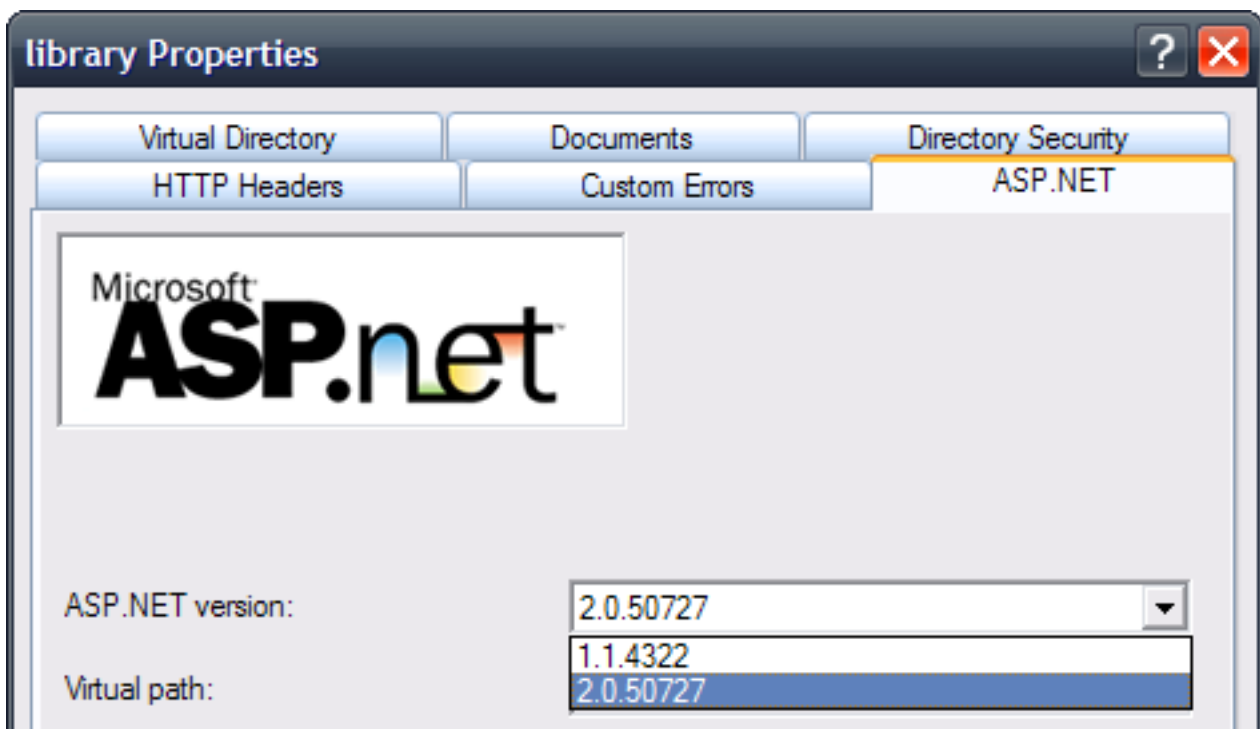


همانطور که مطلع هستید در تنظیمات یک دایرکتوری مجازی در IIS6 یا 5، حتی پس از نصب دات نت فریم ورک سه و نیم، گزینه انتخاب نگارش 3.5 ظاهر نمی‌شود و همان تنظیمات ASP.Net 2.0 [کافی است](#) (شکل زیر) (دات نت 3 و سه و نیم را می‌توان بعنوان افزونه‌هایی با مقیاس سازمانی (WF ، WCF و ...) برای دات نت 2 در نظر گرفت).



هنگام استفاده از VS.Net 2008 و تنظیم نوع پروژه به دات نت فریم ورک 3.5، به صورت خودکار تنظیمات لازم به وب کانفیگ برنامه جهت استفاده از کامپایلرهای مربوطه نیز اضافه می‌شوند که شاید از نظر دور بمانند. برای آزمایش این مورد، فرض کنید صفحه زیر را بدون استفاده از code behind و VS.Net ایجاد کرده ایم (جهت آزمایش سریع یک قطعه کد Linq).

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System" %>
<%@ Import Namespace="System.Linq" %>

<form id="Form1" method="post" runat="server">
  <asp:GridView ID="GridView1" runat="server" />
</form>

<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
    string[] cities = {
      "London", "Amsterdam", "San Francisco", "Las Vegas",
      "Boston", "Raleigh", "Chicago", "Charlestown",
```

```

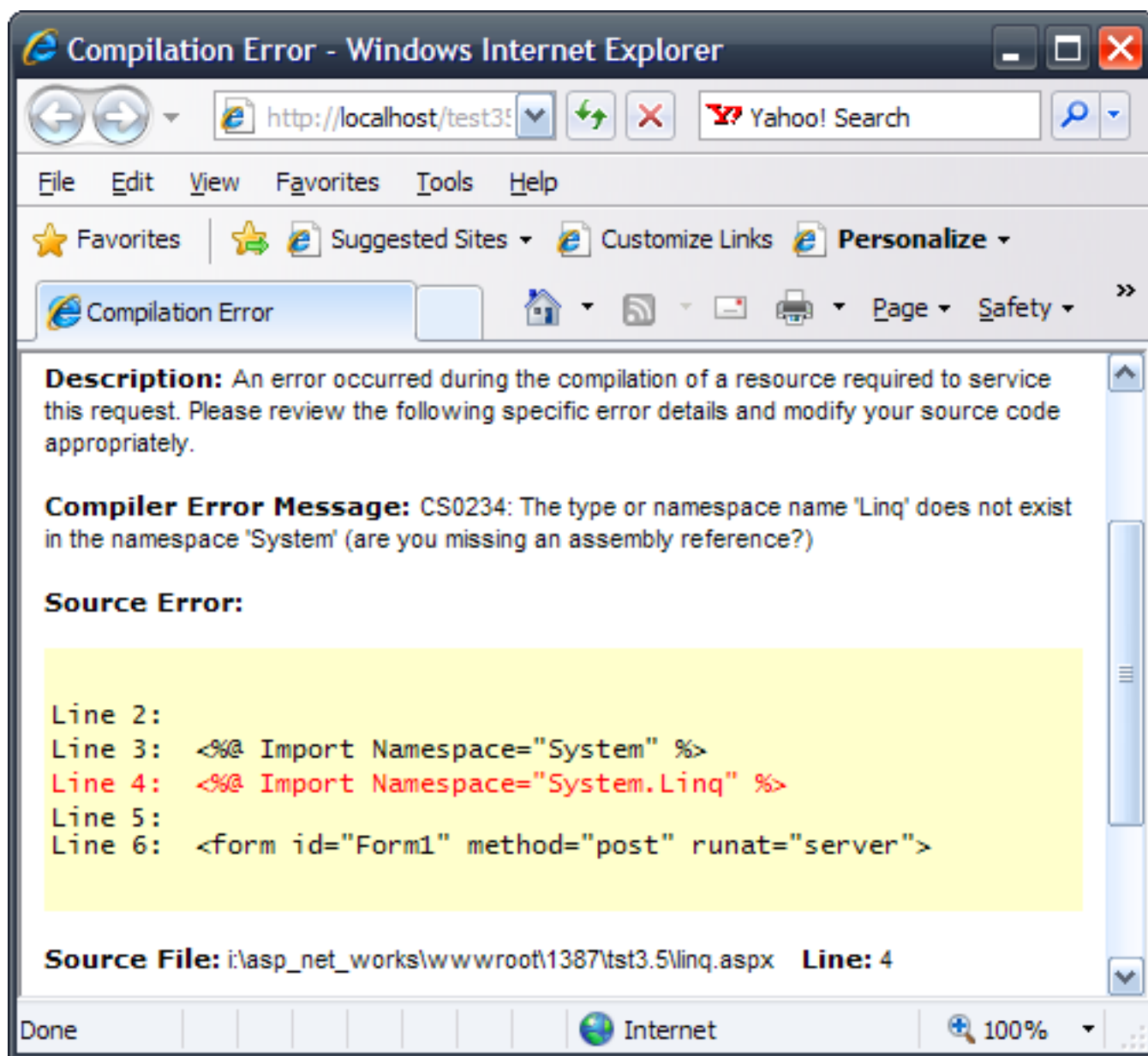
        "Helsinki", "Nice", "Dublin"
    };

    GridView1.DataSource = from city in cities
                           where city.Length > 4
                           orderby city
                           select city.ToUpper();

    GridView1.DataBind();
}
</script>

```

بلافاصله پس از اجرا با خطای زیر روبرو خواهیم شد.



این قطعه کد چون از قابلیت‌های کامپایلر جدید سی شارپ استفاده می‌کند، با کامپایلر پیش فرض و تنظیم شده دات نت 2 کار

نخواهد کرد و باید برای رفع این مشکل، فایل web.config جدیدی را نیز به پوشه برنامه اضافه کنیم:

```
<?xml version="1.0"?>
<configuration>

<system.codedom>
  <compilers>
    <compiler language="c#;cs;csharp" extension=".cs" warningLevel="4"
type="Microsoft.CSharp.CSharpCodeProvider, System, Version=2.0.0.0, Culture=neutral,
PublicKeyToken=b77a5c561934e089">
      <providerOption name="CompilerVersion" value="v3.5"/>
      <providerOption name="WarnAsError" value="false"/>
    </compiler>
  </compilers>
</system.codedom>

<system.web>
  <compilation defaultLanguage="c#">
    <assemblies>
      <add assembly="System.Core, Version=3.5.0.0, Culture=neutral, PublicKeyToken=B77A5C561934E089"/>
    </assemblies>
  </compilation>
</system.web>

</configuration>
```

در اینجا قید اسمبلی System.Core ضروری است و همچنین نگارش کامپایلر نیز به صورت صریح قید شده است تا IIS را وادر کند که از قابلیت‌های جدید دات نت فریم ورک استفاده نماید.

همانطور که ذکر شد اگر از VS.Net 2008 استفاده کنید، هیچ وقت درگیر این مباحث نخواهید شد و همه چیز از پیش تنظیم شده است.

IIS6 فایل‌هایی را که شناسد، سرو نخواهد کرد. بنابراین اگر یکی از کاربران مثلاً یک فایل docx آفیس 2007 را آپلود کرده باشد، به محض کلیک بر روی لینک دریافت فایل، با [خطای زیر](#) متوقف خواهد شد:

HTTP Error 404 - File or directory not found

فایل بر روی سرور موجود است اما کاربر قادر به دریافت آن نیست.

برای شناساندن فرمت‌های جدید به IIS6 می‌توان به یکی از دو روش زیر عمل کرد:

الف) اضافه کردن mime-type جدید از طریق کنسول IIS

ب) ویرایش کردن فایل MetaBase.xml مربوط به IIS

در هر دو روش فوق نیاز است تا با mime-type فایل‌های جدید آشنا بود. برای مثال لیست کامل mime-types مربوط به فایل‌های آفیس 2007 به صورت زیر است:

```
.docm,application/vnd.ms-word.document.macroEnabled.12
.docx,application/vnd.openxmlformats-officedocument.wordprocessingml.document
.dotm,application/vnd.ms-word.template.macroEnabled.12
.dotx,application/vnd.openxmlformats-officedocument.wordprocessingml.template
.potm,application/vnd.ms-powerpoint.template.macroEnabled.12
.potx,application/vnd.openxmlformats-officedocument.presentationml.template
.ppam,application/vnd.ms-powerpoint.addin.macroEnabled.12
.ppsm,application/vnd.ms-powerpoint.slideshow.macroEnabled.12
.ppsx,application/vnd.openxmlformats-officedocument.presentationml.slideshow
.pptm,application/vnd.ms-powerpoint.presentation.macroEnabled.12
.pptx,application/vnd.openxmlformats-officedocument.presentationml.presentation
.xlam,application/vnd.ms-excel.addin.macroEnabled.12
.xlsb,application/vnd.ms-excel.sheet.binary.macroEnabled.12
.xlsm,application/vnd.ms-excel.sheet.macroEnabled.12
.xlsx,application/vnd.openxmlformats-officedocument.spreadsheetml.sheet
.xltm,application/vnd.ms-excel.template.macroEnabled.12
.xltx,application/vnd.openxmlformats-officedocument.spreadsheetml.template
```

روش ب)

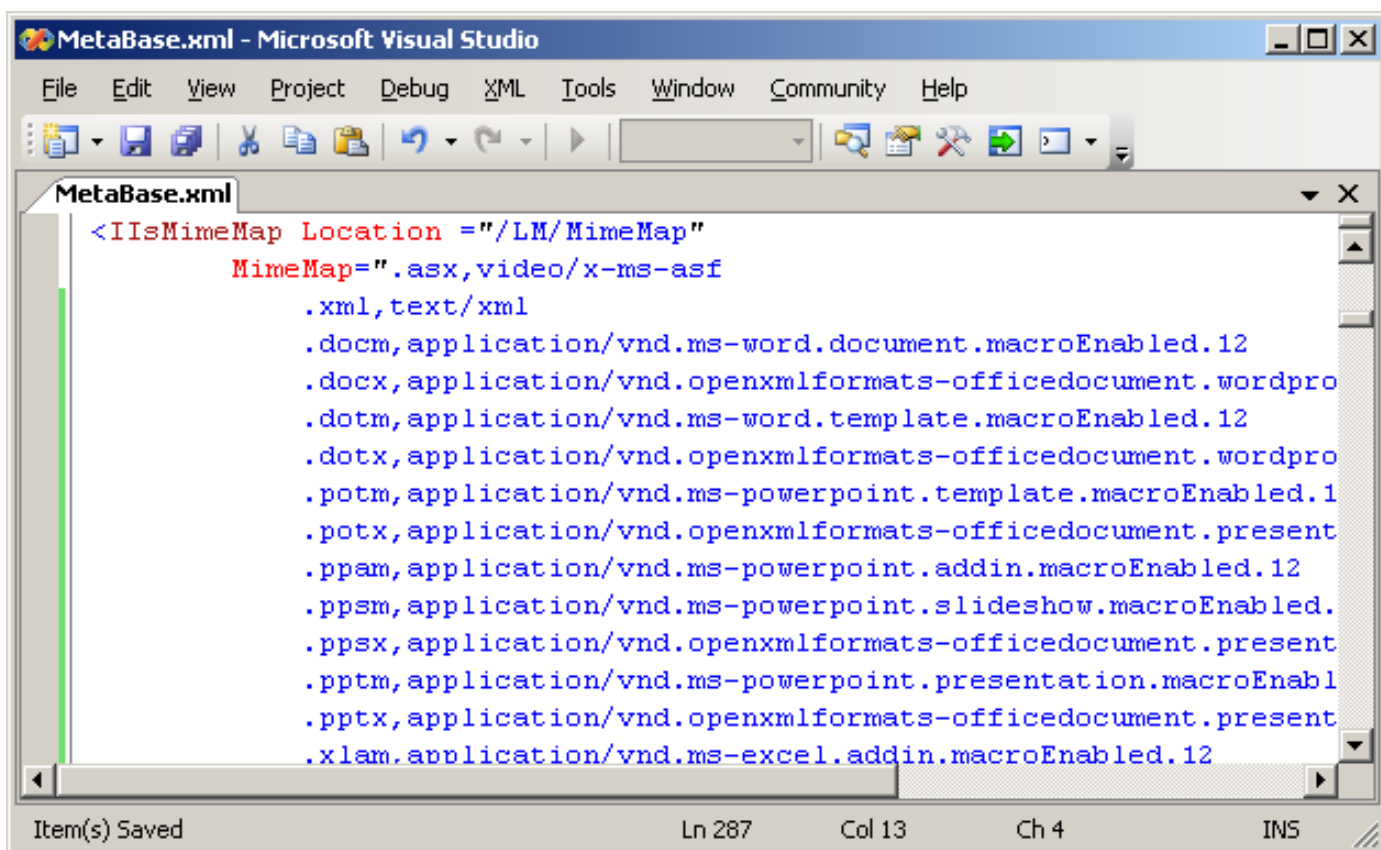
ابتدا IIS6 را stop کنید (در غیر اینصورت قادر به ذخیره سازی تغییرات نخواهید بود):

```
iisreset /stop
```

سپس فایل متابیس آن‌را در یک ادیتور متنی باز کنید. این فایل در آدرس زیر قرار دارد:

```
C:\WINDOWS\system32\inetsrv\MetaBase.xml
```

تگ مربوط به IISMimeMap را یافته و خطوط فوق را دقیقاً به همین صورتیکه ملاحظه می‌کنید به آن اضافه نمایید.



و در آخر IIS را راه اندازی کنید:

```
iisreset /start
```

روش الف)

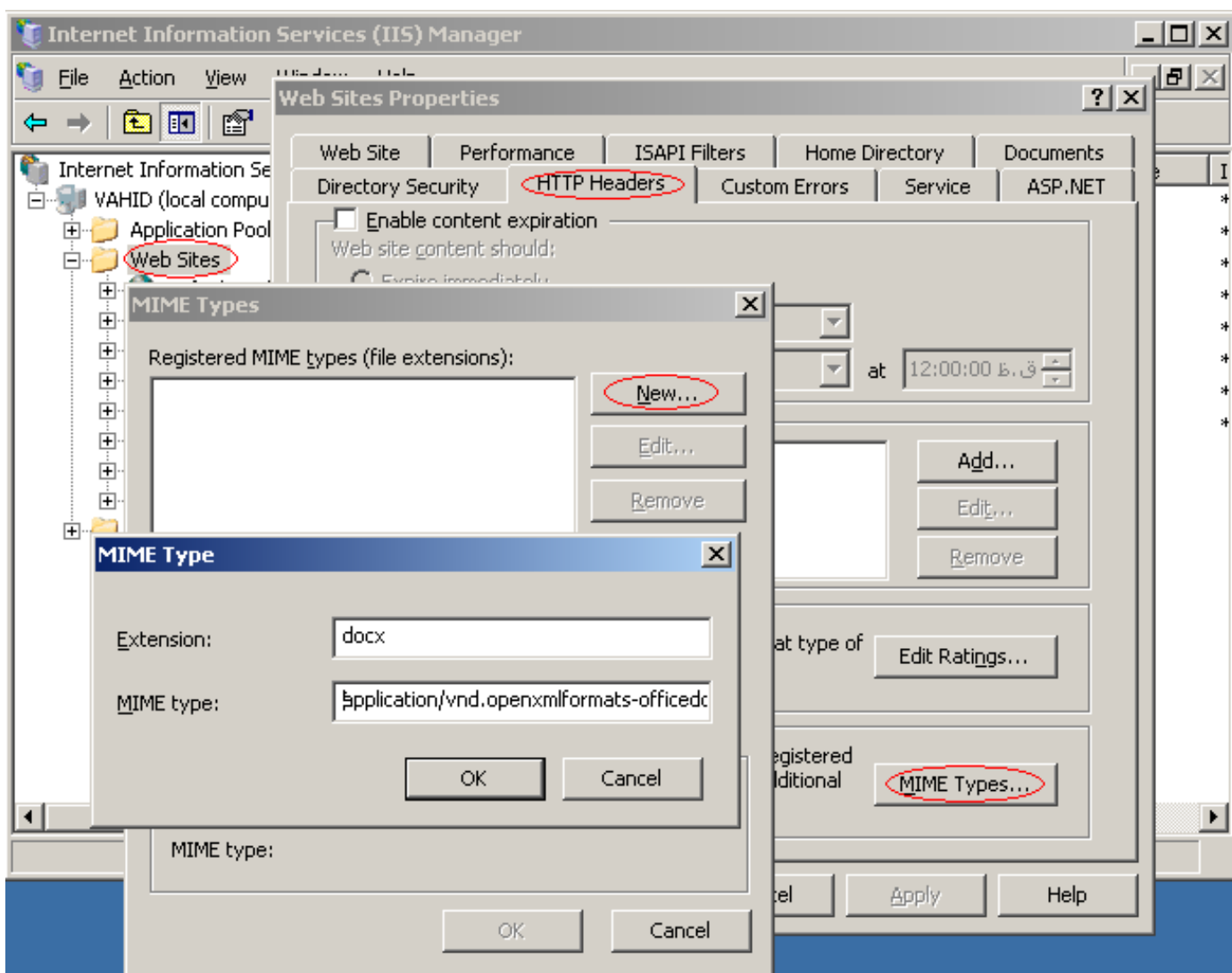
این روش نیازی به stop و start وب سرور ندارد و به محض اضافه شدن، به صورت خودکار اعمال خواهد شد اما کمی طولانی‌تر است:

کنسول IIS را باز کنید

بر روی web sites کلیک راست کنید. (منظور بالاترین سطح ممکن است)

گزینه properties را انتخاب کرده و سپس به برگه http headers مراجعه نمایید.

در اینجا بر روی دکمه mime-types کلیک کرده و در صفحه باز شده باید تک تک موارد جدید را به صورت دستی وارد نمایید (در اینجا نیازی به ذکر نقطه مربوط به پسوند فایل نیست)



لازم به ذکر است که این نوع mime-types به IIS7 اضافه شده‌اند .

عنوان:	لیست تازه‌های IIS 7.5
نویسنده:	وحید نصیری
تاریخ:	۱۳۸۸/۰۱/۱۳ ۱۹:۴۴:۰۰
آدرس:	www.dotnettips.info
برچسب‌ها:	IIS

IIS 7.5 که به همراه ویندوز سرور 2008 ارائه می‌شود شامل تازه‌های زیر است:

بیش از 50 مورد cmdlet جدید مخصوص Powershell جهت مدیریت IIS افزونه‌های جدید مدیریتی: Database Manager (مدیریت اس کیوال سرور از درون IIS و کنسول آن)، Configuration Editor (تولید خودکار اسکریپت‌های مدیریتی جهت اتوماسیون امور مرتبط)، IIS Reports و Request Filtering . پشتیبانی از One-click publishing موجود در Visual Studio 10 Web Deployment Tool یا همان MS Deploy سابق جهت مدیریت بهتر برنامه‌های وب. امکان ردیابی تغییرات در کانفیگ وب سرور گزارشگیری بهتر از وضعیت کارآیی سرور ساپورت دات نت جهت Server Core معرفی شده در ویندوز سرور 2008 WebDav که پیشتر به صورت یک افزونه‌ی آن معرفی شده بود، اکنون جزئی از IIS 7.5 است. یکپارچگی با URLScan 3.0 جهت بالا بردن امنیت وب سرور. FTP server services : با کنسول مدیریتی IIS یکپارچه شده است با بهبودهایی در نحوه‌ی تنظیم کردن و ردیابی آن.

جهت مطالعه بیشتر در مورد تازه‌های ویندوز سرور 2008 نگارش R2 می‌توان به مقالات زیر رجوع کرد:

[Windows Server 2008 R2 new features - the complete list - Part 1: Virtualization](#)

[Windows Server 2008 R2 new features - the complete list - Part 2: Active Directory](#)

[Windows Server 2008 R2 new features - the complete list - Part 3: IIS 7.5 and Performance](#)

[Windows Server 2008 R2 new features - the complete list - Part 4: Administration](#)

نظرات خوانندگان

نویسنده: مهدی پایروند
تاریخ: ۱۱:۳۸:۴۵ ۱۳۸۸/۰۵/۱۴

سلام آیا لیستی برای دانلود و نصب IIS نسخه های 6 به بالا موجود هست؟

نویسنده: وحید نصیری
تاریخ: ۱۴:۳۵:۰۶ ۱۳۸۸/۰۵/۱۴

IIS یک کامپوننت ویندوز است و جدا از ویندوز ارائه نمی‌شود.

با استفاده از IIS6 ویندوز سرور 2003 و تنظیمات ویژه در web.config یک برنامه ASP.Net، حداکثر می‌توان یک فایل 2 گیگابایتی را آپلود کرد (جهت مصارف اینترنتی). برای مثال:

```
<system.web>
  <httpRuntime maxRequestLength="2097151" executionTimeout="900" />
</system.web>
```

2097151 کیلوبایت حداکثر مقداری است که اینجا می‌توان تنظیم کرد و بیش از این با خطای زیر متوقف خواهیم شد:

Parser Error Message: The value for the property 'maxRequestLength' is not valid. The error is: The value must be inside the range 0-2097151

این محدودیت در IIS7 برطرف شده است که تنظیمات آن در وب کانفیگ به صورت زیر می‌باشد:

```
<system.webServer>
  <security>
    <requestFiltering>
      <requestLimits maxAllowedContentLength="4294967295" />
    </requestFiltering>
  </security>
</system.webServer>
```

در اینجا maxAllowedContentLength بر حسب بایت است و نه همانند maxRequestLength بر حسب کیلوبایت (که در IIS7 هیچ تاثیری نخواهد داشت). البته تنظیمات فوق در اینجا به پایان نمی‌رسند زیرا بر اساس تنظیمات امنیتی IIS7، کاربران مجاز به اعمال تنظیمات شخصی خود نیستند و خطای زیر را دریافت خواهند کرد:

The requested page cannot be accessed because the related configuration data for the page is invalid

و یا

The request filtering module is configured to deny a request that exceeds the request content length

برای این منظور باید دستور زیر را با دسترسی مدیریتی در خط فرمان اجرا نمود:
 برای یک برنامه خاص:

```
%windir%\system32\inetsrv\appcmd set config "Default Web Site/<your app>" -section:requestFiltering -requestLimits.maxAllowedContentLength:4294967295
```

و یا برای تمام برنامه‌ها:

```
%windir%\system32\inetsrv\appcmd set config -section:requestFiltering -  
requestLimits.maxAllowedContentLength:4294967295
```

و یا فایل زیر را یافته:

```
%windir%\System32\inetsrv\config\applicationHost.config
```

در آن سطر زیر را

```
<section name="requestFiltering" overrideModeDefault="Deny" />
```

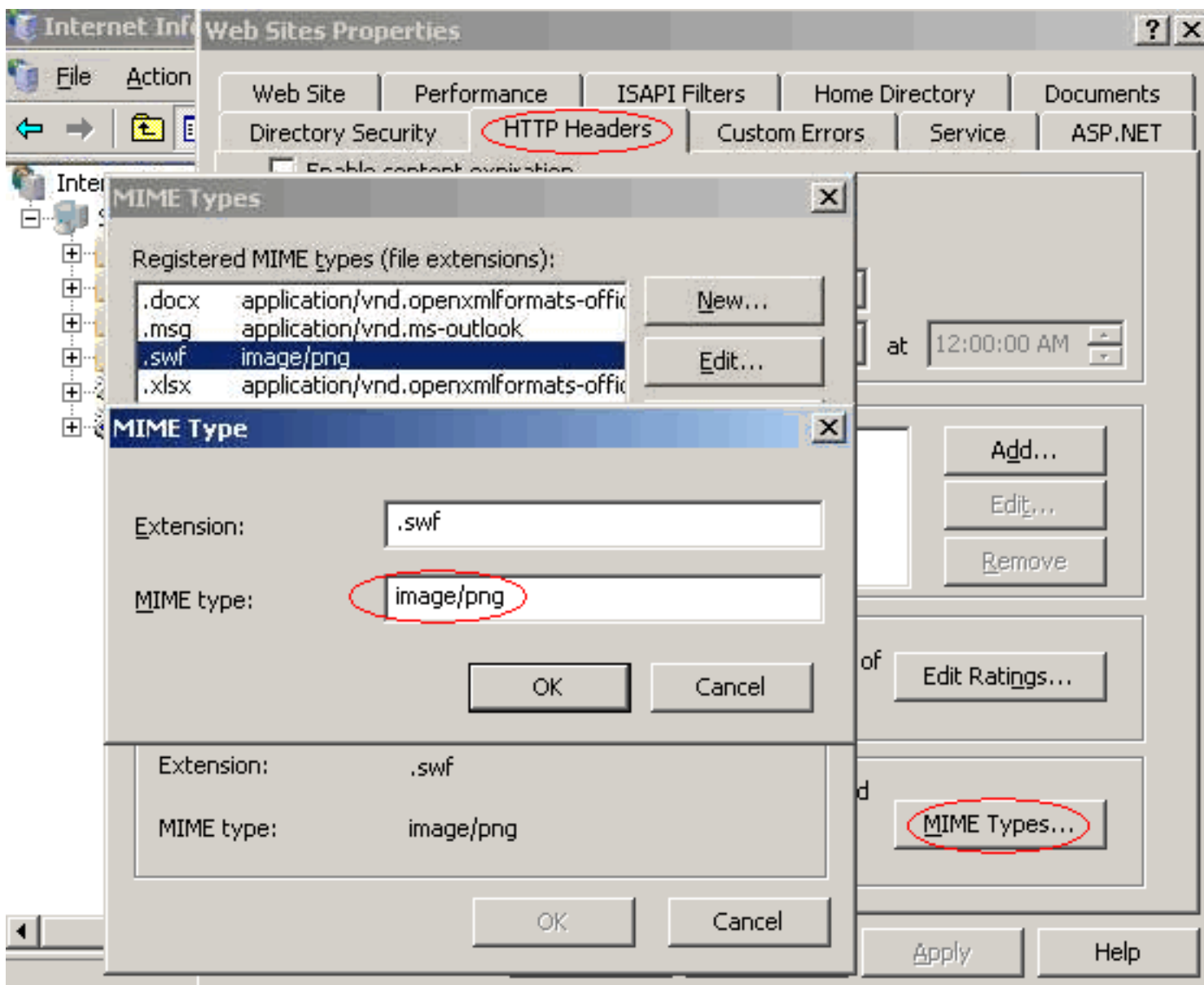
ویرایش کرده و مقدار overrideModeDefault آن را به Allow تنظیم کرد:

```
<section name="requestFiltering" overrideModeDefault="Allow" />
```

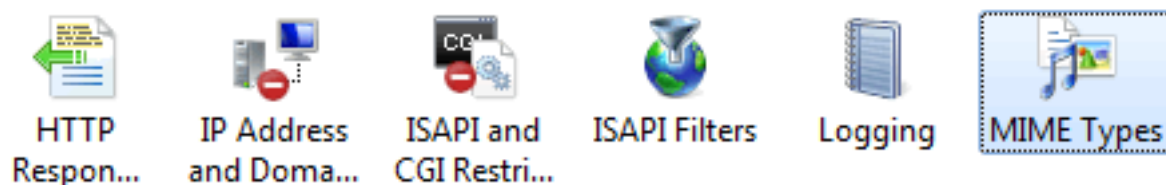
مقدار پیش فرض maxRequestLength در IIS6 مساوی 4 مگابایت و مقدار پیش فرض maxAllowedContentLength در IIS7 مساوی 28.6MB می‌باشد. maxAllowedContentLength از نوع INT32 است یعنی حداکثر تا 4 گیگابایت را توسط آن می‌توان مقدار دهی کرد. maxRequestLength از نوع Int32 است با حداکثر مقدار قابل تنظیم 2 گیگابایت.

چند روزی هست که به دلیل قطعی کابل، فایل‌های فلش سایت‌ها از کار افتاده! فایل‌های فلش را بر اساس mime type آن‌ها فیلتر کرده‌اند یعنی هر چه application/x-shockwave-flash که از طرف وب سرور شما سرو شود فیلتر می‌شود. یک راه حل این است که پسوند تمام فایل‌های فلش را تغییر داد تا وب سرور دیگر این mime type را از خودش بروز ندهد (در یک وب سرور هر mime type دقیقاً به یک یا چند نوع پسوند فایل map شده). این مورد نیاز به اصلاح تمام صفحات سایت نیز خواهد داشت (علاوه بر تغییر پسوند فایل‌های موجود). خوشبختانه فلش پلیر کاری به mime type یا حتی پسوند فایل، جهت نمایش آن ندارد. راه حل ساده‌تر (بدون نیاز به تغییری در فایل‌ها) هم این است که کمی تغییرات وب سرور خود را تغییر داد:

در IIS6 :

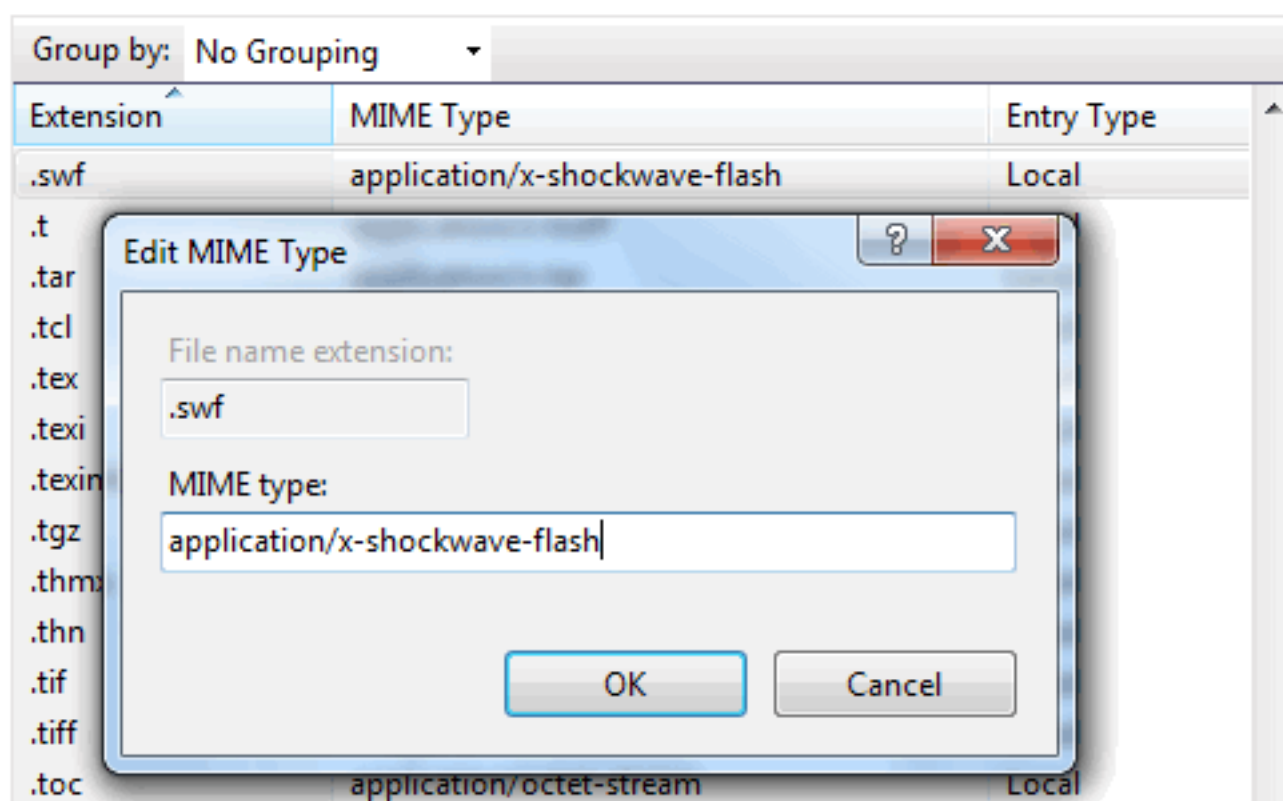


در IIS7 :



MIME Types

Use this feature to manage the list of file name extensions and associated content types that are served as static files by the Web server.



کلا mime type موجود را به برای مثال image/png تغییر دهید تا قسمت‌های از کار افتاده سایت مجدداً برقرار شود و آبروی کاری رفته مجدداً به جوی بازگردد.

نظرات خوانندگان

نویسنده: ms

تاریخ: ۱۶:۰۹:۳۳ ۱۳۸۸/۱۱/۲۰

مونده بودم اینا چه جوری فایل های خاصی رو فیل می کنن
پس به پسوند فایل کاری ندارن ؟

نویسنده: وحید نصیری

تاریخ: ۱۸:۱۸:۵۸ ۱۳۸۸/۱۱/۲۰

فعلا خیر؛ اگر mime type بروز داده شده application/x-shockwave-flash نباشه، پسوند می‌تونه همان swf هم باشد. البته اگر
پسوند را عوض کردید، خودبخود mime type هم چیز دیگری خواهد شد (بر اساس نگاشت‌های موجود در وب سرور).
و اگر لازم باشه بر اساس پسوند هم می‌تونند ...

استفاده از IIS در VS.NET و پروژه‌های ASP.NET داستان خودش را دارد. در نگارش‌های 2002 و 2003 آن، تنها وب سرور قابل استفاده جهت کار با VS.NET همان IIS اصلی بود. مهم‌ترین مشکل این روش، نیاز به داشتن دسترسی مدیریتی بر روی سیستم بود (که در بعضی از شرکت‌ها، این مورد برای عموم کاربران ممنوع است) به همراه نصب جداگانه و تنظیمات مخصوص IIS، صرفاً جهت آزمایش یک برنامه‌ی ساده؛ همچنین با توجه به اینکه IIS جزو کامپوننت‌ها ویندوز بوده و هر نگارشی، IIS خاص خودش را دارد است، این مورد هم مشکلات ویژه‌ای را به همراه دارد (برای مثال IIS5 ویندوز XP را با IIS7 ویندوز سرور 2008 در نظر بگیرید؛ یکی برای توسعه یکی جهت محیط کاری). این روش در VS.Net 2005 کنار گذاشته شد و از وب سرور توکاری به نام Cassini یا ASP.NET Development Server استفاده گردید. به این صورت دیگر نیازی به نصب مجزای IIS کامل جهت آزمایش برنامه‌های ASP.NET نبود و همچنین نیاز به داشتن دسترسی مدیریتی الزامی نیز منتفی گردید. این روش هنوز هم تا نگارش 2010 ویژوال استودیو مرسوم است؛ اما ... اما کسانی که با Cassini کار کرده باشند می‌دانند که یک سری از رفتارهای آن با IIS واقعی تطابق ندارد و اگر برنامه‌ی ASP.NET شما با Cassini خوب نمایش داده می‌شود الزامی ندارد که با IIS واقعی هم به همان نحو رفتار کند، برای نمونه رفتار مسیریابی آدرس‌های نسبی در IIS واقعی و Cassini یکی نیست. علاوه بر آن IIS های 7 و 7.5 هم امکانات و ویژگی‌های خاص خود را دارند که Cassini آن‌ها را پوشش نمی‌دهد؛ به علاوه این دو فقط در ویندوزهای جدید مانند ویندوز سرور 2008 یا ویندوز 7 قابل دسترسی هستند. به همین جهت اخیراً یک نسخه‌ی سبک و express از IIS 7.5 به صورت جداگانه برای برنامه نویسی‌ها فقط جهت آزمون برنامه‌های خود تهیه شده است و البته هدفگیری اصلی آن پروژه‌ی WebMatrix است؛ به همراه ویژگی‌های جدید IIS7 مانند امکان آزمون تنظیمات ویژه IIS7 در وب کانفیگ برنامه، پشتیبانی کامل از SSL، Rewrite و سایر ماژول‌های IIS7، عدم نیاز به دسترسی مدیریتی برای اجرای آن، امکان اجرای آن بر روی پورت‌های مختلف بدون تداخل با وب سرور(های) موجود بر روی سیستم و همچنین برخلاف IIS7 اصلی، بر روی ویندوز XP نیز قابل اجرا است. حجم نگارش IIS Express 7.5 تنها 3.9 مگابایت است:

Internet Information Services (IIS) 7.5 Express

سرویس پک یک ویژوال استودیوی 2010 (که در زمان نگارش این مطلب نسخه‌ی بتای آن ارائه شده)، یک گزینه‌ی جدید را به منوی کلیک راست بر روی نام پروژه در VS.NET به نام Use IIS Express اضافه کرده است تا به سادگی بتوان از این امکان جدید استفاده کرد (یا به عبارتی با IIS Express یکپارچه است و نیاز به تنظیم خاصی ندارد). در سایر حالات (و نسخه‌هایی که این یکپارچگی وجود ندارد و نخواهد داشت) به صورت زیر می‌توان عمل کرد:

روش اول:

دستور زیر را در خط فرمان وارد نمایید:

```
"C:\Program Files\IIS Express\iisexpress.exe" /path:D:\Prog\1389\MySite\ /port:4326 /clr:v4.0
```

به این صورت وب سروری جهت ارائه‌ی سایتی با مسیر ذکر شده بر روی پورت 4326 (<http://localhost:4326>) بر اساس دات نت 4 تشکیل خواهد شد (برای نمونه جهت دات نت سه و نیم مقدار v3.5 را وارد نمایید).

روش دوم (که در حقیقت همان روش اول با ارائه‌ی پشت صحنه‌ی موقت آن است):

الف) ابتدا به مسیر My Documents\IISExpress\config مراجعه کرده و فایل applicationhost.config را باز کنید. سپس گره مربوط به site را یافته (حدود سطر 153) و گزینه‌ی serverAutoStart را حذف کنید:

```
<site name="WebSite1" id="1">
  <application path="/">
    <virtualDirectory path="/" physicalPath="%IIS_SITES_HOME%\WebSite1" />
  </application>
</site>
```

```

</application>
<bindings>
  <binding protocol="http" bindingInformation=":8080:localhost" />
</bindings>
</site>

```

ب) سپس تنظیمات سایت مورد نظر خود را به صورت دستی به این فایل اضافه کنید. برای مثال:

```

<site name="WebSite2" id="2">
  <application path="/" applicationPool="Clr4IntegratedAppPool">
    <virtualDirectory path="/" physicalPath="D:\Prog\1389\MyTestSite\" />
  </application>
  <bindings>
    <binding protocol="http" bindingInformation=":1389:localhost" />
  </bindings>
</site>

```

توضیحات:

Name در اینجا نامی دلخواه است که وارد خواهید نمود.

Id شماره سایتی است که ثبت خواهد شد.

applicationPool در اینجا بسیار مهم است. اگر سایت شما مبتنی بر دات نت 4 است، Clr 4 IntegratedAppPool را وارد نمائید و اگر غیر از این است، Clr 2 IntegratedAppPool باید تنظیم شود.

physicalPath همان مسیر پروژه شما است.

در قسمت bindingInformation هم می‌توان شماره پورت مورد نظر را وارد کرد.

اکنون فایل applicationhost.config را ذخیره کرده و ببندید.

سپس دستور زیر را در خط فرمان ویندوز وارد نمائید:

