

### آشنایی با Virtual Address spaces

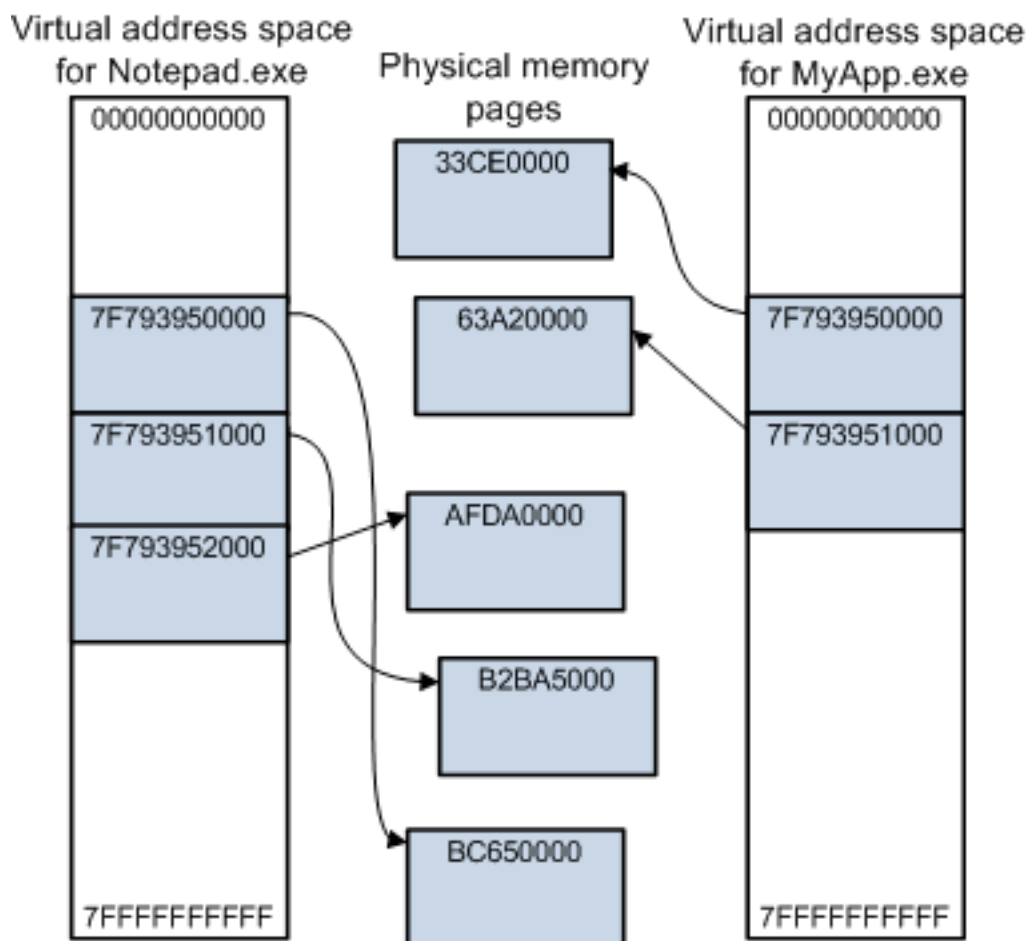
**فضای آدرس‌دهی مجازی:** موقعی که یک پردازشگر در مکانی از حافظه عمل خواندن و نوشتن را آغاز می‌کند، از آدرس‌های مجازی بهره می‌برد. بخشی از عملیات خواندن و نوشتن، تبدیل آدرس‌های مجازی به آدرس‌های فیزیکی در حافظه است. این عمل سه مزیت دارد:

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.

