف شد.

```
if (HttpContext.Current.Request.Headers["X-Requested-With"] != "XMLHttpRequest")
{
}
```