```
"C:\Program Files\IIS Express\iisexpress.exe" /site:WebSite2
```

که در اینجا WebSite2 همان مدخل جدیدی است که به فایل applicationhost.config اضافه شده است. به این صورت آدرس <http://localhost:1389> جهت دسترسی به سایت شما آماده استفاده خواهد بود.

تنظیمات دیباگر VS.NET :

تا اینجا تنها موفق شده‌ایم که این وب سرور آزمایشی را راه اندازی کنیم. اما نکته‌ی مهم امکان دیباگ کردن برنامه توسط آن را از دست داده‌ایم. برای این منظور در VS.NET به خواص پروژه، برگه‌ی Web آن مراجعه کنید. در قسمت Servers گزینه‌ی use custom web server را انتخاب کرده و آدرسی را که در یکی از دو روش فوق ساخته‌اید وارد نمایید. برای مثال <http://localhost:4326> همچنین باید دقت داشت که در همین قسمت هیچکدام از debuggers ذیل گزینه‌ی use custom web server نباید تیک خورده باشند (چون VS.NET دقیقاً نمی‌داند که باید به کدام پروسه در ویندوز attach شود). اکنون برنامه را در حالت دیباگ در VS.NET آغاز کنید (بدیهی است فرض بر این است که iisexpress.exe با تنظیمات ذکر شده باید در حال اجرا باشد).

و ... حداقل مزیت آن بسیار سریع‌تر بودن این روش نسبت به Cassini یا ASP.NET Development Server است.

تا اینجا فقط VS.NET به صورت خودکار مرورگر را باز کرده و سایت نمایش داده می‌شود؛ اما اگر در قسمتی از کدهای خود breakpoint قرار دهیم کار نمی‌کند. برای این منظور باید در حین اجرای برنامه، از منوی debug، گزینه‌ی attach to process را انتخاب کرده و به iisexpress متصل شوید.

نظرات خوانندگان

نویسنده: Abolfazl Hosnaddinov
تاریخ: ۱۲:۲۷:۵۷ ۱۳۸۹/۱۰/۲۶

با سلام...
ممنون و متشکر از بابت وبلاگ پربار شما. مطلب مفید و جالبی بود.
با آرزوی موفقیت روزافزون

نویسنده: مهدی پایروند
تاریخ: ۱۰:۲۰:۱۰ ۱۳۸۹/۱۰/۲۸

سلام اول خواستم که ازتون تشکر کنم و بعدش این قطعه رو بذار تا بقیه هم لذت ببرن:

نو بهاران چو رسد فکر و نظر تازه کنم
باکی نیست هوس کار نو و تازه کنم
نه به اینترنت و وایرلسو کد بازیها
نه به این کد زنی دائم و چت بازی ها
باگ این آبجکت و اکسپشن آن ان تیتی
حذف ریسپانس هدر و افزودن یک پالیسی
قیل و قال منو این هاست خشکیده سواد
که بجز صفحه خطا هیچ نمیداد امداد
کد بی نقص اگر کار کند در لوکال
هر دمی معجزه باید که شود بی اشکال

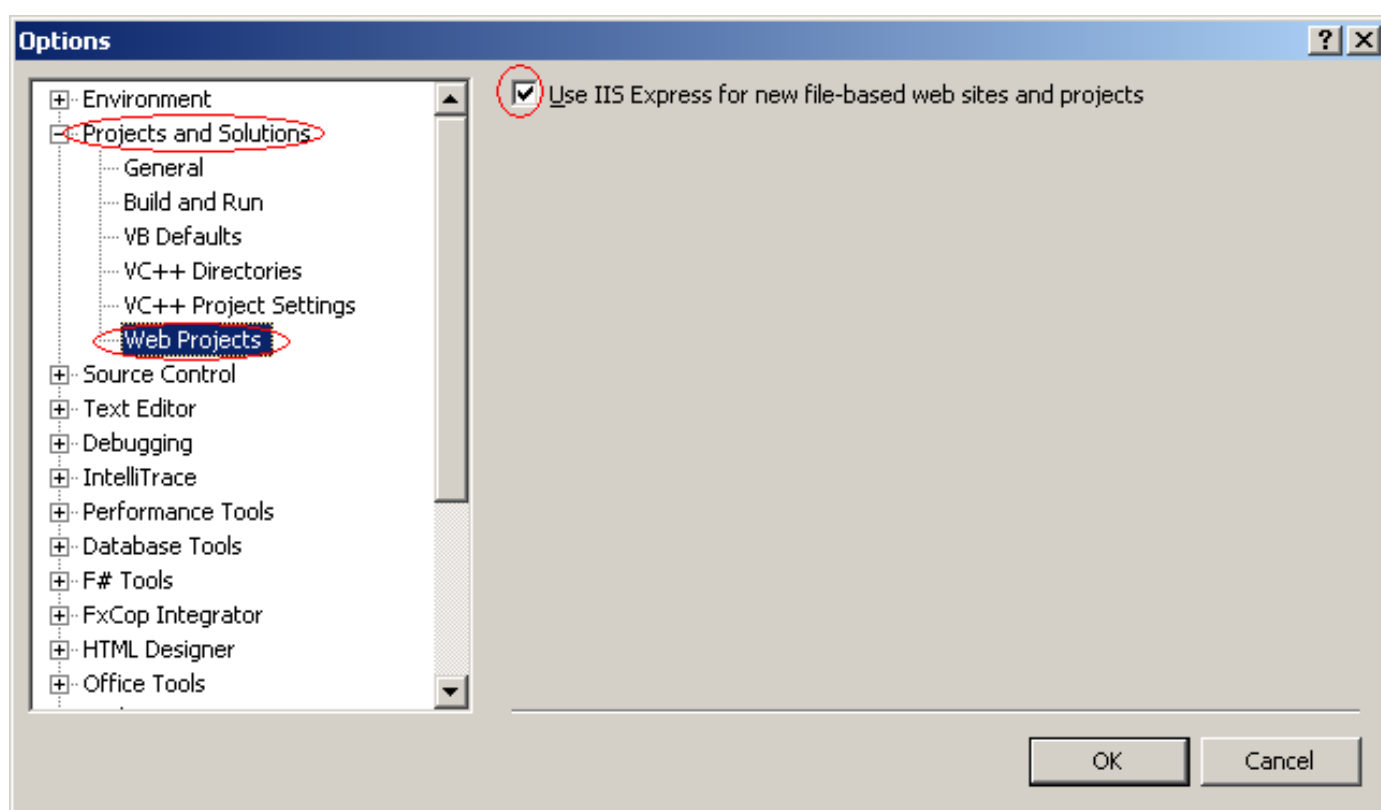
نویسنده: وحید نصیری
تاریخ: ۱۸:۴۰:۲۶ ۱۳۸۹/۱۰/۲۸

خیلی جالب بود! موفق باشید.

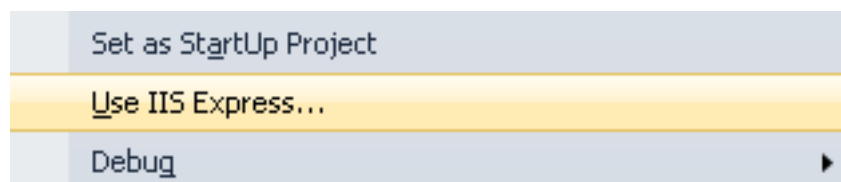
در مورد تنظیمات دستی IIS Express که یک نسخه‌ی سبک IIS 7.5 قابل اجرا بر روی ویندوز XP نیز می‌باشد، پیشتر در این سایت مطلبی را مطالعه کرده‌اید (+). اکنون که سرویس پک یک VS 2010 ارائه شده (+)، دیگر نیازی به آن تنظیمات دستی نبوده و امکان استفاده یکپارچه و خودکار از این نسخه‌ی ساده شده IIS 7.5 به شرح زیر وجود دارد:

ابتدا نیاز است تا هر دو مورد سرویس پک یک VS 2010 و همچنین [IIS Express](#) به صورت جداگانه نصب شوند. سپس:

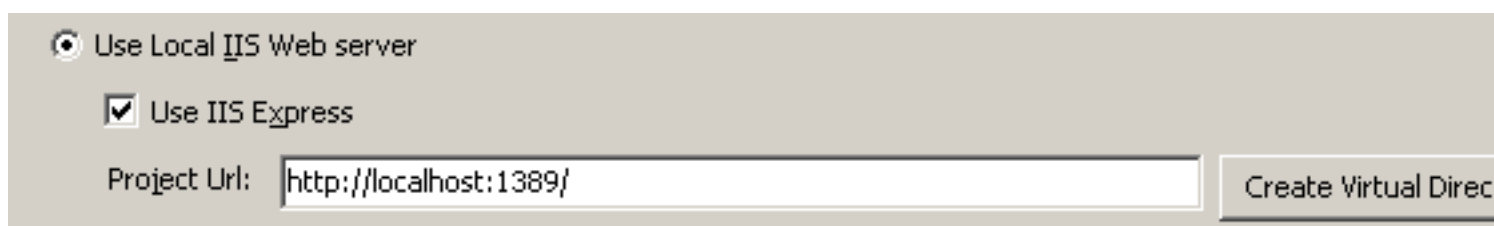
الف) ابتدا از منوی Tools ، گزینه‌ی Options را انتخاب کنید. در صفحه‌ی باز شده در قسمت Projects and solutions ذیل گزینه‌ی Web projects نیاز است تا یکبار مجوز استفاده از IIS express صادر شود:



ب) اکنون بر روی نام پروژه در Solution explorer موجود در Visual studio کلیک راست کرده و گزینه‌ی Use IIS Express را انتخاب نمایید:



به این صورت تنظیمات لازم به صورت خودکار اعمال خواهد گردید و جهت مشاهده آن‌ها می‌توان به خواص پروژه، برگه‌ی Web مراجعه کرد:



نکته مهم:

نسخه‌ی RTM و ویژوال استودیوی 2010 تنظیمات فوق را که در تصویر ملاحظه می‌کنید، ندارد. به عبارتی پس از اعمال تغییرات فوق باید دقت داشت سایرینی که قرار است از پروژه‌ی شما استفاده کنند نیز باید پیشنیازهای ذکر شده را رعایت نمایند و یا جهت توزیع سورس می‌توان مجدداً بر روی نام پروژه کلیک راست کرده و اینبار گزینه‌ی Use Visual Studio Development Server قدیمی را انتخاب کرد.

زمانیکه اولین نگارش ASP.NET حدود 10 سال قبل منتشر شد، تنها سیستم عاملی که از آن پشتیبانی می‌کرد، ویندوز سرور 2000 بود، تنها پروسه‌ای اجرایی آن aspnet_wp نام داشت و تنها معماری پشتیبانی شده هم X86 بود. به پروسه‌ای aspnet_wp محدودیت مصرف حافظه‌ای اعمال شده بود که در حین آغاز آن بر اساس مقدار قابل تغییر `processModel memoryLimit` محاسبه و اعمال می‌شد (تعریف شده در فایل ماشین کانفیگ). این عدد به صورت درصدی از ظرفیت RAM فیزیکی سیستم، قابل تعریف و به صورت پیش فرض به 60 درصد تنظیم شده بود. به این ترتیب این پروسه مجاز نبود تا تمام حافظه‌ی فیزیکی مهیا را مصرف کند و در صورت وجود نشستی حافظه‌ای در برنامه‌ای خاص، این پروسه امکان بازیابی مجدد حافظه را پیدا می‌کرد (recycling). همچنین یک مورد دیگر را هم باید در نظر داشت و آن هم وجود قابلیت است به نام ASP.NET Cache است که امکان ذخیره سازی مقادیر اشیاء را در حافظه‌ی مصرفی این پروسه مهیا می‌سازد. هر زمان که میزان این حافظه‌ی مصرفی به حد نزدیکی از محدودیت تعریف شده برسد، این پروسه به صورت خودکار شروع به حذف آن‌ها خواهد کرد.

محدودیت 60 درصدی تعریف شده، برای سیستم‌هایی با میزان RAM کم بسیار مفید بود اما در سیستم‌هایی با میزان RAM بیشتر، مثلاً 4 گیگ به 2.4GB حافظه مهیا (60 درصد حافظه فیزیکی سیستم) محدود می‌شد و همچنین باید در نظر داشت که میزان user mode virtual address space مهیا نیز تنها 2 گیگابایت بود. بنابراین هیچگاه استفاده مؤثری از تمام ظرفیت RAM مهیا صورت نمی‌گرفت و گاهی مشاهده می‌شد که یک برنامه تنها با مصرف 1.5GB RAM می‌توانست پیغام OutOfMemoryException را صادر کند. در این حالت مطابق بررسی‌های صورت گرفته مشخص شد که اگر مقدار `processModel memoryLimit` به حدود 800 مگابایت تنظیم شود، بهترین عملکرد را برای سیستم‌های مختلف می‌توان مشاهده کرد.

با ارائه‌ی ویندوز سرور 2003 و همچنین ارائه‌ی نسخه‌ی 1.1 دات نت فریم ورک و ASP.NET، این وضعیت تغییر کرد. پروسه‌ی جدید در اینجا w3wp نام دارد و این پروسه تعاریف مرتبط با محدودیت حافظه‌ی خود را از تنظیمات IIS دریافت می‌کند (قسمت Maximum Used Memory در برگه‌ی Recycling مربوط به خواص Application Pool مرتبط). متأسفانه این عدد به صورت پیش فرض محدودیتی ندارد و به ظاهر برنامه مجاز است تا حد امکان از حافظه‌ی مهیا استفاده کند. به همین جهت یکی از مواردی را که باید در نظر داشت، مقدار دهی Maximum Used Memory ذکر شده است. خصوصاً اینکه در نگارش 1.1، تنظیمات میزان مصرف RAM مرتبط با ASP.NET Cache نیز با برنامه یکی است.

در نگارش 2.0 دات نت فریم ورک، تنظیمات مرتبط با ASP.NET cache از تنظیمات میزان مصرفی یک برنامه‌ی ASP.NET جدا شد و این مورد توسط قسمت `cache privateBytesLimit` قابل تنظیم و مدیریت است (در فایل IIS Metabase و همچنین فایل web.config برنامه).

نکته!

اگر `process memory limit` و همچنین `cache memory limit` را تنظیم نکنید، باز به همان عدد 60 درصد سابق بازخواهیم گشت و این مورد به صورت خودکار توسط IIS محاسبه و اعمال می‌شود. البته محدودیت ذکر شده برای پروسه‌های 64 بیتی در این حالت بسیار بهتر خواهد بود. اگر هر دوی این‌ها را تنظیم کنید، عدد حداقل بکارگرفته شده، مبنای کار خواهد بود و اگر تنها یکی را تنظیم کنید، این عدد به هر دو حالت اعمال می‌گردد. برای بررسی بهتر می‌توان به مقدار `Cache.EffectivePrivateBytesLimit` و `Cache.EffectivePercentagePhysicalMemoryLimit` مراجعه کرد.

و ... اکنون بهتر می‌توانید به این سؤال پاسخ دهید که «سرور ما بیشتر از 4 گیگ رم دارد و برنامه‌ی ASP.NET من الان فقط 850 مگ رم مصرف کرده (که البته این هم نشانی از عدم dispose صحیح منابع است یا عدم تعیین تقدم و تاخر و زمان منقضی شدن، حین تعریف اشیاء کش)، اما پیغام out of memory exception را دریافت می‌کنم. چرا؟!»

بنابراین ایجاد یک [Application pool](#) جدید به ازای هر برنامه‌ی ASP.NET امری است بسیار مهم زیرا:

- به این ترتیب هر برنامه‌ی ASP.NET در پروسه‌ای ایزوله از پروسه‌ی دیگر اجرا خواهد شد (این مساله از لحاظ امنیتی هم بسیار مهم است). در اینجا هر برنامه، از پروسه‌ی w3wp.exe مجزای خاص خود استفاده خواهد کرد (شبيه به مرورگرهایی که هر tab را در یک پروسه جدید اجرا می‌کنند).
- اگر پروسه‌ای به حد بالای مصرف حافظه‌ی خود رسید با تنظیمات انجام شده در قسمت recycling مرتبط با Application pool اختصاصی آن، به صورت خودکار کار بازیابی حافظه صورت می‌گیرد و این امر بر روی سایر برنامه‌ها تاثیر نخواهد داشت (کاربران سایر برنامه‌ها مدام شکایت نمی‌کنند که سشن‌ها پرید. کش خالی شد. زیرا در حالت وجود application pool اختصاصی به ازای هر برنامه، مدیریت حافظه برنامه‌ها از هم ایزوله خواهند بود)
- کرش صورت گرفته در یک برنامه به دلیل عدم مدیریت خطاها، بر روی سایر برنامه‌ها تاثیر منفی نخواهد گذاشت. (زمانیکه ASP.NET worker process به دلیل استثنایی مدیریت نشده خاتمه یابد بلافاصله و به صورت خودکار مجدداً «وهله‌ی دیگری» از آن شروع به کار خواهد کرد؛ یعنی تمام سشن‌های قبلی از بین خواهند رفت؛ که در صورت ایزوله سازی ذکر شده، سایر برنامه‌ها در امان خواهند ماند؛ چون در پروسه ایزوله‌ی خود مشغول به کار هستند)
- با وجود application pool اختصاصی به ازای هر برنامه، می‌توان برای سایت‌های کم ترافیک و پرترافیک، زمان‌های recycling متفاوتی را اعمال کرد. به این ترتیب مدیریت حافظه‌ی بهتری قابل پیاده سازی می‌باشد. همچنین در این حالت می‌توان مشخص کرد کدام سایت از تعداد worker process بیشتر یا کمتری استفاده کند.
- کاربری که پروسه‌ی ASP.NET تحت آن اجرا می‌شود نیز همینجا تعریف می‌گردد. بنابراین به این ترتیب می‌توان به برنامه‌ای دسترسی بیشتر و یا کمتر داد، بدون تاثیر گذاری بر روی سایر برنامه‌های موجود.

نتیجه گیری:

- از IIS استفاده می‌کنید؟ آیا می‌دانید Application pool چیست؟
- آیا می‌دانید در صورت عدم مقدار دهی پارامترهای حافظه‌ی یک Application pool، به صورت پیش فرض چند درصد از حافظه‌ی فیزیکی مهیا در اختیار شما است؟

برای مطالعه بیشتر:

[CLR processModel memoryLimit](#)

[Some history on the ASP.NET cache memory limits](#)

[Managing Application Pools in IIS 7](#)

نظرات خوانندگان

نویسنده: محسن
تاریخ: ۱۳۹۰/۰۵/۰۵ ۲۳:۵۹:۱۰

مقاله ی مفیدی نوشتید. جای خالی اینجور مقالات فارسی توی اینترنت احساس میشه. خسته نباشید. دست شما درد نکنه.

نویسنده: Rab Raby
تاریخ: ۱۳۹۰/۰۵/۰۶ ۱۵:۱۹:۳۰

بسیار مهم و مفید بود مثل همیشه .

نویسنده: Amin
تاریخ: ۱۳۹۰/۰۵/۱۰ ۱۰:۰۸:۳۸

سلام آقای نصیری
ممنون از مطلب مفیدتون.
یه سوال: اگر خود این AppPool ها از لحاظ حافظه و CPU به حالتی برسند که بشه گفت به سقف چسبیدن، روشی برای رفع این مشکل وجود دارد؟ ما الان یه چنین مشکلی داریم. من مسئول این کار نیستم و زیاد در جریانش نیستم اما چون این مشکل رو دیدم می خواستم بدونم چه طور میشه این مشکل رو حل کرد.

ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۱۰ ۱۰:۴۵:۰۸

- در مورد بررسی علت بالا بودن CPU Usage اینجا توضیح دادم و روش دیباگ ذکر شده. به این ترتیب می‌تونید نام متدهای مشکل ساز رو دقیقاً پیدا کنید : [\(+\)](#)
- ضمناً یکی از تنظیمات App pool ، مرتبط است با تعیین دقیقاً cpu limit مورد استفاده: [\(+\)](#) البته این تنظیمات مرتبط است به IIS 7 ولی در IIS 6 هم وجود دارد و فرقی نمی‌کند. یعنی به صورت خلاصه می‌تونید تعیین کنید که به سقف نرسند. (در مورد تنظیمات حافظه هم به همین صورت)

نویسنده: Amin
تاریخ: ۱۳۹۰/۰۵/۱۱ ۰۸:۵۵:۵۹

ممنون از راهنماییتون.

نویسنده: Sniper_528
تاریخ: ۱۳۹۰/۰۵/۱۳ ۲۲:۰۸:۰۸

کاش برای SQL Server هم چیزی مثل recycling وجود داشت یعنی هر وقت میزان استفاده اون از RAM به یه حدی می رسید recycle میشد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۱۴ ۰۰:۵۷:۰۳

نه. این خوب نیست؛ چون کش اس کیوال سرور execution plan های زیادی داخل هست و خیلی مسایل دیگه (یعنی این مصرف صحیح حافظه هست نه نشتی حافظه).

در کل می‌شود برای اس کیوال سرور محدودیت حافظه گذاشت؛ در موردش قبلاً مطلب نوشتم در سایت هست : [\(+\)](#)
ضمناً یک سری دستور برای خالی کردن این کش‌ها هم هست: [\(+\)](#) ؛ ولی باز هم توصیه نمی‌شود چون این‌ها نشتی حافظه نیست.

نویسنده: Sniper_528
تاریخ: ۱۳۹۰/۰۵/۱۷ ۰۵:۱۳:۲۹

ممنون

من در سرورم با رم 2 گیگ، IIS، DNS Server و SQL Server رو با هم دارم max memory رو چی پیشنهاد می کنید برای اینها؟
سایت هم بازدید روزانه حدود 400 تا رو داره.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۱۷ ۰۸:۱۳:۳۸

GB 1.2

نویسنده: Sniper_528
تاریخ: ۱۳۹۰/۰۵/۱۷ ۱۴:۱۸:۲۳

در حال حاضر 1.4 گیگ از رم اشغاله که 200 مگ مربوط به اس کیو ال میشه
پس گزاشتمش رو 500 مگ

نویسنده: Nima
تاریخ: ۱۳۹۰/۰۵/۲۲ ۱۳:۱۱:۰۰

با سلام آقای نصیری

در مورد سشن ها چطور؟ آیا محدودیتی برای حجم سشن ها هم هست؟ آیا این محدودیت قابل برداشتن هست؟ فضای سشن ها رو IIS مدیریت میکنه یا Asp.Net ؟ اگر مقدار حافظه مورد نیاز سشن زیاد باشه چه اتفاقی میفته؟ با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۲۲ ۱۳:۳۸:۴۴

بستگی داره Session state به چه صورتی تنظیم شده باشد. می شود آن را طوری تنظیم کرد که در اس کیوال سرور هم حتی ذخیره شود. حالت InProc آن یعنی همان توضیحات فوق و تمام تنظیمات app pool به آن اعمال می شود. اطلاعات بیشتر:

[Session State](#)

نویسنده: میلاد حسینی
تاریخ: ۱۳۹۱/۰۶/۲۱ ۱۳:۲۵

با سلام؛

مشکلی که من دارم نمیدانم مربوط میشود به مدیریت حافظه یا موضوعی دیگر
من یک وب سایت کوچک دارم که با تکنولوژی های زیر ایجاد شده:

ASP.Net MVC 4 , Entity Framework 4 , SQL CE

آن را بر روی یک ویندوز سرور 2012 نسخه دیتاستر نصب کردم

سرور : 2GB Ram و CPU Dual Core 1.8

غیر از این سایت هیچ سایت دیگری بر روی این سرور میزبانی نشده است.

صفحات با سرعت نسبتاً خوبی باز میشوند، اما به هر شکلی iis را تنظیم میکنم، اگر پس از 2 یا 3 دقیقه درخواستی به سمت سرور ارسال نگردد، برنامه از حافظه خارج میشود. اگر درخواستی برای مشاهده صفحه به سرور ارسال شود 15 تا 20 ثانیه طول میکشد تا دوباره کامپایل انجام شود و صفحه درخواستی نمایش یابد.

تصویر تنظیمات Application Pool

General	
.NET Framework Version	v4.0
Enable 32-Bit Applications	False
Managed Pipeline Mode	Integrated
Name	Hand
Queue Length	1000
Start Automatically	True
Start Mode	OnDemand
CPU	
Limit (1/1000 of %)	1000
Limit Action	NoAction
Limit Interval (minutes)	5
Processor Affinity Enabled	True
Processor Affinity Mask	4294967295
Processor Affinity Mask (64-bit option)	4294967295
Process Model	
Generate Process Model Event Log Entry	
Idle Time-out Reached	False
Identity	LocalSystem
Idle Time-out (minutes)	0
Load User Profile	False
Maximum Worker Processes	4
Ping Enabled	True
Ping Maximum Response Time (seconds)	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90
Process Orphaning	
Enabled	False
Executable	
Executable Parameters	
Rapid-Fail Protection	
"Service Unavailable" Response Type	HttpLevel
Enabled	True
Failure Interval (minutes)	5
Maximum Failures	5
Shutdown Executable	
Shutdown Executable Parameters	
Recycling	
Disable Overlapped Recycle	False
Disable Recycling for Configuration Change:	False
Generate Recycle Event Log Entry	
Private Memory Limit (KB)	0
Regular Time Interval (minutes)	0
Request Limit	0
Specific Times	
Virtual Memory Limit (KB)	TimeSpan[] Array
	0

پ.ن: لطفاً اگر امکان دارد بهترین تنظیمات را برای سروری که فقط به یک سایت می‌خواهد سرویس دهد عنوان کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳:۵۷ ۱۳۹۱/۰۶/۲۱

به فرض اینکه تنظیمات specific times فوق که در اینجا مشخص نیست صحیح است (مثلاً تنظیم شده به 2 بامداد)، [این مطلب](#) بیشتر مرتبط است به کار شما.

نویسنده: داود
تاریخ: ۱۳:۱۹ ۱۳۹۲/۰۸/۱۴

سلام؛ ما به سایت داریم که در روز حدود 1000 تا کاربر دارد. در قسمت admin، آپلود فایل هم زیاد داریم.
RAM وب سرور هم 8GB هست.

WinServer 2008 32bit - CPU: Xeon 5160 3GHz

IIS 7

تقریباً روزی یکی دوبار شاید هم هر دو سه روز به بار نیاز به recycle داشته باشیم.

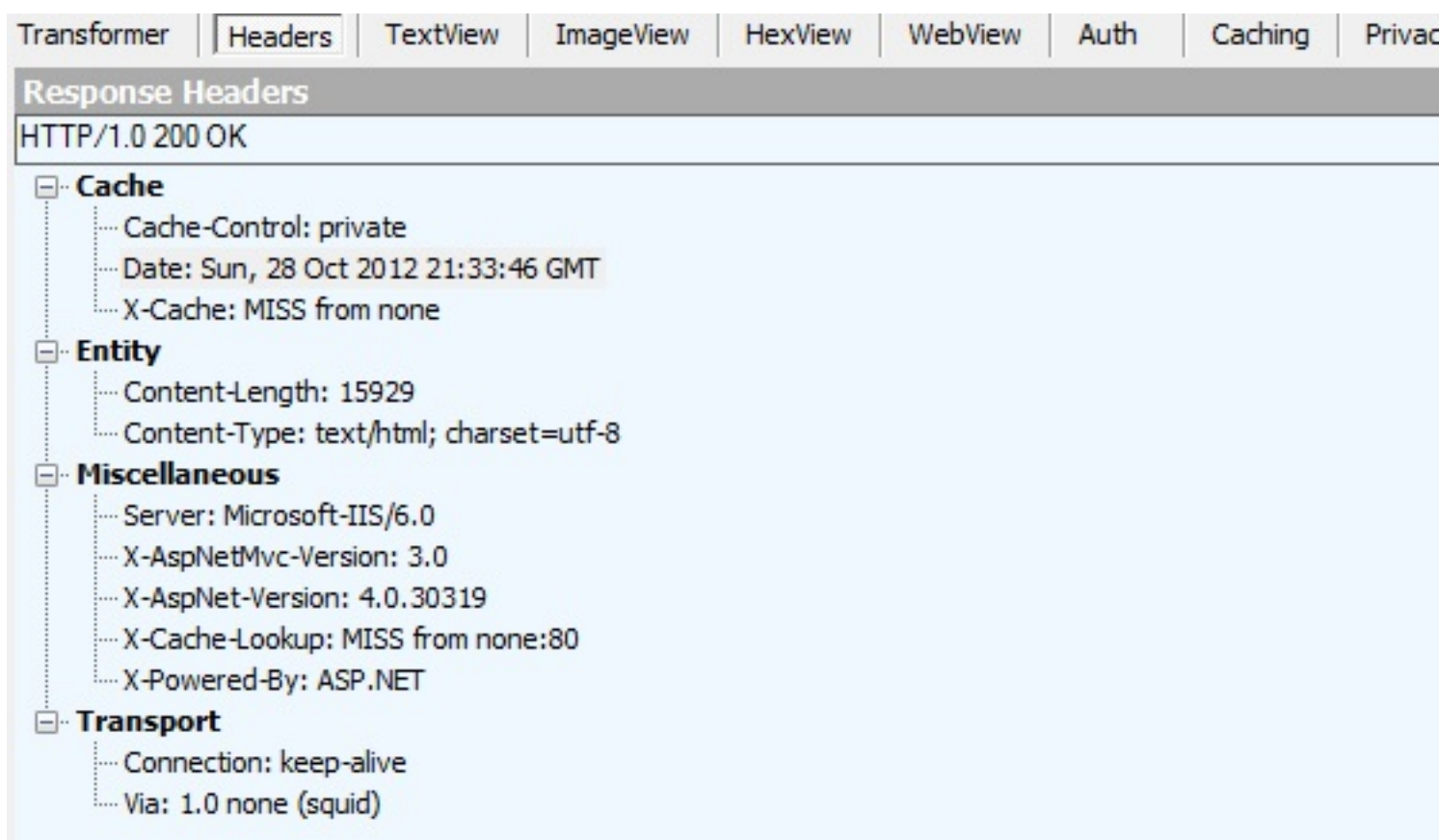
پیشنهاد می‌کنید Virtual Memory Usage و Private Memory Usage چند باشد تا کمترین نیاز رو برای recycle داشته باشیم؟

نویسنده: وحید نصیری
تاریخ: ۱۳:۵۲ ۱۳۹۲/۰۸/۱۴

- سرور 32 بیتی نمی‌تونه از حداکثر میزان RAM سرور شما (بیشتر از 2GB) نهایت استفاده رو انجام بده. [تمهیداتی](#) هم در این زمینه هست ولی ... بهتره به یک سرور 64 بیتی کوچ کنید. بدون *این تمهیدات*، میزان حافظه مهابی جهت یک پروسه 32 بیتی به اندازه address space آن یعنی 2GB محدود است.
- همچنین باید [کش کردن](#) اطلاعات رو فعال کنید و اجازه بدید IIS بجای برنامه این مسایل رو راساً مدیریت کنه؛ یا از یک کش سرور مجزا استفاده کنید.

در تکمیل [این مطلب](#) برای حذف هدرهای مربوط به وب سرور در برنامه‌های ASP.NET MVC از روش زیر می‌توانیم استفاده کنیم.

در حالت پیش فرض تمام پاسخهای که به سمت سرور ارسال میشوند به همراه خود یک سری جزئیات را نیز منتقل میکنند.



برای یک وب اپلیکیشن APS.NET MVC این هدرها را داریم :

Server: که توسط IIS اضافه میشود.

X-AspNet-Version: که در زمان Flush در httpresponse اضافه میشود.

X-AspNetMvc-Version: که توسط MvcHandler در System.Web.dll اضافه میشود.

X-Powered-By: این مورد نیز توسط IIS اضافه میشود.

هکرها از اینکه فریم ورک مورد استفاده چه چیزی است خوشحال خواهند شد: اگر سرور شما برای مدتی Update نشده باشد و یک آسیب پذیری امنیتی بزرگ برای ورژن فریم ورکی که استفاده می‌کنید پیدا شود در نتیجه به هکرها برای رسیدن به هدفشان

کمک کرده اید.

به علاوه این هدرها فضایی را برای تمام پاسخها در نظر میگیرند (البته در حد چندین بایت ولی در اینجا بحث برروی Optimization است).

برای حذف این هدرها باید مراحل زیر را انجام دهیم:

حذف کردن هدر Server : به Global.asax.cs رفته و رویداد Application_PreSendRequestHeaders با کد زیر را به آن اضافه کنید :

```
protected void Application_PreSendRequestHeaders(object sender, EventArgs e)
{
    var app = sender as HttpApplication;
    if (app == null || !app.Request.IsLocal || app.Context == null)
        return;
    var headers = app.Context.Response.Headers;
    headers.Remove("Server");
}
```

حذف کردن هدر X-AspNetMvc-Version : در فایل Global.asax.cs به رویداد Application_Start این کد زیر را اضافه کنید :

```
protected void Application_Start()
{
    ...
    MvcHandler.DisableMvcResponseHeader = true;
    ...
}
```

حذف کردن هدر X-AspNet-Version : به فایل Web.Config مراجعه کرده و این المنت را در داخل system.web اضافه کنید:

```
<system.web>
    ...
    <httpRuntime enableVersionHeader="false" />
    ...
</system.web>
```

حذف کردن هدر X-Powered-By : در داخل فایل Web.Config در داخل system.webServer این خطوط را اضافه کنید:

```
<system.webServer>
    ...
    <httpProtocol>
        <customHeaders>
            <remove name="X-Powered-By" />
        </customHeaders>
    </httpProtocol>
    ...
</system.webServer>
```

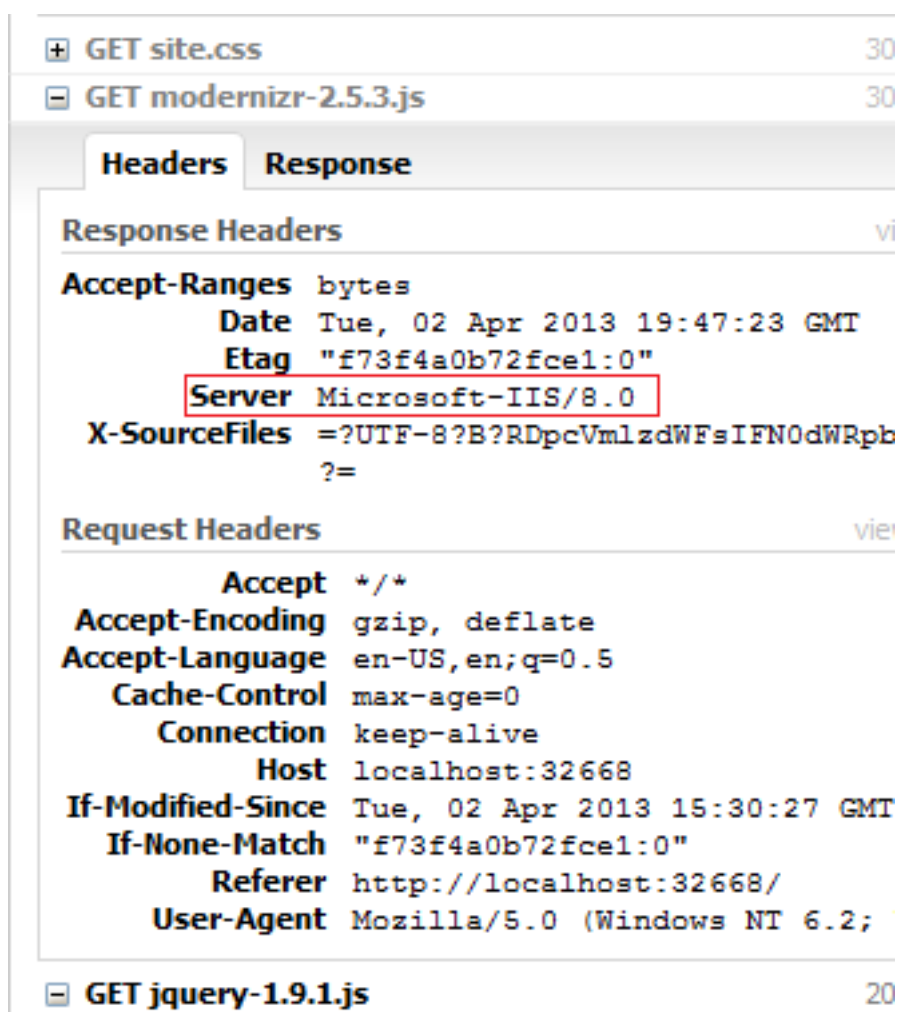
با انجام مراحل فوق پاسخهای سرور سبکتر شده و در نهایت حاوی اطلاعات مهم در مورد ورژن فریم ورک نمیباشد.

نظرات خوانندگان

نویسنده: sysman

تاریخ: ۰۲۲ ۱۳۹۲/۰۱/۱۴

ممون از این مطلب مفید ولی من یک مشکلی دارم کاری که گفتید را انجام دادم و همه چیز خوب انجام شد ولی در بخش GET modernizr-2.5.3.js باز هم اطلاعات سرور رو به من میدهد



نویسنده: علی

تاریخ: ۰۳۶ ۱۳۹۲/۰۱/۱۴

فایل‌های استاتیک رو هم باید به موتور ASP.NET مپ کنید. تا زمانیکه مپ نباشند مستقیماً توسط IIS سرو می‌شن و تنظیمات ASP.NET روی اون‌ها تاثیری نداره.

نویسنده: sysman

تاریخ: ۱۰:۴۰ ۱۳۹۲/۰۱/۱۴

نمی‌دونم این کاری که گفتین رو دقیقاً چطور انجام بدهم ولی این کار باعث ایجاد کندی نمی‌شه؟ اگر در خود IIS تنظیمات [این](#) [پست](#) رو اعمال کنم به نتیجه میرسم؟

نویسنده: وحید نصیری
تاریخ: ۱۱:۱۳ ۱۳۹۲/۰۱/۱۴

- برای فایل‌های جاوا اسکریپت توصیه من این است:
الف) اگر از Web forms استفاده می‌کنید: استفاده از Script manager ([^](#) و [^](#))
ب) اگر از MVC استفاده می‌کنید: استفاده از [Bundling & minification](#)
در هر دو حالت نحوه ارائه اسکریپت‌ها تحت کنترل برنامه ASP.NET در خواهد آمد و مستقیماً و بدون دخالت ASP.NET، توسط IIS توزیع نمی‌شوند.
- برای مپ کردن فایل‌های استاتیک به موتور ASP.NET می‌شود از StaticFileHandler استفاده کرد. اگر کش کردن اطلاعات استاتیک در سمت سرور فعال شود، این مساله بار اضافه‌ای را به سرور تحمیل نخواهد کرد.

```
<system.web>
<httpHandlers>
  <add path="*.js" verb="*" type="System.Web.StaticFileHandler" />
</httpHandlers>
```

نویسنده: یونس دوست
تاریخ: ۱۱:۳۴ ۱۳۹۲/۰۶/۰۲

سلام. من ازش استفاده کردم ولی موقع حذف Header مربوط به سرور ارور زیر رو می‌ده:
This operation requires IIS integrated pipeline mode.
ضمناً من از iis 7.5 استفاده می‌کنم و توی Application Pools هم Classic .Net AppPool رو در حالت pipeline mode: Integrated قرار دادم هم DefaultAppPool ولی همچنان ارور رو دارم. مشکل از کجاست؟
ممنون.

نویسنده: وحید نصیری
تاریخ: ۱۱:۵۴ ۱۳۹۲/۰۶/۰۲

- محل تست کردن واقعی این کدها بر روی ویندوز سرور است و نه دیباگر و وب سرور آزمایشی ویژوال استودیو.
- شما اگر تا این حد دسترسی به IIS دارید، اصلاً نیازی به کدنویسی برای حذف هدرهای وب سرور نخواهید داشت. به قسمت [HTTP response headers](#) کنسول مدیریتی مراجعه و مداخل موجود را حذف یا ویرایش کنید.
+ حذف هدر مربوط به نام Server کار ساده‌ای نیست. خیلی‌ها از روش HTTP module هم جواب نگرفته‌اند، اما با استفاده از [URL Scan](#) خود مایکروسافت قابل حذف است (این برنامه روی ویندوزهای سرور 2003 به بعد قابل نصب است). بعد از نصب به فایل `C:\Windows\System32\inetssrv\urlscan\UrlScan.ini` مراجعه و `RemoveServerHeader` را با 1 مقدار دهی کنید. ضمناً قبل از نصب URLScan تغییر زیر را هم امتحان کنید (بجای Remove از Set استفاده شده):

```
void OnPreSendRequestHeaders(object sender, EventArgs e)
{
    HttpContext.Current.Response.Headers.Set("Server", "CERN httpd");
}
```

نویسنده: پیام دات نت
تاریخ: ۲۰:۳۵ ۱۳۹۳/۰۳/۱۵

با سلام؛ ممنون از مطلب خوبتون. من تو IIS 8 Local اجرا کردم درست کار میکنه و بسیار عالیه. اما تو اجرای نهایی روی سرور هدر Server حذف نشد و از سمت سرور ارسال میشد. [بعد از کلی جستجو](#) به نتیجه زیر رسیدم.

```
<system.webServer>
  <rewrite>
    <outboundRules>
      <rule name="Remove RESPONSE_Server" >
        <match serverVariable="RESPONSE_Server" pattern=".*" />
        <action type="Rewrite" value="" />
      </rule>
    </outboundRules>
  </rewrite>
</system.webServer>
```

نویسنده: دانش پژوه
تاریخ: ۱۱:۳۹ ۱۳۹۳/۱۰/۰۲

ممنون از مطلبتون

فقط یک نکته رو میخوام بگم نام کوکی سشن بطور پیشفرض ASP.NET_SessionId هست و میشه از این نام فهمید که سرور asp.net هستش. از طریق کد زیر توی وب کانفیگ میتونید تغییرش بدید:

```
<system.web>  
  <sessionState cookieName="foo" />  
</system.web>
```

همان طور که می‌دانید در css امکان استفاده از فونت‌های فارسی مهیاست. برای این کار کافیت با دستور زیر فونت را در فایل css خود تعریف کنیم و در صورتیکه فونت روی سیستم کاربر موجود نباشد ابتدا فونت روی سیستم دانلود شده و سپس نمایش داده می‌شود. استفاده از سه پسوند مختلف نیز برای مرورگرهای مختلف در نظر گرفته شده است تا خروجی در تمامی مرورگرها به درستی نمایش داده شود.

```
@font-face {
    src: url('Font/BYekan.eot?#') format('eot'), /* IE6-8 */
         url('Font/BYekan.woff') format('woff'), /* FF3.6+, IE9, Chrome6+, Saf5.1+ */
         url('Font/BYekan.ttf') format('truetype'); /* Saf3-5, Chrome4+, FF3.5, Opera 10+ */
}
```

در یک پروژه با مشکل عجیبی روبرو شدم. پروژه روی لوکال به خوبی کار می‌کرد اما بعد از آپلود روی سرور فونت‌ها بسیار دیر لود می‌شد. یعنی ابتدا تمامی اجزای صفحه لود شده و با تاخیری چند ثانیه ای فونت نمایش داده می‌شد. مرورگری که در حال کار روی آن برای تست پروژه بودم باید از پسوند woff مربوط به آن فونت استفاده می‌کرد.

بعد از تست پروژه با فایر باگ متوجه شدم که متاسفانه اصلا فونت روی سیستم دانلود نشده و خروجی 404 که مربوط به File Not Found است برگشت داده می‌شود. سعی کردم با دادن آدرس فونت در برنامه download manager فونت را دانلود کنم. اما باز هم فونت دانلود نمی‌شد! (یعنی فایل روی سرور موجود بود ولی هیچ دسترسی به آن ممکن نبود) برای دو پسوند دیگر یعنی eot و ttf هم این کار را تست کردم و متوجه شدم روی این دو پسوند مشکل خاصی وجود ندارد. با تماس با پشتیبانی هاست متوجه شدم اصلا این پسوند در Mime Type های سرور تعریف نشده است. بنابراین به صورت دستی Mime Type مورد نظر را روی سرور تعریف کردم و مشکل حل شد:

Home Folder	Virtual Dirs	Secured Folders	Extensions	Custom Errors	Headers	Web Publishing	MIME Types
Add MIME							
Extension		MIME-Type					
.woff		application/x-font-woff					
Update		Cancel		Delete			

همچنین لیست کامل Mime Types ها در آدرس زیر موجود است:

[لیست کامل Mime Types](#)

نظرات خوانندگان

نویسنده: آموزش مجازی
تاریخ: ۱۵:۲۷ ۱۳۹۱/۱۲/۲۹

شما اگر این تنظیمات رو توی پروژه خودت انجام بدی ، احتیاجی نیست برای IIS حتما MimeType تعریف کنی. البته فقط قسمتهایی از این تنظیمات در مورد مساله ای که گفتی لازم هست اما من برای ایده گرفتن در مورد تنظیمات IIS 7 و به بالا در web.config که کلا مهجور مونده ، تنظیماتی که خودم معمولا می‌ذارم رو گذاشتم اینجا.