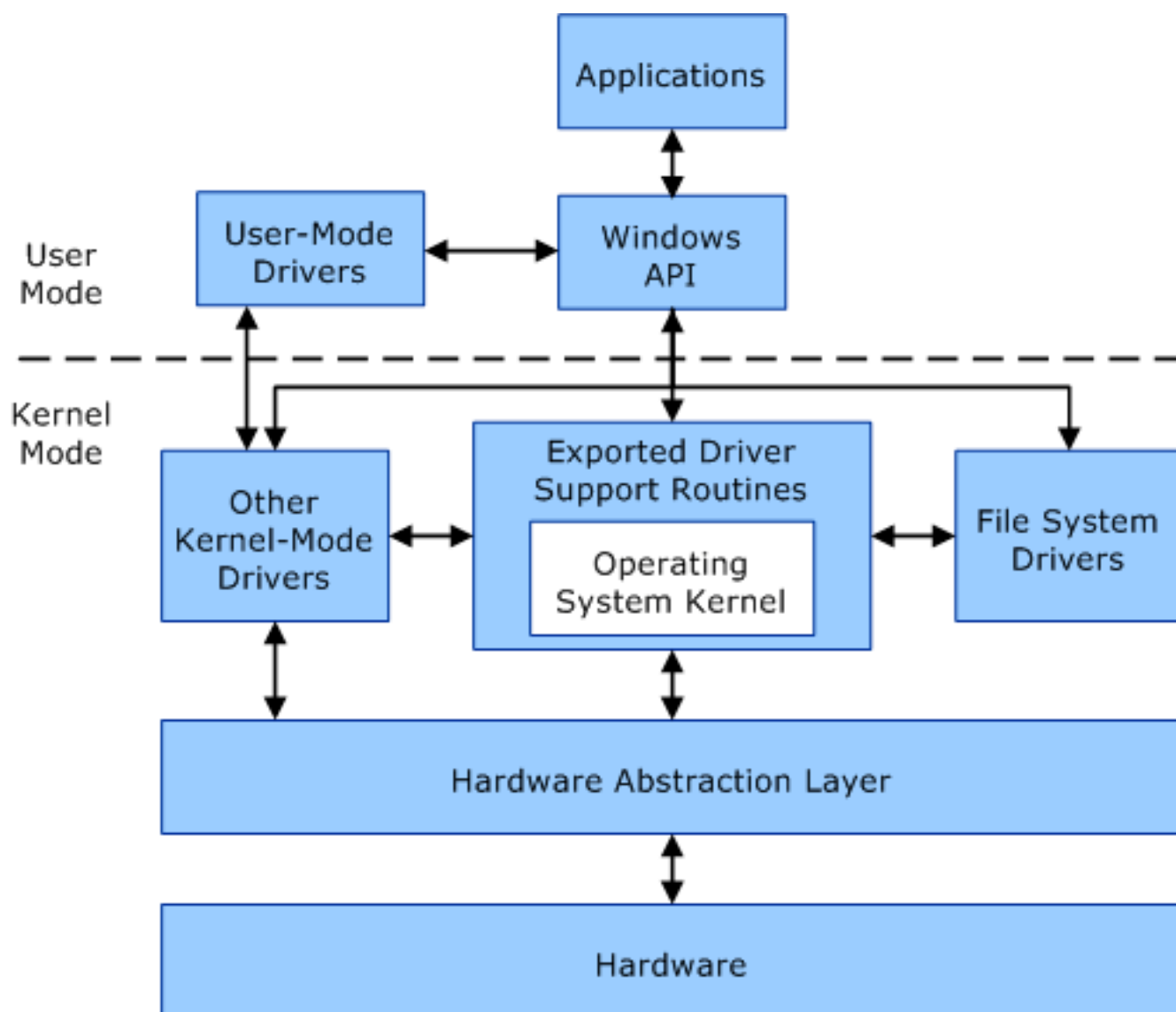
#### تفاوت kernel mode و user mode

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.

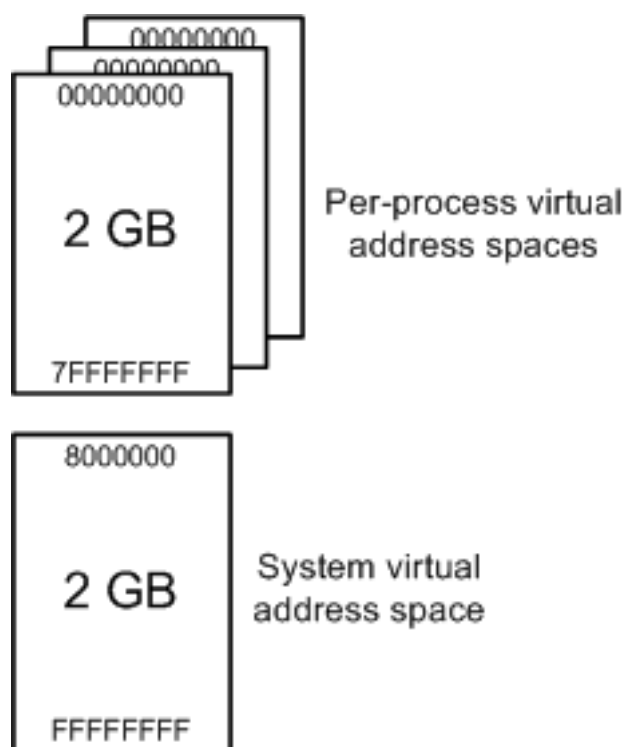
تصویر زیر به خوبی ارتباط بین مد کاربری و مد کرنل را نشان می‌دهد:



#### 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.



## نظرات خوانندگان

نویسنده: وحید محمدطاهری  
تاریخ: ۱۴:۵۱ ۱۳۹۴/۰۸/۰۵

سلام و خدا قوت  
آیا این درست است؟

در سیستم‌های 32 بیتی Pagedpool توانایی 128 گیگ فضای آدرس دهی مجازی و در سیستم‌های 64 بیتی توانایی 128 گیگ فضای آدرس دهی مجازی

نویسنده: علی یگانه مقدم  
تاریخ: ۱۷:۴۳ ۱۳۹۴/۰۸/۰۵

ممنون حق با شماست  
میتوانید اندازه این موارد در [اینجا](#) و [اینجا](#) ببینید. در سیستم‌های 32 بیتی 470 مگابایت برای PagedPool و 256 مگابایت برای nonpaged pool و در سیستم‌های 64 بیتی برای هر دو همان 128 گیگابایت است