```
<system.webServer>
  <urlCompression doStaticCompression="true" doDynamicCompression="true" />
  <staticContent>
    <remove fileExtension=".js"/>
    <mimeTypeMap fileExtension=".js" mimeType="text/javascript" />
    <mimeTypeMap fileExtension=".mp4" mimeType="video/mp4" />
    <mimeTypeMap fileExtension=".m4v" mimeType="video/m4v" />
    <mimeTypeMap fileExtension=".ogg" mimeType="video/ogg" />
    <mimeTypeMap fileExtension=".ogv" mimeType="video/ogg" />
    <mimeTypeMap fileExtension=".webm" mimeType="video/webm" />
    <mimeTypeMap fileExtension=".oga" mimeType="audio/ogg" />
    <mimeTypeMap fileExtension=".spx" mimeType="audio/ogg" />
    <mimeTypeMap fileExtension=".svg" mimeType="images/svg+xml" />
    <mimeTypeMap fileExtension=".svgz" mimeType="images/svg+xml" />
    <mimeTypeMap fileExtension=".eot" mimeType="application/vnd.ms-fontobject" />
    <mimeTypeMap fileExtension=".otf" mimeType="font/otf" />
    <mimeTypeMap fileExtension=".woff" mimeType="font/x-woff" />
  </staticContent>
```

نویسنده: آرمان فرقانی
تاریخ: ۱۶:۳۹ ۱۳۹۱/۱۲/۲۹

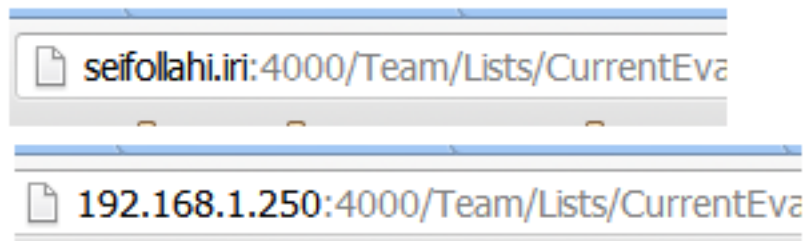
البته با توجه به پردازش سلسه مراتبی فایل‌های config در وب سرور احتمال دارد در برخی میزبان‌های اشتراکی برخی از قابلیت‌ها توسط مدیر سرور Lock شود.

نویسنده: ناصر فرجی
تاریخ: ۱۵:۴۱ ۱۳۹۲/۰۱/۰۱

بله. تمامی تنظیمات iis رو به نوعی میشه از طریق web.config هم انجام داد ولی برای اعمال سراسری تنظیمات بهتره از iis استفاده بشه.

در صورتی که نیاز به مشاهده یک سایت شیرپوینت خارج از یک دامین باشد باید تنظیمات مسیر یابی آن فعال باشد تا بتوان آن را بدون نمایش خطا مشاهده کرد . در این پست کوتاه تنظیمات مرتبط به Alternate Access Mappings یا AAM بیان می‌شوند .

در صورتی که هنگام ایجاد application مسیر دامنه را برای آن مشخص کرده باشید و نیاز داشته باشید از خارج از دامنه نیز آن را مشاهده کنید ممکن است در مواردی به مشکل برخورد کنید . مثلاً با مشاهده لیست به خطای زیر برخورد کنید :



Server Error in '/' Application.

nts

Runtime Error

Description: An application error occurred on the server. The current custom error settings for this application prevent it from displaying the details of the error.

Details: To enable the details of this specific error message to be viewable on the local server machine, please create a custom error page. This <customErrors> tag should then have its "mode" attribute set to "RemoteOnly". To enable the details to be viewable on the local server machine, please create a custom error page. This <customErrors> tag should then have its "mode" attribute set to "RemoteOnly". To enable the details to be viewable on the local server machine, please create a custom error page. This <customErrors> tag should then have its "mode" attribute set to "RemoteOnly".

```
<!-- Web.Config Configuration File -->
<configuration>
  <system.web>
    <customErrors mode="RemoteOnly"/>
  </system.web>
</configuration>
```

Notes: The current error page you are seeing can be replaced by a custom error page by modifying the "defaultRedirect" attribute of the <customErrors> tag to point to the path of the custom error page. For example, to replace the current error page with a custom error page located at "mycustompage.htm", the <customErrors> tag should be configured as follows:

```
<!-- Web.Config Configuration File -->
<configuration>
  <system.web>
    <customErrors mode="On" defaultRedirect="mycustompage.htm"/>
  </system.web>
</configuration>
```

برای رفع این مشکل باید مسیر نامعتبر را برای Application به عنوان یک مسیر جایگزین تعریف کرد .

یک راه تعریف AAM برای Application می باشد . برای این منظور روی Configure alternate access mappings در بخش مدیریت مرکزی شیرپوینت (در قسمت های مشخص شده در تصویر زیر) بروید :

/_admin/AlternateUrlCollections.aspx

Central Administration ▶ Application Management



Web Applications

Manage web applications |

[Configure alternate access mappings](#)



Site Collections

Create site collections |

Delete a site collection |

Confirm site use and deletion |

Spe

Configure the internal and public URL mapping for web applications in the farm

Central Administration ▶ System Settings



Servers

Manage servers in this farm |

Manage services on server



E-Mail and Text Messages (SMS)

Configure outgoing e-mail settings |

Configure incoming e-mail settings |

Co



Farm Management

[Configure alternate access mappings](#) |

Manage farm features |

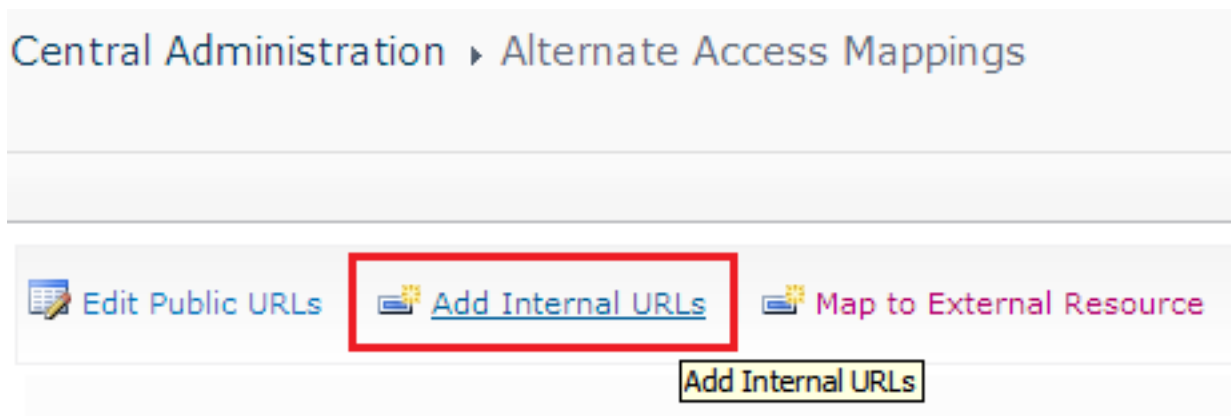
Manage farm

Configure privacy options |

Configure cross-firewall access zone

Configure the internal and public URL mapping for web applications in the farm

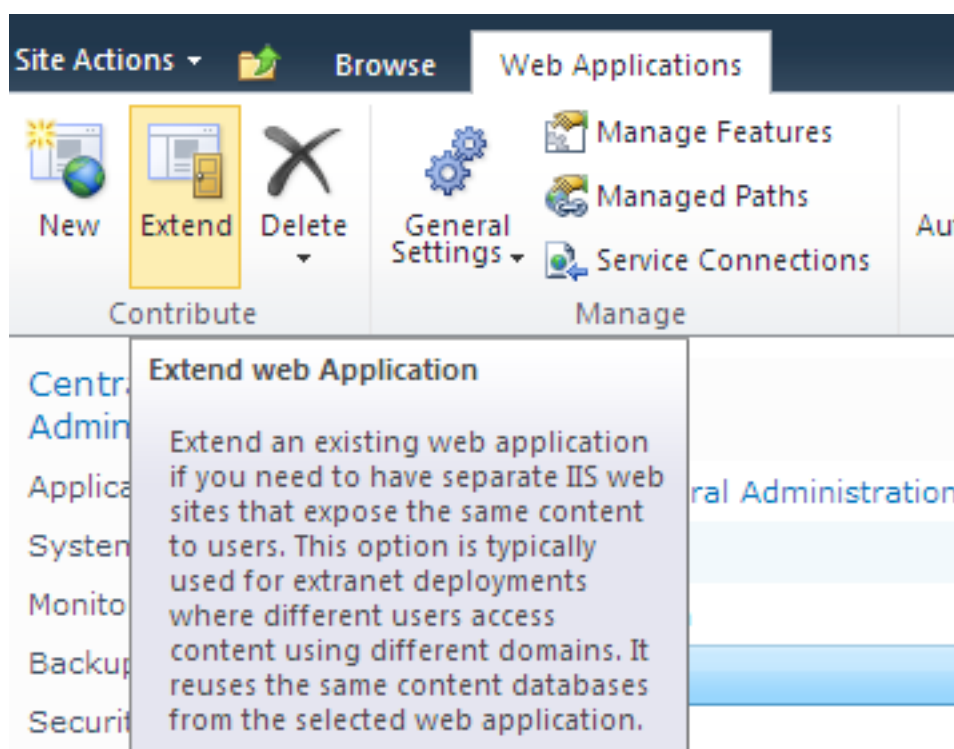
سپس روی Add Internal Urls کلیک کنید و در فیلد Url مسیر آدرس را با IP و پورت آن وارد کنید (در صورت وجود) و zone آن را روی Internet قرار دهید. (باید SiteCollection را نیز در همین بخش مشخص کنید)



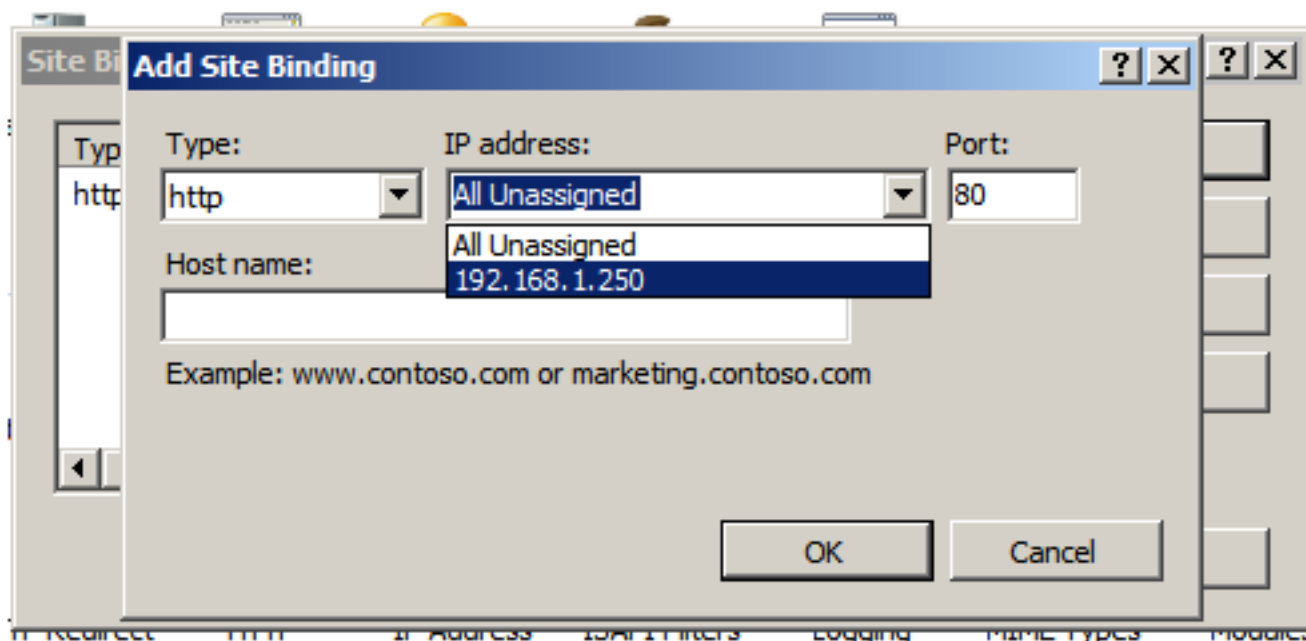
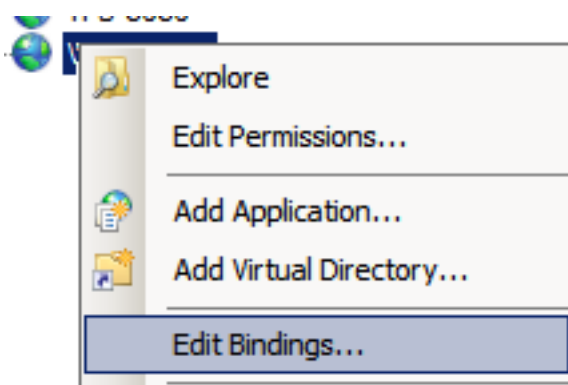
البته این تنظیم از بخش Edit Public Urls نیز با کمی تغییر قابل انجام می‌باشد .

پس از اعمال تنظیمات مشکل عدم نمایش محتویات رفع خواهد شد .

راه حل دیگر اعمال یک Extend برای Application جاری است .



در صورتی که از روش دوم یعنی Extend این کار را انجام دهید برای شما یک Application در زیر مجموعه سایت‌های IIS ایجاد می‌شود که ممکن است باز هم همان مشکل رخ دهد . برای رفع آن می‌توان Binding آن را ویرایش کرد و آی پی مورد نظر را به آن Assign کرد



[موفق باشید](#)

عنوان: کدام w3wp.exe مرتبط با Application جاری من است ؟

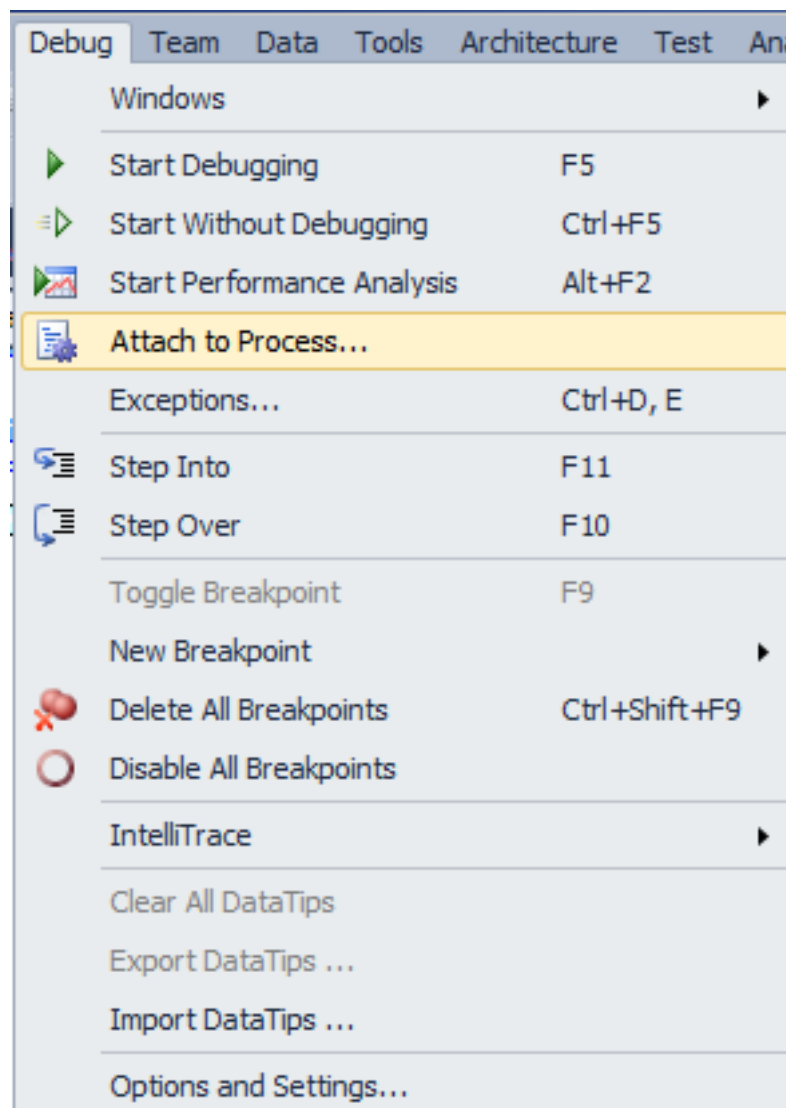
نویسنده: محمد باقر سیف الهی

تاریخ: ۱۵:۱۰ ۱۳۹۲/۰۱/۲۳

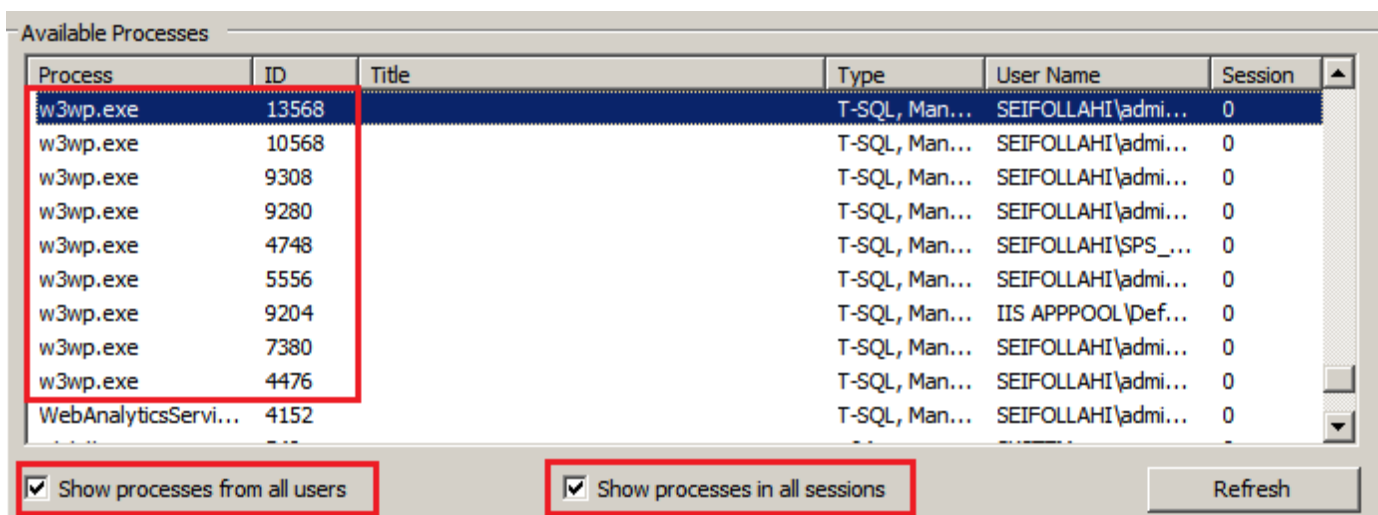
آدرس: www.dotnettips.info

برچسب‌ها: Debugging, IIS, Visual Studio, Visual Studio 2012

مواردی وجود دارد که نیاز به Attach کردن یک پروسس به Application خود دارید. برای این منظور باید از بین w3wp‌های موجود که IIS اجرا کرده پردازش مرتبط را یافته و آن را Attach نمایید در غیر این صورت امکان debug کردن Application مشکل خواهد بود. در این پست راه حلی برای این مورد بیان شده است .
فرض کنید می‌خواهید از بین تعدادی پروسس یکی را برای debug کردن انتخاب نمایید .
(برای انتخاب پروسس از منوی Debug روی Attach to Process کلیک کنید و تیک نمایش تمام پروسس‌ها را بزنید)



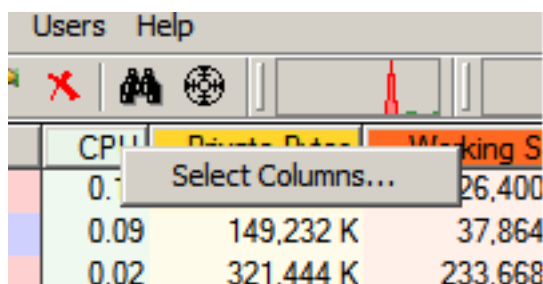
کدام w3wp.exe مرتبط با Application جاری من است ؟



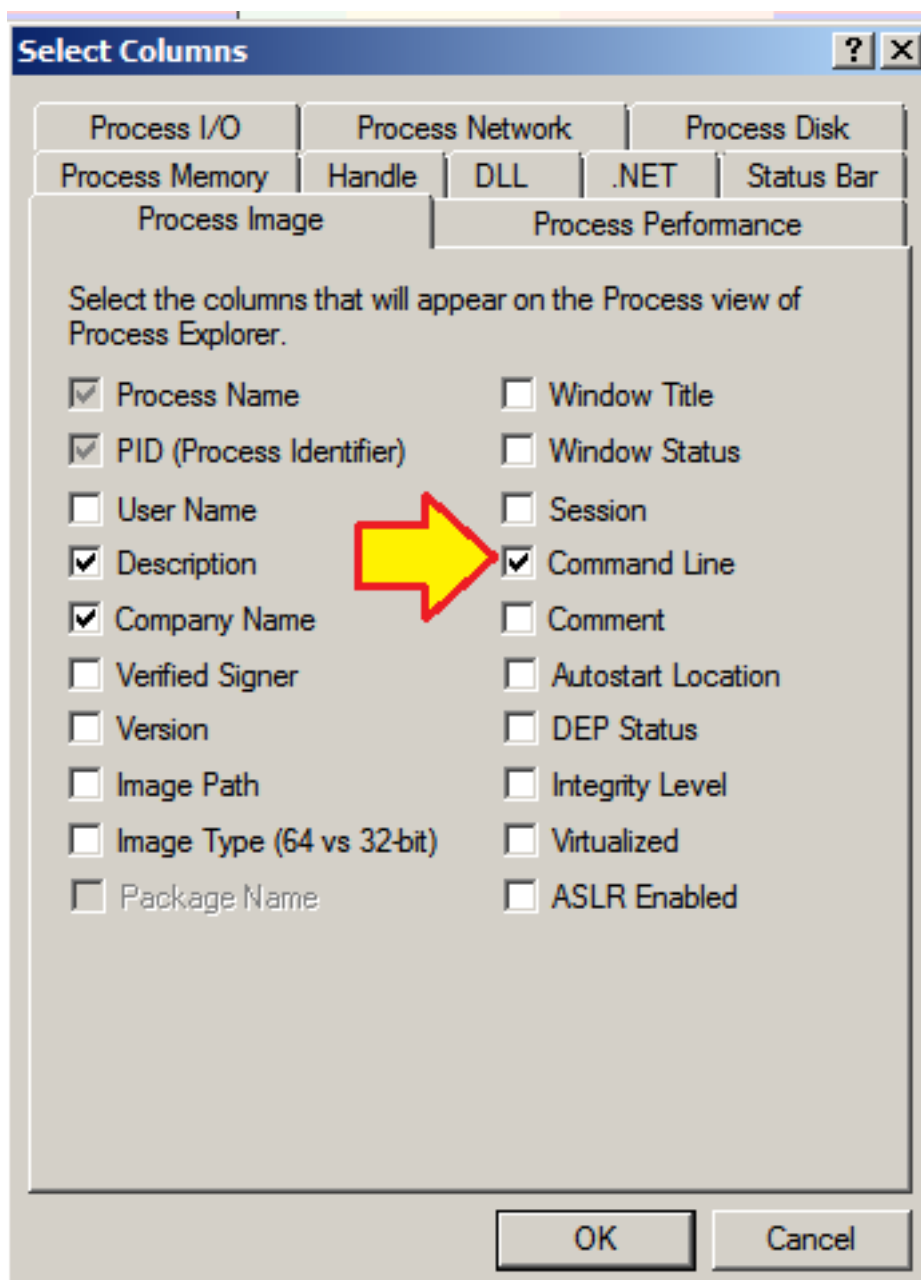
Process	ID	Title	Type	User Name	Session
w3wp.exe	13568		T-SQL, Man...	SEIFOLLAHI\admi...	0
w3wp.exe	10568		T-SQL, Man...	SEIFOLLAHI\admi...	0
w3wp.exe	9308		T-SQL, Man...	SEIFOLLAHI\admi...	0
w3wp.exe	9280		T-SQL, Man...	SEIFOLLAHI\admi...	0
w3wp.exe	4748		T-SQL, Man...	SEIFOLLAHI\SPS_...	0
w3wp.exe	5556		T-SQL, Man...	SEIFOLLAHI\admi...	0
w3wp.exe	9204		T-SQL, Man...	IIS APPPOOL\Def...	0
w3wp.exe	7380		T-SQL, Man...	SEIFOLLAHI\admi...	0
w3wp.exe	4476		T-SQL, Man...	SEIFOLLAHI\admi...	0
WebAnalyticsServi...	4152		T-SQL, Man...	SEIFOLLAHI\admi...	0

☒ Show processes from all users ☒ Show processes in all sessions Refresh

برای پیدا کردن اینکه کدام پروسس متعلق به Application مورد نظر ماست باید از برنامه [Process Explorer](#) کمک بگیریم . پس از اجرای این برنامه روی ستون‌های آن کلیک سمت راست کنید و Select Column را انتخاب نمایید



گزینه Command Line را انتخاب نمایید و پس از OK کردن به دنبال پردازش‌های w3wp در حال اجرا بگردید .



حال می‌توانید پروسه مورد نظر خود را براحتی بیابید و شناسه پردازش را از آنجا بخوانید

Process	PID	Command Line
svchost.exe	3400	C:\Windows\system32\svchost.exe -k iissvcs
w3wp.exe	4476	c:\windows\system32\inetsrv\w3wp.exe -ap "SecurityTokenServiceApplicationPool" -v "v2.0" -f ...
w3wp.exe	7380	c:\windows\system32\inetsrv\w3wp.exe -ap "6f3e04326881487090916c07abd51b7b" -v "v2.0" -f ...
w3wp.exe	9204	c:\windows\system32\inetsrv\w3wp.exe -ap "DefaultAppPool" -v "v2.0" -f "webengine4.dll" -a ...
w3wp.exe	5556	c:\windows\system32\inetsrv\w3wp.exe -ap "TSF8080" -v "v2.0" -f "webengine4.dll" -a ...
w3wp.exe	4748	c:\windows\system32\inetsrv\w3wp.exe -ap "SPS2010 - 3000 - AP" -v "v2.0" -f "webengine4.dll" ...
w3wp.exe	9280	c:\windows\system32\inetsrv\w3wp.exe -ap "SharePoint - MySiteCollection2000" -v "v2.0" -f "we...
w3wp.exe	9308	c:\windows\system32\inetsrv\w3wp.exe -ap "SharePoint - 9090" -v "v2.0" -f "webengine4.dll" -a ...
w3wp.exe	10568	c:\windows\system32\inetsrv\w3wp.exe -ap "SharePoint Central Administration v4" -v "v2.0" -f "w...
w3wp.exe	13568	c:\windows\system32\inetsrv\w3wp.exe -ap "SharePoint - 4000" -v "v2.0" -f "webengine4.dll" -a ...

اکنون شناسه مشخص شده و می‌توانید به Debug کردن پردازش

vas.exe	5792
vssphost4.exe	12640
w3wp.exe	13568
w3wp.exe	10568
w3wp.exe	9308
w3wp.exe	9280
w3wp.exe	4748
w3wp.exe	5556
w3wp.exe	9204
w3wp.exe	7380

[موفق باشید](#)

نظرات خوانندگان

نویسنده: مهدی موسوی
تاریخ: ۱۶:۸ ۱۳۹۲/۰۱/۲۳

سلام.

روش‌های ساده‌تری هم برای اینکار وجود داره. کافیه تا اونجاییکه علاقمند هستید کدتون break بخوره، این کد رو بنویسید:

```
if (Debugger.IsAttached)
    Debugger.Break();
else
    Debugger.Launch();
```

بدین ترتیب هر وقت اجرا به این خط برسه، پنجره Visual Studio Just-In-Time Debugger باز میشه و Debugger بطور خودکار به App شما Attach میشه و ...

موفق باشید.

نویسنده: محمد باقر سیف الهی
تاریخ: ۱۸:۴۷ ۱۳۹۲/۰۱/۲۳

به خاطر بعضی دست کاری هایی که روی IIS انجام داده بودم (+) VS در مود Debug قرار نمی‌گرفت و پس از فشردن F5 پیغام خطا نمایش می‌داد (با این مضمون که امکان Attach کردن به پروسس وجود ندارد).

نویسنده: مهدی
تاریخ: ۱۳:۵۱ ۱۳۹۲/۰۱/۲۴

راه حلی دیگر

```
cd c:\windows\system32\inetsrv

appcmd list wp
```

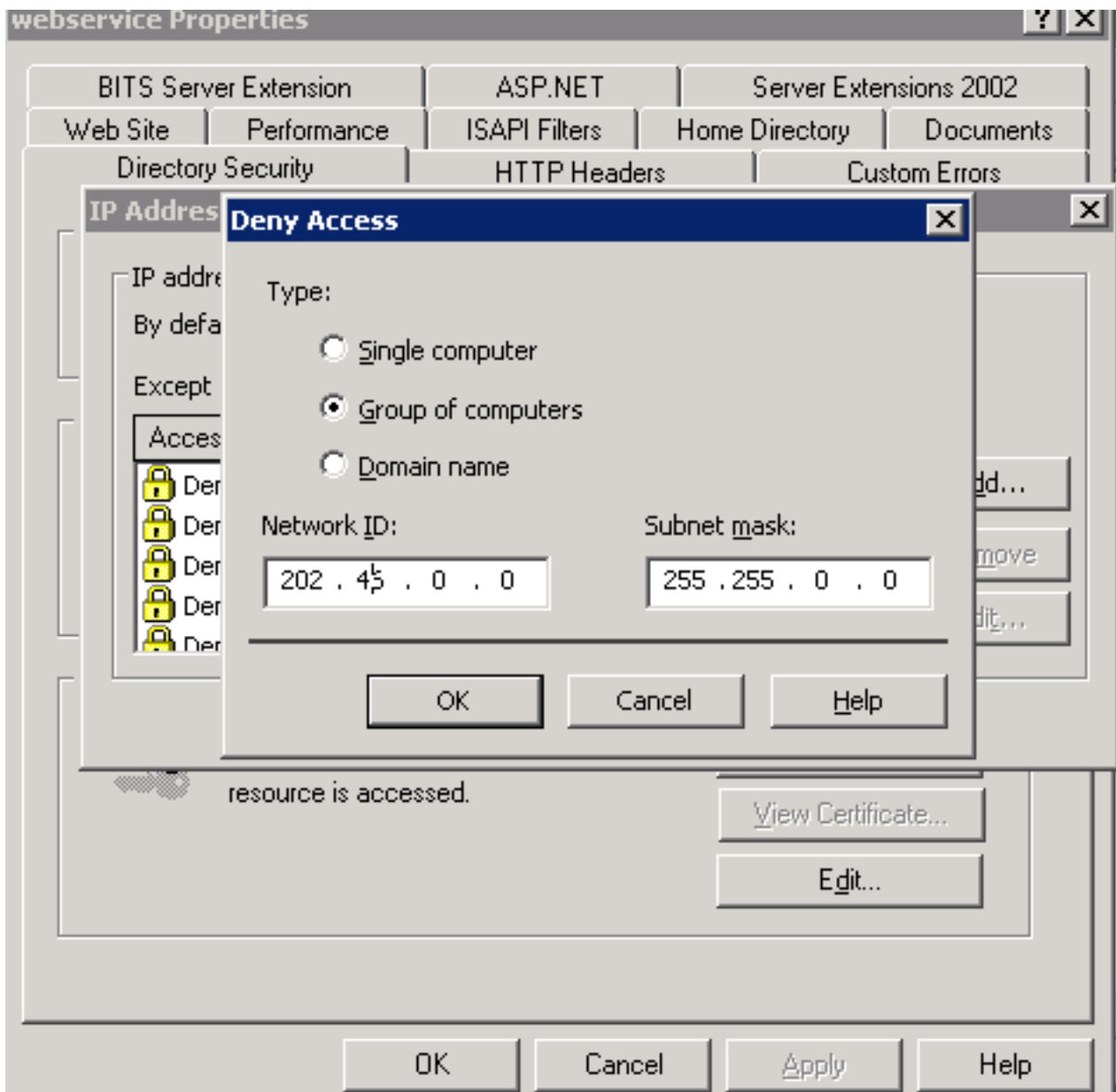
نویسنده: حمید
تاریخ: ۱۸:۳۱ ۱۳۹۲/۰۱/۲۹

بجای Process Explorer از تسک منیجر هم میتونید استفاده کنید، ستون‌های مورد نظر رو فقط شو کنید.

یک سری از ربات‌ها مدام سایت‌ها را برای یافتن یک سری از اسکریپت‌های خاص اسکن می‌کنند. IP‌های آن‌ها نیز عموماً متعلق است به چین و هسایگان آن. مشکلی که با این ربات‌ها وجود دارد این است که از یک IP خاص نشات نمی‌گیرند و به نظر صدها سرور آلوده را جهت مقاصد خود مورد استفاده قرار می‌دهند. به همین جهت نیاز است بتوان یک بازه‌ی IP را در IIS بست.

بستن یک بازه‌ی IP در IIS 6

در IIS6 باید به خواص وب سایت و برگه‌ی Directory security آن مراجعه کرد. سپس در این قسمت، در حین افزودن IP مد نظر، باید گزینه‌ی Group of computers را انتخاب نمود.



در اینجا برای مثال برای بستن IP هایی که با 194 شروع می‌شوند، باید 194.0.0.0 را وارد کرد و در این حالت subnet mask را نیز باید 255.0.0.0 انتخاب کرد. با این subnet mask خاص، اعلام می‌کنیم که فقط اولین قسمت IP وارد شده برای ما اهمیت دارد و سه قسمت بعدی خیر. به این ترتیب تمام IP های شروع شده با 194 با هر سه جزء دیگر دلخواهی، بسته خواهند شد.

بستن یک بازه‌ی IP در IIS های 7 به بعد

در IIS های 7 به بعد نیاز است مراحل زیر را طی کرده و نقش « [IP and Domain Restrictions](#) » را نصب کنید.

```
Administrative Tools -> Server Manager -> expand Roles  
-> Web Server (IIS) -> Role Services -> Add Role Services -> select IP and Domain Restrictions
```

پس از آن با استفاده از تنظیمات ذیل در فایل web.config می‌توان یک IP و یا بازه‌ای از IP ها را بست:

```
<system.webServer>  
  <security>  
    <ipSecurity>  
      <add ipAddress="192.168.100.1" />  
      <add ipAddress="169.254.0.0" subnetMask="255.255.0.0" />  
    </ipSecurity>  
  </security>  
</system.webServer>
```

البته علاوه بر نصب نقش یاد شده، باید Feature Delegation نیز جهت استفاده از آن فعال گردد:

```
IIS 7 -> root server -> Feature Delegation -> IP and Domain Restrictions  
-> Change the delegation to Read/Write
```

نظرات خوانندگان

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۲:۴۶

ضمن تشکر بفرمایید چرا از طریق فایروال دسترسی IPهای یاد شده را مسدود نمی‌کنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۲:۴۷

این هم یک روش است؛ البته برای کسی که ادمین است. اگر ادمین نباشد و ادمین قبلاً قابلیت ذکر شده مخصوص IIS 7 را افزوده باشد، کاربران یک هاست اشتراکی هم می‌توانند راساً و بدون نیاز به ادمین، فقط با ویرایش کردن فایل web.config، اقدام کنند.

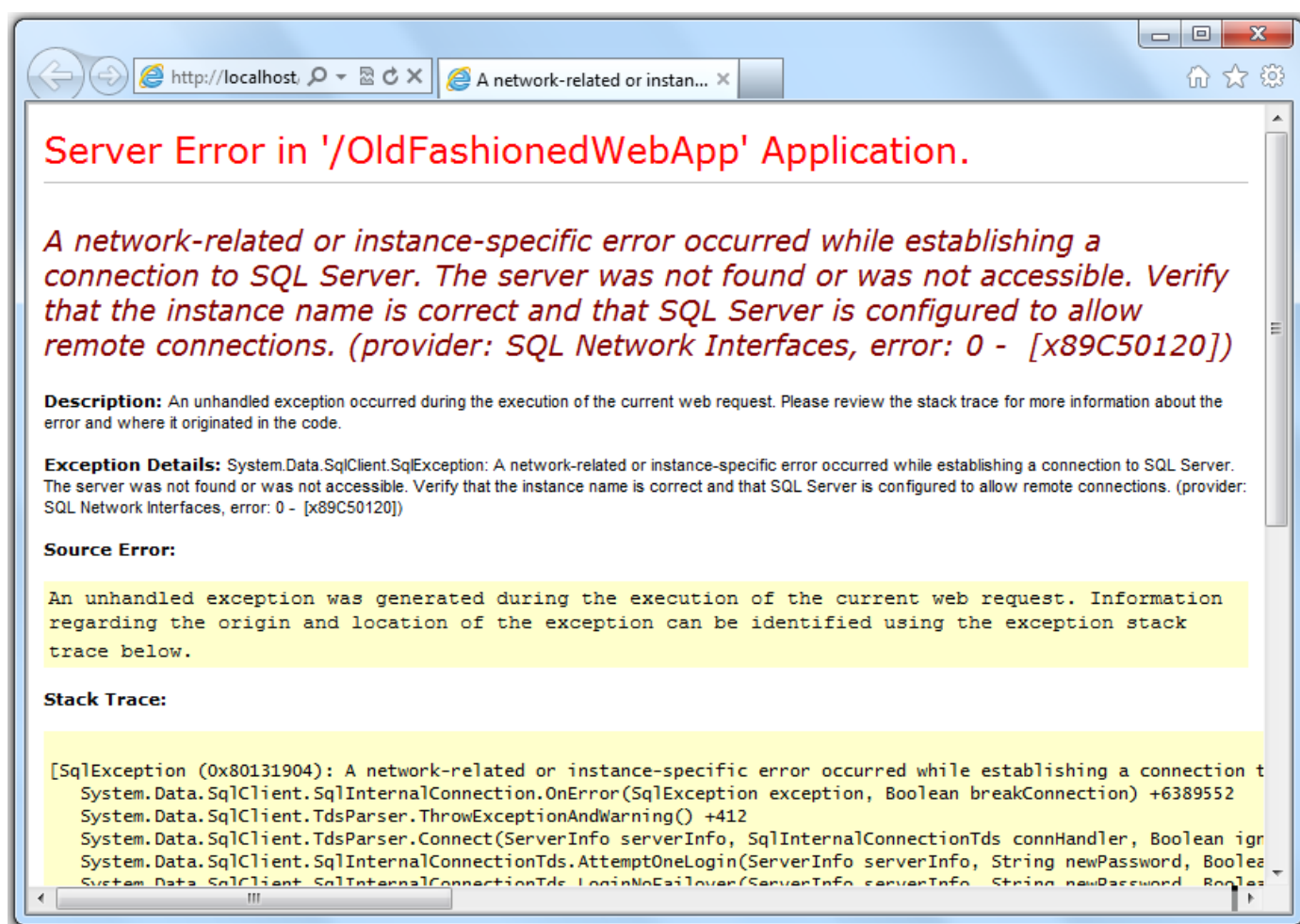
نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۳:۰۲

بله. البته منابع سیستمی که اسکن‌های یاد شده مصرف می‌کنند، معمولاً مسدود نمودن آنها را در حیطه وظایف مدیر سرور قرار می‌دهد و البته در مورد IIS 6 دسترسی به وب سرور لازم است. و احتمالاً مسدود نمودن از طریق فایروال از نظر سربرار و مصرف منابع ارجح است. از طرفی باید در نظر داشت که بسیار دیده شده که همان سرورهای چینی علاقه مند به اسکن سایت‌ها علاقه مند به حملات DDOS و ... هم هستند که الزاماً از مسیر IIS نمی‌گذرد.

در هر صورت ممنون از بیان مطلب فوق که به هر حال دانستن آن بهتر از ندانستن است. سوال کردم چون گفتم شاید دلیل خاصی دارد که این روش را بیان فرمودید.

این مقاله قسمت اول یک سری دو قسمتی است، که در آن به نحوه استفاده از LocalDB در IIS می‌پردازیم.

LocalDb دیتابیس توصیه شده برای ویژوال استودیو است و برای انواع پروژه‌ها مانند اپلیکیشن‌های وب می‌تواند استفاده شود. هنگام استفاده از این دیتابیس در IIS Express یا Cassini همه چیز طبق انتظار کار میکند. اما به محض آنکه بخواهید از آن در Full IIS استفاده کنید با خطاهایی مواجه می‌شوید. مقصود از Full IIS همان نسخه ای است که به همراه ویندوز عرضه می‌شود و در قالب یک Windows Service اجرا می‌گردد.



هنگام استفاده از Full IIS دو خاصیت از LocalDb باعث بروز مشکل می‌شوند:

LocalDb نیاز دارد پروفایل کاربر بارگذاری شده باشد

بصورت پیش فرض، وهله LocalDb متعلق به یک کاربر بوده و خصوصی است

در ادامه این مقاله روی بارگذاری پروفایل کاربر تمرکز می‌کنیم، و در قسمت بعدی به مالکیت وهله LocalDb می‌پردازیم.

بارگذاری پروفایل کاربر

بگذارید دوباره به خطای موجود نگاهی بیاندازیم:

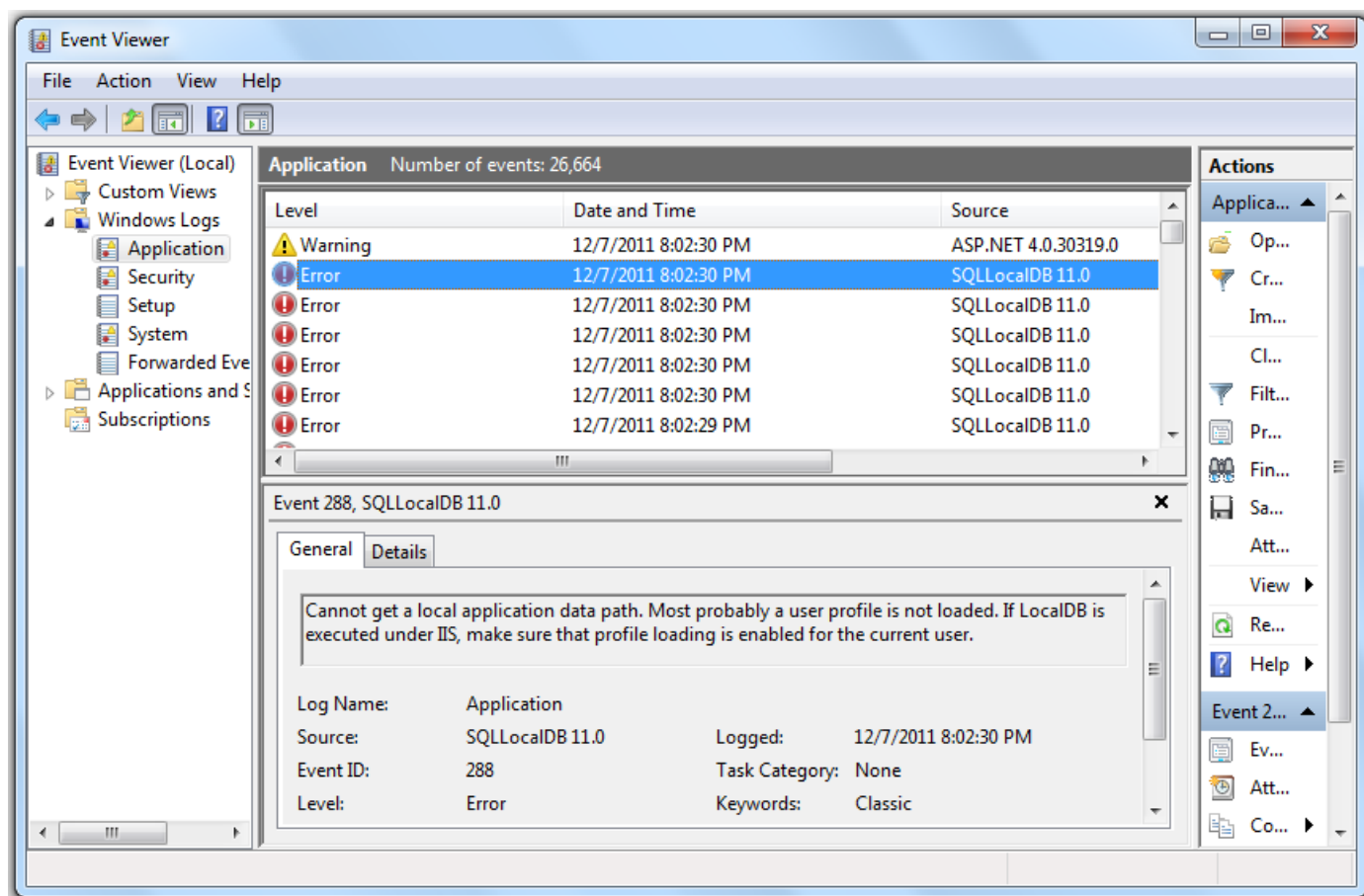
System.Data.SqlClient.SqlException: A network-related or instance-specific error occurred while establishing a connection to SQL Server. The server was not found or was not accessible. Verify that the instance name is correct and that SQL Server is configured to allow remote connections. (provider: SQL Network Interfaces, error: (0 - [x89C50120

این پیغام خطا زیاد مفید نیست، اما LocalDb اطلاعات بیشتری در Event Log ویندوز ذخیره می‌کند. اگر **Windows Logs** را باز کنید و به قسمت **Application** بروید پیغام زیر را مشاهده خواهید کرد.

*Windows API call SHGetKnownFolderPath returned error code: 5. Windows system error message is: Access is denied
Reported at line: 400*
بعلاوه این پیام خطا:

Cannot get a local application data path. Most probably a user profile is not loaded. If LocalDB is executed under IIS, make sure that profile loading is enabled for the current user

به احتمال زیاد تعداد بیشتری از این دو خطا در تاریخچه وقایع وجود خواهد داشت، چرا که منطق کانکشن ADO.NET چند بار سعی می‌کند در بازه‌های مختلف به دیتابیس وصل شود.

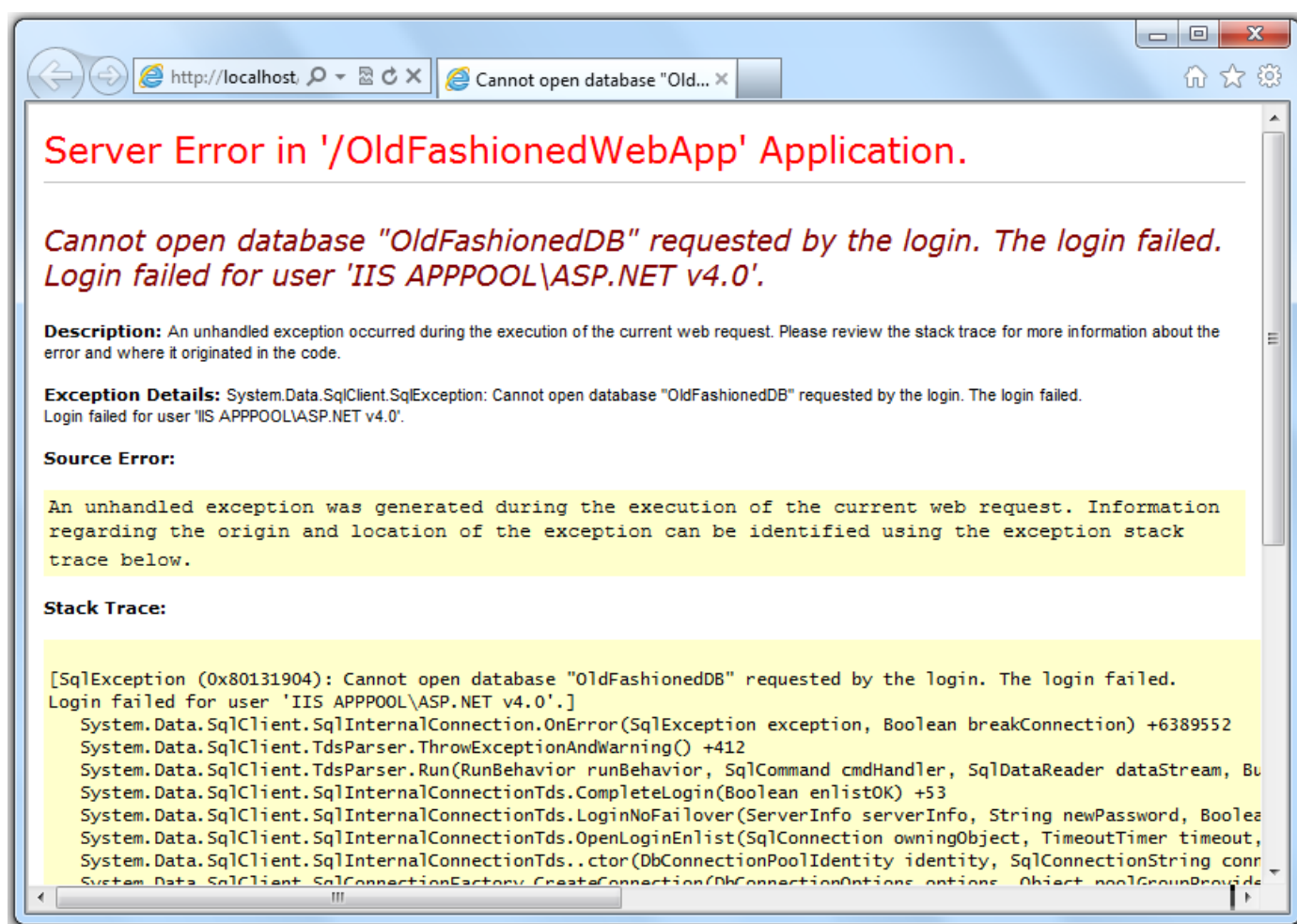


پیغام خطای دوم واضح است، نیاز است پروفایل کاربر را بارگذاری کنیم. انجام اینکار زیاد مشکل نیست، هر Application Pool در IIS تنظیماتی برای بارگذاری پروفایل کاربر دارد که از قسمت Advanced Settings قابل دسترسی است. متأسفانه پس از انتشار سرویس پک 1 برای Windows 7 مسائل کمی پیچیده‌تر شد. در حال حاضر فعال کردن **loadUserProfile** برای بارگذاری

کامل پروفایل کاربر به تنهایی کافی نیست، و باید **setProfileEnvironment** را هم فعال کنیم. برای اطلاعات بیشتر در این باره به مستندات [KB 2547655](https://msdn.microsoft.com/en-us/library/aa479546.aspx) مراجعه کنید. بدین منظور باید فایل **applicationHost.config** را ویرایش کنید. فایل مذکور در مسیر `C:\Windows\System32\inetsrv\config` قرار دارد. همانطور که در مستندات KB 2547655 توضیح داده شده، باید پرچم هر دو تنظیمات را برای **ASP.NET 4.0** فعال کنیم:

```
<add name="ASP.NET v4.0" autoStart="true" managedRuntimeVersion="v4.0"
managedPipelineMode="Integrated">
  <processModel identityType="ApplicationPoolIdentity" loadUserProfile="true"
setProfileEnvironment="true" />
</add>
```

پس از بروز رسانی این تنظیمات، Application Pool را Restart کنید و اپلیکیشن را مجدداً اجرا نمایید. اگر همه چیز طبق انتظار پیش برود، با خطای جدیدی مواجه خواهیم شد:



جای هیچ نگرانی نیست، چرا که این پیغام خطا انتظار می‌رود. همانطور که در ابتدا گفته شد، دو خاصیت LocalDb باعث بروز این خطاها می‌شوند و ما هنوز به خاصیت دوم نپرداخته ایم. بصورت پیش فرض وهله‌های LocalDb خصوصی (private) هستند و در Windows account جاری اجرا می‌شوند. بنابراین ApplicationPoolIdentity در IIS به وهله‌های دیتابیس دسترسی نخواهد داشت. در قسمت دوم این مقاله، راه‌های مختلفی را برای رفع این مشکل بررسی می‌کنیم.

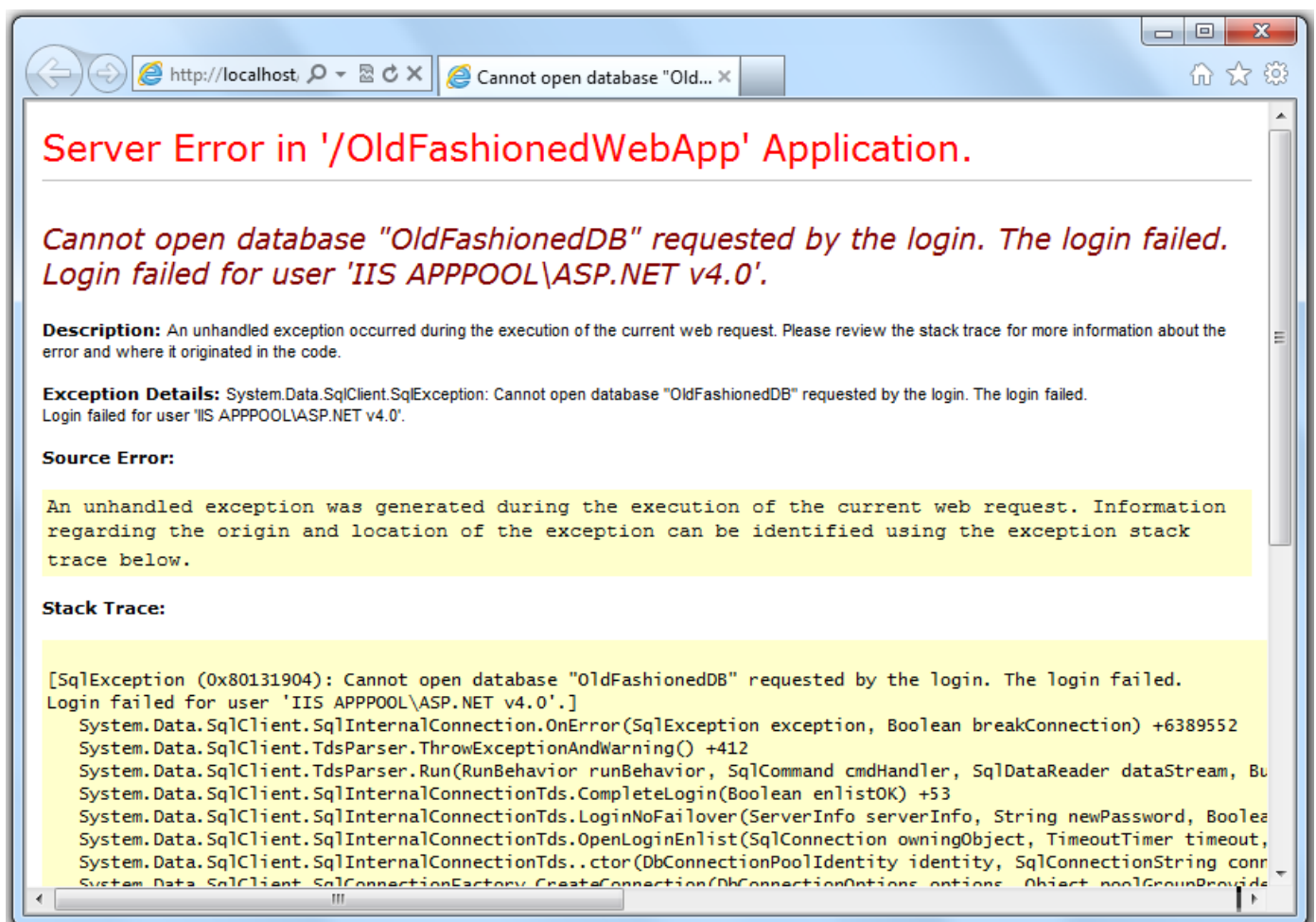
در [قسمت قبلی](#) این مقاله گفتیم که دو خاصیت از LocalDb هنگام استفاده از Full IIS باعث بروز خطا می شوند:

LocalDb نیاز دارد که پروفایل کاربر بارگذاری شده باشد
بصورت پیش فرض وهله LocalDb متعلق به یک کاربر بوده، و خصوصی است

در قسمت قبل دیدیم چگونه باید پروفایل کاربر را بدرستی بارگذاری کنیم. در این مقاله به مالکیت وهله ها (instance ownership) می پردازیم.

مشکل وهله خصوصی

در پایان قسمت قبلی، اپلیکیشن وب را در این حالت رها کردیم:

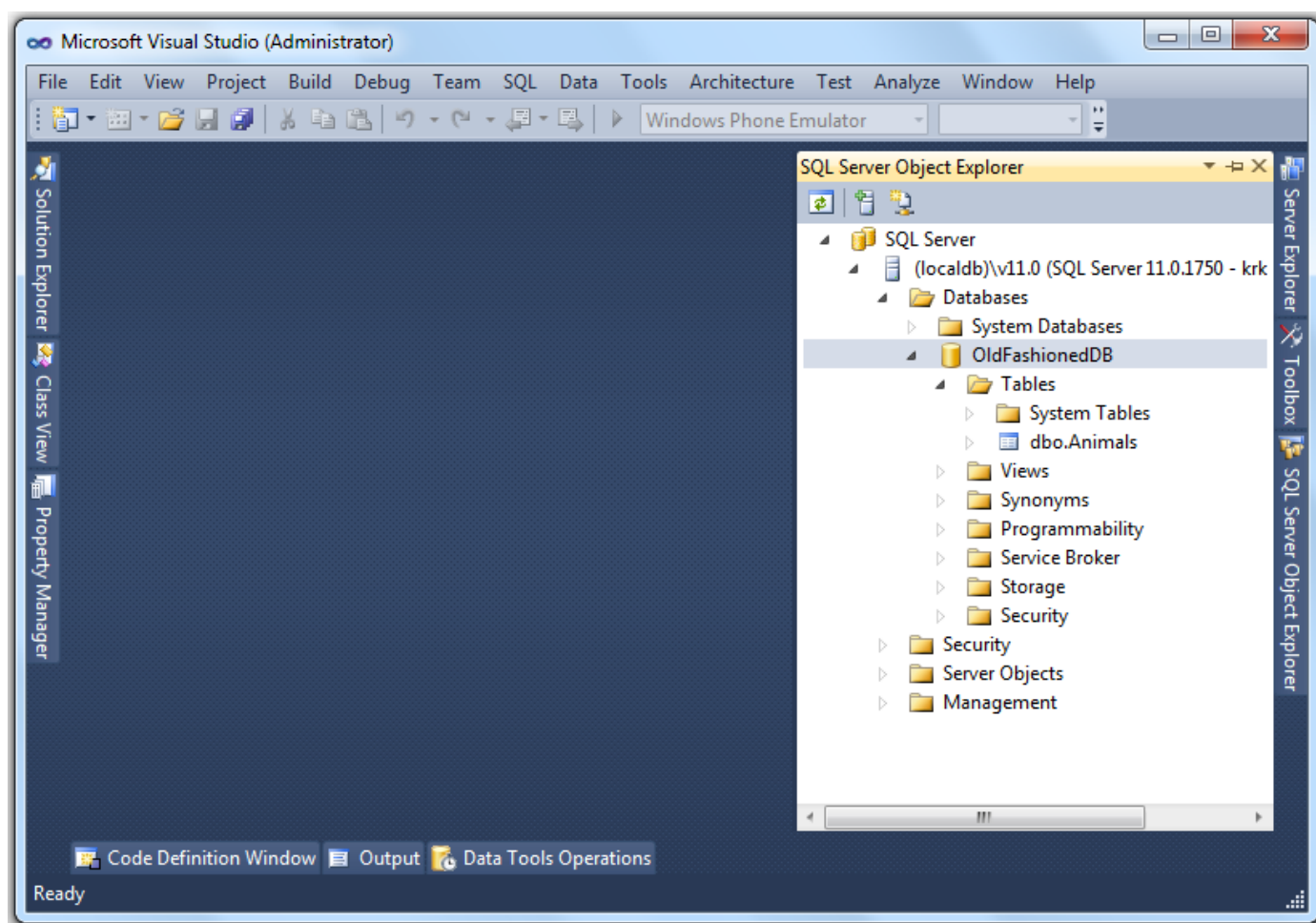


همانطور که مشاهده می کنید با خطای زیر مواجه هستیم:

System.Data.SqlClient.SqlException: Cannot open database "OldFashionedDB" requested by the login. The login failed.

'Login failed for user 'IIS APPPOOL\ASP.NET v4.0

این بار پیغام خطا واضح و روشن است. LocalDb با موفقیت اجرا شده و اپلیکیشن وب هم توانسته به آن وصل شود، اما این کانکشن سپس قطع شده چرا که دسترسی به وهله جاری وجود نداشته است. اکانت ApplicationPoolIdentity (در اینجا IIS APPPOOL\ASP.NET v4.0) نتوانسته به دیتابیس LocalDb وارد شود، چرا که دیتابیس مورد نظر در رشته اتصال اپلیکیشن (OldFashionedDB) وجود ندارد. عجیب است، چرا که وصل شدن به همین دیتابیس با رشته اتصال جاری در ویژوال استودیو با موفقیت انجام می‌شود.



همانطور که در تصویر بالا مشاهده می‌کنید از ابزار SQL Server Object Explorer استفاده شده است. این ابزار توسط [SQL Server Data Tools](#) معرفی شد و در نسخه‌های بعدی ویژوال استودیو هم وجود دارد و توسعه یافته است. چطور ممکن است ویژوال استودیو براحتمی بتواند به دیتابیس وصل شود، اما اپلیکیشن وب ما با همان رشته اتصال نمی‌تواند دیتابیس را باز کند؟ در هر دو صورت رشته اتصال ما بدین شکل است:

```
Data Source=(localdb)\v11.0;Initial Catalog=OldFashionedDB;Integrated Security=True
```

پاسخ این است که در اینجا، دو وهله از LocalDb وجود دارد. برخلاف وهله‌های SQL Server Express که بعنوان سرویس‌های ویندوزی اجرا می‌شوند، وهله‌های LocalDb بصورت پروسس‌های کاربری (user processes) اجرا می‌شوند. هنگامی که کاربران مختلفی سعی می‌کنند به LocalDb متصل شوند، برای هر کدام از آنها پروسس‌های مجزایی اجرا خواهد شد. هنگامی که در ویژوال استودیو به (localdb)\v11.0 وصل می‌شویم، وهله ای از LocalDb ساخته شده و در حساب کاربری ویندوز جاری اجرا می‌شود.

اما هنگامی که اپلیکیشن وب ما در IIS می‌خواهد به همین دیتابیس وصل شود، وهله دیگری ساخته شده و در **ApplicationPoolIdentity** اجرا می‌شود. گرچه ویژوال استودیو و اپلیکیشن ما هر دو از یک رشته اتصال استفاده می‌کنند، اما در عمل هر کدام به وهله‌های متفاوتی از LocalDb دسترسی پیدا خواهند کرد. پس مسلماً دیتابیزی که توسط وهله ای در ویژوال استودیو ساخته شده است، برای اپلیکیشن وب ما در IIS در دسترس نخواهد بود.

یک مقایسه خوب از این وضعیت، پوشه **My Documents** در ویندوز است. فرض کنید در ویژوال استودیو کدی بنویسیم که در این پوشه یک فایل جدید می‌سازد. حال اگر با حساب کاربری دیگری وارد ویندوز شویم و به پوشه My Documents برویم این فایل را نخواهیم یافت. چرا که پوشه My Documents برای هر کاربر متفاوت است. بهمین شکل، وهله‌های LocalDb برای هر کاربر متفاوت است و به پروسس‌ها و دیتابیس‌های مختلفی اشاره می‌کنند.

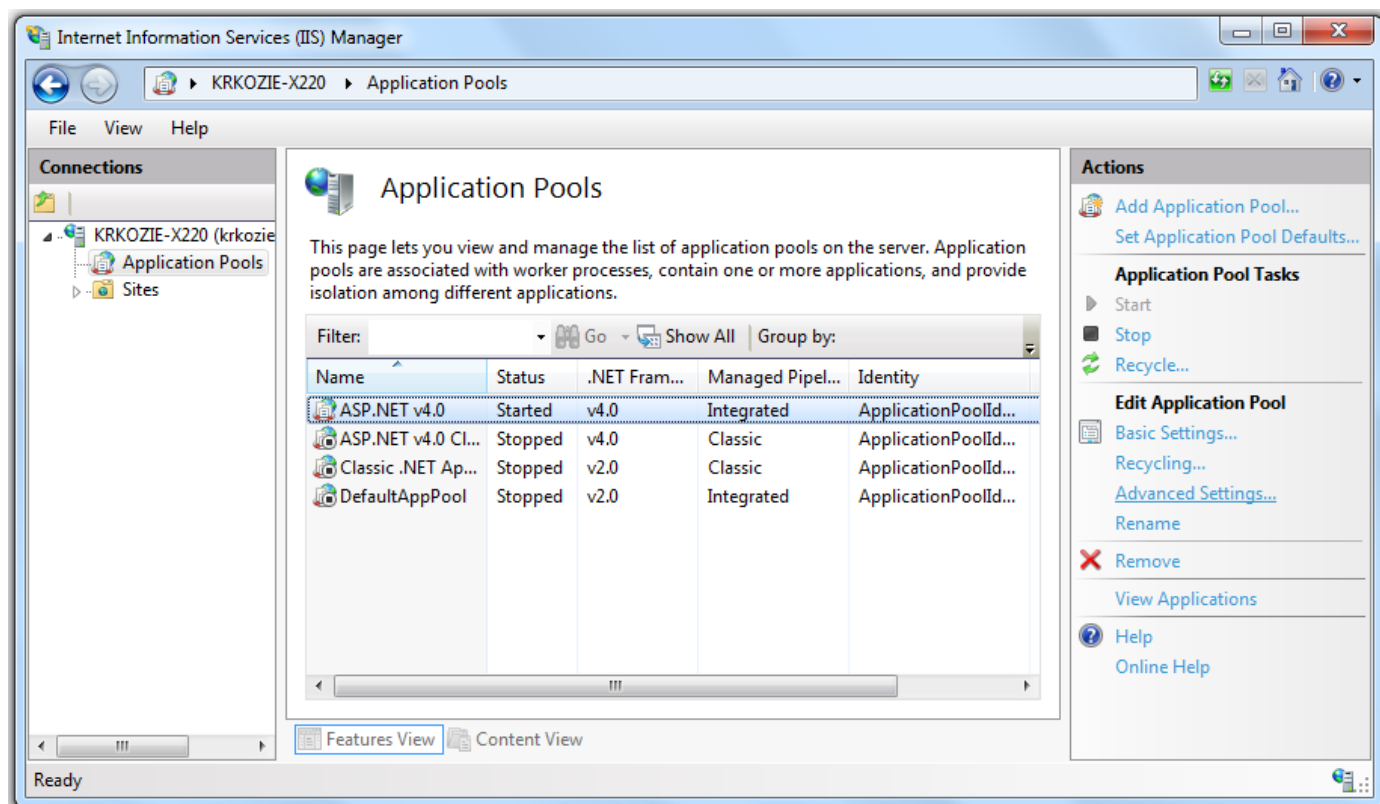
به همین دلیل است که اپلیکیشن وب ما می‌تواند بدون هیچ مشکلی روی IIS Express اجرا شود و دیتابیس را باز کند. چرا که IIS Express درست مانند LocalDb یک پروسس کاربری است. IIS Express توسط ویژوال استودیو راه اندازی می‌شود و روی حساب کاربری جاری اجرا می‌گردد، پس پروسس آن با پروسس خود ویژوال استودیو یکسان خواهد بود و هر دو زیر یک اکانت کاربری اجرا خواهند شد.

راه حل ها

درک ماهیت مشکل جاری، راه حال‌های مختلفی را برای رفع آن بدست می‌دهد. از آنجا که هر راه حل مزایا و معایب خود را دارد، بجای معرفی یک راه حال واحد چند راهکار را بررسی می‌کنیم.

رویکرد 1: اجرای IIS روی کاربر جاری ویندوز

اگر مشکل، حساب‌های کاربری مختلف است، چرا خود IIS را روی کاربر جاری اجرا نکنیم؟ در این صورت ویژوال استودیو و اپلیکیشن ما هر دو به یک وهله از LocalDb وصل خواهند شد و همه چیز بدرستی کار خواهد کرد. ایجاد تغییرات لازم نسبتاً ساده است. IIS را اجرا کنید و Application Pool مناسب را انتخاب کنید، یعنی همان گزینه که برای اپلیکیشن شما استفاده می‌شود.



قسمت Advanced Settings را باز کنید:

Advanced Settings

(General)

.NET Framework Version	v4.0
Enable 32-Bit Applications	False
Managed Pipeline Mode	Integrated
Name	ASP.NET v4.0
Queue Length	1000
Start Automatically	True

CPU

Limit	0
Limit Action	NoAction
Limit Interval (minutes)	5
Processor Affinity Enabled	False
Processor Affinity Mask	4294967295

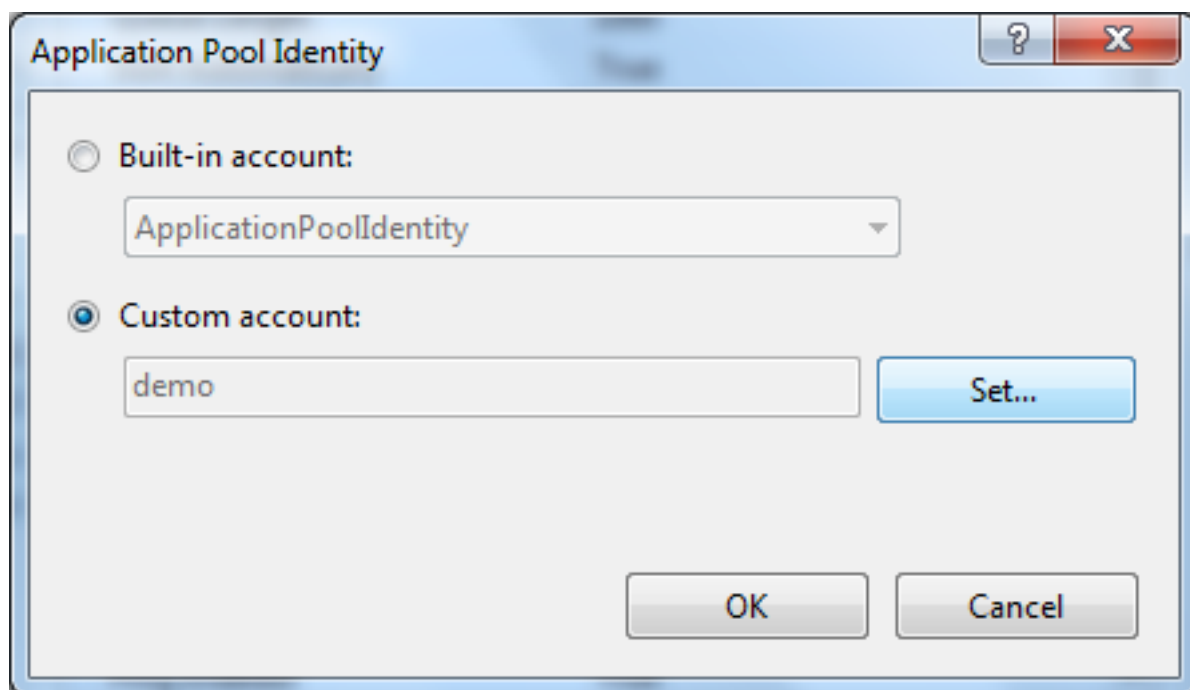
Process Model

Identity	ApplicationPoolIdentity
Idle Time-out (minutes)	20
Load User Profile	True
Maximum Worker Processes	1
Ping Enabled	True

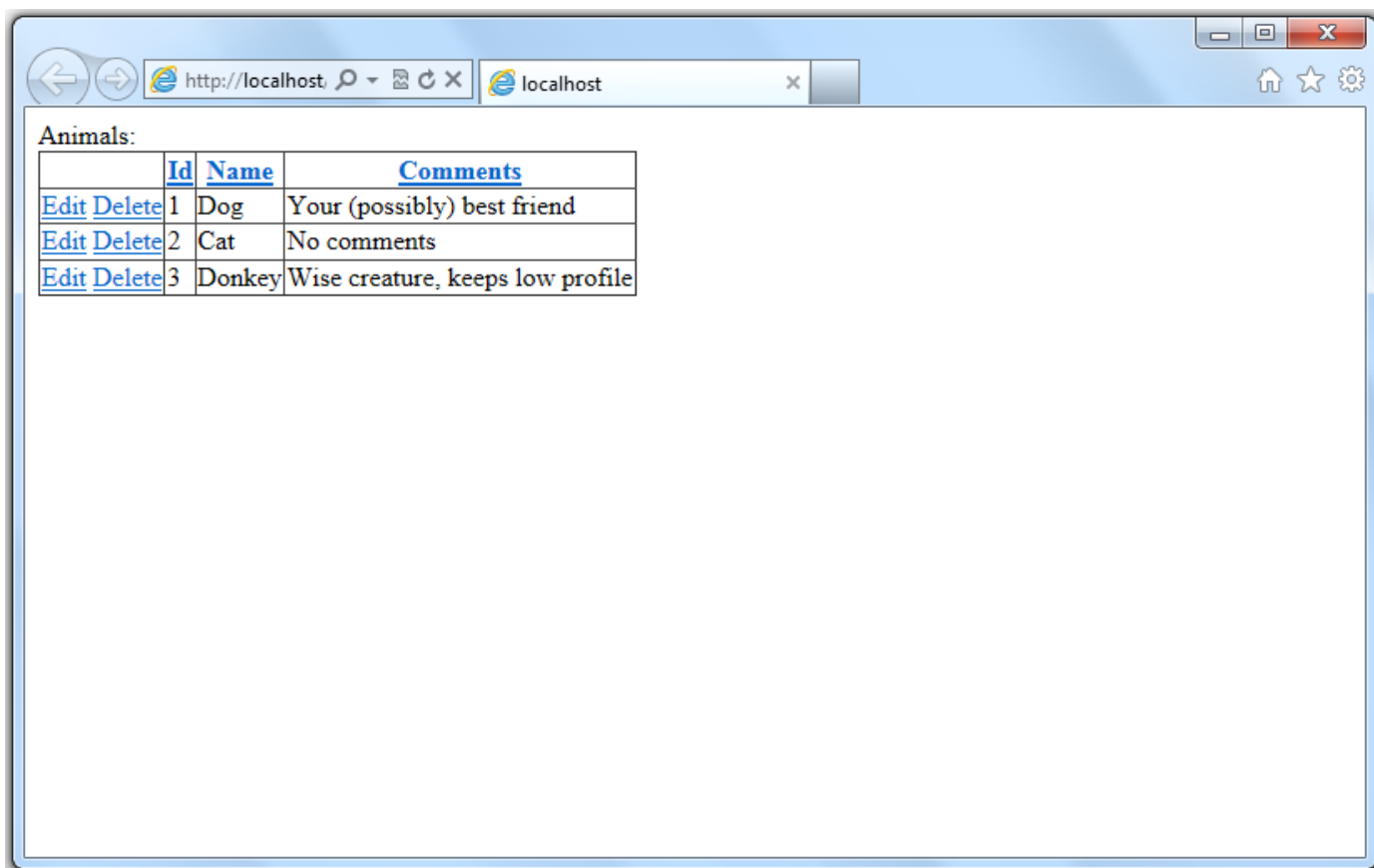
Identity
 [identityType, username, password] Configures the application pool to run as built-in account, i.e. Application Pool Identity (recommended), Network Service, Local System, Local Service, or as a specific user identity.

OK Cancel

روی دکمه سه نقطه کنار خاصیت **Identity** کلیک کنید تا پنجره Application Pool Identity باز شود:



در این قسمت می‌توانید از حساب کاربری جاری استفاده کنید. روی دکمه Set کلیک کنید و نام کاربری و رمز عبور خود را وارد نمایید. حال اگر اپلیکیشن را مجدداً اجرا کنید، همه چیز باید بدرستی اجرا شود.



خوب، معایب این رویکرد چیست؟ مسلماً اجرای اپلیکیشن وب روی اکانت کاربری جاری، ریسک‌های امنیتی متعددی را معرفی می‌کند. اگر کسی بتواند اپلیکیشن وب ما را هک کند، به تمام منابع سیستم که اکانت کاربری جاری به آنها دسترسی دارد، دسترسی خواهد داشت. اما اجرای اپلیکیشن مورد نظر روی ApplicationPoolIdentity امنیت بیشتری را ارائه می‌کند، چرا که اکانت‌های ApplicationPoolIdentity دسترسی بسیار محدودتری به منابع سیستم محلی دارند. بنابراین استفاده از این روش بطور کلی توصیه نمی‌شود، اما در سناریوهای خاصی با در نظر داشتن ریسک‌های امنیتی می‌تواند رویکرد خوبی باشد.

رویکرد 2: استفاده از و هله مشترک

یک راه حال دیگر استفاده از قابلیت instance sharing است. این قابلیت به ما این امکان را می‌دهد تا یک و هله LocalDb را بین کاربران یک سیستم به اشتراک بگذاریم. و هله به اشتراک گذاشته شده، توسط یک نام عمومی (public name) قابل دسترسی خواهد بود.

ساده‌ترین راه برای به اشتراک گذاشتن و هله‌های LocalDb استفاده از ابزار [SqlLocalDB.exe](#) است. بدین منظور Command Prompt را بعنوان مدیر سیستم باز کنید و فرمان زیر را اجرا نمایید:

```
sqllocaldb share v11.0 IIS_DB
```

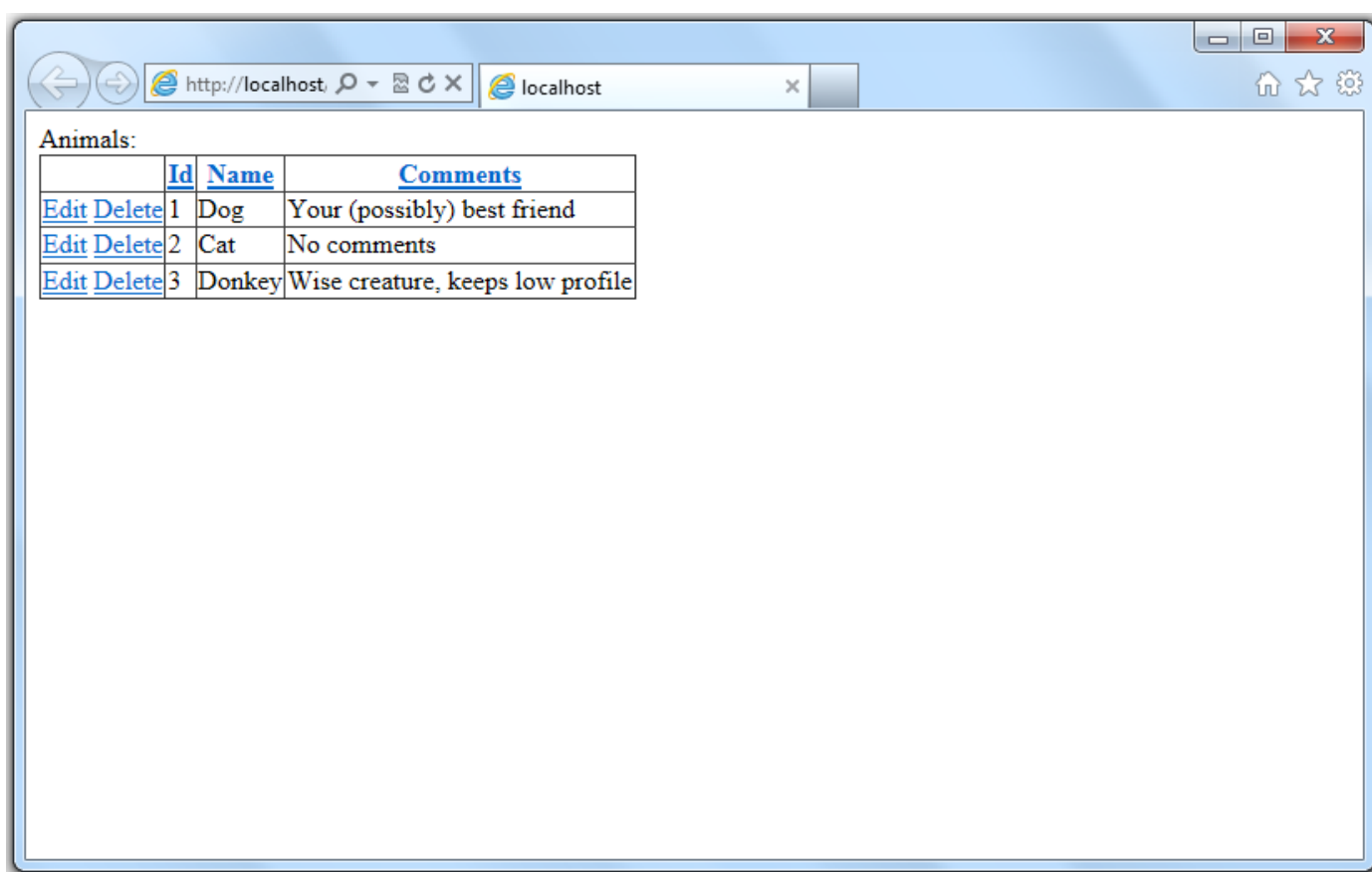
این فرمان و هله خصوصی LocalDb را با نام عمومی **IIS_DB** به اشتراک می‌گذارد. حال تمام کاربران سیستم می‌توانند با آدرس **(localdb)\.IIS_DB** به این و هله وصل شوند. این فرمت آدرس دهی سرور دیتابیس، مشخص می‌کند که از یک و هله shared استفاده می‌کنیم. رشته اتصال جدید مانند لیست زیر خواهد بود:

```
Data Source=(localdb)\.IIS_DB;Initial Catalog=OldFashionedDB;Integrated Security=True
```

پیش از آنکه اپلیکیشن وب ما بتواند به این وهله متصل شود، باید لاگین‌های مورد نیاز برای ApplicationPoolIdentity را ایجاد کنیم. راه اندازی وهله ساده است، کافی است دیتابیس را در SQL Server Object Explorer باز کنید. این کار اتصالی به دیتابیس برقرار می‌کند و آن را زنده نگاه می‌دارد. برای ایجاد لاگین مورد نظر، می‌توانیم در SQL Server Object Explorer یک کوئری اجرا کنیم:

```
create login [IIS APPPOOL\ASP.NET v4.0] from windows;  
exec sp_addsrvrolemember N'IIS APPPOOL\ASP.NET v4.0', sysadmin
```

اسکرپت بالا به اکانت ApplicationPoolIdentity سطح دسترسی کامل می‌دهد. در صورت امکان بهتر است از سطوح دسترسی محدودتری استفاده کنید، مثلاً دسترسی به دیتابیس یا جداولی مشخص. حالا می‌توانید اپلیکیشن را مجدداً اجرا کنید و همه چیز بدون خطا باید کار کند.



معایب این روش چیست؟ مشکل اصلی در این رویکرد این است که پیش از آنکه اپلیکیشن ما بتواند به وهله مشترک دسترسی داشته باشد، باید وهله مورد نظر را راه اندازی و اجرا کنیم. بدین منظور، حساب کاربری ویندوزی که مالکیت وهله را دارد باید به آن وصل شود و کانکشن را زنده نگه دارد، در غیر اینصورت وهله LocalDb قابل دسترسی نخواهد بود.

رویکرد 3: استفاده از SQL Server Express

از آنجا که نسخه کامل SQL Server Express بعنوان یک سرویس ویندوزی اجرا می‌شود، شاید بهترین راه استفاده از همین روش باشد. کافی است یک نسخه از SQL Server Express را نصب کنیم، دیتابیس مورد نظر را در آن بسازیم و سپس به آن متصل شویم. برای این کار حتی می‌توانید از ابزار جدید SQL Server Data Tools استفاده کنید، چرا که با تمام نسخه‌های SQL Server سازگار است. در صورت استفاده از نسخه‌های کامل تر، رشته اتصال ما بدین شکل تغییر خواهد کرد:

```
Data Source=.\SQLEXPRESS;Initial Catalog=OldFashionedDB;Integrated Security=True
```

مسلما در این صورت نیز، لازم است اطمینان حاصل کنیم که ApplicationPoolIdentity به وهله SQL Server Express دسترسی کافی دارد. برای این کار می‌توانیم از اسکریپت قبلی استفاده کنیم:

```
create login [IIS APPPOOL\ASP.NET v4.0] from windows;  
exec sp_addsrvrolemember N'IIS APPPOOL\ASP.NET v4.0', sysadmin
```

حال اجرای مجدد اپلیکیشن باید با موفقیت انجام شود. استفاده از این روش مسلما امکان استفاده از LocalDb را از ما می‌گیرد. ناگفته نماند که وهله‌های SQL Server Express همیشه در حال اجرا خواهند بود چرا که بصورت سرویس‌های ویندوزی اجرا می‌شوند. همچنین استفاده از این روش ممکن است شما را با مشکلاتی هم مواجه کند. مثلا خرابی رجیستری ویندوز می‌تواند SQL Server Express را از کار بیاندازد و مواردی از این دست. راهکارهای دیگری هم وجود دارند که در این مقاله به آنها نپرداختیم. مثلا می‌توانید از AttachDbFilename استفاده کنید یا از اسکریپت‌های T-SQL برای استفاده از وهله خصوصی ASP.NET کمک بگیرید. اما این روش‌ها دردسرهای زیادی دارند، بهمین دلیل از آنها صرفنظر کردیم.

مطالعه بیشتر درباره LocalDb

[Introducing LocalDB and LocalDB Q&A](#)

[Upgrading .NET Framework 4 to support LocalDB connections and using SQL Server Management Studio to work with](#)

[LocalDB](#)

[Using LocalDB in Visual Studio 2010](#)

[Where are LocalDB database files located](#)

نظرات خوانندگان

نویسنده: رضا گرمارودی
تاریخ: ۲۰:۳۱۳۹۲/۱۰/۲۹

سلام؛ ممنون از مقاله جالبتون. یک سوال داشتم. مقالات شما رو در خصوص Identity و Localdb دنبال کردم. شما تجربه کاری با WindowsAzor داریو و اینجا هم از Localdb صحبت کردین. می‌خواستم برای هر کاربر یک دیتابیس مجزا داشته باشم اما هاست‌ها نهایتاً دو یا سه تا دیتابیس sqlserver در اختیار شما می‌گذارند. Localdb همان طور که از اسمش برمیاد Local هست یعنی همیشه روی هاست ولو این که از این دیتابیس تنها یک نفر آنلاین استفاده کنه؟ استفاده از windowsAzor چطور؟ مشکل من و حل می‌کنه یا شما راه حل بهتری معرفی می‌کنید؟

نویسنده: محسن خان
تاریخ: ۲۲:۳۱۳۹۲/۱۰/۲۹

[SQL Server CE](#) داخل پروسه برنامه اجرا میشه (و پروسه مجزایی نداره). [نسخه LocalDB](#) خارج از پروسه برنامه اما به صورت یک [child process](#) اجرا میشه. برنامه وب شما می‌تونه با هر تعداد فایل sdf مربوط به SQL CE یا فایل mdf بانک اطلاعاتی LocalDB کار کنه (در رشته اتصالی آن [AttachDbFileName](#) قابل تعریف هست).

نویسنده: رضا گرمارودی
تاریخ: ۱۶:۵۷۱۳۹۲/۱۱/۰۱

Attachdb در رشته اتصالی موجود است و بر روی لوکال مشکلی ندارم. اما بر روی هاست بانک را Attach نمی‌کند. با سه تا هاست مختلف تست گرفتم ولی نشد! ممکنه Localdb و SQL Server CE تنها بر روی لوکال کار کنند و نه بر روی هاست؟ یا ارائه دهنده سرویس می‌بایست این سرویس‌ها را بر روی سرور خود نصب کنه؟

نویسنده: محسن خان
تاریخ: ۱۷:۵۱۳۹۲/۱۱/۰۱

SQL CE با bin deploy کار می‌کنه و نیازی به نصب نداره (البته نصاب داره؛ ولی مهم و الزامی نیست). فایل‌هاش رو بسته نیوگت زیر به پروژه اضافه می‌کنه:

<http://www.nuget.org/packages/Microsoft.SqlServer.Compact>

اما LocalDB رو باید سرور دار حتما خودش یکبار نصب کنه.

مطمئناً اکثر شما برنامه نویسان با معماری IIS و ASP.NET کمابیش آشنایی دارید
Request از سمت کلاینت به IIS ارسال می‌شود، و عموماً بسته به نوع درخواست کلاینت یا به یک Static File مپ می‌شود (مثلاً به یک عکس)، و یا به یک ISAPI

ISAPI کدی است که عموماً با ++C نوشته می‌شود، و برای درخواست آمده از سمت کلاینت کاری را انجام می‌دهد
یکی از این ISAPI‌ها برای ASP.NET است، که درخواست کلاینت را به یک کد مبتنی بر .NET مپ می‌کند (به همین علت به آن ASP.NET می‌گویند)

نکته ای که در خطوط فوق به وضوح دیده می‌شود، وابستگی شدید ASP.NET به IIS است
بدیهتاً کدی که بر روی بستر ASP.NET نوشته می‌شود نیز وابستگی فوق العاده ای به IIS دارد، که یکی از بدترین نوع این وابستگی‌ها در ASP.NET Web Forms دیده می‌شود.

خب، این مسئله چه مشکلاتی را ایجاد می‌کند ؟
مشکل اول که شاید کمتر به چشم بیاید، بحث کندی اجرای بار اول برنامه‌های ASP.NET است.
اما مشکل دوم عدم توانایی در نوشتن کد برنامه، بدون وابستگی به وب سرور (در اینجا IIS) است، که این مشکل دوم روز به روز در حال جدی‌تر شدن است.

این مشکل دوم را برنامه نویسان جاوا سالهاست که با آن درگیرند، نکته این است که بین دو وب سرور در نحوه پردازش یک درخواست کلاینت تفاوت‌هایی وجود دارد، که بالطبع این تفاوت‌ها در نحوه اجرای کد بالاخره خودش را جاهایی نشان می‌دهد، این که بگوییم رفتار وب سرورها نباید متفاوت باشد کمی مسخره است، زیرا تفاوت آنها با یکدیگر باعث شده که سرعت یکسان و امکانات یکسانی نداشته باشند و هر کدام برای یک سناریوی خاص مناسب‌تر باشند
این مسئله برای ما نیز روز به روز دارای اهمیت بیشتری می‌شود، دیگر این که Web Server ما فقط IIS صرف باشد، سناریوی متداول در پروژه‌های Enterprise نیست

در چه جاهایی می‌توان یک برنامه را هاست کرد ؟

IIS به همراه ASP.NET

IIS بدون ASP.NET (می‌خواهیم برنامه بر روی IIS هاست شود، ولی کاری با ASP.NET نداریم) CLR AppDomains

و وب سرورهای لینوکسی در صورت اجرای برنامه بر روی Mono

و ...

هم اکنون به میزان زیادی مشکل شفاف شده است، مطابق با معماری فعلی داریم

Request >> IIS >> aspnet_isapi.dll >> System.Web.dll >> Your codes

مشکل دیگری که وجود دارد این است که اگر تیمی بخواهد فریم ورکی برای برنامه نویسان نهایی فراهم کند، باید آنرا بر روی اکثر گزینه‌های هاست موجود سازگار کنید، برای مثال مشاهده می‌کنید که ASP.NET SignalR را هم می‌توان بر روی IIS و ASP.NET هاست کرد و هم بر روی یک App Domain کاملاً معمولی و علاوه بر این که تیم SignalR باید این هزینه مضاعف را پرداخت کند، خروجی برای ما نیز چندان خوشایند نیست، برای مثال اجرای همزمان ASP.NET SignalR و ASP.NET Web API اگر چه که بر روی هاستی به غیر از ASP.NET نیز امکان پذیر است، اما متأسفانه به عنوان دو بازیگر جدا از هم کار می‌کنند و عملاً تعاملی با یکدیگر ندارند، مگر این که بر روی ASP.NET هاست شوند، و یا بسیاری از امکانات Routing موجود در WCF بر روی بستری غیر از ASP.NET کار نمی‌کند. بدیهی است که این بازار پر آشوب به نفع هیچ کس نیست. و اما راه حل چیست ؟ تعدادی از برنامه نویسان حرفه ای .NET دور یکدیگر جمع شدند و طی بررسی هایشان به این نتیجه رسیدند که هاست‌های مختلف نقاط اشتراک بسیار زیادی دارند و تفاوت‌ها نباید باعث این میزان مشکل شود.

پس استانداری را طراحی کردند با نام [OWIN](http://owin.org) یا Open Web Interface for .NET

این استاندارد به صورت کاملاً ریز به طراحی هر چیزی را که شما به آن فکر کنید پرداخته است، Request, Cookie, Response, ... و Web Sokcet

اما همانطور که از نامش مشخص است این یک استاندارد است و پیاده سازی ندارد، و هر هاستی باید یک بار این استاندارد را بر

روی خود پیاده سازی کند

خبر خوش این است که تا این لحظه اکثر هاست‌های مهم این استاندارد را پیاده سازی کرده اند و یا در دست پیاده سازی دارند
 پروژه [Helios](#) برای IIS

پروژه [Katana](#) برای IIS به در کنار و سازگار با ASP.NET برای پروژه هایی که تا این لحظه از امکانات سطح پایین ASP.NET استفاده
 زیادی کرده اند و فرصت تغییر ساختاری ندارند

پروژه هایی برای App Domains و ...

مرحله‌ی بعدی این است که فریم ورک‌ها خوشان را با Owin سازگار کنند

معروف‌ترین فریم ورک هایی که تا این لحظه اقدام به انجام این کار کرده اند، عبارتند از:

ASP.NET Web API

ASP.NET MVC

ASP.NET Identity

ASP.NET Signal R (در حال حاضر Signal R فقط بر روی Owin قابلیت استفاده دارد)

بدیهی است که زمانی که پروژه ASP.NET Web API بر روی استاندارد OWIN نوشته می‌شود، دیگر نیازی به تحمل هزینه مضاعف
 برای سازگاری خود با انواع هاست‌ها ندارد و این مسئله توسط Katana, Helios و ... انجام شده است، که بالطبع بزرگترین نفع آن
 برای ما جلوگیری از چند باره کاری توسط تیم Web API و ... است که بالطبع در زمان کمتر امکانات بیشتری را به ما ارائه می‌دهند.
 البته واضح است فریم ورک هایی که با کلاینت و درخواست‌ها کاری ندارند، با این مقولات کاری ندارند، پس Entity Framework و ...
 از این داستان مستثنا هستند. و علاوه بر این فریم ورک هایی با طراحی اشتباه و قدیمی مانند ASP.NET Web Forms به صورت کلی
 قابلیت سازگار شدن با این استاندارد را ندارند، زیرا کاملاً به ASP.NET وابسته هستند

و در نهایت در مرحله‌ی بعدی لازم است شما نیز از فریم ورک هایی استفاده کنید که مبتنی بر OWIN هستند، یعنی برای مثال پروژه
 بعدی تان را مبتنی بر ASP.NET MVC و ASP.NET Web API و ASP.NET Identity پیاده سازی کنید، در این صورت شما می‌توانید برنامه
 ای بنویسید که به Web Server هیچ گونه وابستگی ندارد.

به این صورت کد زدن چند مزیت بزرگ دیگر هم دارد که از کم اهمیت‌ترین آنها شروع می‌کنیم:

1- سرعت بسیار بالاتر برنامه در هاست‌های غیر ASP.NET ای، مانند زمانی که شما از IIS به صورت مستقیم و بدون وابستگی به
 System.Web.dll استفاده می‌کنید.

توجه کنید که حتی در این حالت هم می‌توانید از ASP.NET Web API و Signal R و Identity استفاده کنید و تا 25% سرعت
 بیشتری داشته باشید (بسته به سناریو) 2- قابلیت توسعه آسانتر و با قابلیت نگهداری بالاتر پروژه‌های Enterprise، برای مثال در
 یکی از پروژه‌ها من مجبور بودم از ASP.NET Web API به صورتی استفاده کنم که هم توسط کلاینت JavaScript ای استفاده شود، و
 هم توسط کدهای Controller های MVC (بدون استفاده مستقیم از کد سرویس با رفرنس زدن به سرویس‌ها البته) که خوشبختانه

OWIN به خوبی از پس این کار بر آمد، و عملاً یک سرویس Web API را هم بر روی IIS هاست کردم و هم داخل یک AppDomain
 3- در چند سال آینده که اکثریت مطلق سایت‌ها از این روش استفاده کنند (شما چه بدانید و چه ندانید اگر در برنامه خودتان از
 Signal R نسخه 2 دارید استفاده می‌کنید، حتماً از OWIN استفاده کرده اید)، میکروسافت می‌تواند دست به تغییرات اساسی‌تری
 بزند، برای مثال معماری جدیدی از IIS ارائه دهد که مشکلات ساختاری فراوان فعلی IIS را که از حوصله توضیح این مقاله خارج
 است را نداشته باشد، و فقط یک پیاده سازی OWIN جدید بر روی آن ارائه دهد و برنامه‌های ما بدون تغییر بر روی آن نیز کار کنند،
 و یا این که بتواند تعدادی از فریم ورک‌های با طراحی قدیمی را راحت‌تر از دور خارج کند، مانند Web Forms

نکته پایانی، اگر هم اکنون پروژه ای دارید که در داخل آن از ASP.NET استفاده شده، و برای مثال تعدادی فرم ASP.NET Web Forms
 نیز دارد، نگران نباشید، کماکان می‌توانید از Owin برای سایر قسمت‌ها مانند Web API استفاده کنید، البته در این حالت تأثیری در
 بهبود سرعت اجرای برنامه مشاهده نخواهید کرد، اما برای مهاجرت و اعمال تغییرات این آسانترین روش ممکن است در قسمت
 بعدی، مثالی را شروع می‌کنیم مبتنی بر ASP.NET Web API، ASP.NET Identity و Helios

نظرات خوانندگان

نویسنده:

ناظم

تاریخ:

۱۰:۵۱ ۱۳۹۲/۱۲/۰۳

سلام

ممنون بابت مطلب مفیدتون.

بدون وابستگی به IIS یعنی هر web server ی که OWIN را پیاده سازی کند امکان اجرای برنامه هایی که مثلا با asp.net mvc نوشته شدن رو خواهند داشت؟

همین که مثلا با asp.net mvc برنامه نوشته شده به معنی این هست که برنامه بر اساس استاندارد OWIN هست؟ یا کارهایی برای این منظور باید انجام داد؟

نویسنده:

مسعود پاکدل

تاریخ:

۱۱:۴۸ ۱۳۹۲/۱۲/۰۳

بدون وابستگی به IIS یعنی شما امکان هاست کردن سرویس‌های Web API رو به صورت Windows Service یا پروژه Console هم خواهید داشت.

به صورت پیش فرض یک پروژه MVC بدون وابستگی به Owin پیاده سازی می‌شود و برای این منظور می‌توانید یکی از موارد زیر را انجام دهید:

«امکان هاست سرویس‌ها روی IIS. در این صورت Owin فقط به صورت یک Middleware عمل خواهد نمود و در این حالت دیگر نیاز به نوشتن HttpModule ها نخواهید داشت. البته این روش به System.Web وابستگی دارد (Microsoft.Owin.Host.SystemWeb)»
 «استفاده از OwinHost.Exe که در واقع یک پیاده سازی دیگر برای Owin است و عملیات bootstrapping را بر عهده خواهد داشت. در نتیجه شما فقط موارد مربوط به middleWare در application انجام خواهید داد.»
 «استفاده از Owin Self Hosting برای هاست سرویس‌ها در قالب برنامه Console یا Windows Service (Microsoft.Owin.Host.HttpListener)»

نویسنده:

یاسر مرادی

تاریخ:

۱۲:۱۳ ۱۳۹۲/۱۲/۰۳

بله، به همین معنی است

البته دقت کنید، پیاده سازی OWIN کار ساده ای نیست، و به سرعت نمی‌توان شاهد پیاده سازی آن بر روی هاست‌های مختلف بود، و این پروسه با سرعت فعلی از نظر من مدتی طول خواهد کشید.

برای مثال Katana که یک پیاده سازی قابل استفاده و خوب از آن به شمار می‌رود کار شرکت مایکروسافت است و سایر پیاده سازی Open Source سایر تیم‌ها که بالطبع امکان مانور شرکت مایکروسافت را ندارند، کمی طول می‌کشد تا واقعا آماده استفاده شود.

و همچنین پیاده سازی‌های فعلی در قسمت هایی مانند Web Socket ها و سایر مسائل پیچیده دارای ضعف هایی هستند.

درست مانند استاندارد HTML 5 که بر روی مرورگرهای مختلف به میزان‌های مختلفی پیاده سازی شده است.

به بیان دیگر پیاده سازی OWIN صفر و صدی نیست، بلکه هر پیاده سازی ممکن است در داخل خود دارای ضعف‌ها و یا نواقصی باشد.

علاوه بر این اگر شما در کد نویسی ASP.NET MVC خود، بی جهت به امکانات پایه ASP.NET ایجاد وابستگی کنید، نیز در این عمل دچار مشکل خواهید شد، برای همین بدیهتا کاری را که می‌توانید با Action Filter انجام دهید را نباید با یک Http Module انجام دهید و ...

مهم‌ترین کار طراحی برنامه هایی که می‌نویسید به صورت سازگار با OWIN است که در پست‌های بعدی قرار است به همین قسم از مطالب بپردازیم

البته من آینده خوبی برای OWIN قائلم، و نفع آن در کوتاه مدت و بلند مدت کاملا آشکار و واضح است، کما این که در مطلب به آن اشاره شد.

برای مشاهده پیاده سازی های مختلف OWIN می توانید به سایت owin.org مراجعه کنید.
موفق و پایدار باشید

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۲/۱۲/۰۳ ۱۹:۲۶

ممنون از پاسختون، البته این رو در نظر داشته باشید که استفاده از IIS به همراه Owin لزوماً به پیاده سازی Katana یا همان Microsoft.Owin.Host.SystemWeb وابسته نیست، در این حالت شما هیچ گونه بهبود سرعتی رو مشاهده نخواهید کرد و حتی به علت اضافه شدن Owin Middleware بر روی ASP.NET حتی کندتر هم خواهید شد، این حالت فقط برای پروژه هایی توصیه می شود که با استفاده از مواردی مانند Module های ASP.NET و یا Web form به ASP.NET وابسته اند. برای پروژه های جدید استفاده از Helios که نه به System.Web احتیاجی دارد و نه به Owin.Host.SystemWeb توصیه می شود، به همراه Web API ، SignalR و MVC، که به نظر من این ۳ آنقدر کامل و کافی هستند که لزومی به استفاده از ASP.NET System.Web.dll و پیاده سازی Owin مربوطه ای که نام بردید نباشد، تا بتوان بیشتر از مزایای Owin به خصوص کارامدی بیشتر برنامه ها بهره برد

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۰۳ ۲۱:۵۵

ممنونم.
در حال حاضر من استفاده از helios رو پیشنهاد نمی کنم چون اولین محدودیتی که در helios جلب توجه می کند Minimum system requirements مورد نظر است.
برای توسعه پروژه های helios :
« Windows 8 یا Windows Server 2012
« NET Framework 4.5.1
« Visual Studio 2012 یا Visual Studio 2013
و برای Web Server نیز :
« Windows Server 2012
« NET Framework 4.5.1
« Full trust مورد نیاز است.
البته به گفته تیم توسعه پروژه helios، احتمال رفع این محدودیت ها در آینده وجود دارد. در نتیجه به نظر من
Microsoft.AspNet.WebApi.OwinSelfHost گزینه بهتری برای Owin Self Hosting است و از آن جا که در حالت Owin Self Hosting هیچ گونه وابستگی به IIS و البته System.Web نیز وجود ندارد در نتیجه مشکل performance نیز برطرف خواهد شد.

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۲/۱۲/۰۴ ۰:۲

روش برنامه نویسی مایکروسافت بیش از دو سالی می شود که به این شکل شده است که هر امکان و قابلیت جدیدی بر روی آخرین نسخه NET Framework ارائه می شود و البته سپس به نسخ قبلی نیز تعمیم می یابد، در همین جا است که می بینید اکثر امکانات 5 & 6 Entity Framework ابتدا بر روی NET Framework 4.5 ارائه شدند، و سپس بر روی 4
اگر ما بخواهیم به NET Framework به عنوان یک پیش نیاز دردرس را نگاه کنیم، در اولین قدم خودمان را به دردرس انداخته ایم، چون نه برای Helios، بلکه برای صدها امکان دیگر مانند Data Flow های جدید و ... نیز باید صبر کنیم، که عملاً هزینه به فایده آن نمی صرفد. پس همیشه با فراغ بال از آخرین نسخه NET Framework استقبال کنیم
نکته ای دیگر را که باید مد نظر داشته باشیم، این است که مطابق با سیاست هایی که مایکروسافت جدیداً اتخاذ کرده است، دیگر نباید خیلی نگران نسخه های جدید NET Framework باشیم، چون دیگر از آن نسخه دهی های پشت سر هم و با حجم تغییرات بالا خبری نیست، بلکه اکثر فریم ورک های مهم جدا از NET ارائه و به روز رسانی می شوند.
علاوه بر این، ارتقا به آخرین نسخه سیستم عامل ویندوز نیز به هیچ وجه مانند قبل (IIS 6 به IIS 7) دردرس را نیست، و خوشبختانه این ارتقا (و یا تغییر هاست) بدون دردرس است.

به نظر من این ارتقاء را انجام دهید، چون نه فقط Helios که خیلی چیزهای دیگری را دارید از دست می‌دهید، مانند سرعت بالاتر توسعه برنامه بر روی Visual Studio 2013 و Windows 8.1 برای توسعه برنامه‌های وبی، سرعت و کارآمدی بسیار بالاتر NET Framework 4.5.1. با IIS 8.5 برای مشتری‌های برنامه و ...

به نظر من آنقدر این ارتقاء ارزشمند است، که ارزش Helios این میان کمتر ارزشش به چشم می‌آید.

یکی از دلایلی که برنامه‌های سمت وب به سرعت بر برنامه‌های دسکتاپی قدیمی چیره شدند، همین است: امکان ارتقای سرورها در مدت زمان کم و به شکل مدیریت شده و با کمترین تاثیر روی مشتری‌های نهایی، بارها این تصمیم را که در ابتدایش کمی سخت به نظر می‌آید را گرفته ام و در نهایت از مشتری تا برنامه نویس همه را راضی دیده ام، چون هیچ کسی از امکانات جدید که بدون دردسر حاصل شود بدش نمی‌آید، و خوشبختانه کیفیت محصولات مایکروسافت واقعا بهبود یافته و دیگر آن زمانی که از NET 2. به 3.5 می‌رفتیم و گرفتار چندین مشکل می‌شدیم گذشته است.

از این نگذريد که بالاخره روزی باید این مهاجرت‌ها را انجام دهید، پس چه بهتر که از سود آن زودتر بهره مند شوید، البته بی مهابا عمل کردن توصیه نمی‌شود، بد نیست زمانی شروع به ارتقاء کنید که صفحه Release Notes و سوالات موجود در سایت Stack over flow در رابطه با اشکالات رخ داده در زمان ارتقاء و Breaking Changes را از بر باشید، به این صورت عمل کنید تماما برد کرده اید.

شاید شما هم قصد داشته باشید تا از برخی درخواست ها به وب سایت یا اپلیکیشن خود ممانعت عمل بیاورید. نظیر درخواست های SQL Injection یا برخی Query String های خاص یا برخی درخواست های مزاحم.

یکی از مزاحمت هایی که گریبانگیر وب سایت هاست، Bot های متفاوتی است که برای کپی اطلاعات، درج کامنت به صورت خودکار و مواردی از این دست، به آنها مراجعه میکنند. شاید در نگاه اول بد نباشد که این Bot ها به سراغ وب سایت ما بیایند و باعث افزایش تعداد ویزیت سایتمان شوند؛ ولی ضررهای ناشی از کپی و سرقت مطالب سایت، آنهم با سرعت بالا، بیشتر از منافع ناشی از بالا رفتن رنک سایت است. به طور مثال همین سایت NET Tips دارای تعداد زیادی مقالات مفید است که افراد متعددی در نگارش و تهیه آنها زحمت کشیده اند، یا وب سایتی برای جلب اعتماد مشتریان جهت درج اطلاعاتشان و یا آگهی هایشان زحمت زیادی کشیده است، Bot های آماده ی زیادی وجود دارد که با چند دقیقه صرف وقت جهت تنظیم شدن آماده میشوند تا مطالب را طبق ساختار تعیین شده، مورد به مورد کپی کنند.

برای خلاصی از این موارد روش های متعددی وجود دارد که از جمله آنها می توان به تنظیمات فایل htaccess در وب سرورهایی نظیر Apache و یا web.config در IIS اشاره کرد. در این مقاله این امکان را با IIS مرور میکنیم و برای فعال سازی آن کافی است در:

IIS 7.5 و بالاتر، همراه با انتخاب Request Filtering در مراحل نصب IIS

IIS 7.0 پس از نصب بسته آپدیت [Microsoft Knowledge Base Article 957508](https://support.microsoft.com/kb/957508).

IIS 6.0 با نصب URLScan 3.0

در بخش <system.webServer> و سپس <security>، تگ requestFiltering را استفاده کنیم، در این تگ دستورالعمل های ویژه ی پالایشگر درخواست ها را مینویسیم (filteringRules) هر دستورالعمل پالایش دارای خصیصه های (Attributes) زیر است:

denyUnescapedPercent

مقدار Boolean و انتخابی

اگر برابر با true تنظیم گردد، درخواست هایی که دارای کاراکتر "درصد" (%) هستند و به وسیله escape character ها پوشش داده نشده باشند، رد می شوند. (جهت جلوگیری از حملات XSS و...) مقدار پیش فرض true است.

name

عنوان دستورالعمل.

مقدار پیش فرض نداشته و درج کردن آن اجباری است.

scanAllRaw

مقدار Boolean و انتخابی

اگر برابر با true تنظیم گردد، پالایشگر درخواست ها موظف است تا با بررسی متن header های درخواست، در صورت یافتن یکی از واژه هایی که در خصیصه denyStrings ذکر کرده اید، درخواست را رد کند. مقدار پیش فرض false است.

scanQueryString

مقدار Boolean و انتخابی

اگر برابر با true تنظیم گردد، پالایشگر درخواستها موظف است تا Query string را بررسی کند تا در صورتی که یکی از واژه‌های درج شده در خصیصه denyStrings را بیابد، درخواست را رد کند.

اگر خصیصه‌ی unescapeQueryString از تگ < requestFiltering > برابر با true باشد، query string دوبار بررسی می‌شود: یکبار متن query string برای یافتن عبارات ممنوعه و بار دیگر برای یافتن کاراکترهای بدون پوشش scaped. مقدار پیشفرض false است.

scanUrl

مقدار Boolean و انتخابی

اگر برابر با true تنظیم گردد، پالایشگر درخواستها URL را برای یافتن واژه‌های ممنوعه‌ی ذکر شده در خصیصه denyStrings بررسی می‌نماید. مقدار پیش فرض false است.

چند مثال:

مثال 1: در این مثال عنوان User-Agent هایی را که در موارد متعدد برای وب سایت هایی که روی آنها کار می‌کردم مزاحمت ایجاد میکردند را پالایش میکنیم. (لیست این Bot ها آپدیت میشود)

```
<requestFiltering>
  <filteringRules>
    <filteringRule name="BlockSearchEngines" scanUrl="false" scanQueryString="false">
      <scanHeaders>
        <clear />
        <add requestHeader="User-Agent" />
      </scanHeaders>
      <appliesTo>
        <clear />
      </appliesTo>
      <denyStrings>
        <clear />
        <add string="Python Urllib" />
        <add string="WGet" />
        <add string="Apache HttpClient" />
        <add string="Unknown Bot" />
        <add string="Yandex Spider" />
        <add string="libwww-perl" />
        <add string="Nutch" />
        <add string="DotBot" />
        <add string="CCBot" />
        <add string="Majestic 12 Bot" />
        <add string="Java" />
        <add string="Link Checker" />
        <add string="Baiduspider" />
        <add string="Exabot" />
        <add string="PHP" />
      </denyStrings>
    </filteringRule>
  </filteringRules>
</requestFiltering>
```

مثال 2: ممانعت از SQL Injection

```
<requestFiltering>
  <filteringRules>
    <filteringRule name="SQLInjection" scanUrl="false" scanQueryString="true">
      <appliesTo>
        <clear />
        <add fileExtension=".asp" />
        <add fileExtension=".aspx" />
        <add fileExtension=".php" />
      </appliesTo>
      <denyStrings>
        <clear />
        <add string="--" />
        <add string=";" />
      </denyStrings>
    </filteringRule>
  </filteringRules>
</requestFiltering>
```

```

        <add string="*" />
        <add string="@" />
        <add string="char" />
        <add string="alter" />
        <add string="begin" />
        <add string="cast" />
        <add string="create" />
        <add string="cursor" />
        <add string="declare" />
        <add string="delete" />
        <add string="drop" />
        <add string="end" />
        <add string="exec" />
        <add string="fetch" />
        <add string="insert" />
        <add string="kill" />
        <add string="open" />
        <add string="select" />
        <add string="sys" />
        <add string="table" />
        <add string="update" />
    </denyStrings>
    <scanHeaders>
        <clear />
    </scanHeaders>
</filteringRule>
</filteringRules>
</requestFiltering>

```

مثال 3: ممانعت از درخواست انواع خاصی از فایل ها

```

<requestFiltering>
    <filteringRules>
        <filteringRule name="Block Image Leeching" scanUrl="false" scanQueryString="false"
scanAllRaw="false">
            <scanHeaders>
                <add requestHeader="User-agent" />
            </scanHeaders>
            <appliesTo>
                <add fileExtension=".zip" />
                <add fileExtension=".rar" />
                <add fileExtension=".exe" />
            </appliesTo>
            <denyStrings>
                <add string="leech-bot" />
            </denyStrings>
        </filteringRule>
    </filteringRules>
</requestFiltering>

```

[اطلاعات بیشتر در وب سایت رسمی IIS](#)

نظرات خوانندگان

نویسنده: امین مصباحی
تاریخ: ۱۳۹۳/۰۲/۰۱ ۶:۳

تکمیلی 1: AhrefsBot هم از جمله ی Bot های مزاحم است، لذا:

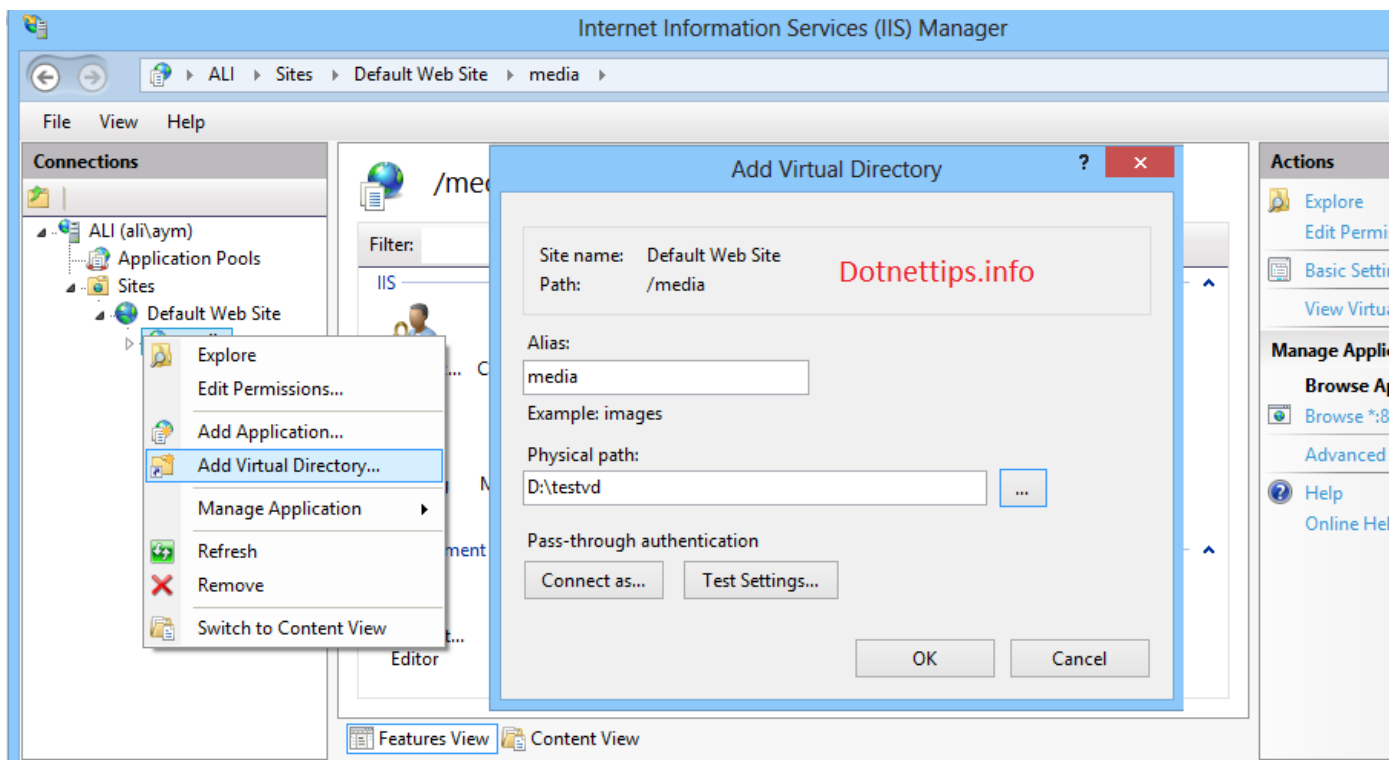
```
</ "add string="AhrefsBot">
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۱ ۹:۳۵

لیست User-Agent هایی است که من در این سایت بستم تا امروز (از لاگ های خطای برنامه استخراج شدند):
[bots.txt](#)

نمیدانم آیا تا به حال برای شما پیش آمده است که بخواهید اطلاعاتتان را در جایی غیر از زیرشاخه‌های wwwroot ذخیره نمایید یا خیر؟

یادم هست برای یکی از مشتریانم یک سرور خریده بودیم که دو پارتیشن داشت و آن موقع به ذهنم خطور کرد که اگر بخواهم مثلاً فایل‌های سیستم مدیریتی را داخل یک پارتیشن دیگر قرار بدهم چگونه انجام می‌شود؛ چطوری میتوانم به مکانی غیر از شاخه‌ی wwwroot، عمل mappath را انجام بدم؟ چگونه میتوانم یک لینک مستقیم، به مکانی دیگر داشته باشم؟ جواب تمام این سوالات در امکانی از IIS به اسم virtual Directory بود. در این روش ما یک مکان فیزیکی را به iis معرفی می‌کنیم و به آن یک نام مجازی می‌دهیم که از آن در برنامه استفاده خواهیم کرد. در iis، در بخش sites، همه سایت‌های ایجاد شده بر روی IIS لیست می‌شوند. یک context menu بر روی سایت مورد نظر خود باز کنید و گزینه‌ی add virtual directory انتخاب کنید. یک نام مجازی به آن بدهید و مکان مورد نظر را انتخاب کنید و کادر را تایید کنید.



برای ویرایش آن یعنی تغییر مسیر فیزیکی هم میتوان از طریق مسیر زیر عمل کرد

Right Click On virtual Directory>Manage Virtual Directory >Advanced Settings

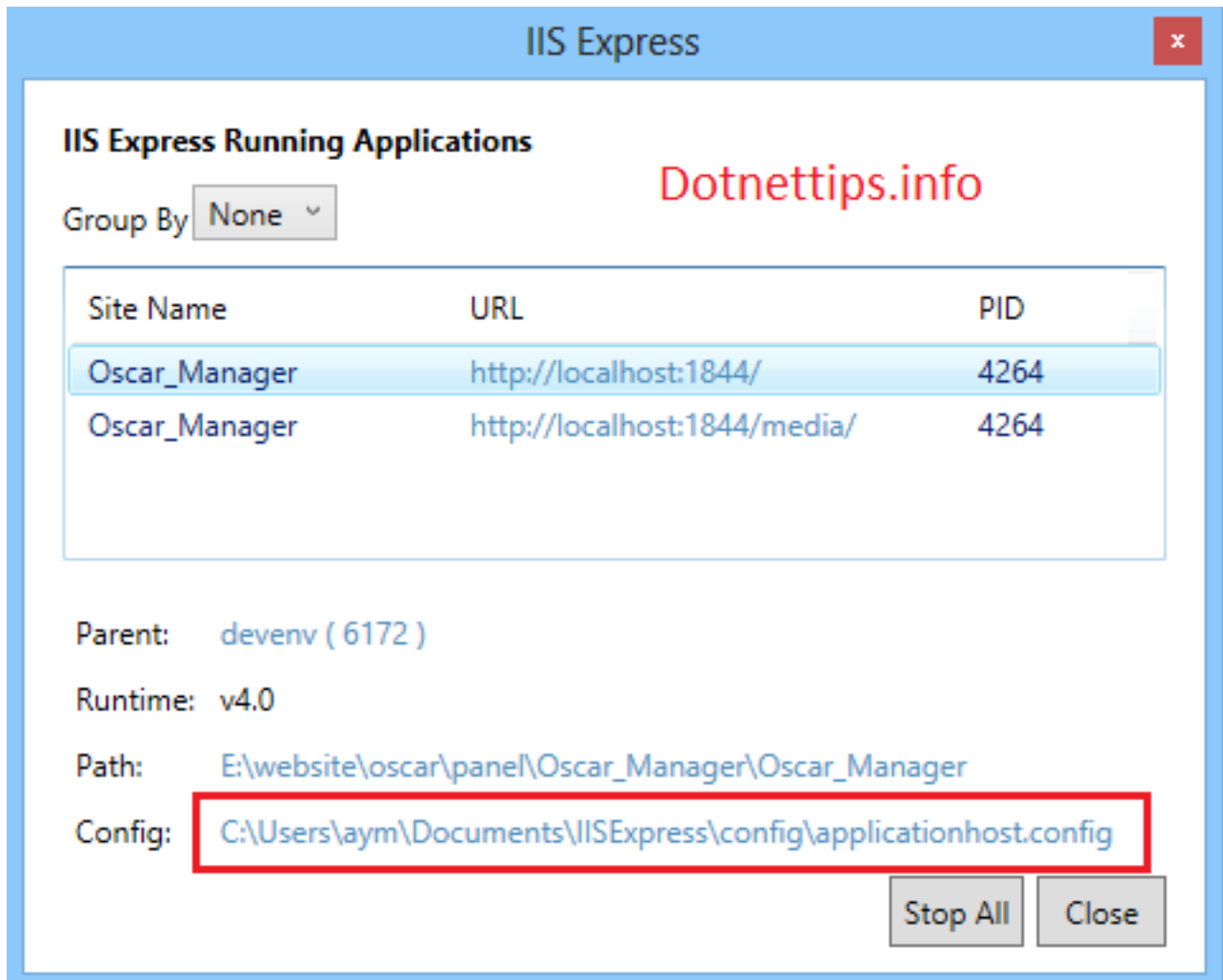
از این پس در IIS دسترسی به پوشه، از طریق ~media میسر هست؛ ولی بسیاری از ما برای تست، برنامه را از طریق IIS Express اجرا و تست می‌کنیم. پس بهتر هست این گزینه در آنجا هم مورد بررسی قرار بگیرد.

برای دسترسی به کانفیگ iis express عموماً مسیرهای زیر معتبر هستند:

```
%userprofile%\documents\iisexpress\config\applicationhost.config
%userprofile%\my documents\iisexpress\config\applicationhost.config
```

فایل applicationhost.config فایلی است که قرار است تغییر بدهیم.

ولی اگر مسیرهای بالا راهگشا نبود، برای پیدا کردن فایل‌های کانفیگ می‌توانید از طریق آیکن IIS Express که حین اجرای پروژه در notification area باز می‌شود نیز اقدام کرد.



یک context menu از طریق این آیکن باز کرده و گزینه‌ی show all applications را انتخاب کنید. لیست تمامی سایت‌های در حال اجرا نمایش داده می‌شود. یکی را انتخاب کنید تا در زیر اطلاعات نمایش یابد. در قسمت کانفیگ، آدرس فایل کانفیگ داده شده است و مسیر نیز مشخص است. با کلیک بر روی آن فایل applicationhost.config باز می‌شود.

فایل مورد نظر را باز کنید. این فایل را می‌توان با نوت پد یا یک xml editor گشود. در آن یک تگ sites وجود دارد که تمامی پروژه‌های تحت وبی را که تا الان دارید، درون خودش ذخیره کرده‌است. به ازای هر سایت یک تگ site هست و خصوصیات هر کدام، داخل این تگ قرار گرفته‌است. اگر دقت کنید هر پروژه شما یا همان تگ site، شامل تگ زیر می‌باشد:

```
<application path="/" applicationPool="Clr4IntegratedAppPool">
  <virtualDirectory path="/"
  physicalPath="E:\website\oscar\panel\Oscar_Manager\Oscar_Manager" />
</application>
```

در اینجا خود IIS Express اقدام به ساخت یک دایرکتوری مجازی که همان مسیر ذخیره پروژه هست کرده. برای معرفی دایرکتوری مجازی جدید، یک کپی از تگ application را ایجاد کنید.

برای مثال من قصد دارم یک دایرکتوری مجازی به اسم media بسازم تا تصاویر و دیگر فایل‌های چندرسانه‌ای را در آن ذخیره کنم و محل فیزیکی آن نیز D:\testvd می‌باشد. پس تگ کپی شده را به نحو زیر تغییر می‌دهم:

```
<application path="/media" applicationPool="Clr4IntegratedAppPool">
  <virtualDirectory path="/" physicalPath="d:\testvd" />
</application>
```

بنابراین در کل، تگ site من به شکل زیر تغییر پیدا می‌کند:

```
<site name="Oscar_Manager" id="23">
  <application path="/" applicationPool="Clr4IntegratedAppPool">
    <virtualDirectory path="/"
physicalPath="E:\website\oscar\panel\Oscar_Manager\Oscar_Manager" />
  </application>
  <application path="/media" applicationPool="Clr4IntegratedAppPool">
    <virtualDirectory path="/" physicalPath="d:\testvd" />
  </application>
  <bindings>
    <binding protocol="http" bindingInformation="*:1844:localhost" />
  </bindings>
</site>
```

الان مسیر مجازی ما ساخته شده است. برای تست صحت این کار، یک فایل تصویری را در آن قرار می‌دهم و در کنترلر مورد نظر، این کد را می‌نویسم تا یک تصویر به سمت کلاینت در یک virtual directory ارسال شود.

```
var dir = Server.MapPath("~/media");
var path = System.IO.Path.Combine(dir, "1.jpg");
return base.File(path, "image/jpeg");
```

کنترلر را صدا زده تا نتیجه کار را ببینید.

همانطور که حتما متوجه شدید IIS Express محیط GUI ندارد. البته مدتی است افزونه‌ای برای این کار محیا شده ولی بیشتر کاربرد آن ایجاد یک سایت جدید و اجرا و توقف IIS می‌باشد که می‌توانید آن را از [اینجا](#) دریافت نمایید.

نکته: البته بنده این نکته را تایید نمی‌کنم ولی شنیده‌ام که در نسخه‌های بالاتر ویژوال استادیو با راست کلیک بر روی نام پروژه، گزینه Use IIS Express وجود دارد که به یک محیط گرافیکی ختم می‌شود و از آنجا که بنده نسخه 2012 را دارم این مورد را تست نکردم.

ایجاد virtual Directory از طریق Appcmd

دسترسی به appcmd از طریق مسیر زیر امکان پذیر است:

```
%systemroot%\system32\inetsrv\AppCmd.exe
```

این دستور به تمامی اشیاء سرور، از قبیل سایت‌ها و اپلیکیشن‌ها دسترسی دارد و میتواند هر نوع مدتی را بر روی اشیاء سرور، از قبیل ثبت، ویرایش و حذف را انجام دهد.

یکی از این دستورات، برای ساخت Virtual Directory استفاده می‌شود:

```
APPCMD add vdir /app.name:"default we site/" /path:/vdir1 /physicalPath:d:\testvd
```

سوییچ app/ برای نام وب سایت یا پروژه است و حتما در انتهای نام، علامت / قرار گیرد. سوییچ بعدی هم که path/ هست برای دادن مسیر مجازی و سوییچ آخری هم آدرس محل فیزیکی است. بعد از اجرای دستور، پیام زیر نمایش داده می‌شود:

```
VDIR object "Default Web Site/vdir1" added
```

پنجره commandprompt باید به صورت *Run as Administrator* باز شده باشد.

برای تغییر محل فیزیکی یک virtual directory میتوان از دستور زیر استفاده کرد:

```
appcmd.exe set vdir "Default Web Site/vdir1/" -physicalPath:"D:\Files"
```

از این پس مسیر فیزیکی این آدرس مجازی مسیر D:\Files خواهد بود.

البته مباحث پیشرفته‌تری چون application pool ها نیز وجود دارد که از حوصله این مقاله خارج هست و در وقت و مقاله دیگر بررسی خواهیم کرد.

برای ارسال دستور به IIS Express هم از طریق مسیر زیر appcmd را باز کنید:

```
%ProgramFiles%\IIS Express\appcmd.exe
```

امکان ایجاد virtual directory عموماً در سرورهای مجازی و اختصاصی در پنل مربوطه نیز وجود دارد.