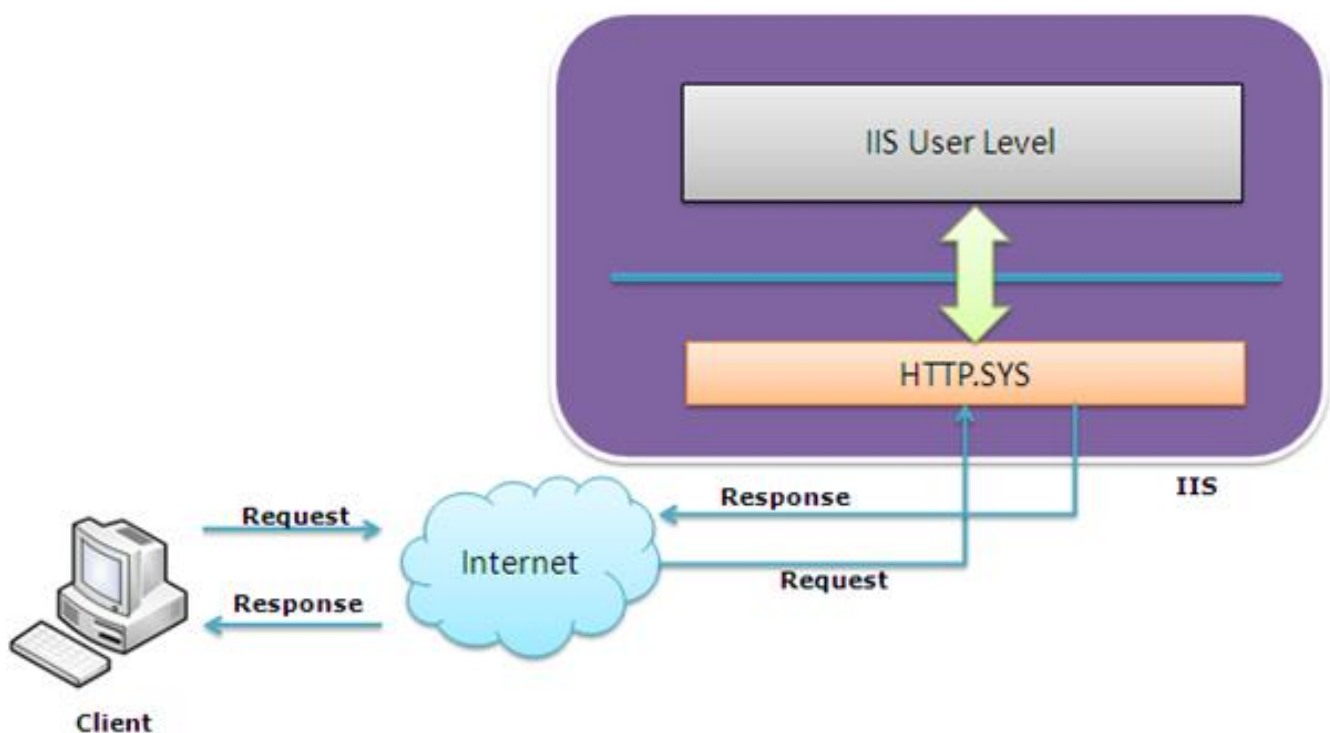
در [مقاله قبل](#) در مورد نحوه ذخیره سازی در حافظه نوشتیم و به user mode و kernel mode اشاراتی کردیم که می‌توانید به آن رجوع کنید.

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**

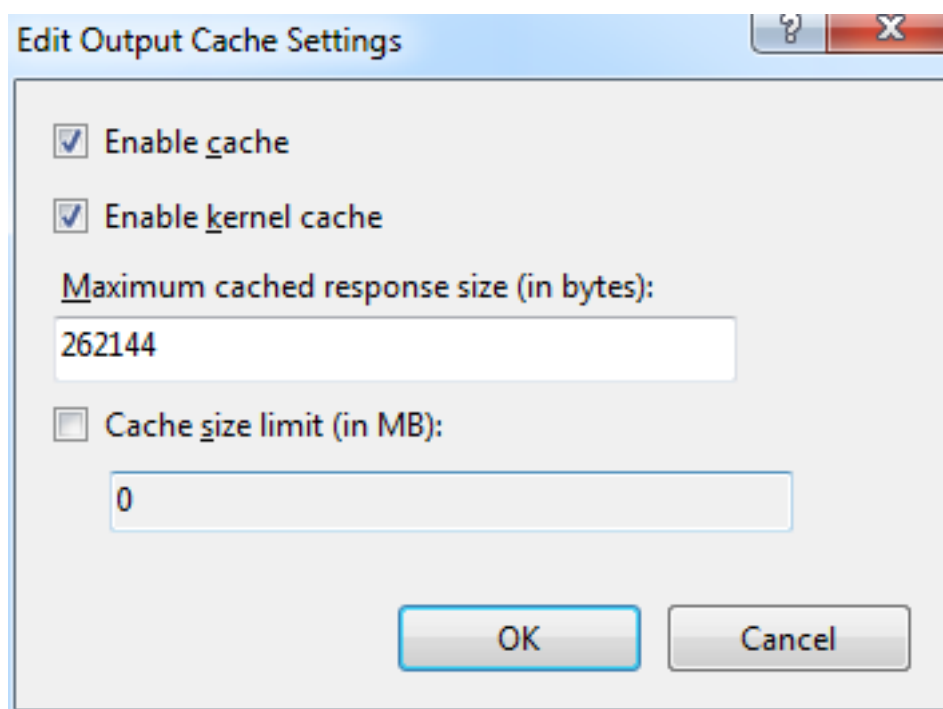
☒ Using file change notifications

☐ 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 به شاخه کلاینت را نوشته تا تست آغاز شود. بعد از انجام تست به شاخه نصب کنترلر WCAT رفته و فایل log را بخوانید و اگر دوباره دستور نمایش کش کرنل را بزنید باید خالی باشد. حالا کش را فعال کنید و دوباره عملیات تست را از سر بگیرید و اگر دستور netsh ارسال 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.

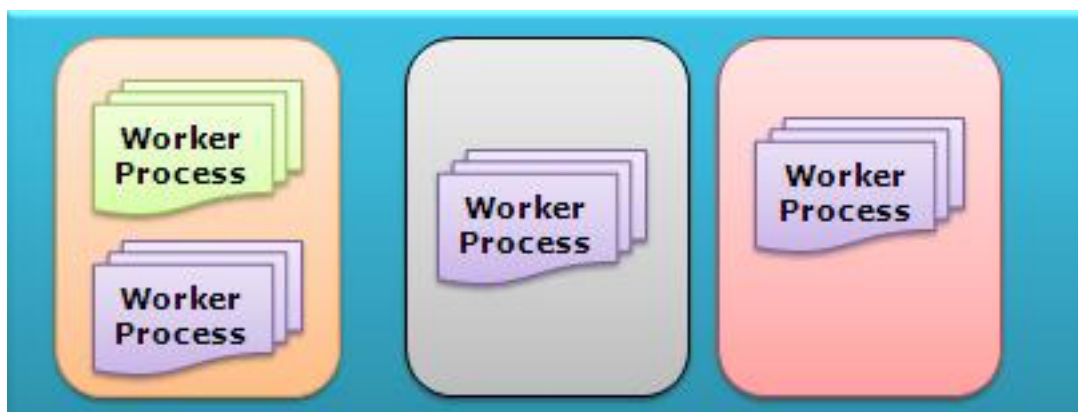
در قسمت قبلی گفتیم که IIS از تعدادی کامپوننت تشکیل شده است و به یکی از آن‌ها به نام Http.sys پرداختیم. در این قسمت قصد داریم به WWW Services بپردازیم. اجازه بدهید قبل از هر چیزی به دو مفهوم اصلی در IIS بپردازیم:

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 سروریس 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 پروتکل 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 پروتکل‌های 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.

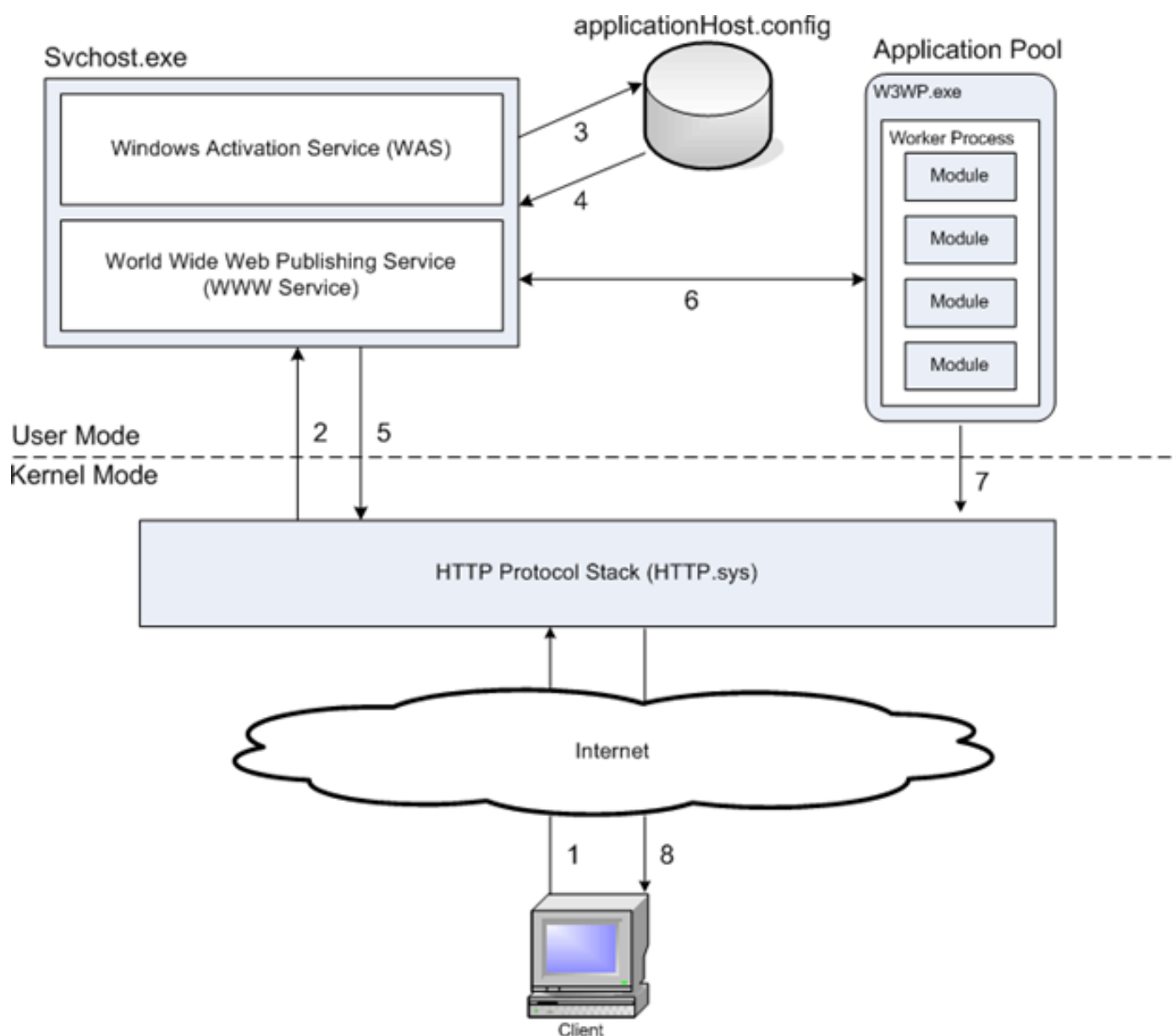
WAS هم اطلاعات پیکربندی شده را از محل ذخیره داده‌ها که applicationHost.config هست، می‌خواند. WWW Service که یک آداپتور برای Http.sys هست، اطلاعات را از WAS دریافت می‌کند. این اطلاعات شامل پیکربندی application pool و سایت می‌باشد.

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.