[ساخت virtual Directory در وب سایت پنل](#)

[ساخت virtual Directory در پلسک](#)

آشنایی با Virtual Address spaces

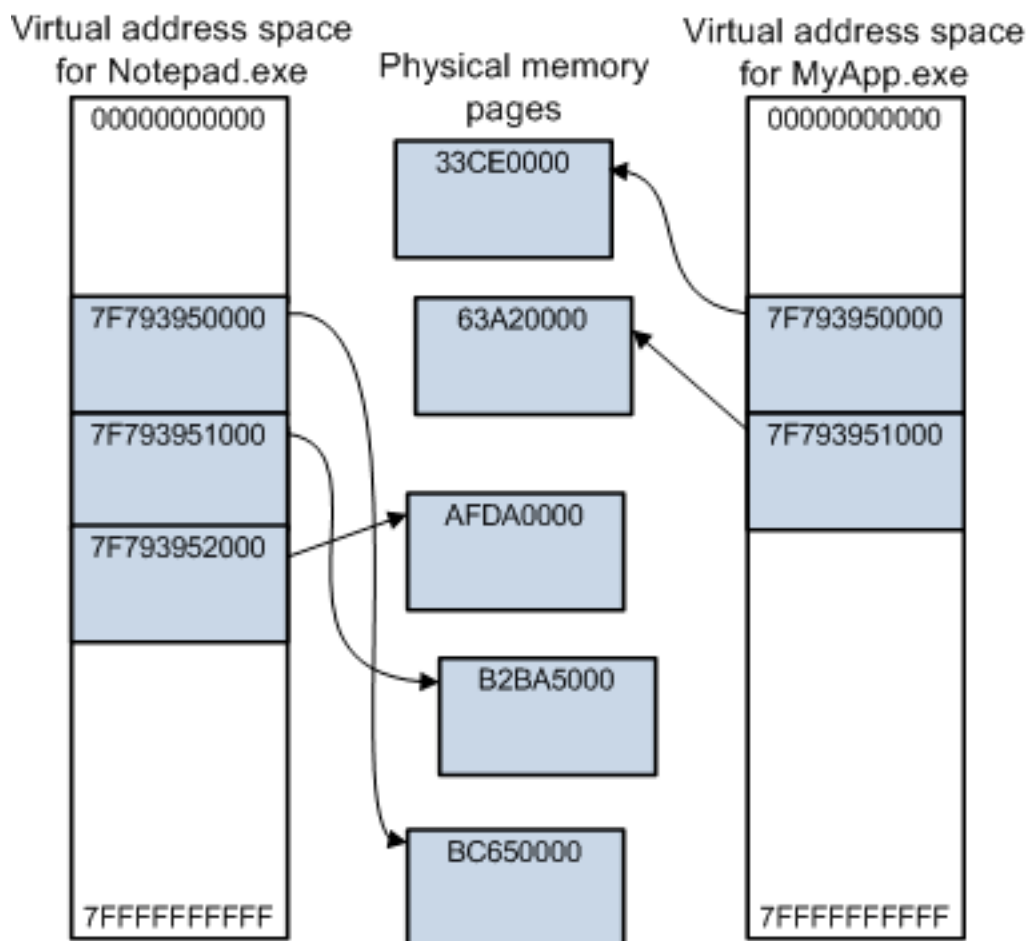
فضای آدرس‌دهی مجازی: موقعی که یک پردازشگر در مکانی از حافظه عمل خواندن و نوشتن را آغاز می‌کند، از آدرس‌های مجازی بهره می‌برد. بخشی از عملیات خواندن و نوشتن، تبدیل آدرس‌های مجازی به آدرس‌های فیزیکی در حافظه است. این عمل سه مزیت دارد:

آدرس‌های مجازی به صورت پیوسته و پشت سر هم هستند و آدرس دهی بسیار راحت است ولی داده‌ها بر روی یک حافظه به صورت متصل به هم یا پیوسته ذخیره یا خوانده نمی‌شوند و کار آدرس دهی مشکل است. پس یکی از مزایای داشتن آدرس دهی مجازی پشت سر هم قرار گرفتن آدرس هاست.

برنامه از آدرس‌های مجازی برای دسترسی به بافر حافظه استفاده می‌کند که بزرگتر از حافظه فیزیکی موجود هست. موقعی که نیاز به حافظه بیشتر باشد و حافظه سیستم کوچکتر یا کمتر از تقاضا باشد، مدیر حافظه، صفحات حافظه فیزیکی را به صورت یک فایل (عموما 4 کیلویی) بر روی دیسک سخت ذخیره می‌کند و صفحات داده‌ها در موقع نیاز بین حافظه فیزیکی و دیسک سخت جابجا می‌شود.

هر پردازشی که بر روی آدرس‌های مجازی کار می‌کند ایزوله شده است. یعنی یک پروسه هیچ گاه نمیتواند به آدرس‌های یک پروسه دیگر دسترسی داشته باشد و باعث تخریب داده‌های آن شود.

به محدوده شروع آدرس‌های مجازی تا پایان آن محدوده، فضای آدرس‌دهی مجازی گویند. هر پروسه ای که در مد کاربر آغاز میشود از یک فضای آدرس خصوصی یا مختص به خود استفاده می‌کند. برای سیستم‌های 32 بیتی این فضا میتواند دو گیگ باشد که از آدرس 0x00000000 شروع می‌شود و تا 0x7FFFFFFF ادامه پیدا می‌کند و برای یک سیستم 64 بیتی تا 8 ترابایت می‌باشد که از آدرس 0x000'00000000 تا آدرس 0x7FF'FFFFFFFF ادامه می‌یابد. گاهی اوقات به محدوده آدرس‌های مجازی، حافظه مجازی می‌گویند. شکل زیر اصلی‌ترین خصوصیات فضای آدرس‌های مجازی را نشان می‌دهد:



در شکل بالا دو پروسه 64 بیتی به نام‌های *notepad.exe* و *myapp.exe* قرار دارند که هر کدام فضای آدرس‌های مجازی خودشان را دارند و از آدرس 0x000'0000000 شروع و تا آدرس 0x7FF'FFFFFFFF ادامه می‌ابند. هر قسمت شامل یک صفحه 4 کیلویی از حافظه مجازی یا فیزیکی است. به برنامه نوت‌پد دقت کنید که از سه صفحه پشت سر هم یا پیوسته تشکیل شده که آدرس شروع آن 0x7F7'93950000 می‌باشد ولی در حافظه فیزیکی خبری از پیوسته بودن دیده نمی‌شود و حتما این نکته را متوجه شدید که هر دو پروسه از یک آدرس شروع استفاده کرده‌اند، ولی به آدرسی متفاوت از حافظه فیزیکی نگاشت شده‌اند.

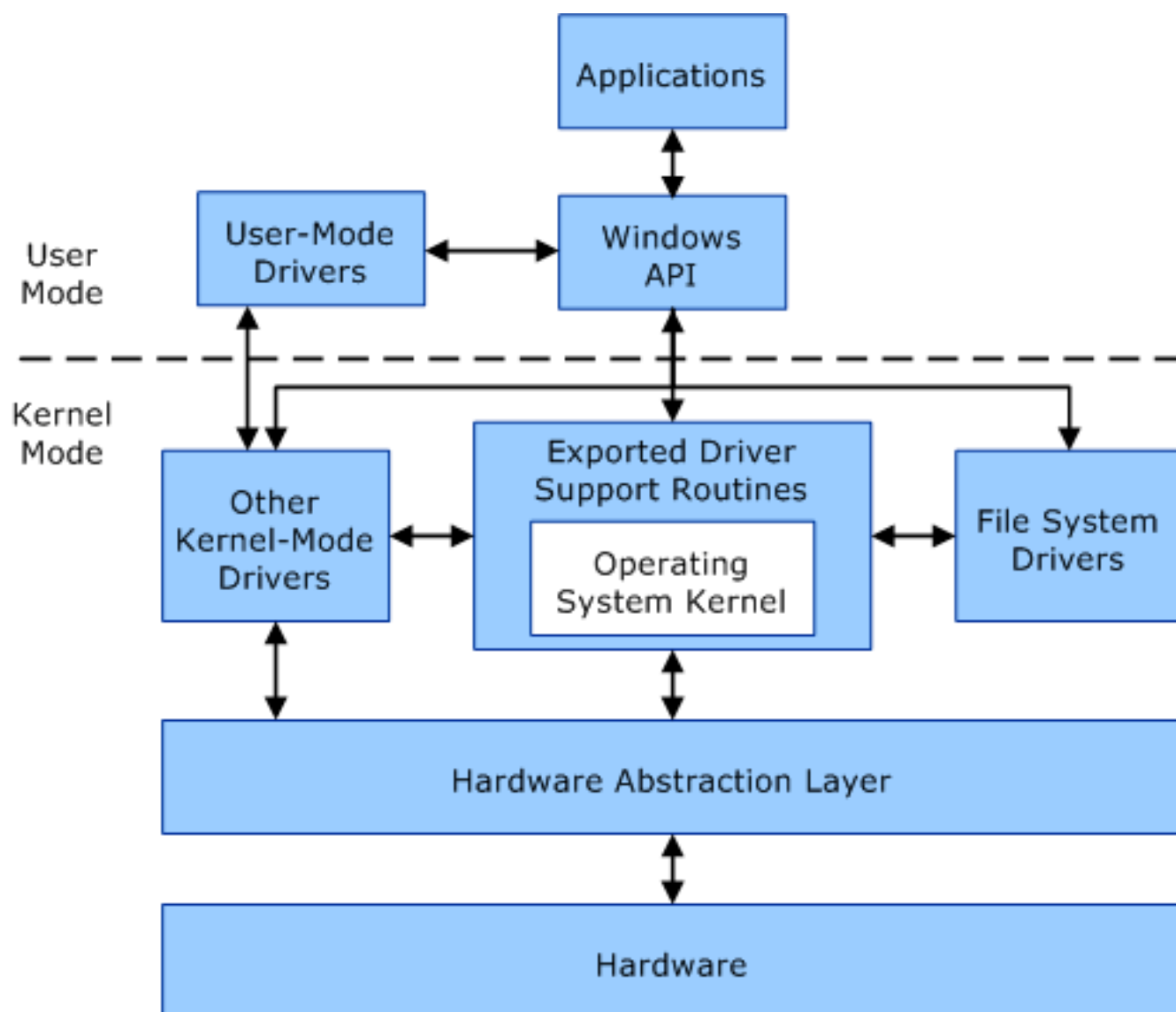
تفاوت kernel mode و user mode

هر پردازش در سیستم بر اساس *user mode* مد کاربر یا *kernel mode* مد کرنل اجرا می‌شود. پردازش‌ها بر اساس هر نوع کد بین این دو بخش سوییچ می‌کنند. اپلیکیشن‌ها بر اساس مد کاربر و هسته سیستم عامل و اکثر درایورها بر اساس مد کرنل کار می‌کنند؛ ولی تعدادی از آن‌ها هم در مد کاربر.

هر برنامه یا اپلیکیشنی که اجرا می‌شود، در یک مد کاربری قرار می‌گیرد. ویندوز هم برای هر برنامه یک پروسه یا فرآیندی را ایجاد می‌کند. پروسه برای برنامه یک فضای آدرس‌دهی مجازی و یک جدول مدیریت به صورت خصوصی یا مختص همین برنامه تشکیل می‌دهد. به این ترتیب هیچ برنامه دیگری نمی‌تواند به داده‌های برنامه دیگر دسترسی داشته باشد و هر برنامه در یک محیط ایزوله شده برای خودش قرار می‌گیرد و این برنامه اگر به هر ترتیبی کرش کند، برنامه‌های دیگر به کار خود ادامه می‌دهند و هیچ تاثیری بر برنامه‌های دیگر نمی‌گذارند.

البته استفاده از این آدرس‌های مجازی محدودیت‌هایی هم دارد، چرا که بعضی از آن‌ها توسط سیستم عامل رزرو شده‌اند و برنامه نمی‌تواند به آن قسمت‌ها دسترسی داشته باشد و این باعث می‌شود که داده‌های برنامه از خسارت و آسیب دیدن حفظ شوند. تمام برنامه‌هایی در حالت کرنل ایجاد می‌شوند، از یک فضای آدرس مجازی استفاده می‌کنند. به این معنی که یک درایور مد کرنل نسبت به دیگر درایورها و خود سیستم عامل به هیچ عنوان در یک محیط ایزوله قرار ندارد. بنابراین ممکن است یک کرنل درایور تصادفاً در یک آدرس مجازی اشتباه که می‌تواند متعلق به سیستم عامل یا یک درایور دیگر باشد بنویسد. یعنی اگر یک درایور کرنل کرش کند کل سیستم عامل کرش می‌کند.

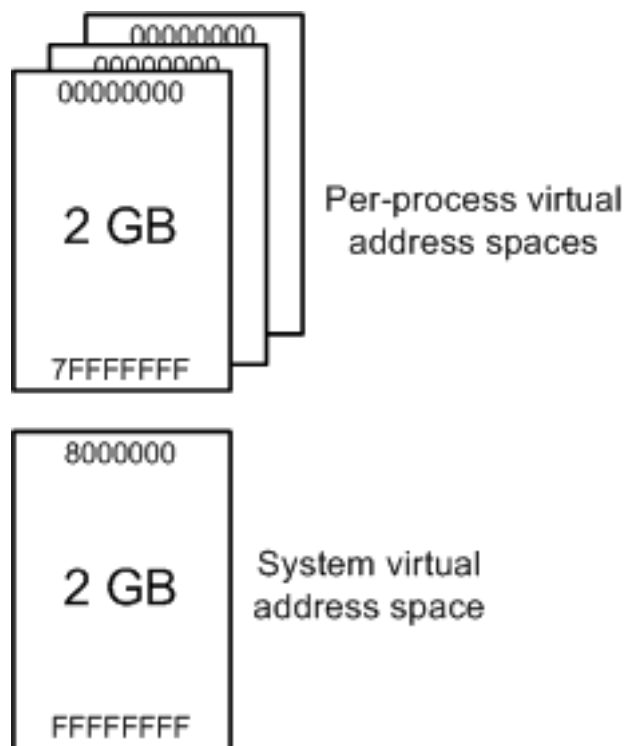
تصویر زیر به خوبی ارتباط بین مد کاربری و مد کرنل را نشان می‌دهد:



فضای کاربری و فضای سیستمی User space and system space

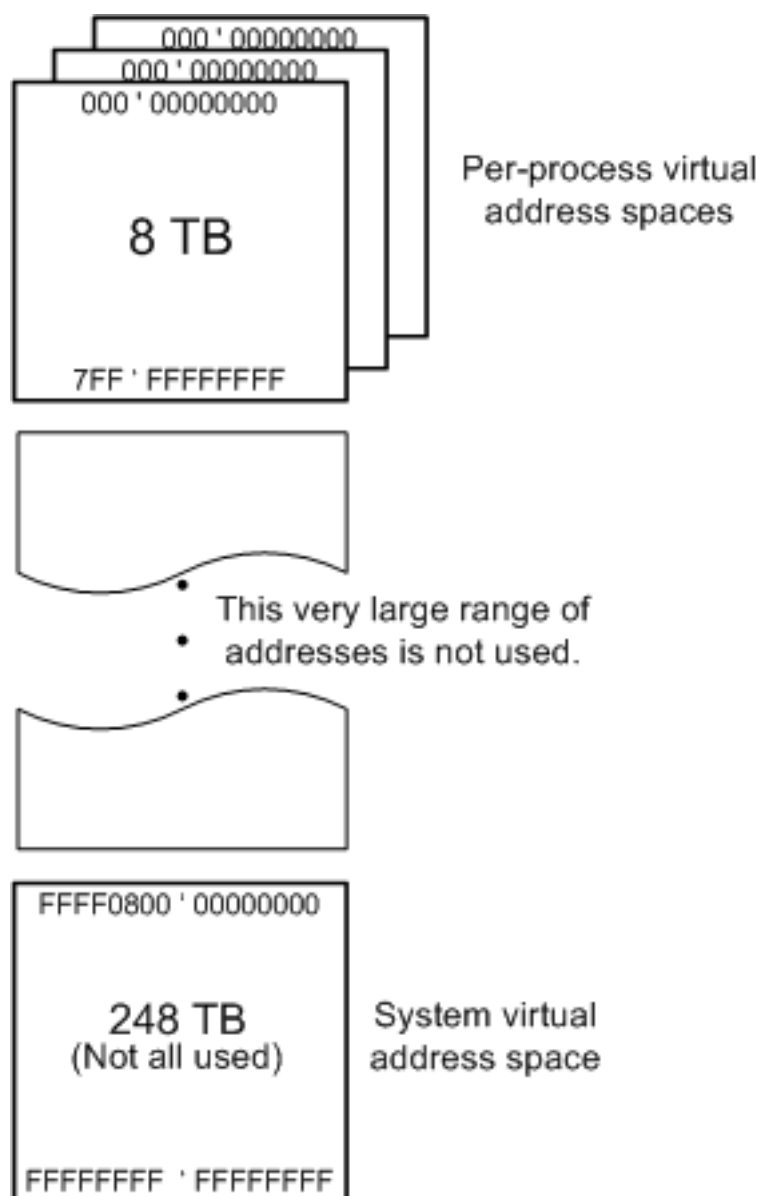
گفتیم بسیاری از پروسه‌ها در حالت user mode و پروسه‌های هسته سیستم عامل و درایورها در حالت kernel mode اجرا می‌شوند. هر پروسه مد کاربر از فضای آدرس دهی مجازی خودش استفاده می‌کند ولی در حالت کرنل همه از یک فضای آدرس دهی استفاده می‌کنند که به آن فضای سیستمی می‌گویند و برای مد کاربری می‌گویند فضای کاربری.

در سیستم‌های 32 بیتی نهایتاً تا 4 گیگ حافظه می‌توان به این‌ها تخصیص داد؛ 2 گیگ ابتدایی به user space و دو گیگ بعدی به system space :



در ویندوزهای 32 بیتی شما امکان تغییر این مقدار حافظه را در میان بوت دارید و می‌توانید حافظه کاربری را تا 3 گیگ مشخص کنید و یک گیگ را برای فضای سیستمی. برای اینکار می‌توانید از برنامه [bcdedit](#) استفاده کنید.

در سیستم‌های 64 بیتی میزان حافظه‌های مجازی به صورت تئوری تا 16 اگزابایت مشخص شده است؛ ولی در عمل تنها بخش کوچکی از آن یعنی 8 ترابایت استفاده می‌شود.



کدهایی که در user mode اجرا می‌شوند فقط به فضای کاربری دسترسی دارند و دسترسی آن‌ها به فضای سیستمی به منظور جلوگیری از تخریب داده ممکن نیست. ولی در حالت کرنل می‌توان به دو فضای سیستمی و کاربری دسترسی داشت. درایورهایی که در مد کرنل نوشته شده اند باید تمام دقت خود را در زمینه نوشتن و خواندن از فضای سیستمی در حافظه به کار گیرند. سناریوی زیر به شما نشان می‌دهد که چرا باید مراقب بود:

برنامه جهت اجرا در مد کاربر یک درخواست را برای خواندن داده‌های یک device را آماده می‌کند. سپس برنامه آدرس شروع یک بافر را برای دریافت داده، مشخص می‌کند.

وظیفه این درایور یک قطعه در مد کرنل این است که عملیات خواندن را شروع کرده و کنترل را به درخواست کننده ارسال می‌کند.

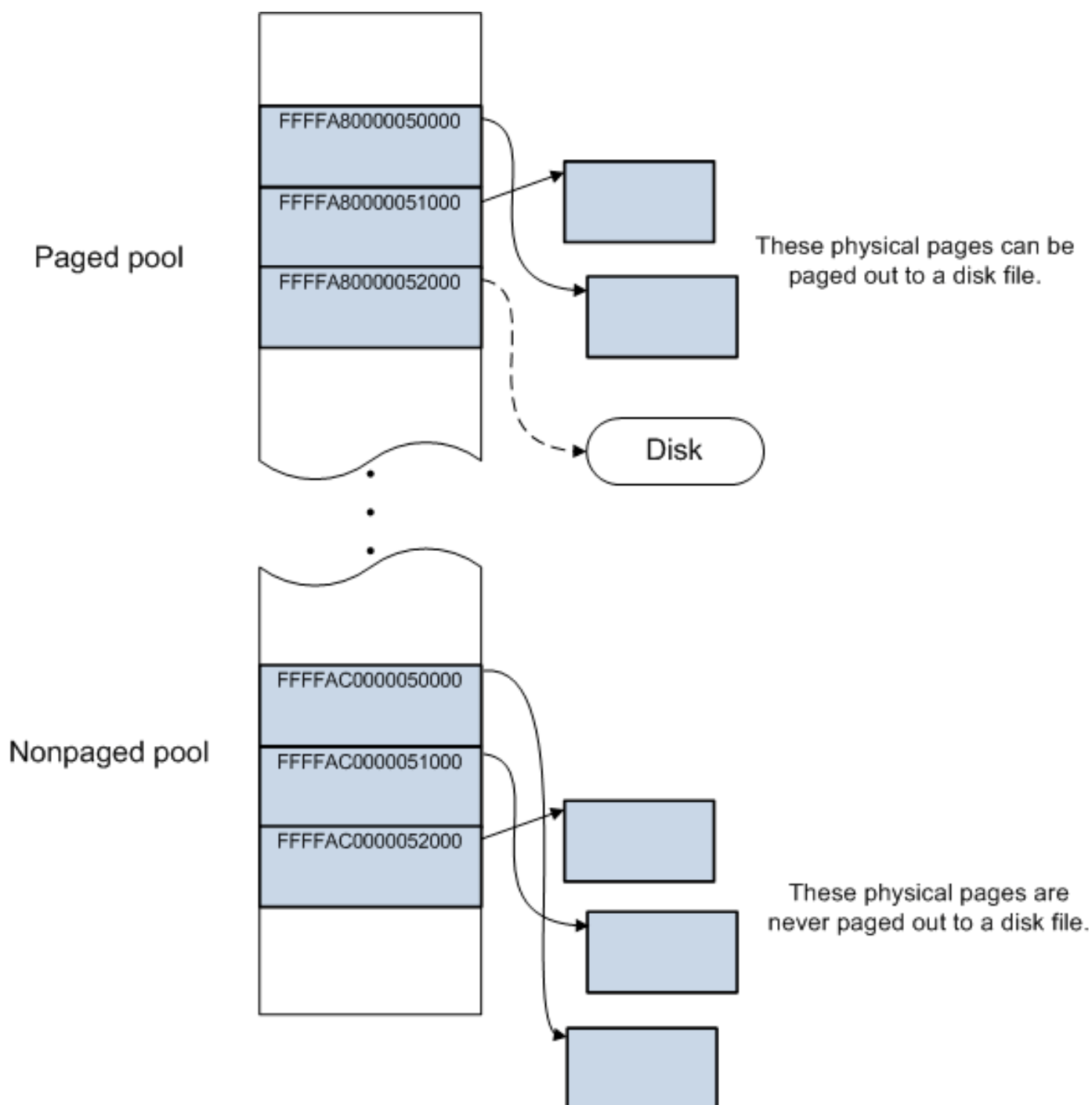
بعد device یک وقفه را به هر تردی thread که در حال اجراست ارسال می‌کند تا بگوید، عملیات خواندن پایان یافته است. این وقفه توسط ترد درایور مربوطه دریافت می‌شود.

حالا دیگر درایور نباید داده‌ها را در همان جایی که گام اول برنامه مشخص کرده است ذخیره کند. چون این آدرس که برنامه در مد کاربری مشخص کرده است، با نمونه‌ای که این فرآیند محاسبه می‌کند متفاوت است.

Paged Pool and NonPaged Pool

در فضای کاربری تمام صفحات در صورت نیاز توانایی انتقال به دیسک سخت را دارند ولی در فضای سیستمی همه بدین صورت نیستند. فضای سیستمی دو ناحیه حافظه تخصیصی پویا دارد که به نام‌های *paged pool* و *nonpaged pool* شناخته می‌شوند. در سیستم‌های 32 بیتی *Pagedpool* توانایی 128 گیگ فضای آدرس دهی مجازی را از آدرس 0xFFFFAC00'00000000 تا آدرس

در سیستم‌های 64 بیتی توانایی 128 گیگ فضای آدرس دهی مجازی را از 0xFFFFA800'00000000 تا 0xFFFFAC1F'FFFFFFFF دارد. حافظه ای که به صورت *paged pool* تخصیص شده باشد می‌تواند صفحات حافظه را بر روی دیسک سخت ذخیره کند؛ ولی حافظه ای که به صورت *nonpaged* تخصیص یافته باشد، هرگز نمی‌تواند.



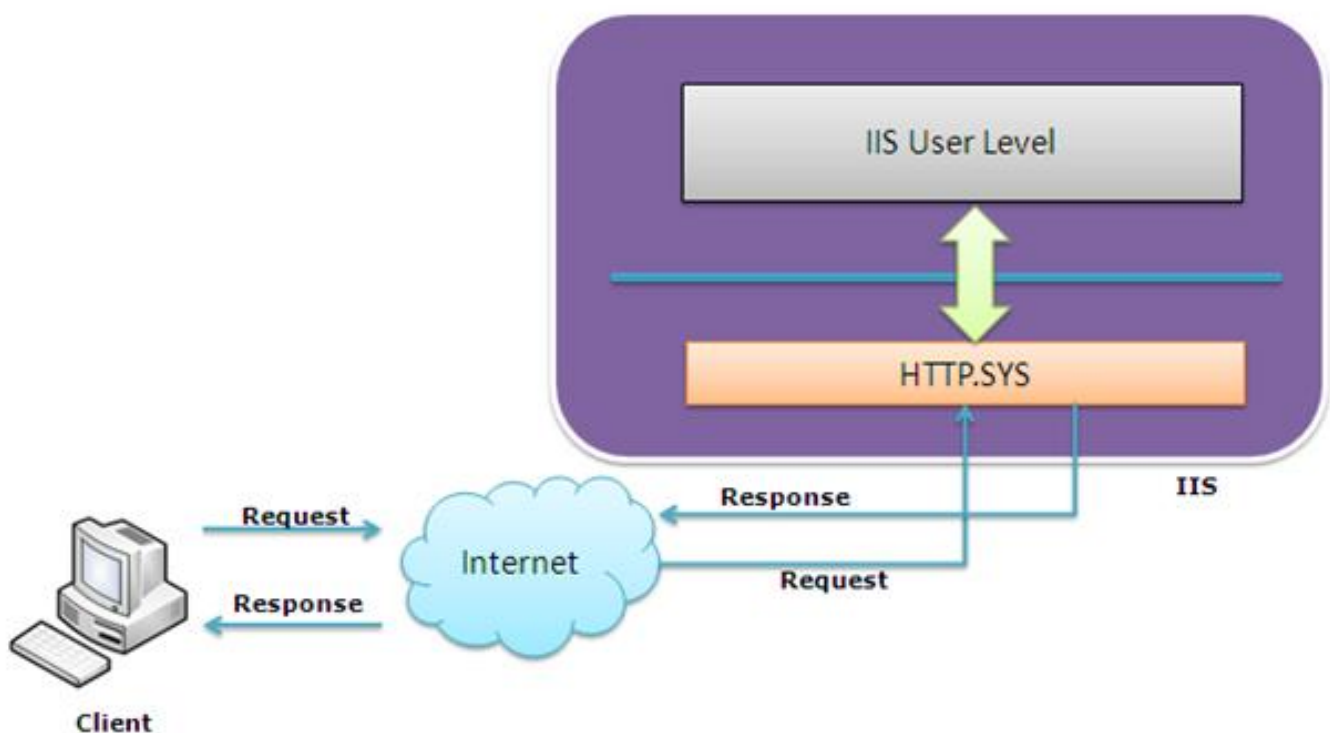
در [مقاله قبل](#) در مورد نحوه ذخیره سازی در حافظه نوشتیم و به user mode و kernel mode اشاراتی کردیم که می‌توانید به آن رجوع کنید.

در این سری مقالات قصد داریم به بررسی اجزا و روند کاری موجود در IIS بپردازیم که چگونه IIS کار می‌کند و شامل چه بخش هایی می‌شود. مطمئناً آشنایی با این بخش‌ها در روند شناسایی رفتارهای وب اپلیکیشن‌ها و واکنش‌های سرور، کمک زیادی به ما خواهد کرد. در اینجا نسخه IIS7 را به عنوان مرجع در نظر گرفته‌ایم.

وب سرور IIS در عبارت مخفف Internet information services به معنی سرویس‌های اطلاعاتی اینترنت می‌باشد. IIS شامل کامپوننت‌های زیادی است که هر کدام از آن‌ها کار خاصی را انجام می‌دهند؛ برای مثال گوش دادن به درخواست‌های ارسال شده به سرور، مدیریت فرآیندها Process و خواندن فایل‌های پیکربندی Configuration؛ این اجزا شامل Http.sys، protocol listener و WSA و .. می‌شوند. **Protocol Listeners**

این پروتکل‌ها به درخواست‌های رسیده گوش کرده و آن‌ها را مورد پردازش قرار می‌دهند و پاسخی را به درخواست کننده، ارسال می‌کنند. هر listener بر اساس نوع پروتکل متفاوت هست. به عنوان مثال کلاینتی، درخواست صفحه‌ای را می‌کند و http listener که به آن Http.sys می‌گویند به آن پاسخ می‌دهد. به طور پیش فرض Http.sys به درخواست‌های http و https گوش فرا می‌دهد، این کامپوننت از IIS6 اضافه شده است ولی در نسخه 7 از SSL نیز پشتیبانی می‌کند. **Http.sys یا Hypertext transfer protocol stack**

کار این واحد در سه مرحله دریافت درخواست، ارسال آن به واحد پردازش IIS و ارسال پاسخ به کلاینت است؛ قبل از نسخه 6 از Winsock یا windows socket api که یک کامپوننت user-mod بود استفاده می‌شد ولی Http.sys یک کامپوننت Kernel-mod هست.



Http.sys مزایای زیر را به همراه دارد:

صف درخواست مد کرنل: به خاطر اینکه کرنل مستقیماً درخواست‌ها را به پروسه‌های مربوطه می‌فرستد و اگر پروسه موجود نباشد، درخواست را در صف گذاشته تا بعداً پروسه مورد نظر آن را از صف بیرون بکشد. برای درخواست‌ها یک پیش پردازش و همچنین اعمال فیلترهای امنیتی اعمال می‌گردد. عملیات کش کردن تماماً در محیط کرنل مد صورت می‌گیرد؛ بدون اینکه به حالت یوزرمد سویچ کند. مد کرنل دسترسی بسیار راحت و مستقیمی را برای استفاده از منابع دارد و لازم نیست مانند مد کاربر به لایه‌های زیرین، درخواست کاری را بدهد؛ چرا که خود مستقیماً وارد عمل می‌شود و برداشته شدن واسط در سر راه، موجب افزایش عمل caching می‌شود. همچنین دسترسی به کش باعث می‌شود که مستقیماً پاسخ از کش به کاربر برسد و توابع پردازشی در حافظه بارگذاری نشوند. البته این کش کردن محدودیت‌هایی را هم به همراه دارد:

کش کرنل به صورت پیش فرض بر روی صفحات ایستا فعال شده است؛ نه برای صفحاتی با محتوای پویا که البته این مورد قابل تغییر است که نحوه این تغییر را پایینتر توضیح خواهیم داد.

اگر آدرس درخواستی شامل کوئری باشد صفحه کش نخواهد شد: <http://www.site.info/postarchive.htm?id=25>

برای پاسخ از مکانیزم‌های فشرده سازی پویا استفاده شده باشد مثل gzip کش نخواهد شد

صفحه درخواست شده صفحه اصلی سایت باشد کش نخواهد شد: <http://www.dotnettip.info> ولی اگر درخواست بدین صورت باشد <http://www.domain.com/default.htm> کش خواهد کرد.

درخواست به صورت ناشناس anonymous نباشد و نیاز به authentication داشته باشد کش نخواهد شد (یعنی در هدر شامل گزینه authorization می‌باشد).

درخواست باید از نوع نسخه http1 به بعد باشد.

اگر درخواست شامل [Entity-body](#) باشد کش نخواهد کرد.

درخواست شامل [If-Range/Range header](#) باشد کش نمی‌شود.

کل حجم response بیشتر از اندازه تعیین شده باشد کش نخواهد گردید، این اندازه در کلید رجستری UriMaxUriBytes قرار دارد. [اطلاعات بیشتر](#)

اندازه هدر بیشتر از اندازه تعیین شده باشد که عموماً اندازه تعیین شده یک کیلو بایت است.

کش پر باشد، کش انجام نخواهد گرفت.

برای فعال سازی کش کرنل راهنمای زیر را دنبال کنید:

گزینه output cache را در IIS، فعال کنید و سپس گزینه Add را بزنید. کادر add cache rule که باز شود، از شما می‌خواهد یکی از دو نوع کش مد کاربر و مد کرنل را انتخاب کنید و مشخص کنید چه نوع فایل‌هایی (مثلاً aspx) از این قوانین پیروی کنند و مکانیزم کش کردن به سه روش جلوگیری از کش کردن، کش زمان دار و کش بر اساس آخرین تغییر فایل انجام گردد.

Add Cache Rule

File name extension:
.aspx
Example: .aspx or .axd

☐ User-mode caching

File Cache Monitoring

☒ Using file change notifications

☐ At time intervals (hh:mm:ss):
00:00:30

☐ Prevent all caching

Advanced...

☒ Kernel-mode caching

File Cache Monitoring

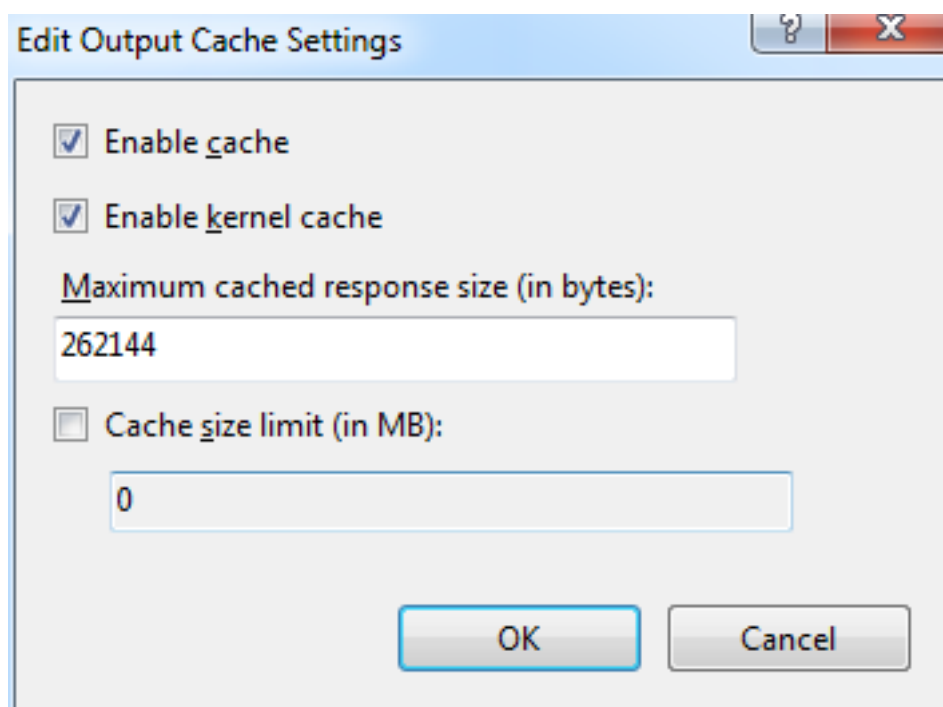
☒ Using file change notifications

☐ At time intervals (hh:mm:ss):
00:00:30

☐ Prevent all caching

OK Cancel

برای تعیین مقدار سائز کش response که در بالا اشاره کردیم می‌توانید در همان پنجره، گزینه edit feature settings را انتخاب کنید.



این قسمت از مطلب که به نقل از مقاله آقای Karol Jarkovsky در این [آدرس](#) است یک سری تست هایی با نرم افزار (Web Capacity Analysis Tool (WCAT گرفته است که به نتایج زیر دست پیدا کرده است:

Kernel Cache Disabled 4 clients/160 threads/30 sec 257 req/sec
Kernel Cache Enabled 4 clients/160 threads/30 sec 553 req/sec

همانطور که می بینید نتیجه فعال سازی کش کرنل پاسخ به بیش از دو برابر درخواست در حالت غیرفعال آن است که یک عدد فوق العاده به حساب میاد.

برای اینکه خودتان هم تست کرده باشید در این [آدرس](#) برنامه را دانلود کنید و به دنبال فایل request.cfg و از صحت پارامترهای server و url اطمینان پیدا کنید. در گام بعدی [5 پنجره خط فرمان](#) باز کرده و در یکی از آن ها دستور netsh http show cachestate را بنویسید تا تمامی ورودی های entry که در کش کرنل ذخیره شده اند لیست شوند. البته در اولین تست کش را غیرفعال کنید و به این ترتیب نباید چیزی نمایش داده شود. در همان پنجره فرمان wcctl -a localhost -c config.cfg -s request.cfg را زده تا کنترلر برنامه در وضعیت listening قرار بگیرد. در 4 پنجره دیگر فرمان wcclient localhost را بنویسید تا تست آغاز شود. بعد از انجام تست به شاخه نصب کنترلر WCAT رفته و فایل log را بخوانید و اگر دوباره دستور netsh http show cachestate را اجرا کنید باید کش کرنل را بزنید باید خالی باشد. حالا کش را فعال کنید و دوباره عملیات تست را از سر بگیرید و اگر دستور netsh http show cachestate را اجرا کنید باید کش کرنل دارای ورودی باشد. برای تغییرات در سطح http.sys می توانید از رجستری کمک بگیرید. در [اینجا](#) تعداد زیادی از تنظیمات ذخیره شده در رجستری برای http.sys لیست شده است.

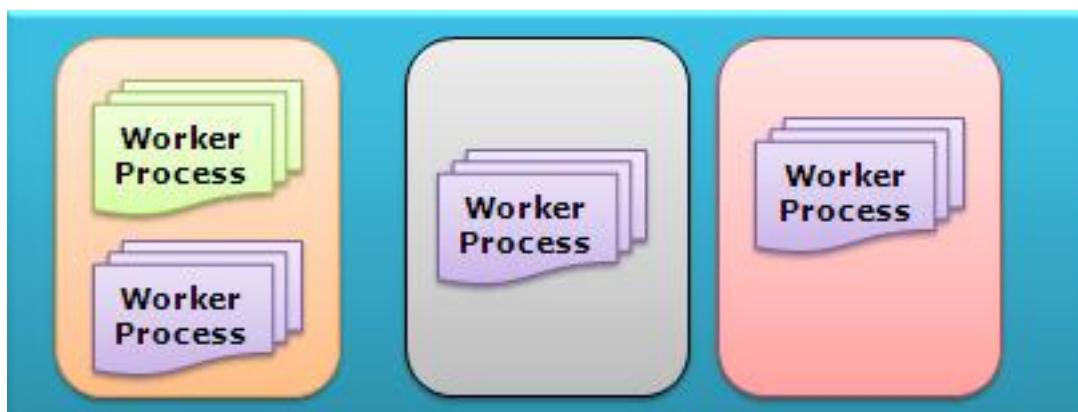
در قسمت قبلی گفتیم که IIS از تعدادی کامپوننت تشکیل شده است و به یکی از آن‌ها به نام Http.sys پرداختیم. در این قسمت قصد داریم به WWW Services بپردازیم. اجازه بدهید قبل از هر چیزی به دو مفهوم اصلی در IIS بپردازیم:

1. Worker Process

2. Application Pool

پرونده‌های کارگر w3wp.exe وظیفه‌ی اجرای برنامه‌های asp.net را در IIS، به عهده دارند. این پرونده‌ها مسئولیت پردازش تمامی درخواست و پاسخ‌ها از/به کلاینت را دارند. هر کاری که باید در asp.net انجام بشود، توسط این‌ها صورت می‌گیرد. به بیان ساده‌تر این پرونده‌ها قلب برنامه‌های ASP.Net بر روی IIS هستند.

Application Pool: این پول‌ها در واقع ظرفی یا در برگیرنده‌ای برای پرونده‌های کارگر به حساب می‌آیند. این پول‌ها پرونده‌های کارگر را از هم جدا و دسته‌بندی می‌کنند تا قابلیت اعتماد، امنیت و در دسترس بودن بدهند. موقعی که یک پرونده یا حتی یک پول دچار مشکل می‌شود، این اطمینان داده می‌شود که تاثیری بر دیگر پول‌ها یا پرونده‌های کارگر، ندارد. یعنی موقعی که یک web application دچار مشکل شود، هیچ تاثیری بر اجرای web application های دیگر ندارد. به یک application pool با چند پرونده کارگر web garden می‌گویند.



World Wide Web Publishing Services

یکی از قدیمی‌ترین امکانات موجود در IIS هست که از نسخه 7 به بعد، کار خود را با یک سرورس جدید به اسم Windows Process Activation Service یا به اختصار WAS که به صورت local system بر روی پرونده Svchost.exe با یک کد باینری یکسان اجرا می‌شود، شریک شده است. ممکن است در بعضی جاها WWW Service به صورت W3SVC هم نوشته شود.

اصلا این WWW Service چه کاری انجام می‌دهد و به چه دردی می‌خورد؟

این سرویس در سه بخش مهم IIS 6 به فعالیت می‌پردازد:

HTTP administration and configuration

Performance monitoring

Process management

HTTP Administration and Configuration

سرویس WWW وظیفه خواندن اطلاعات پیکربندی IIS از متابیس را بر عهده دارد و از این اطلاعات خوانده شده برای پیکربندی و به روز کردن Http.sys استفاده می‌کند. به غیر از این کار، وظیفه آغاز و توقف و نظارت یا مانیتورینگ و همچنین مدیریت کامل پروسه‌های کارگر در زمینه http request را هم عهده دار است.

Performance Monitoring

سرویس WWW بر کارایی وب سایت‌ها و کش IIS نظارت می‌کند و البته یک شمارنده کارایی performance counter هم ایجاد می‌کند. کار شمارنده کارایی این است که اطلاعات یک سرویس یا سیستم عامل یا یک برنامه کاربردی را جمع آوری می‌کند تا به ما بگوید که این بخش‌ها به چه میزانی بهینه کار خود را انجام می‌دهند و به ما کمک می‌کنند که سیستم را به بهترین کارایی برسانیم. سیستم عامل، شبکه و درایورها، داده‌های شمارشی را تهیه و در قالب یک سیستم نظارتی گرافیکی به کارشناس سیستم یا شبکه نشان می‌دهند. برنامه نویسی‌ها هم از این طریق می‌توانند برنامه‌های خود را بنویسند که در [اینجا](#) لیستی از شمارنده‌ها در دانت نت را می‌توانید ببینید و بیشتر آن‌ها از طریق فضای نام system.diagnostics در دسترس هستند.

Process Management

سرویس www مدیریت application pool و پروسه‌های کارگر را هم به عهده دارد. این مدیریت شامل شروع و توقف و بازیابی پروسه‌های کارگر می‌شود. به علاوه اینکه این سرویس کار نظارت بر صحت انجام عملیات پروسه‌های کارگر را هم جز وظایف خود می‌داند. وقتی که چندین بار کار پروسه‌های کارگر در یک دوره زمانی که در فایل پیکربندی مشخص شده با مشکل مواجه شود، از شروع یک پروسه کارگر دیگر جلوگیری می‌کند.