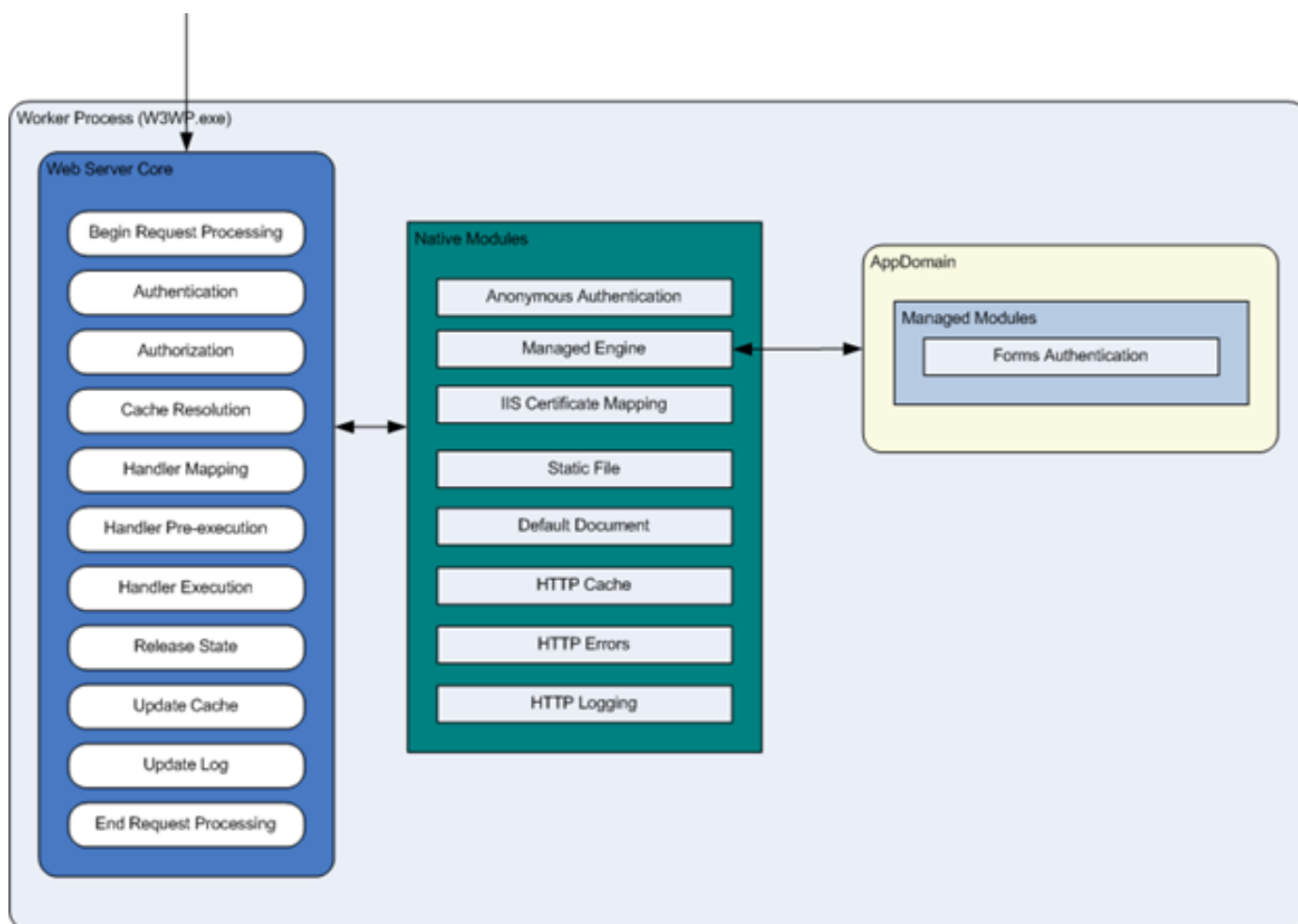
Http.sys نتیجه را دریافت و برای کلاینت می‌فرستد.



#### 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 مازول های مدیریت شده](#)

[دانلود مازول های native](#)

**در مطالب قبلی** در مورد ماژولار 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 ماژول باید دو مرحله را انجام داد:

نصب ماژول

فعال سازی ماژول

نکته ای که در مورد ماژول‌های native وجود دارد این هست که این ماژول‌ها دسترسی بدون محدودیتی به منابع سروری دارند و از این رو حتما باید این نکته را دقت کنید که ماژول 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 ماژول‌ها 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 ماژول به راحتی می‌توانید مدخل ورودی یک ماژول را به صورت دستی حذف نمایید و برای جایگزینی هم بعد از حذف، ماژول خود را معرفی کنید. ولی توجه داشته باشید که این حذف به معنی حذف این ماژول از تمامی اپلیکیشن‌های موجود بر روی IIS هست و سپس اضافه کردن یک ماژول به این بخش. همچنین اگر قصد شما فقط حذف یک ماژول 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 ماژول مختص این اپلیکیشن و یا حذف یک ماژول خاص اقدام نمایید.

#### PreConditions

این ویژگی می‌تواند در خط معرفی ماژول، 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 ماژول 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

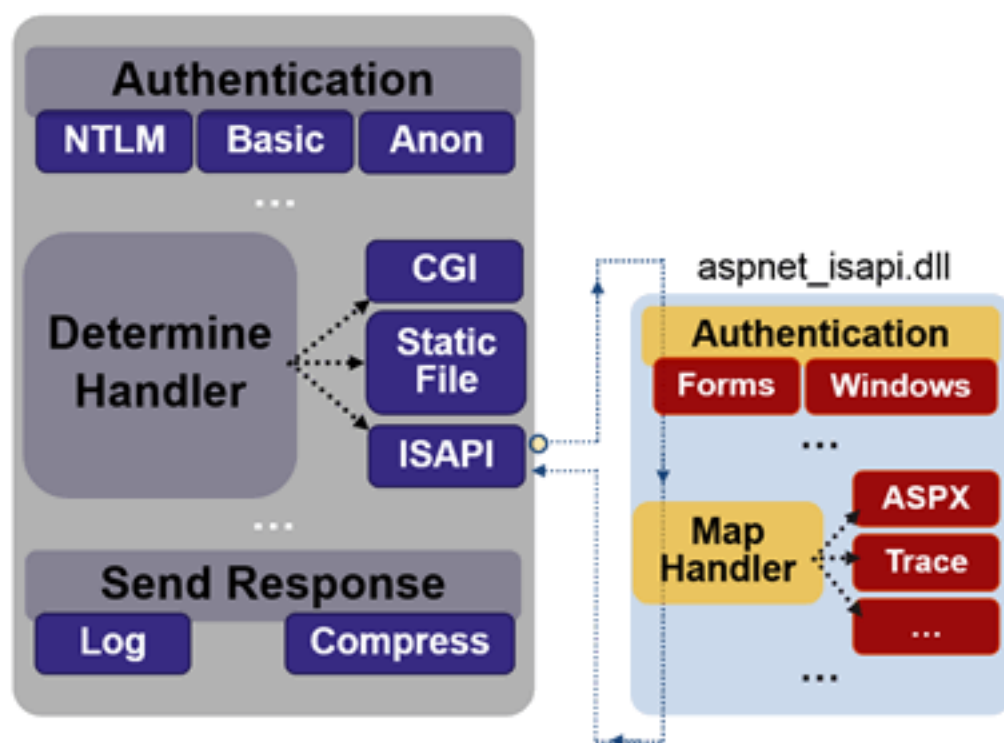
تا به الان گفتیم که چگونه می‌توانیم یک ماژول و یا هندلر مدیریت شده‌ای را به Pipeline اضافه کنیم؛ ولی IIS7 به بالا نیاز دارد که تا پروسه‌های کارگر را به روشی خاص به این منظور اجرا کند و فریم ورک دات نت را برای اجرای آن‌ها بارگزاری کند. همچنین به اجرای ماژولی به اسم webengine.dll برای مدیریت ماژول‌های 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.

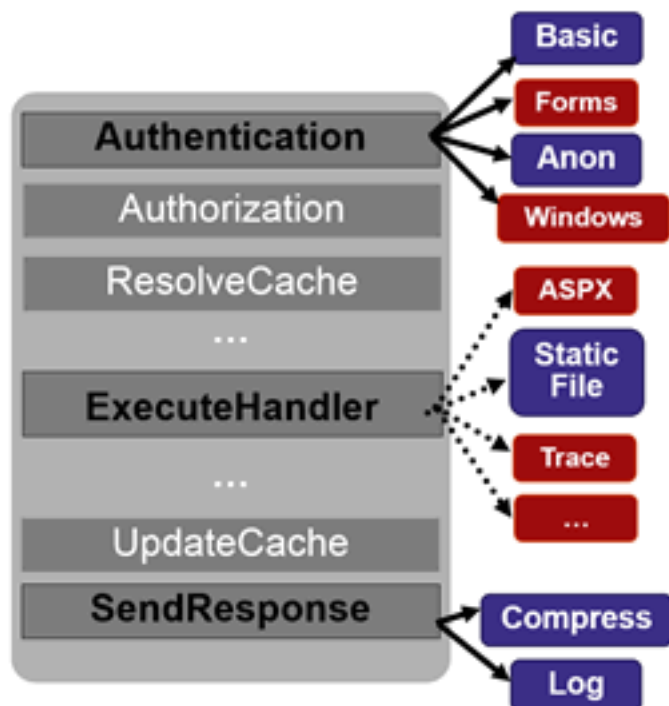
شکل زیر نمونه ای از حالت کلاسیک را نشان می‌دهد که در آن دو بار عمل auth دارد انجام می‌گیرد.