در نسخه‌های جدیدتر IIS چکاری بر عهده WWW Service است؟

در IIS7 به بعد، دیگر مدیریت پروسه‌های کارگر را به عهده ندارد؛ ولی به جای آن سمتی جدید را به اسم listener adapter دریافت کرده است که یک listener adapter برای http listener یعنی Http.sys است. اصلی‌ترین وظیفه فعلی را که انجام می‌دهد پیکربندی Http.sys می‌باشد. موقعی که اطلاعات پیکربندی به روز می‌شوند باید این تغییرات بر روی Http.sys اعمال شوند. دومین وظیفه آن این است موقعی که درخواست جدیدی وارد صف درخواست‌ها می‌شود این مورد را به اطلاع WAS برساند. WAS در قسمت سوم این مقاله توضیح داده خواهد شد.

همانطور که در مطلب قبلی گفتیم، در این مطلب قرار است به WAS بپردازیم؛ در دنباله متن قبلی گفتیم که دومین وظیفه WWW Service این است: موقعی که یک درخواست جدید در صف درخواست‌ها وارد شد، به اطلاع WAS برساند.

WAS یا Windows Process Activation Service

در نسخه 7 به بعد، WAS مدیریت پیکربندی application pool و پروسه‌های کارگر را به جای WWW Service به عهده گرفته است. این مورد شما را قادر می‌سازد تا همان پیکربندی که برای Http در نظر گرفته‌اید، بر روی درخواست‌هایی که Http نیستند هم اعمال کنید. همچنین موقعی که سایت شما نیازی به درخواست‌های Http ندارد می‌توانید WAS را بدون WWW Service راه اندازی کنید. به عنوان یک مثال فرض کنید شما یک وب سرویس WCF را از طریق WCF Listener Adapter مدیریت می‌کنید و احتیاجی به درخواست‌های نوع Http listener ندارید و http.sys کاری برای انجام ندارد پس نیازی هم به راه اندازی www service نیست.

پیکربندی مدیریتی در WAS

در زمان شروع کار IIS، سرویس WAS اطلاعاتی را از فایل ApplicationHost.config می‌خواند و آن‌ها را به دست listener adapterهای مربوطه می‌رساند و listener adapter ارتباط بین WAS و listenerهای مختلف را در IIS، برقرار می‌کند. آداپتورها اطلاعات لازم را از WAS می‌گیرند و به listenerهای مربوطه انتقال می‌دهند تا listenerها بر اساس آن تنظیمات یا پیکربندی‌ها، به درخواست‌ها گوش فرا دهند.

در مورد WCF، ابتدا WAS تنظیمات را برای آداپتور WCF که NetTcpActivator نام دارد ارسال کرده و این آداپتور بر اساس آن listener مربوطه را پیکربندی کرده تا به درخواست‌هایی که از طریق پروتوکول net.tcp می‌رسد گوش فرا دهد. لیست زیر تعدادی از اطلاعاتی را که از فایل پیکربندی می‌خواند و ارسال می‌کند را بیان کرده است:

Global configuration information

Protocol configuration information for both HTTP and non-HTTP protocols

Application pool configuration, such as the process account information

Site configuration, such as bindings and applications

Application configuration, such as the enabled protocols and the application pools to which the applications belong

نکته پایانی اینکه اگر فایل ApplicationHost.config تغییر می‌کند، WAS یک اعلان دریافت کرده و اطلاعات آداپتورها را به روز می‌کند.

مدیریت پروسه‌ها Process Managment

گفتیم که مدیریت پول و پروسه‌های کارگر جزء وظایف این سرویس به شمار می‌رود. موقعی که یک protocol listener درخواستی را دریافت می‌کند، WAS چک می‌کند که آیا یک پروسه کارگر در حال اجراست یا خیر. اگر application pool پروسه‌ای داشته باشد که در حال سرویس دهی به درخواست‌هاست، آداپتور درخواست را به پروسه کارگر ارسال می‌کند. در صورتی که پروسه‌ای در application pool در حال اجرا نباشد، WAS یک پروسه جدید را آغاز می‌کند و آداپتور درخواست را به آن پاس می‌کند.

نکته: از آنجایی که WAS هم پروسه‌های http و هم non-http را مدیریت می‌کند، پس می‌توانید از یک applicatio pool برای چندین protocol استفاده کنید. به عنوان مثال شما یکی سرویس XML دارید که می‌توانید از آن برای سرویس دهی به پروتوکول‌های Http و net.tcp بهره بگیرید.

ماژول‌ها در IIS

قبلاً مقاله ای در مورد moduleها با نام "[کمی در مورد httpmoduleها](#)" قرار داده بودیم که بهتر است برای آشنایی بیشتر، به آن رجوع کنید. به غیر از وب کانفیگ که برای معرفی ماژول‌ها استفاده می‌کردیم، می‌توانید به صورت گرافیکی و دستی هم این کار را

انجام بدهید. ابتدا یک پروژه class library ایجاد کرده و ماژول خود را بنویسید و سپس آن را به یک dll تبدیل کنید و dll را در شاخه bin که این شاخه در ریشه وب سایتتان قرار دارد کپی کنید. سپس در IIS قسمت module گزینه Add را انتخاب کنید و در قسمت اول نامی برای آن و در قسمت بعدی دقیقاً همان قوانین type که در وب کانفیگ مشخص می‌کردید را مشخص کنید:

Namespace.ClassName

گزینه invoke only for requests to asp.net and manage handlers را هم تیک بزنید. کار تمام است.

ماژول‌های کد ماشین یا native

این ماژول‌ها به صورت پیش فرض به سیستم اضافه شده‌اند و در صورتی که می‌خواهید جایگزینی به منظور خصوصی سازی انجام دهید آن‌ها را پاک کنید و ماژول جدید را اضافه کنید.

جدول ماژول‌های HTTP

نام ماژول	توضیحات	نام فایل منبع
CustomErrorModule	موقعی که هنگام response، کد خطایی تولید می‌گردد، پیام خطا را پیکربندی و سپس ارسال می‌کند.	Inetsrv\Custerr.dll
HttpRedirectionModule	تنظیمات redirection برای درخواست‌های http را در دسترس قرار می‌دهد.	Inetsrv\Redirect.dll
ProtocolSupportModule	انجام عملیات مربوط به پروتوکول‌ها بر عهده این ماژول است؛ مثل تنظیم کردن قسمت هدر برای response.	Inetsrv\Protsup.dll
RequestFilteringModule	این ماژول از IIS 7.5 به بعد اضافه شد. درخواست‌ها را فیلتر می‌کند تا پروتوکول و رفتار محتوا را کنترل کند.	Inetsrv\modrqflt.dll
WebDAVModule	این ماژول از IIS 7.5 به بعد اضافه شد. امنیت بیشتر در هنگام انتشار محتوا روی HTTP SSL	Inetsrv\WebDAV.dll

ماژول‌های امنیتی

نام ماژول	توضیحات	نام فایل منبع
AnonymousAuthenticationModule	موقعی که هیچ کدام از عملیات authentication با موفقیت روبرو نشود، عملیات Anonymous authentication انجام می‌شود.	Inetsrv\Authanon.dll
BasicAuthenticationModule	عمل ساده و اساسی authentication را انجام می‌دهد.	Inetsrv\Authbas.dll
CertificateMappingAuthenticationModule	انجام عمل Certificate Mapping در authentication Active Directory	Inetsrv\Authcert.dll
DigestAuthenticationModule	Digest authentication	Inetsrv\Authmd5.dll
IISCertificateMappingAuthenticationModule	همان Certificate Mapping authentication ولی اینبار با IIS Certificate	Inetsrv\Authmap.dll

نام ماژول	توضیحات	نام فایل منبع
RequestFilteringModule	عملیات اسکن URL از قبیل نام صفحات و دایرکتوری‌ها، نوع verb و یا کاراکترهای مشکوک و خطرآفرین	Inetsrv\Modrqflt.dll
UrlAuthorizationModule	عمل URL authorization	Inetsrv\Urlauthz.dll
WindowsAuthenticationModule	عمل NTLM integrated authentication	Inetsrv\Authsspi.dll
IpRestrictionModule	محدود کردن IP‌های نسخه 4 لیست شده در IP Security در قسمت پیکربندی	Inetsrv\iprestr.dll

ماژول‌های محتوا

نام ماژول	توضیحات	نام فایل منبع
CgiModule	ایجاد پردازش‌های (Common Gateway Interface (CGI response به منظور ایجاد خروجی	Inetsrv\Cgi.dll
DefaultDocumentModule	تلاش برای ساخت یک سند پیش فرض برای درخواست‌هایی که دایرکتوری والد ارسال می‌شود	Inetsrv\Defdoc.dll
DirectoryListingModule	لیست کردن محتوای یک دایرکتوری	Inetsrv\dirlist.dll
IsapiModule	میزبانی فایل‌های ISAPI	Inetsrv\Isapi.dll
IsapiFilterModule	پشتیبانی از فیلترهای ISAPI	Inetsrv\Filter.dll
ServerSideIncludeModule	پردازش کدهای include شده سمت سرور	Inetsrv\Iis_ssi.dll
StaticFileModule	ارائه فایل‌های ایستا	Inetsrv\Static.dll
FastCgiModule	پشتیبانی از CGI	Inetsrv\iisfcgi.dll

ماژول‌های فشرده سازی

DynamicCompressionModule	فشرده سازی پاسخ response با gzip	Inetsrv\Compdyn.dll
StaticCompressionModule	فشرده سازی محتوای ایستا	Inetsrv\Compstat.dll

ماژول‌های کش کردن

FileCacheModule	تهیه کش در مد کاربری برای فایل‌ها.	Inetsrv\Cachfile.dll
HTTPCacheModule	تهیه کش مد کاربری و مد کرنل برای http.sys	Inetsrv\Cachhttp.dll
TokenCacheModule	تهیه کش مد کاربری بر اساس جفت نام کاربری و یک token که توسط Windows user principals تولید شده است.	Inetsrv\Cachtokn.dll
UriCacheModule	تهیه یک کش مد کاربری از اطلاعات URL	Inetsrv\Cachuri.dll

ماژول‌های عیب‌یابی و لاگ کردن

Inetsrv\Logcust.dll	بارگزاری ماژول‌های خصوصی سازی شده جهت لاگ کردن	CustomLoggingModule
Inetsrv\Iisfrieb.dll	برای ردیابی درخواست‌های ناموفق	FailedRequestsTracingModule
Inetsrv\Loghttp.dll	دریافت اطلاعات و پردازش وضعیت http.sys برای لاگ کردن	HttpLoggingModule
Inetsrv\Iisreqs.dll	ردیابی درخواست‌هایی که در حال حاضر در پروسه‌های کارگر در حال اجرا هستند و گزارش اطلاعاتی در مورد وضعیت اجرا و کنترل رابط برنامه نویسی کاربردی.	RequestMonitorModule
Inetsrv\Iisetw.dll	گزارش رخدادهای Microsoft Event Tracing for Windows یا به اختصار ETW	TracingModule

ماژول‌های مدیریتی و نظارتی بر کل ماژول‌ها

Microsoft.NET\Framework\v2.0.50727\webengine.dll	مدیریتی بر ماژول‌های غیر native که در پایین قرار دارند.	ManagedEngine
Inetsrv\validcfg.dll	اعتبارسنجی خطاها، مثل موقعی که برنامه در حالت integrated اجرا شده و ماژول‌ها یا هندلرها در system.web تعریف شده‌اند.	ConfigurationValidationModule

از IIS6 به بعد در حالت integrated و ماقبل، در حالت کلاسیک می‌باشند. اگر [مقاله ماژول‌ها](#) را خوانده باشید می‌دانید که تعریف آن‌ها در وب کانفیگ در بین این دو نسخه متفاوت هست و رویداد سطر آخر در جدول بالا این موقعیت را چک می‌کند و اگر به خاطر داشته باشید با اضافه کردن یک خط اعتبارسنجی آن را قطع می‌کردیم. در مورد هندلرها هم به همین صورت می‌باشد. به علاوه ماژول‌های native بالا، IIS این امکان را فراهم می‌آورد تا از ماژول‌های کد مدیریت شده (یعنی CLR) برای توسعه توابع و کارکرد IIS بهره مند شوید:

ماژول	توضیحات	منبع
AnonymousIdentification	مدیریت منابع تعیین هویت برای کاربران ناشناس مانند asp.net profile	System.Web.Security.AnonymousIdentificationModule
DefaultAuthentication	اطمینان از وجود شی Authentication در context مربوطه	System.Web.Security.DefaultAuthenticationModule
FileAuthorization	تایید هویت کاربر برای دسترسی به فایل درخواست	System.Web.Security.FileAuthorizationModule
FormsAuthentication	با این قسمت که باید کاملاً آشنا باشید؛ برای تایید هویت کاربر	System.Web.Security.FormsAuthenticationModule

ماژول	توضیحات	منبع
		tionModule
OutputCache	مدیریت کش	System.Web.Caching.OutputCacheModule
Profile	مدیریت پروفایل کاربران که تنظیماتش را در یک منبع داده‌ای چون دیتابیس ذخیره و بازیابی می‌کند.	System.Web.Profile.ProfileModule
RoleManager	مدیریت نقش و سمت کاربران	System.Web.Security.RoleManagerModule
Session	مدیریت session ها	System.Web.SessionState.SessionStateModule
UrlAuthorization	آیا کاربر جاری حق دسترسی به URL درخواست را دارد؟	System.Web.Security.UrlAuthorizationModule
UrlMappingsModule	تبدیل یک Url واقعی به یک Url کاربرپسند	System.Web.UrlMappingsModule
WindowsAuthentication	شناسایی و تایید و هویت یک کاربر بر اساس لاگین او به ویندوز	System.Web.Security.WindowsAuthenticationModule

پردازش درخواست‌های HTTP در IIS

بگذارید در این قسمت خلاصه‌ای از درخواست‌های نوع HTTP را که تا به الان گفته‌ایم، به همراه شکل بیان کنیم: موقعی که کلاینت درخواست خود را مبنی بر یکی از منابع سرور ارسال می‌کند، Http.sys این درخواست را می‌گیرد. Http.sys با WAS تماس گرفته و درخواست می‌کند تا اطلاعات پیکربندی یا تنظیمات IIS را برای نحوه‌ی برخورد با درخواست، برایش بفرستد.

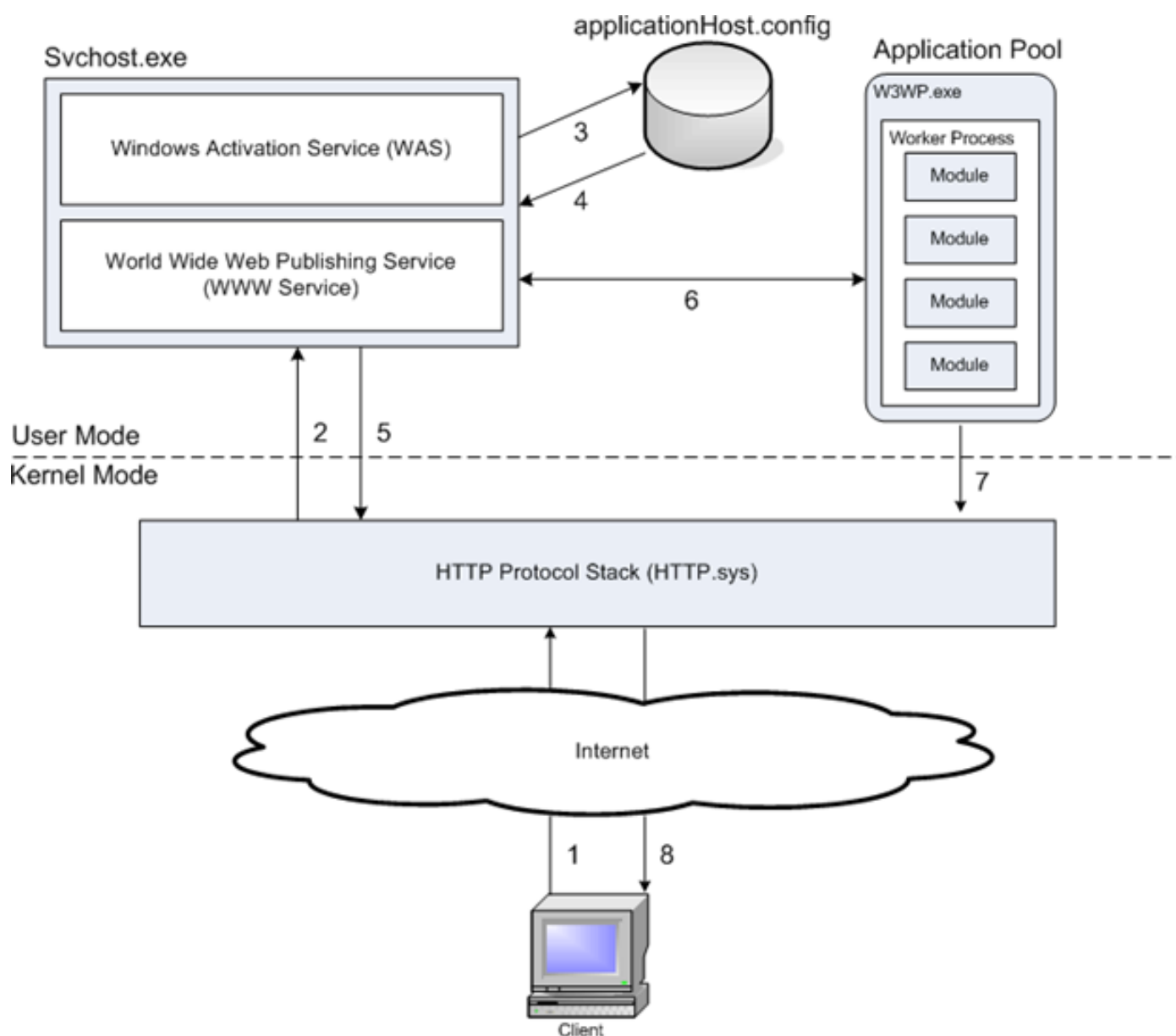
WAS هم اطلاعات پیکربندی شده را از محل ذخیره داده‌ها که applicationHost.config هست، می‌خواند. WWW Service که یک آداپتور برای Http.sys هست، اطلاعات را از WAS دریافت می‌کند. این اطلاعات شامل پیکربندی application pool و سایت می‌باشد.

WWW Service اطلاعات را برای Http.sys می‌فرستد.

WAS یک پروسه کارگر را در application pool ایجاد می‌کند تا درخواست رسیده مورد پردازش قرار بگیرد.

پروسه‌های کارگر درخواست را پردازش کرده و خروجی یا response مورد نظر را تولید می‌کنند.

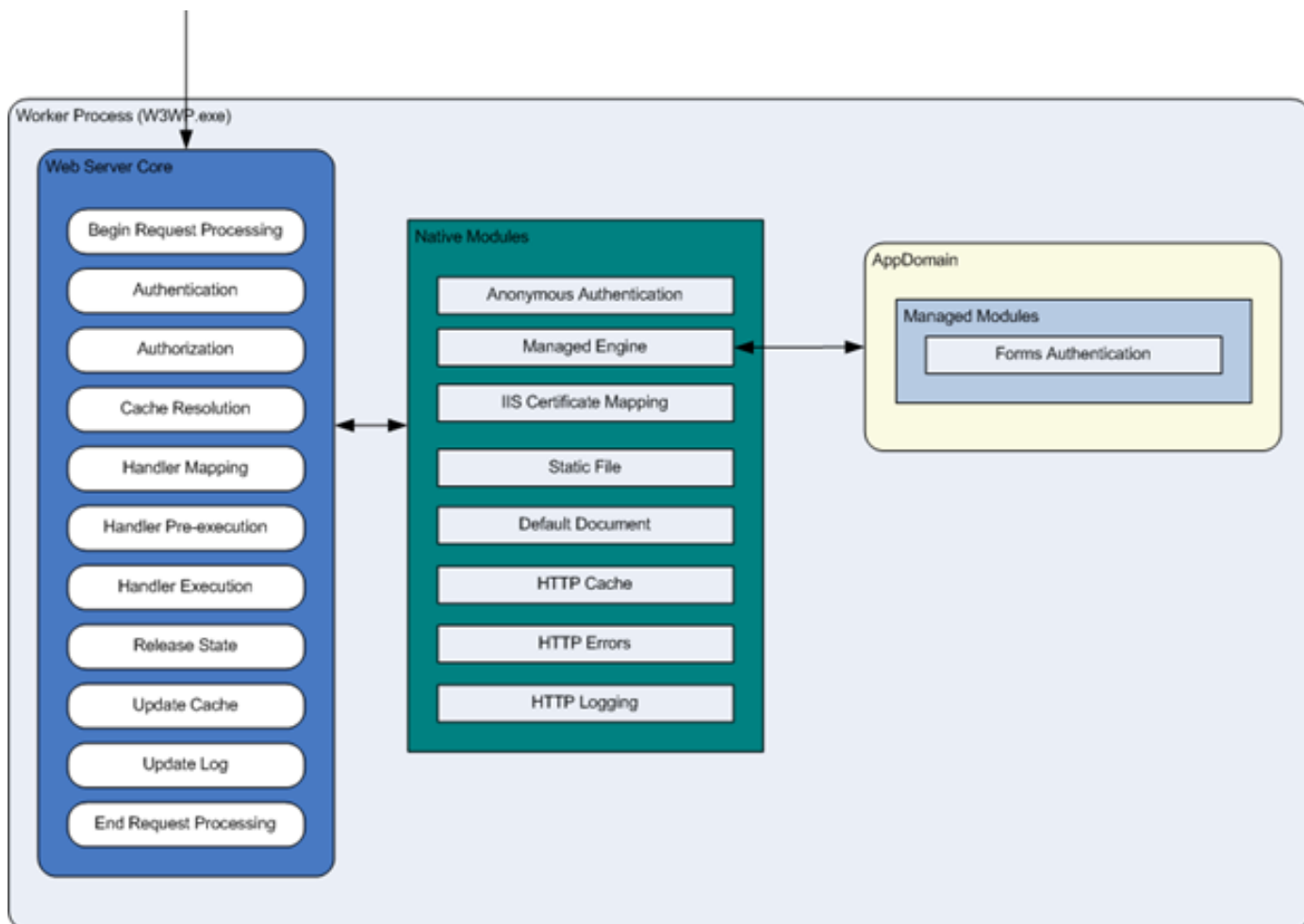
Http.sys نتیجه را دریافت و برای کلاینت می‌فرستد.



حال بیایید ببینیم موقعی که درخواست وارد پروسه‌ی کارگر میشود چه اتفاقی می‌افتد؟

در پروسه‌های کارگر، یک درخواست از مراحل لیست شده‌ای به ترتیب عبور می‌کند. در هسته وب سرور، رویدادهایی را فراخوانی می‌کند که در هر رویداد چندین ماژول native برای کارهایی چون authentication یا events logs دارد و در صورتیکه درخواستی نیاز به یک ماژول مدیریت شده CLR داشته باشد، از ماژول native managedEngine کمک گرفته و یک app domain را ایجاد می‌کند تا ماژول‌های مدیریت شده، عملیات لازم خودشان را انجام دهند. مثل authentication form و ...

موقعی هم که درخواست، از تمامی این رویدادها عبور کند، response برای http.sys ارسال می‌شود تا به کلاینت بازگشت داده شود. شکل زیر نحوه ورود یک درخواست به پروسه کارگر را نشان می‌دهد.



از نسخه 7 به بعد، IIS از یک معماری ماژولار استفاده می‌کند و این ویژگی، سه فایده دارد:

Componentization یا کامپوننت سازی
Extensibility یا توسعه پذیری یا قابل گسترش

ASP.NET Integration

Componentization

همه خصوصیات و ویژگی‌های این وب سرور، توسط کامپوننت‌ها مدیریت می‌شوند که باعث می‌شود شما به راحتی بتوانید کامپوننتی را اضافه، حذف یا جایگزین کنید و این باعث می‌شود که چندین امتیاز از IIS قبلی جلوتر باشد:

باعث کاهش [attack surface](#) می‌شود که در نتیجه امنیت سیستم را بالا می‌برد. با ویژگی حذف کامپوننت‌ها شما می‌توانید ویژگی‌های غیرقابل استفاده IIS را حذف کنید تا ورودی‌های سیستم کاهش یابد. پس با کاهش ویژگی‌هایی که از آن هرگز استفاده نخواهید کرد، مدخل ورود هکر را از بین برده تا امنیت سرور بالاتر برود.

افزایش کارایی و کاهش مصرف حافظه. با حذف ویژگی‌هایی که هرگز استفاده نمی‌کنید، در مصرف حافظه و بهینه استفاده شدن منابع سرور صرفه جویی کنید.

با وجود ویژگی افزودن و جایگزینی کامپوننت‌ها، ناخودآگاه ذهن ما به سمت کاستوم سازی یا خصوصی سازی کشیده می‌شود. با این کار شما به راحتی یک custom server ایجاد می‌کنید که این سرور بر اساس علایق شما کارش را انجام می‌دهد و به راحتی امکاناتی چون افزودن third party را به توسعه دهنده می‌دهد.

Extensibility

با توجه به موارد بالا، خصوصی سازی باعث گسترش امکانات IIS می شود که می تواند به دلایل زیر اتفاق بیفتد:

قدرت بخشی به برنامه های وب. امکانات و قدرتی که می تواند در این حالت به برنامه های در حال اجرا داد به مراتب بیشتر از استفاده از لایه های داخلی خود برنامه هست. برای اینکار شما می توانید کدهای خود را با ASP.Net نوشته یا از کدهای native چون C++ استفاده کنید.

تجربه ای از توسعه پذیری ساده تر و راحت تر

استفاده از قدرت و تمامی امکانات را به شما می دهد و می توانید تمام دستورات را برای همه منابع حتی فایل های ایستا، ASP، CGI و دیگر منابع اجرا کنید.

ASP.NET Integration

تمامی موارد گفته شده بالا در این گزینه خلاصه می شود : محیط ASP.Net Integration به شما امکان استفاده از تمامی امکانات و منابع را به طور کامل می دهد. [دانلود ماژول های مدیریت شده](#)

[دانلود ماژول های native](#)

در مطالب قبلی در مورد مازولار بودن IIS زیاد صحبت کردیم، ولی اجازه بدهید این مورد را به صورت کاربردی‌تر و موشکافانه‌تر بررسی کنیم. برای اینکه به مشکلی در طول این سری از مطالب برخوردید، IIS را به صورت کامل یعنی full feature نصب نمایید. از بخش Turn Windows features on or off > programs & features > control panel اقدام نمایید و هرچه زیر مجموعه Internet information service هست را برگزینید. در صورتی که از نسخه‌های ویندوز سرور 2008 استفاده می‌کنید از طریق server manager > roles > web server اقدام نمایید.

برای نصب یک مازول باید دو مرحله را انجام داد:

نصب مازول

فعال سازی مازول

نکته ای که در مورد مازول‌های native وجود دارد این هست که این مازول‌ها دسترسی بدون محدودیتی به منابع سروری دارند و از این رو حتما باید این نکته را دقت کنید که مازول native شما از یک منبع مورد اعتماد دریافت شده باشد.

نصب یک native module

برای نصب می‌توانید یکی از سه راه زیر را استفاده کنید:

ویرایش دستی فایل کانفیگ و از نسخه IIS7.5 به بعد هم می‌توانید از configuration editor هم استفاده کنید.

استفاده از محیط گرافیکی IIS

استفاده از خط فرمان با دستور Appcmd

مزیت روش دستی این هست که شما دقیقا می‌دانید در پشت صحنه چه اتفاقی می‌افتد و نتیجه هر کدام از این سه روش، اضافه شدن یک مدخل ورودی به تگ <globalmodules> است. برای اعمال تغییرات، مسیر زیر را بروید:

```
%windir%\system32\inetsrv\config\applicationhost.config
```

کسی که نیاز به دسترسی به این مسیر و انجام تغییرات دارد باید در بالاترین سطح مدیریتی سرور باشد.

اگر فایل را باز کنید و تگ globalmodule را پیدا کنید متوجه می‌شوید که تمامی مازول‌ها در این قسمت معرفی شده‌اند و برای خود یک مدخل ورودی یا همان تگ add را دارند که در آن مسیر فایل dll هم ذکر شده است:

```
<globalModules>
<addname="DefaultDocumentModule"image="%windir%\system32\inetsrv\defdoc.dll"/>
<addname="DirectoryListingModule"image="%windir%\system32\inetsrv\dirlist.dll"/>
<add name="StaticFileModule"image="%windir%\system32\inetsrv\static.dll"/>
...
</globalModules>
```

برای حذف یا جایگزینی یک مازول به راحتی می‌توانید مدخل ورودی یک مازول را به صورت دستی حذف نمایید و برای جایگزینی هم بعد از حذف، مازول خود را معرفی کنید. ولی توجه داشته باشید که این حذف به معنی حذف این مازول از تمامی اپلیکیشن‌های موجود بر روی IIS هست و سپس اضافه کردن یک مازول به این بخش. همچنین اگر قصد شما فقط حذف یک مازول از روی یکی از اپلیکیشن‌ها باشد باید از طریق فایل کانفیگ سایت از مسیر تگ‌های <modules> <system.webserver> و با استفاده از تگ‌های add و remove به معرفی یک مازول مختص این اپلیکیشن و یا حذف یک مازول خاص اقدام نمایید.

PreConditions

این ویژگی می‌تواند در خط معرفی مازول، مورد استفاده قرار بگیرد. اگر به فایل نگاه کنید می‌بینید که در بعضی خطوط این ویژگی ذکر شده است. تعریف این ویژگی به هسته IIS می‌گوید که این مازول در چه مواردی و به چه شیوه ای باید به کار گرفته شود.

```
<add name="ManagedEngine64" image="%windir%\Microsoft.NET\Framework64\v2.0.50727\webengine.dll"
preCondition="integratedMode, runtimeVersionv2.0, bitness64" />

<add name="ManagedEngine" image="%windir%\Microsoft.NET\Framework\v2.0.50727\webengine.dll"
preCondition="integratedMode, runtimeVersionv2.0, bitness32" />

<add name="ManagedEngineV4.0_32bit" image="%windir%\Microsoft.NET\Framework\v4.0.30319\webengine4.dll"
preCondition="integratedMode, runtimeVersionv4.0, bitness32" />

<add name="ManagedEngineV4.0_64bit"
image="%windir%\Microsoft.NET\Framework64\v4.0.30319\webengine4.dll"
preCondition="integratedMode, runtimeVersionv4.0, bitness64" />
```

مقادیری که precondition میتواند بگیر شامل موارد زیر هستند: **bitness**

این گزینه به دو صورت bitness32 و bitness64 یافت می‌شود. امروزه پردازنده‌های 64 بیتی بسیار متداول شده اند و بسیاری از تولید کنندگان دارند به سمت عرضه ابزارهای 64 بیتی رو می‌آورند و به زودی عرضه‌های 32 بیتی را متوقف می‌کنند و به سمت سیستم عامل‌های 64 بیت سوییچ خواند کرد ولی باز هم هنوز برنامه‌های 32 بیتی زیادی هستند که مورد استفاده قرار می‌گیرند و نمی‌توان آن‌ها را نادیده گرفت. برای همین ویندوزهای 64 بیتی مایکروسافت در کنار محیط 64 بیتی‌شان از یک محیط 32 بیت به اسم WOW64 استفاده می‌کنند. در این حالت این امتیاز به شما داده می‌شود که از پروسه‌های کارگر 32 بیتی در کنار پروسه‌های کارگر 64 بیتی استفاده کنید و PreCondition به bitness32 یا bitness64 تنظیم می‌شود تا از صحت بارگزاری dll در یک محیط درست مطمئن شود. در صورتی که این خصوصیت ذکر نشود یک هندلر 32 بیتی و 64 بیتی و یک module map اجرا می‌شود.

Runtime version

اگر ماژول خاصی برای اجرا به ورژن خاصی از .net framework نیاز دارد، این ویژگی ذکر می‌شود. در صورتی که ماژولی قصد اجرای بر روی فریم ورک اشتباهی داشته باشد سبب خطا خواهد شد.

ManagedHandler

با معرفی IIS7 ما با یک مدل توسعه پذیر روبرو شدیم و می‌توانستیم ماژول‌ها و هندلرهای خود را بنویسیم و مستقیماً در Pipeline قرار دهیم ولی سوییچ کردن بین دو بخش کدهای مدیریت شده و native یک عمل سنگین برای سیستم به شمار می‌آید و به منظور کاهش این بار گزینه managedhandler قرار داده شده است تا تعیین کند مواقعی که درخواست نیازی به این ماژول ندارد، این ماژول اجرا نگردد. به عنوان مثال فایل‌های ایستا چون jpg یا html ... شامل این ماژول نخواهند شد. واضح‌ترین مثال در این زمینه Forms Authentication می‌باشد و مواقعی اجرا می‌شوند که درخواست فایل‌های aspx شده باشد و اگر یک فایل html را درخواست کنید این ماژول امنیتی روی آن اثری ندارد و عملیات شناسایی هویت روی آن اجرا نمی‌شود و اگر می‌خواهید روی همه فایل‌ها، این عملیات شناسایی انجام شود باید خصوصیت precondition="managedhandler" حذف شود. در صورتی که تگ module را به صورت زیر بنویسید تمامی ماژول‌ها برای تمامی درخواست‌ها اجرا خواهد شد، صرف نظر از اینکه آیا ماژولی به صورت precondition="managedmodule" مقداردهی شده است یا خیر.

```
<modules runAllManagedModulesForAllRequests="true"/>
```

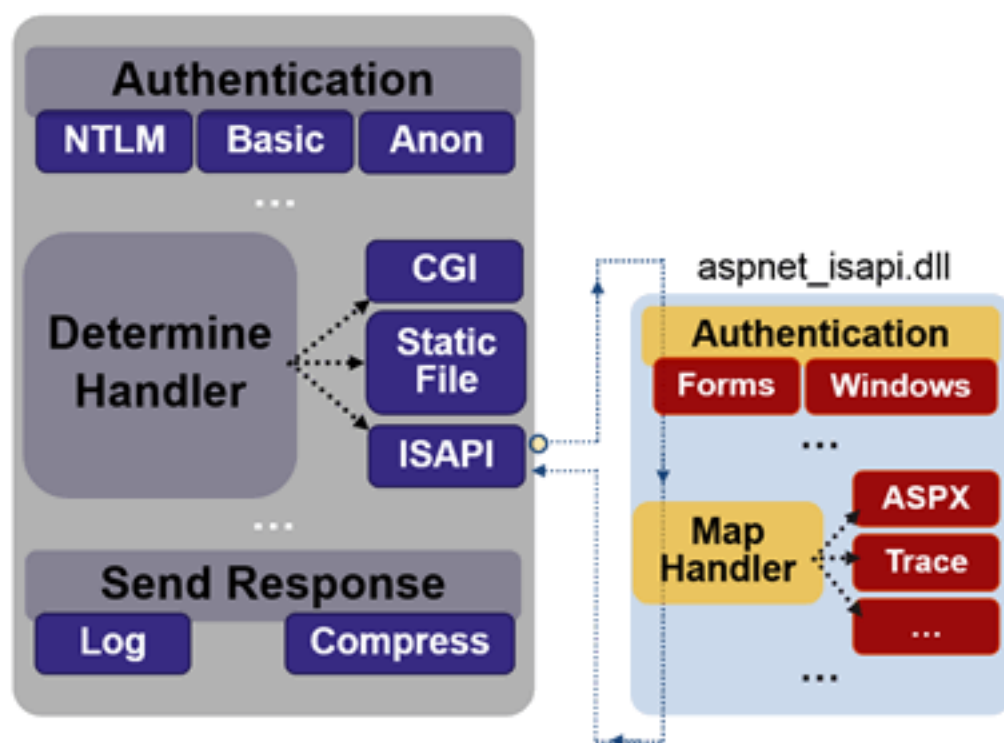
The Mode Precondition

تا به الان گفتیم که چگونه می‌توانیم یک ماژول و یا هندلر مدیریت شده‌ای را به Pipeline اضافه کنیم؛ ولی IIS7 به بالا نیاز دارد که تا پروسه‌های کارگر را به روشی خاص به این منظور اجرا کند و فریم ورک دات نت را برای اجرای آن‌ها بارگزاری کند. همچنین به اجرای ماژولی به اسم webengine.dll برای مدیریت ماژول‌های مدیریت شده نیازمند است و خود IIS در مورد کدهای مدیریت شده چیزی متوجه نمی‌شود. پس ما برای اینکه IIS را متوجه این موضوع نمائیم، باید Integrated mode را به آن معرفی کنیم. در نسخه‌های قبلی IIS یک روش قدیمی برای کدهای مدیریت شده وجود داشت که از طریق اینترفیسی به نام ISAPI صورت می‌گرفت. در این حالت ASPNET_ISAPI.DLL مسئول این کار بود و اگر هنوز هم می‌خواهید از این dll در نسخه‌های جدیدتر IIS کمک بگیرید باید به جای معرفی classic mode ، integration mode را معرفی کنید. با برابر کردن precondition به مقدار "integratedmode" هندلر یا ماژول شما در یک پول با خصوصیت integrated بارگزاری خواهد شد و اگر مقدار آن "classicmode" باشد در یک پول بدون خاصیت integrated بارگزاری می‌شود. تفاوت بین دو روش کلاسیک و مجتمع integrated بر سر این هست که در روش جدید، ماژول شما به عنوان یک پلاگین برای IIS

دیده نمی‌شود و کد شما را جزئی از کامپوننت‌های خود به شمار می‌آورد. به صورت واضح‌تر در حالت کلاسیک موقعی که درخواستی وارد pipeline میشد ابتدا از کامپوننت‌ها و ماژول‌های داخلی خود IIS عبور داده میشد و بعد فایل ASPNET_ISAPI.DLL جهت پردازش کدهای مدیریت شده صدا زده میشد و با توجه به کدهای شما، بعضی مراحل تکرار میشد؛ مثلاً اگر کد شما در مورد Authentication بود و بعد از گذر از مراحل auth داخل خود IIS و بقیه موارد دوباره نوبت کد شما و گذر از مراحل authentication بود. یعنی وجود دو pipeline؛ ولی در حالت مجتمع این دوبار انجام وظیفه از بین رفته است چرا که کدهای شما به طور مستقیم در pipeline قرار دارند و آن‌ها را جزئی از خود می‌دانند، نه یک پلاگین که افزون بر فعالیت خودشان، اجرای کدهای شما رو هم بر دوش بکشند.

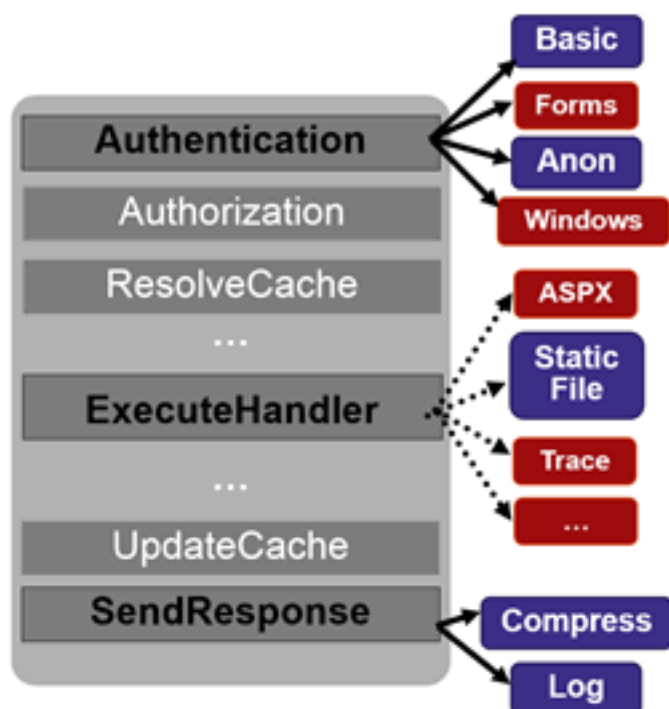
شکل زیر نمونه‌ای از حالت کلاسیک را نشان می‌دهد که در آن دو بار عمل auth دارد انجام می‌گیرد.

IIS6 ASP.NET Integration



شکل زیر هم نمونه‌ای از حالت مجتمع هست:

IIS7 ASP.NET Integration



- Classic Mode
 - runs as ISAPI
- Integrated Mode
 - .NET modules / handlers plug directly into pipeline
 - Process all requests
 - Full runtime fidelity

در کل امروزه دیگر استفاده از روش کلاسیک راهکار درستی نیست و این ویژگی تنها به عنوان یک سازگاری با نمونه کارهای قدیمی است.