## IIS6 ASP.NET Integration



شکل زیر هم نمونه ای 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

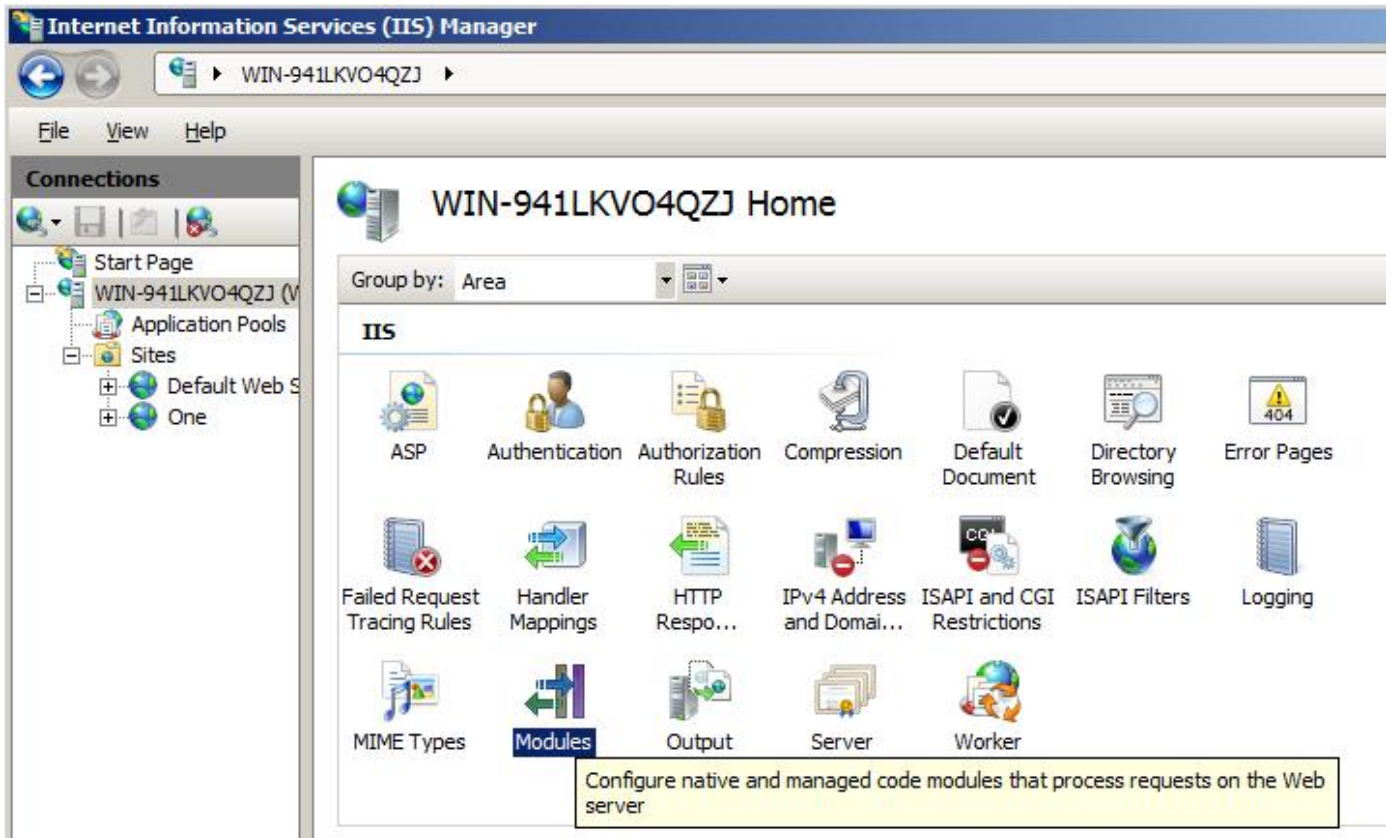
handlers

modules

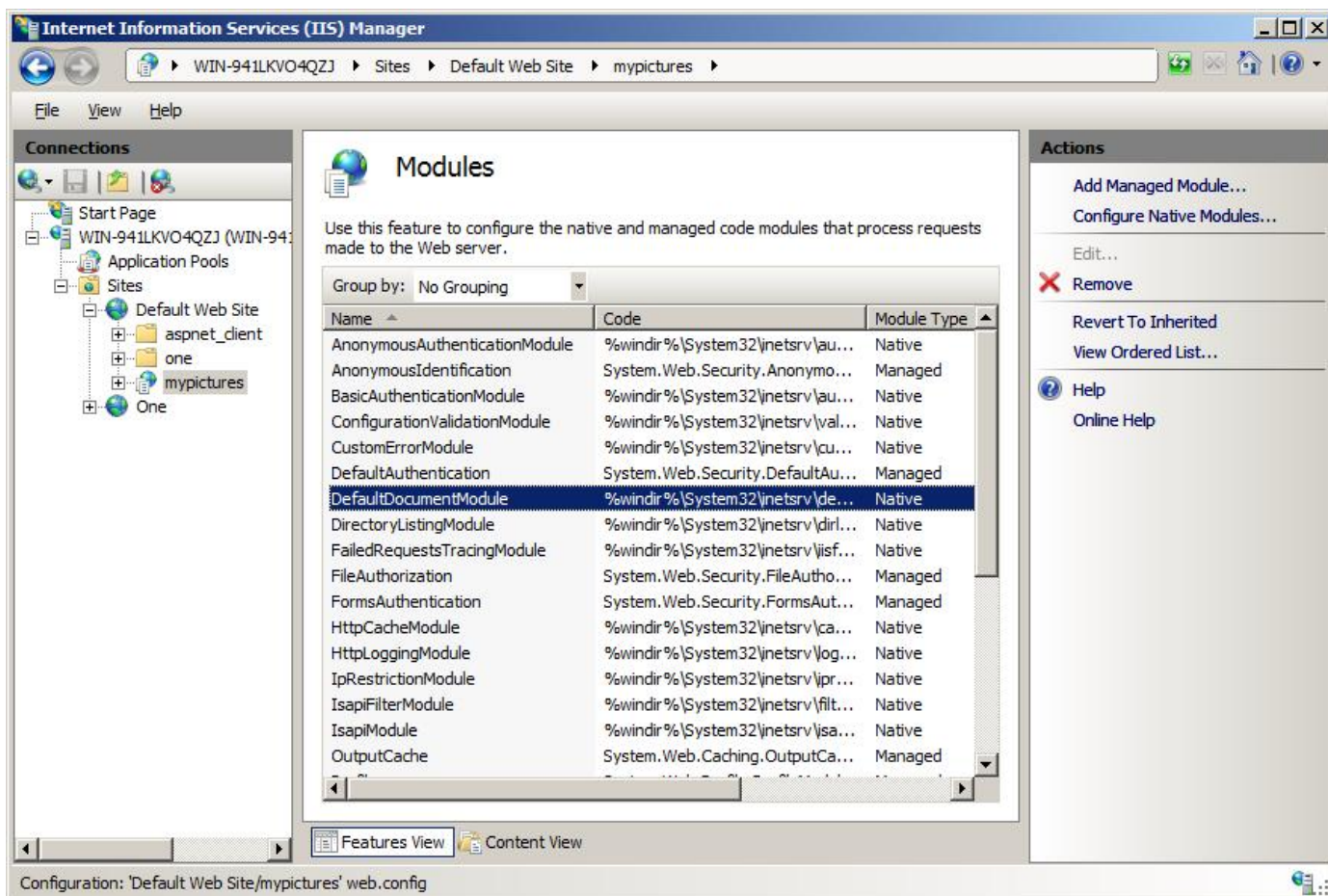
در مورد بقیه تگ‌ها در آینده بیشتر بحث می‌کنیم. بهتر هست مطلب را با توضیح precondition جهت ممانعت از طولانی و طومار شدن در اینجا ببندیم و در قسمت آینده دیگر روش‌های نصب ماژول‌مان را دنبال کنیم.

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 <a href="#">in-memory compression</a> 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

<b>HttpRedirectionModule</b>	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 :
<b>ServerSideIncludeModule</b>	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 :

<b>StaticFileModule</b>	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 :
<b>AnonymousAuthenticationModule</b>	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 <a href="#">RFC 2617</a> .
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 <a href="#">RFC 2617</a> .
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.