تگ‌هایی که از خصوصیت precondition استفاده می‌کنند به شرح زیر هستند:

ISAPI filters

globalModules

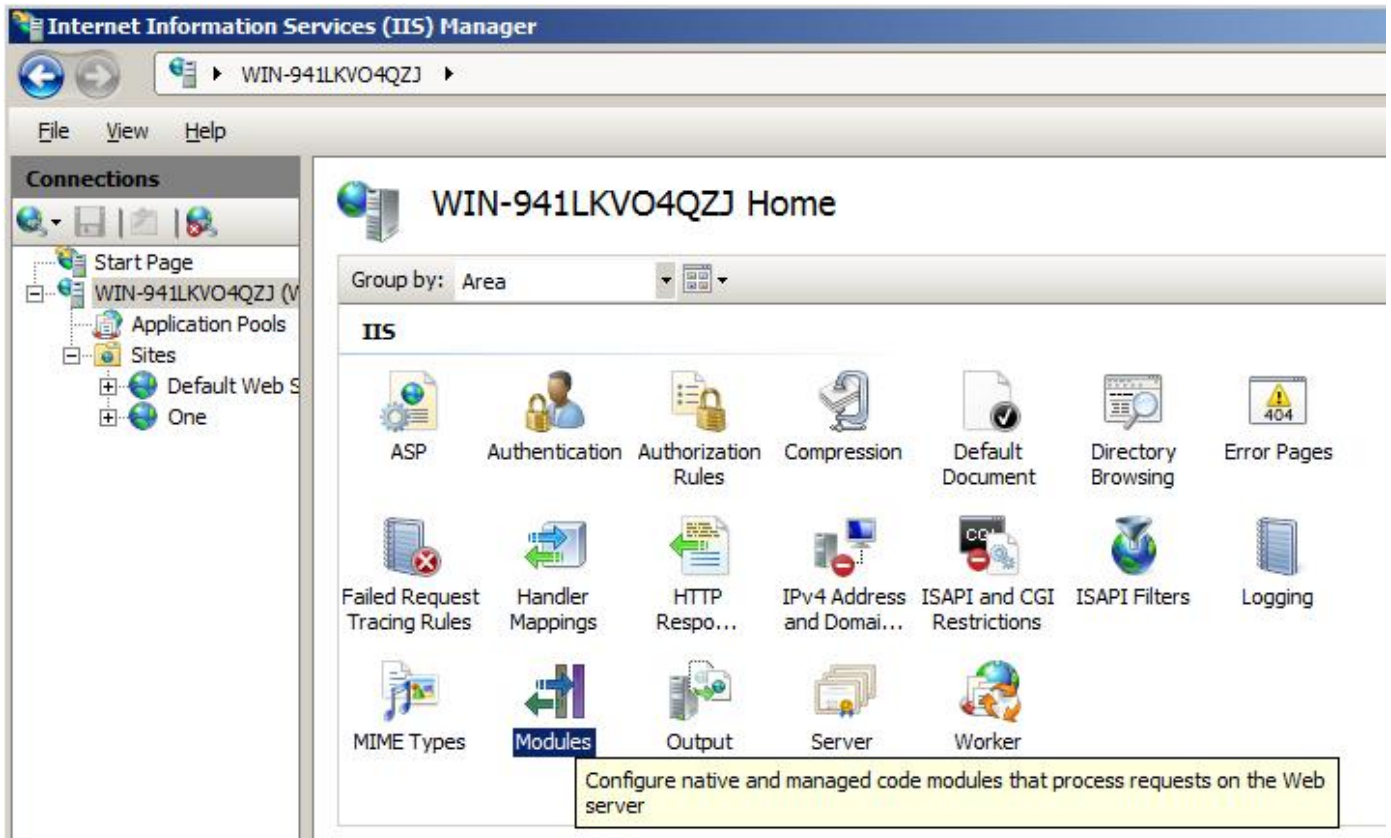
handlers

modules

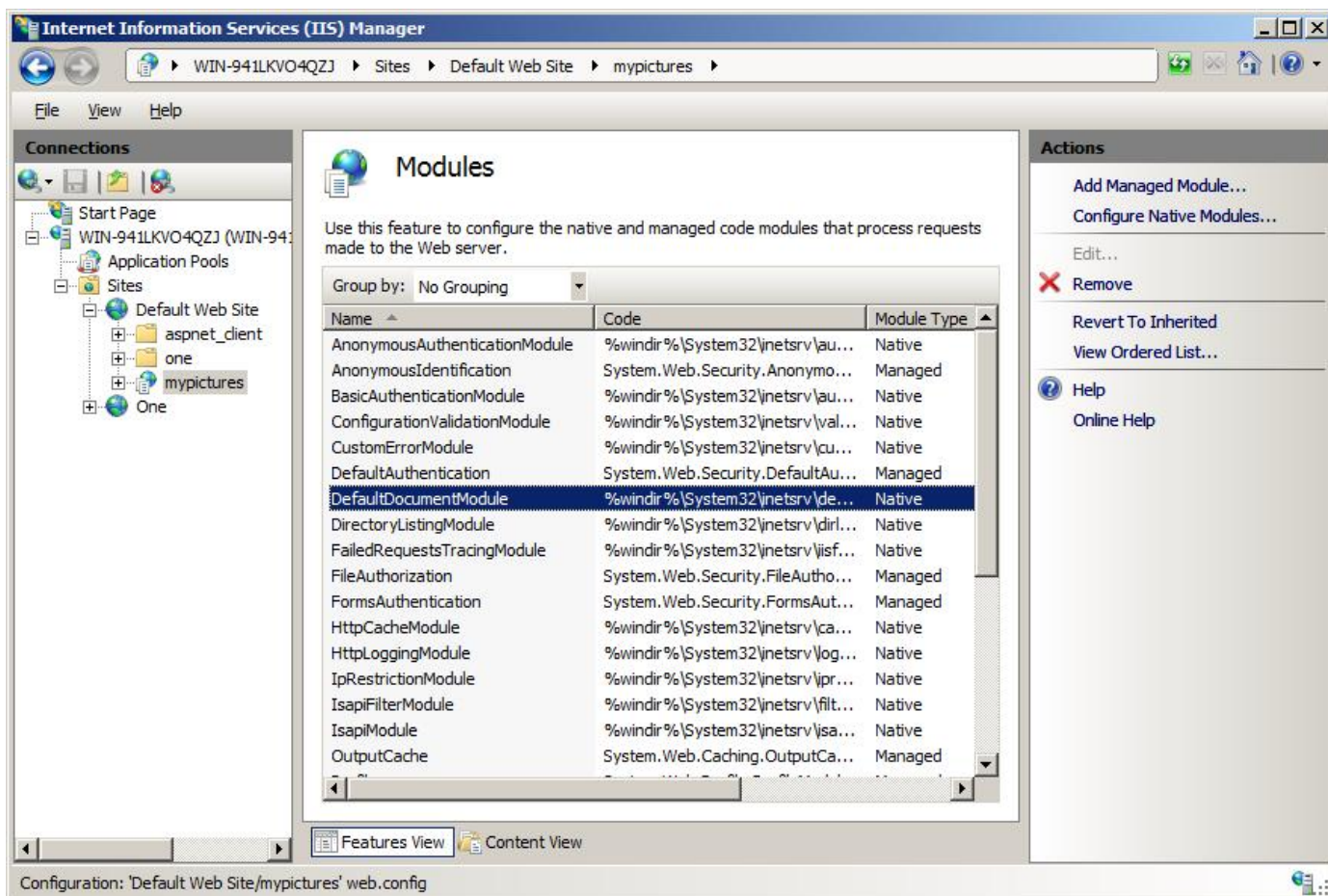
در مورد بقیه تگ‌ها در آینده بیشتر بحث می‌کنیم. بهتر هست مطلب را با توضیح precondition جهت ممانعت از طولانی و طومار شدن در اینجا ببندیم و در قسمت آینده دیگر روش‌های نصب ماژول‌مان را دنبال کنیم.

در مطلب قبلی روش دستی را برای اضافه کردن ماژول‌های خود، نام بردیم. در اینجا به روش‌های دیگر اضافه کردن ماژول‌ها می‌پردازیم.

استفاده از محیط گرافیکی IIS جهت لیست کردن، اضافه و حذف ماژول‌ها



به بخش modules در IIS بروید. در پنل سمت راست همه امکانات جهت افزودن و ویرایش و حذف وجود دارند:



روش معرفی ماژول در خط فرمان با استفاده از دستور Appcmd

```
Appcmd.exe install module /name:MODULE_NAME /image:PATH_TO_DLL
```

قسمت name که نام ماژول است و قسمت image هم مسیر قرار گرفتن فایل dll هست.

برای نمونه:

```
%windir%\system32\inetsrv\appcmd.exe install module /name:DefaultDocumentModule /image:%windir%\system32\inetsrv\defdoc.dll
```

در صورتیکه ماژولی که قبلا افزوده شده باشد را بخواهید اضافه کنید، خطای زیر را دریافت خواهید کرد:

```
ERROR ( message:Failed to add duplicate collection element "DefaultDocumentModule". )
```

جهت حذف ماژول دستور زیر را صادر کنید:

```
Appcmd.exe uninstall module MODULE_NAME
```

نمونه:

```
%windir%\system32\inetsrv\appcmd.exe uninstall module DefaultDocumentModule
```


گرفتن کوئری یا لیستی از ماژول‌های فعال برای یک اپلیکیشن یا عمومی:

```
Appcmd.exe list modules [/app.name:APPLICATION_NAME]
```

سوئیچ `app.name` اختیاری است ولی اگر نام یک اپلیکیشن را به آن بدهید، فقط ماژول‌هایی را که روی این اپلیکیشن اجرا می‌شوند، لیست می‌کند.

نمونه:

```
%windir%\system32\inetsrv\appcmd.exe list modules /app.name:"Default Web Site"
```

کد زیر هم نمونه ای برای لیست کردن تمامی ماژول‌های عمومی که بر روی تمامی اپلیکیشن‌ها اجرا می‌شوند:

```
%windir%\system32\inetsrv\appcmd.exe list modules
```

خط زیر یک ماژول را برای همه اپلیکیشن‌ها یا اپلیکیشن خاصی فعال می‌کند که بستگی دارد سوئیچ `type` چگونه مقداردهی شده باشد:

```
Appcmd.exe add module /name:MODULE_NAME /type:MGD_TYPE
```

برای مثال خط زیر باعث می‌شود ماژول Forms Authentication فقط برای وب اپلیکیشن default web site فعال شود:

```
%windir%\system32\inetsrv\appcmd.exe add module /name:FormsAuthentication  
/type:System.Web.Security.FormsAuthenticationModule /app.name:"Default Web Site"
```

یا در پایین آن را به صورت عمومی یا global فعال می‌کند:

```
%windir%\system32\inetsrv\appcmd.exe add module /name:FormsAuthentication  
/type:System.Web.Security.FormsAuthenticationModule
```

برای غیرفعال کردن یک ماژول از دستور زیر استفاده می‌شود:

```
Appcmd.exe delete module MODULE_NAME [/app.name:APPLICATION_NAME]
```

اگر غیر فعال کردن یک ماژول در یک اپلیکیشن خاص مدنظر شما باشد دستور زیر نمونه آن است:

```
%windir%\system32\inetsrv\appcmd.exe delete module FormsAuthentication /app.name:"Default Web Site"
```

اگر قصد دارید آن‌را بر روی تمامی اپلیکیشن‌ها غیرفعال کنید، دستور زیر نمونه آن است:

```
%windir%\system32\inetsrv\appcmd.exe delete module FormsAuthentication
```

حفظ کردن یا به خاطر سپردن دستورات بالا ممکن است کار سخت و دشواری باشد، به همین جهت از `help` کمک بگیرید:

```
Appcmd.exe module /?
```

یا به شکل اختصاصی‌تر برای یک دستور

```
Appcmd.exe install module /?Appcmd add module /?
```

در این قسمت بیشتر یک سری از ماژول‌ها را به شما در قالب جداول گروه بندی شده معرفی خواهیم کرد :

همانطور که در قسمت‌های قبلی گفتیم سرور IIS آماده خصوصی سازی و کار بر اساس علائق شماست؛ ولی توجه داشته باشید حذف تمامی ماژول‌ها ممکن است اثرات جانبی هم داشته باشد. در اینجا ما ماژول هایی را به شما معرفی می‌کنیم که بدانید کار هر ماژول چیست تا مثلا با حذف ماژولی، امنیت وب سایت خود را به خطر نیندازید :

ماژول‌های سودمند یا utility

نام ماژول:	UriCacheModule
توضیح:	این ماژول نوعی کش برای URLها به شمار می‌رود. موقعی که url درخواست می‌شود، اطلاعات در اولین درخواست خوانده شده و کش می‌شود و اگر دوباره همان url درخواست شود، بدون خواندن تنظیمات و بر اساس تنظیمات قبلی، کار url مربوطه را انجام میدهد تا اطلاعات پیکربندی تغییر کند و بر اساس اطلاعات جدید، خود را به روز کند.
تگ قابل پیکربندی:	لازم ندارد
وابستگی:	ندارد
اثرات حذف آن:	کارایی سیستم کاهش می‌یابد و سیستم مجبور است برای هر درخواست فایل پیکربندی را بخواند.
نام ماژول :	FileCacheModule
توضیح :	فایل هندل فایل‌هایی که قبلا در سرور باز شده‌اند را کش می‌کند تا در صورت نیاز در دفعات بعدی سریعتر عمل کند.
تگ قابل پیکربندی :	لازم ندارد .
وابستگی :	ندارد.
اثرات حذف آن :	کارایی سیستم کاهش می‌یابد. سیستم در هر اجرای دستور مربوط به فایل‌ها باید فایل هندل را به دست آورد.

نام ماژول :	TokenCacheModule
توضیح :	توکن‌های امنیتی ویندوز که پسوردهایی بر اساس authentication schemes هستند را کش می‌کند (anonymous authentication, basic authentication, IIS client authentication, certificate authentication)
تگ قابل پیکربندی :	لازم ندارد
وابستگی :	ندارد
اثرات حذف آن :	کارایی سیستم به شدت پایین می‌آید. کاربر باید با هر درخواستی لاگین کند. یکی از اصلی‌ترین ضربه‌ها با حذف این ماژول این است که اگر مثلاً یک پسورد از یک فایل html محافظت می‌کند و این صفحه به 50 تصویر ارجاع دارد، 51 بار باید درخواست لاگین اجرا گردد یا شاید هم بدتر

MANAGED ENGINE: ASP.NET INTEGRATION

نام ماژول :	ManagedEngine
توضیح :	مدیریت ماژول‌های native و مدیریت شده
تگ قابل پیکربندی :	
وابستگی :	ندارد
اثرات حذف آن :	مشخصاً غیرفعال شدن asp.net integrated و غیر فعال شدن تمامی ماژول‌ها و هندلرهای تگ وب کانفیگ یا داخل فایل کانفیگ IIS که در مقالات قبلی به تفصیل بیان کرده‌ایم.

IIS 7 NATIVE MODULES

نام ماژول :	HttpCacheModule
توضیح :	مدیریت کش خروجی در http.sys بر اساس پیکربندی مثل تعریف سایز کش و ...
تگ قابل پیکربندی :	System.webServer/caching

نام ماژول :	HttpCacheModule
وابستگی :	ندارد.
اثرات حذف آن :	محتوا دیگر به صورت کرنل مد، کش نمی‌شود و کش پروفایل هم ندید گرفته می‌شود و احتمالاً بر کارایی و استفاده از منابع هم اثر می‌گذارد.
نام ماژول :	DynamicCompressionModule
توضیح :	پیاده سازی in-memory compression در محتوای پویا
تگ قابل پیکربندی :	system.webServer/httpCompression and system.webServer/urlCompression.
وابستگی :	وابستگی ندارد چرا که به طور پیش فرض غیرفعال است.

نام ماژول :	StaticCompressionModule
توضیح :	پیداسازی فشرده سازی در محتوای ایستا و برای فایل‌های سیستمی از نوع in memory
تگ قابل پیکربندی :	system.webServer/httpCompression and system.webServer/urlCompression
وابستگی :	ندارد.
اثرات حذف آن :	در صورت عدم فشرده سازی بر مصرف ترافیک تاثیر گذار است.
نام ماژول :	DefaultDocumentModule
توضیح :	پیاده سازی یک لیست سند پیش فرض. درخواست‌ها مدام پشت سر هم می‌آیند و این درخواست‌های پشت سرهم، به سند پیش فرض هدایت می‌شوند. همان پنجره ای که شما به ترتیب فایل‌های index.htm,index.asp,default.aspx و ... را تعیین می‌کنید.
تگ قابل پیکربندی :	system.webServer/defaultDocument

نام ماژول :	StaticCompressionModule
وابستگی :	ندارد.
اثرات حذف آن :	درخواست را به ریشه هدایت می‌کند. مثلاً برای localhost صفحه 404 باز میگرداند و اگر directoryBrowsing فعال باشد لیستی از دایرکتوری ریشه را باز میگرداند.

نام ماژول :	DirectoryListingModule
توضیح :	پیادی سازی لیستی از محتویات یک دایرکتوری
تگ قابل پیکربندی :	system.webServer/directoryBrowse
وابستگی :	ندارد.
اثرات حذف آن :	اگر این ماژول و ماژول قبلی غیرفعال باشند response نهایی خالی است.
نام ماژول :	ProtocolSupportModule
توضیح :	پیاده سازی اختصاصی از response header پیاده سازی تنظیمات trace و HTTP verbs. پیاده سازی تنظیمات مربوطه به keep-alive بر اساس پیکربندی
تگ قابل پیکربندی :	system.webServer/httpProtocol
وابستگی :	ندارد.
اثرات حذف آن :	بازگرداندن پیام خطای "Method not allowed 405".

نام ماژول :	HttpRedirectionModule
توضیح :	پیاده سازی عملیات انتقال یا redirect

HttpRedirectionModule	نام ماژول :
system.webServer/httpRedirect	تگ قابل پیکربندی :
ندارد.	وابستگی :
خطر امنیتی: اگر منابعی با redirect کردن محافظت می‌شوند، از این پس در دسترسند.	اثرات حذف آن :
ServerSideIncludeModule	نام ماژول :
حمایت از فایل shtml یا shtml و ...	توضیح :
system.webServer/serverSideInclude	تگ قابل پیکربندی :
ندارد.	وابستگی :
این فایل‌ها به صورت متنی نمایش داده می‌شوند	اثرات حذف آن :

StaticFileModule	نام ماژول :
فایل‌های ایستا را به همراه پسوند ارسال می‌کند. مثل jpg,html و نوع محتوا را بر اساس staticContent/mimeMap پیکربندی می‌کند.	توضیح :
system.webServer/staticContent	تگ قابل پیکربندی :
ندارد.	وابستگی :
فایل‌های ایستا دیگر ارائه نشده و به جای آن خطای 404 بازگشت داده می‌شود.	اثرات حذف آن :
AnonymousAuthenticationModule	نام ماژول :
پیاده سازی سیستم شناسایی افراد ناشناس. همانطور که میدانید در یک وب سایت حداقل محتوایی برای افرادی بدون داشتن اکانت هم وجود دارد. برای اینکار یک شیء httpuser ایجاد می‌کند.	توضیح :
	تگ قابل پیکربندی :

نام ماژول :	StaticFileModule
	system.webServer/security/authentication/anonymousAuthentication
وابستگی :	ندارد.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار داده ای در IIS می باشد و در صورت نبودن هیچ سیستم شناسایی وجود نداشته و در نبود شیء httpuser سیستم خطای 401.2 را تولید می کند.

نام ماژول :	CertificateMappingAuthenticationModule
توضیح :	مجوز SSL را به Active Directory نگاشت می کند.
تگ قابل پیکربندی :	system.webServer/security/authentication/clientCertificateMappingAuthentication
وابستگی :	برای اینکه این ماژول وظیفه خود را انجام دهد باید تنظیمات SSL انجام شود و همچنین سیستم IIS جزئی از دامنه Active directory باشد
اثرات حذف آن :	درخواست ها، نرمال رسیدگی میشوند انگار SSL وجود ندارد.
نام ماژول :	BasicAuthenticationModule
توضیح :	پیاده سازی پایه ای و روتین شناسایی کاربران بر اساس آن چیزی که در استاندارد زیر آمده است RFC 2617 .
تگ قابل پیکربندی :	system.webServer/security/authentication/basicAuthentication
وابستگی :	None.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار

نام ماژول :	CertificateMappingAuthenticationModule
	داده‌ای در IIS می‌باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می‌کند.

نام ماژول :	WindowsAuthenticationModule
توضیح :	((windows Authentication (NTLM or Negotiate (Kerberos
تگ قابل پیکربندی :	system.webServer/security/authentication/windowsAuthentication
وابستگی :	ندارد.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار داده ای در IIS می‌باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می‌کند.
نام ماژول :	DigestAuthenticationModule
توضیح :	پیاده سازی سیستم شناسایی دیاجست بر اساس RFC 2617 .
تگ قابل پیکربندی :	system.webServer/security/authentication/digestAuthentication
وابستگی :	IIS باید بخشی از دامنه Active Directory باشد.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار داده ای در IIS می‌باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می‌کند.

نام ماژول :	IISCertificateMappingAuthenticationModule
توضیح :	پیاده سازی نگاشت مجوزهای IIS، نگهداری و ذخیره اطلاعات همه نگاشت ها و مجوزهای کاربری چون SSL client certificates
تگ قابل پیکربندی :	system.webServer/iisClientCertificateMappingAuthentication
وابستگی :	اطلاعات SSL به همراه دریافت client certificates جهت پیکربندی این ماژول
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناسایی یا authenticate وجود داشته باشد. httpuser یک ساختار داده ای در IIS می باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می کند.
نام ماژول :	UrlAuthorizationModule
توضیح :	پیاده سازی authorization بر اساس قوانین پیکربندی شده
تگ قابل پیکربندی :	system.webServer/security/authorization
وابستگی :	ندارد.
اثرات حذف آن :	محتوای محافظت شده توسط authorization دیگر محافظت نمی شوند.

نام ماژول :	IsapiModule
توضیح :	پیاده سازی ISAPI Extension
تگ قابل پیکربندی :	system.webServer/isapiCgiRestriction
وابستگی :	ندارد.

IsapiModule	نام ماژول :
هندلرهای معرفی شده در بخش IsapiModule و تگ handlers دیگر اجرا نمی‌شوند	اثرات حذف آن :
IsapiFilterModule	نام ماژول :
پایاده سازی ISAPI filter	توضیح :
system.webServer/isapiFilters	تگ قابل پیکربندی :
ندارد.	وابستگی :
اگر برنامه ای از ISAPI filter استفاده می‌کند، در اجرا دچار مشکل خواهد شد.	اثرات حذف آن :

IpRestrictionModule	نام ماژول :
یک سیستم تشخیص دسترسی بر اساس آی پی‌های ورژن 4	توضیح :
system.webServer/security/ipSecurity	تگ قابل پیکربندی :
IPv4 stack باید نصب شود.	وابستگی :
کلاینت هایی که IP هایشان در IPsecurity لیست شده‌اند ندید گرفته میشوند	اثرات حذف آن :
RequestFilteringModule	نام ماژول :
پایاده سازی یک مجموعه قدرتمند از قوانین امنیتی که درخواست‌های مشکوک را پس می‌زند.	توضیح :
system.webServer/security/requestFiltering	تگ قابل پیکربندی :
ندارد.	وابستگی :
دیگر قوانین امنیتی اجرا نخواهند شد و سبب وجود مشکلات امنیتی میشود.	اثرات حذف آن :

نام ماژول :	CustomLoggingModule
توضیح :	<p>پیاده سازی اینترفیس ILogPlugin در سمت IIS، به مشتریان اجازه میدهد تا لاگ‌های خود را توسعه دهند. هر چند این روش توصیه نمی‌شود و توصیه کارشناس مایکروسافت استفاده از یک ماژول دست نویس از نوع RQ_LOG_REQUEST می باشد.</p> <p>Implements the ILogPlugin interface on top of IIS.</p> <p>ILogPlugin is a previous COM implementation that allowed customers to extend IIS logging. We do not recommend extending IIS using this interface. Instead, customers should write a module and subscribe to the RQ_LOG_REQUEST notification.</p>
تگ قابل پیکربندی :	system.webServer/httpLogging and system.applicationhost/sites/site/logFile/customLogPluginClsid
وابستگی :	ندارد.
اثرات حذف آن :	مسئله پلاگین‌های این اینترفیس از کار می‌افتند که سیستم ODBC Logging هم جز آن است.
نام ماژول :	CustomErrorModule
توضیح :	پیاده سازی مدیریت خطاهای ویژه
تگ قابل پیکربندی :	system.webServer/httpErrors
وابستگی :	None.
اثرات حذف آن :	در صورتی که خطایی از هسته باشد، نتیجه یک صفحه، با توضیح مختصری از خطا خواهد بود. در غیر این صورت اگر خطا از برنامه یا کامپوننتی باشد جزئیات خطا فاش خواهد شد
نام ماژول :	HttpLoggingModule
توضیح :	پیاده سازی سیستم logging استاندارد http.sys
تگ قابل پیکربندی :	system.applicationHost/log and

نام ماژول :	HttpLoggingModule
	system.webServer/httpLogging
وابستگی :	ندارد.
اثرات حذف آن :	از کار افتادن سیستم لاگ
نام ماژول :	FailedRequestsTracingModule
توضیح :	پیاده سازی سیستم ردیابی درخواست‌های ناموفق و اجرای قوانین، طبق پیکربندی
تگ قابل پیکربندی :	system.webServer/tracing and system.webServer/httpTracing
وابستگی :	ندارد.
اثرات حذف آن :	Tracing http requests will no longer work.

نام ماژول :	RequestMonitorModule
توضیح :	پیاده سازی IIS Run-time State and Control Interface یا به اختصار RSCA . به کاربران اجازه می‌دهد از اطلاعات، حین اجرا، کوئری بگیرند. مثل درخواست در حال اجرای جاری، آغاز به کار یا توقف وب سایت و دامنه‌های اپلیکیشن در حال اجرای جاری
تگ قابل پیکربندی :	ندارد.
وابستگی :	ندارد.
اثرات حذف آن :	ابزارهای مرتبط با این موضوع از کار می‌افتند
نام ماژول :	CgiModule
توضیح :	پیاده سازی CGI در سمت IIS
تگ قابل پیکربندی :	system.webServer/cgi and system.webServer/isapiCgiRestriction

RequestMonitorModule	نام ماژول :
ندارد.	وابستگی :
برنامه‌های CGI متوقف می‌شوند	اثرات حذف آن :

TracingModule	نام ماژول :
پیاده سازی سیستم ردیابی ETW	توضیح :
system.webServer/httpTracing	تگ قابل پیکربندی :
ندارد.	وابستگی :
باعث از کار افتادن سیستم مربوطه می‌شود	اثرات حذف آن :
ConfigurationValidationModule	نام ماژول :
اعتبارسنجی تنظیمات برنامه ASP.Net که به حالت integrate انتقال یافته است	توضیح :
system.webServer/Validation	تگ قابل پیکربندی :
ندارد.	وابستگی :
عدم اعتبارسنجی و در نتیجه عدم نمایش خطاها	اثرات حذف آن :

:MANAGED MODULES

OutputCache	نام ماژول :
پیاده سازی output caching	توضیح :
system.web/caching/outputCache	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
عدم اجرای output cache	اثرات حذف آن :

OutputCache	نام ماژول :
Session	نام ماژول :
مدیریت سشن ها	توضیح :
system.web/sessionState	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
سشن ها از دسترس خارج می شوند.	اثرات حذف آن :

WindowsAuthentication	نام ماژول :
اینجا	توضیح :
system.web/authentication	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
این حالت قابل اجرا نخواهد بود	اثرات حذف آن :
FormsAuthentication	نام ماژول :
اینجا	توضیح :
system.web/authentication	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
این حالت قابل اجرا نیست و کاربران مجوز دار هم نمی توانند به منابع محافظت شده دسترسی داشته باشند.	اثرات حذف آن :

نام ماژول :	DefaultAuthentication
توضیح :	اطمینان از وجود شی Authentication در context مربوطه
تگ قابل پیکربندی :	system.web/authentication
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	اگر مد Forms authentication انتخاب شده باشد بر روی بعضی از کاربران ناشناس کار نخواهد کرد و رویداد DefaultAuthentication.OnAuthenticate اجرا نخواهد شد.
نام ماژول :	RoleManager
توضیح :	اینجا
تگ قابل پیکربندی :	
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	این قابلیت در دسترس نمی‌باشد

نام ماژول :	UrlAuthorization
توضیح :	اینجا
تگ قابل پیکربندی :	system.web/authorization.
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	باعث از کار افتادن asp.net authorization و فاش شدن بعضی اطلاعات و همچنین دیگر تهدیدات امنیتی

UrlAuthorization	نام ماژول :
AnonymousIdentification	نام ماژول :
اینجا	توضیح :
	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
The anonymous identification feature used by the ASP.NET Profile will not work.	اثرات حذف آن :

Profile	نام ماژول :
اینجا	توضیح :
	تگ قابل پیکربندی :
ManagedEngine module must be installed.	وابستگی :
ASP.Net Profile از کار خواهد افتاد	اثرات حذف آن :
UrlMappingsModule	نام ماژول :
تبدیل یک Url واقعی به یک Url کاربرپسند	توضیح :
	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
نگاشت Urlها صورت نمی‌گیرد	اثرات حذف آن :

پس از بررسی مفاهیم، بهتر هست وارد یک کار عملی شویم. مثال مورد نظر، یک مثال از وب سایت شرکت مایکروسافت است که هنگام نمایش تصاویر، بر حسب پیکربندی موجود، یک پرچسب یا تگی را در گوشه‌ای از تصویر درج می‌کند. البته تصویر را ذخیره نمی‌کنیم و تگ را بر روی تصویر اصلی قرار نمی‌دهیم. تنها هنگام نمایش به کاربر، روی response خروجی آن را درج می‌کنیم.

قبلا ما در این [مقاله](#) به بررسی httpHandler پرداخته‌ایم، ولی بهتر هست در این مثال کمی حالت پیشرفته‌تر آن را بررسی کنیم.

ابتدا اجازه دهید کمی قابلیت‌های فایل کانفیگ IIS را گسترش دهیم.

مسیر زیر را باز کنید:

```
%windir%\system32\inetsrv\config\schema
```

یک فایل xml را با نام imagecopyright.xml ساخته و تگ‌های زیر را داخلش قرار دهید:

احتمال زیاد دسترسی برای ویرایش این دایرکتوری به خاطر مراتب امنیتی با مشکل برخورد برای ویرایش این نکته امنیتی از [اینجا](#) یا به خصوص از [اینجا](#) کمک بگیرید.

```
<configSchema>
  <sectionSchema name="system.webServer/imageCopyright">
    <attribute name="enabled" type="bool" defaultValue="false" />
    <attribute name="message" type="string" defaultValue="Your Copyright Message" />
    <attribute name="color" type="string" defaultValue="Red"/>
  </sectionSchema>
</configSchema>
```

با این کار ما یک شِما یا اسکیمای ایجاد کردیم که دارای سه خصوصیت زیر است:

enabled: آیا این هندلر فعال باشد یا خیر.

message: پیامی که باید به عنوان تگ درج شود.

color: رنگ متن که به طور پیش فرض قرمز رنگ است.

به هر کدام از تگ‌های بالا یک مقدار پیش فرض داده ایم تا اگر مقداردهی نشدند، ماژول طبق مقادیر پیش فرض کار خود را انجام دهد.

بعد از نوشتن شما، لازم هست که آن را در فایل applicationhost.config نیز به عنوان یک section جدید در زیر مجموعه system.webserver معرفی کنیم:

```
<configSections>
...
<sectionGroup name="system.webServer">
  <section name="imageCopyright" overrideModeDefault="Allow"/>
...
</sectionGroup>
</configSections>
```

تعریف کد بالا به شما اجازه می‌دهد تا در زیر مجموعه تگ system.webserver، برای هندلر خود تگ تعریف کنید. در کد بالا، شما خود را بر اساس نام فایل مشخص می‌کنیم و خصوصیت overrideModeDefault، یک قفل گذار امنیتی برای تغییر محتوای section است. در صورتی که allow باشد هر کسی در هر مرحله‌ی دسترسی در سیستم و در هر فضای نامی، در فایل‌های وب کانفیگ می‌تواند به مقادیر این section دسترسی یافته و آن‌ها را تغییر دهد. ولی اگر با Deny مقادیری شده باشد، مقادیر قفل شده و هیچ دسترسی برای تغییر آن‌ها وجود ندارد.

در مثال زیر ما به ماژول windows Authentication اجازه می‌دهیم که هر کاربری در هر سطح دسترسی به این section

دسترسی داشته باشد؛ از تمامی سایت‌ها یا اپلیکشین‌ها یا virtual directories موجود در سیستم و در بعضی موارد این گزینه باعث افزایش ریسک امنیتی می‌گردد.

```
<section name="windowsAuthentication" overrideModeDefault="Allow" />
```

در کد زیر اینبار ما دسترسی را بستیم و در تعاریف دامنه‌های دسترسی، دسترسی را فقط برای سطح مدیریت سایت AdministratorSite باز گذاشته‌ایم:

```
<location path="AdministratorSite" overrideMode="Allow">
  <security>
    <authentication>
      <providers>
        <windowsAuthentication enabled="false">
          </providers>
          <add value="Negotiate" />
          <add value="NTLM" />
        </windowsAuthentication>
      </authentication>
    </security>
```

برای خارج نشدن بیش از اندازه از بحث، به ادامه تعریف هندلر می‌پردازیم. بعد از معرفی یک section برای هندلر خود، می‌توانیم به راحتی تگ آن را در قسمت system.webserver تعریف کنیم. این کار می‌تواند از طریق فایل web.config سایت یا applicationhost.config صورت بگیرد یا می‌تواند از طریق ویرایش دستی یا خط فرمان appcmd صورت گیرد؛ ولی در کل باید به صورت زیر تعریف شود:

```
<system.webServer>
  <imageCopyright />
</system.webServer>
```

در کد بالا این تگ تنها معرفی شده است؛ ولی مقادیر آن پیش فرض می‌باشند. در صورتی که بخواهید مقادیر آن را تغییر دهید کد به شکل زیر تغییر می‌کند:

```
<system.webServer>
  <imageCopyright enabled="true" message="an example of www.dotnettips.info" color="Blue" />
</system.webServer>
```

در صورتی که می‌خواهید از خط فرمان کمک بگیرید به این شکل بنویسید:

```
%windir%\system32\inetsrv\appcmd set config -section:system.webServer/imageCopyright /color:yellow /message:"Dotnettips.info" /enabled:true
```

برای اطمینان از این که دستور شما اجرا شده است یا خیر، یک کوئری یا لیست از تگ مورد نظر در system.webserver بگیرید:

```
%windir%\system32\inetsrv\appcmd list config -section:system.webServer/imageCopyright
```

در این مرحله یک دایرکتوری برای پروژه تصاویر ایجاد کنید و در این مثال ما فقط تصاویر jpg را ذخیره می‌کنیم و در هنگام درج تگ، تصاویر jpg را هندل می‌کنیم؛ برای مثال ما:

```
c:\inetpub\mypictures
```

در این مرحله دایرکتوری ایجاد شده را به عنوان یک application معرفی می‌کنیم:

```
%windir%\system32\inetsrv\appcmd add app -site.name:"Default Web Site" -path:/mypictures -physicalPath:%systemdrive%\inetpub\mypictures
```

و برای آن ماژول DirectoryBrowse را فعال می‌کنیم. برای اطلاعات بیشتر به [مقاله قبلی](#) که به تشریح وظایف ماژول‌ها پرداختیم رجوع کنید. فقط به این نکته اشاره کنم که اگر کاربر آدرس localhost/mypictures را درخواست کند، فایل‌های این قسمت را برای ما لیست می‌کند. برای فعال سازی، کد زیر را فعال می‌کنیم:

```
%windir%\system32\inetsrv\appcmd set config "Default Web Site/mypictures" -section:directoryBrowse -enabled:true
```

حال زمان این رسیده است تا کد نوشته و فایل cs آن را در مسیر زیر ذخیره کنیم:

c:\inetpub\mypictures\App_Code\imagecopyrighthandler.cs

هندل مورد نظر در زبان سی شارپ :

```
#region Using directives
using System;
using System.Web;
using System.Drawing;
using System.Drawing.Imaging;
using Microsoft.Web.Administration;
#endregion

namespace IIS7Demos
{
    public class imageCopyrightHandler : IHttpHandler
    {
        public void ProcessRequest(HttpContext context)
        {
            ConfigurationSection imageCopyrightHandlerSection =
                WebConfigurationManager.GetSection("system.webServer/imageCopyright");

            HandleImage(
                context,
                (bool)imageCopyrightHandlerSection.Attributes["enabled"].Value,
                (string)imageCopyrightHandlerSection.Attributes["message"].Value,
                (string)imageCopyrightHandlerSection.Attributes["color"].Value
            );
        }

        void HandleImage(
            HttpContext context,
            bool enabled,
            string copyrightText,
            string color
        )
        {
            try
            {
                string strPath = context.Request.PhysicalPath;
                if (enabled)
                {
                    Bitmap bitmap = new Bitmap(strPath);
                    // add copyright message
                    Graphics g = Graphics.FromImage(bitmap);
                    Font f = new Font("Arial", 50, GraphicsUnit.Pixel);
                    SolidBrush sb = new SolidBrush(Color.FromName(color));
                    g.DrawString(
                        copyrightText,
                        f,
                        sb,
                        5,
                        bitmap.Height - f.Height - 5
                    );
                    f.Dispose();
                    g.Dispose();
                    // slow, but good looking resize for large images
                    context.Response.ContentType = "image/jpeg";
                    bitmap.Save(
                        context.Response.OutputStream,
                        System.Drawing.Imaging.ImageFormat.Jpeg
                    );
                    bitmap.Dispose();
                }
                else
                {
                    context.Response.WriteFile(strPath);
                }
            }
            catch (Exception e)
            {
                context.Response.Write(e.Message);
            }
        }

        public bool IsReusable
        {
            get { return true; }
        }
    }
}
```

در خط `WebConfigurationManager.GetSection` تگ `imagecopyright` تعریف شده باشد، همه اطلاعات این تگ را از فایل کانفیگ بیرون کشیده و داخل شیء `imageCopyrightHandlerSection` از نوع `ConfigurationSection` قرار می‌دهیم. سپس اطلاعات هر سه گزینه را خوانده و به همراه `context` (اطلاعات درخواست) به تابع `handleimage` که ما آن را نوشته ایم ارسال می‌کنیم. کار این تابع درج تگ می‌باشد.

در خطوط اولیه تابع، ما آدرس فیزیکی منبع درخواست شده را به دست آورده و در صورتیکه مقدار گزینه `enable` با `true` مقدار دهی شده باشد، آن را به شیء `bitmap` نسبت می‌دهیم و با استفاده از دیگر کلاس‌های گرافیکی، تگ مورد نظر را با متن و رنگ مشخص شده ایجاد می‌کنیم. در نهایت شیء `bitmap` را ذخیره و نوع خروجی `response` را از نوع `image/jpeg` تعریف می‌کنیم تا مرورگر بداند که خروجی ما یک تصویر است. ولی در صورتی که `enabled` با `false` مقداردهی شده باشد، همان تصویر اصلی را بدون درج تگ ارسال می‌کنیم.

فضای نام `Microsoft.Web.Administration` برای اجرای خود نیاز دارد تا اسمبلی آن رفرنس شود. برای اینکار به درون دایرکتوری `mypictures` رفته و در داخل فایل `web.config` که بعد از تبدیل این دایرکتوری به اپلیکیشن ایجاد شده بنویسید:

```
<system.web>
  <compilation>
    <assemblies>
      <add assembly="Microsoft.Web.Administration, Version=7.0.0.0,
Culture=neutral, PublicKeyToken=31bf3856ad364e35, processorArchitecture=MSIL"/>
    </assemblies>
  </compilation>
</system.web>
```

در صورتی که کلاس خود را کامپایل کنید می‌توانید آن را داخل پوشه‌ی `Bin` به جای `App_Code` قرار دهید و نیاز به رفرنس کرده اسمبلی `Microsoft.Web.Administration` نیز ندارید.

در آخرین مرحله فقط باید به IIS بگویید که تنها فایل‌های `jpg` را برای این هندلر، هندل کن. این کار را از طریق خط فرمان انجام می‌دهیم:

```
appcmd set config "Default Web Site/mypictures/" -section:handlers
/+[name='JPGImageCopyrightHandler',path='*.jpg',verb='GET',type='IIS7Demos.imageCopyrightHandler']
```

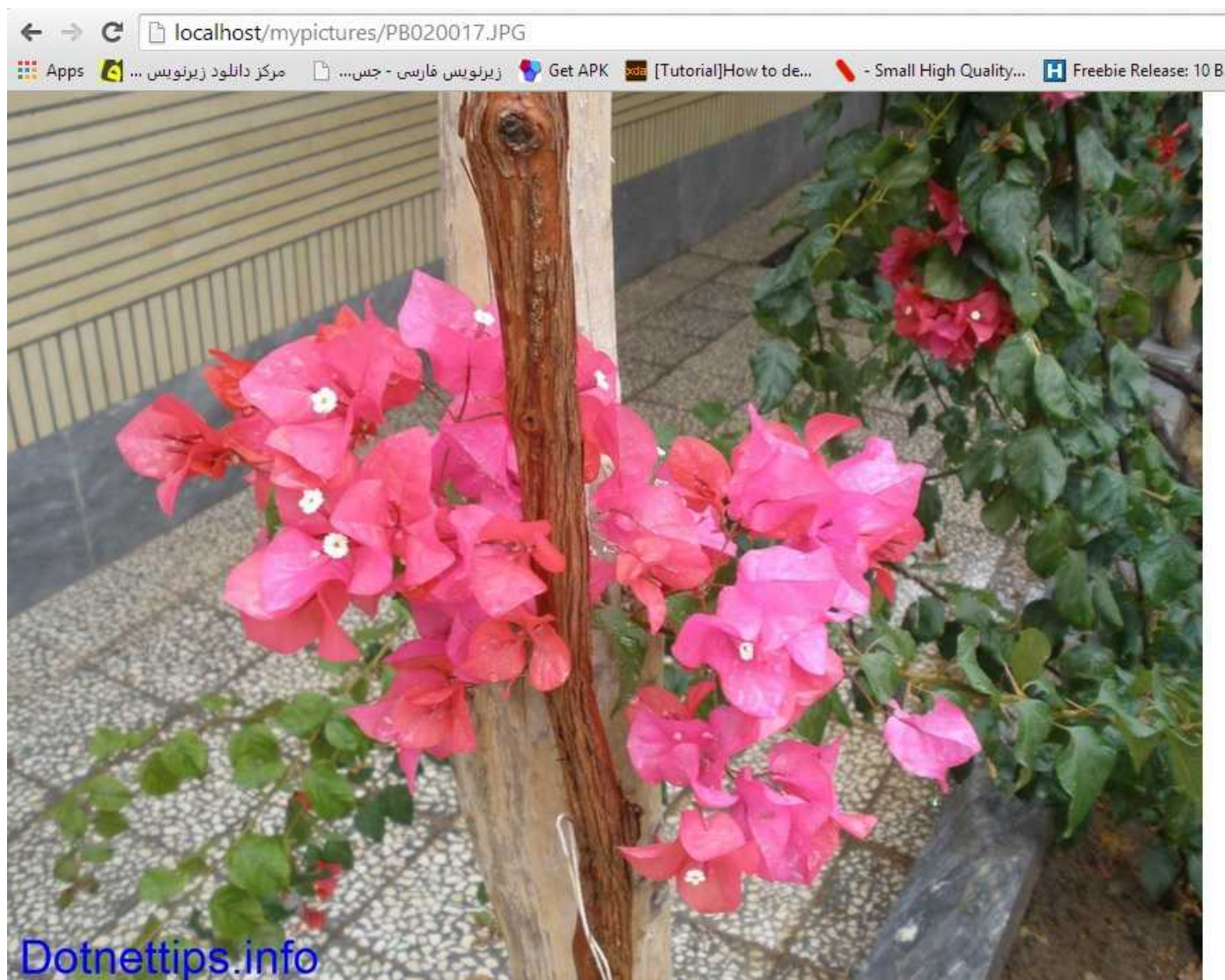
هندلر مورد نظر تنها برای این اپلیکیشن و در مسیر `mypicture` فعال شده و در قسمت `name`، یک نام اختیاری بدون فاصله و `unique` بر می‌گزینیم. در قسمت `path` نوع فایل‌هایی را که نیاز به هندلر هست، مشخص کردیم و در قسمت `verb` گفته‌ایم که تنها برای درخواست‌های نوع `GET`، هندلر را اجرا کن و در قسمت `type` هم که اگر [مقاله httphandler](#) را خوانده باشید می‌دانید که به معرفی هندلر می‌پردازیم؛ اولی نام فضای نام هست و بعد از . نام کلاس، که در اینجا می‌شود:

```
'IIS7Demos.imageCopyrightHandler'
```

الان همه چیز برای اجرا آماده است و فقط یک مورد برای احتیاط الزامی است و آن هم این است که پروسه‌های کارگر، ممکن است از قبل در حال اجرا بوده باشند و هنوز شمای جدید ما را شناسایی نکرده باشند، برای همین باید آن‌ها را با تنظیمات جدیدمان آشنا کنیم تا احیاناً برایمان استثناء صادر نشود:

```
appcmd recycle AppPool DefaultAppPool
```

کارمان تمام شده، چند تصویر داخل دایرکتوری قرار داده و درخواست تصاویر موجود را بدهید تا تگ را ببینید:



فعلا تا بدین جا کافی است. در قسمت آینده این هندلر را کمی بیشتر توسعه خواهیم داد.