<b>IsapiModule</b>	نام ماژول :
هندلرهای معرفی شده در بخش IsapiModule و تگ handlers دیگر اجرا نمی‌شوند	اثرات حذف آن :
<b>IsapiFilterModule</b>	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 :

<b>IpRestrictionModule</b>	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 :
<b>RequestFilteringModule</b>	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



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

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

#### :MANAGED MODULES

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

<b>OutputCache</b>	نام ماژول :
<b>Session</b>	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 :

<b>WindowsAuthentication</b>	نام ماژول :
<a href="#">اینجا</a>	توضیح :
system.web/authentication	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
این حالت قابل اجرا نخواهد بود	اثرات حذف آن :
<b>FormsAuthentication</b>	نام ماژول :
<a href="#">اینجا</a>	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 :	<a href="#">اینجا</a>
تگ قابل پیکربندی :	
وابستگی :	نیاز به ManagedEngine .
اثرات حذف آن :	این قابلیت در دسترس نمی‌باشد

نام ماژول :	UrlAuthorization
توضیح :	<a href="#">اینجا</a>
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

<b>UrlAuthorization</b>	نام ماژول :
<b>AnonymousIdentification</b>	نام ماژول :
<a href="#">اینجا</a>	توضیح :
	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
The anonymous identification feature used by the ASP.NET Profile will not work.	اثرات حذف آن :

<b>Profile</b>	نام ماژول :
<a href="#">اینجا</a>	توضیح :
	تگ قابل پیکربندی :
ManagedEngine module must be installed.	وابستگی :
ASP.Net Profile از کار خواهد افتاد	اثرات حذف آن :
<b>UrlMappingsModule</b>	نام ماژول :
تبدیل یک Url واقعی به یک Url کاربرپسند	توضیح :
	تگ قابل پیکربندی :
نیاز به ManagedEngine .	وابستگی :
نگاشت Url ها صورت نمی گیرد	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 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
```

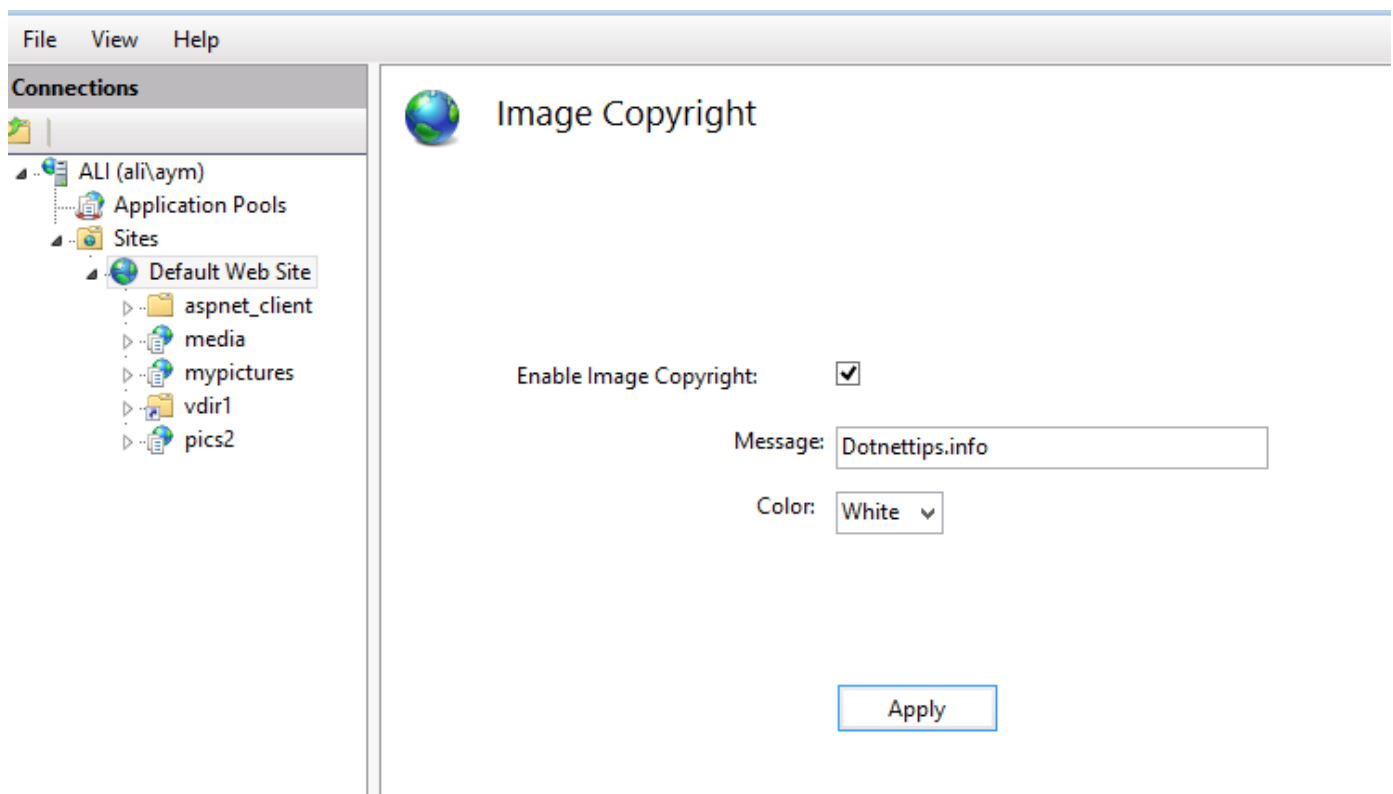
کارمان تمام شده، چند تصویر داخل دایرکتوری قرار داده و درخواست تصاویر موجود را بدهید تا تگ را ببینید:





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

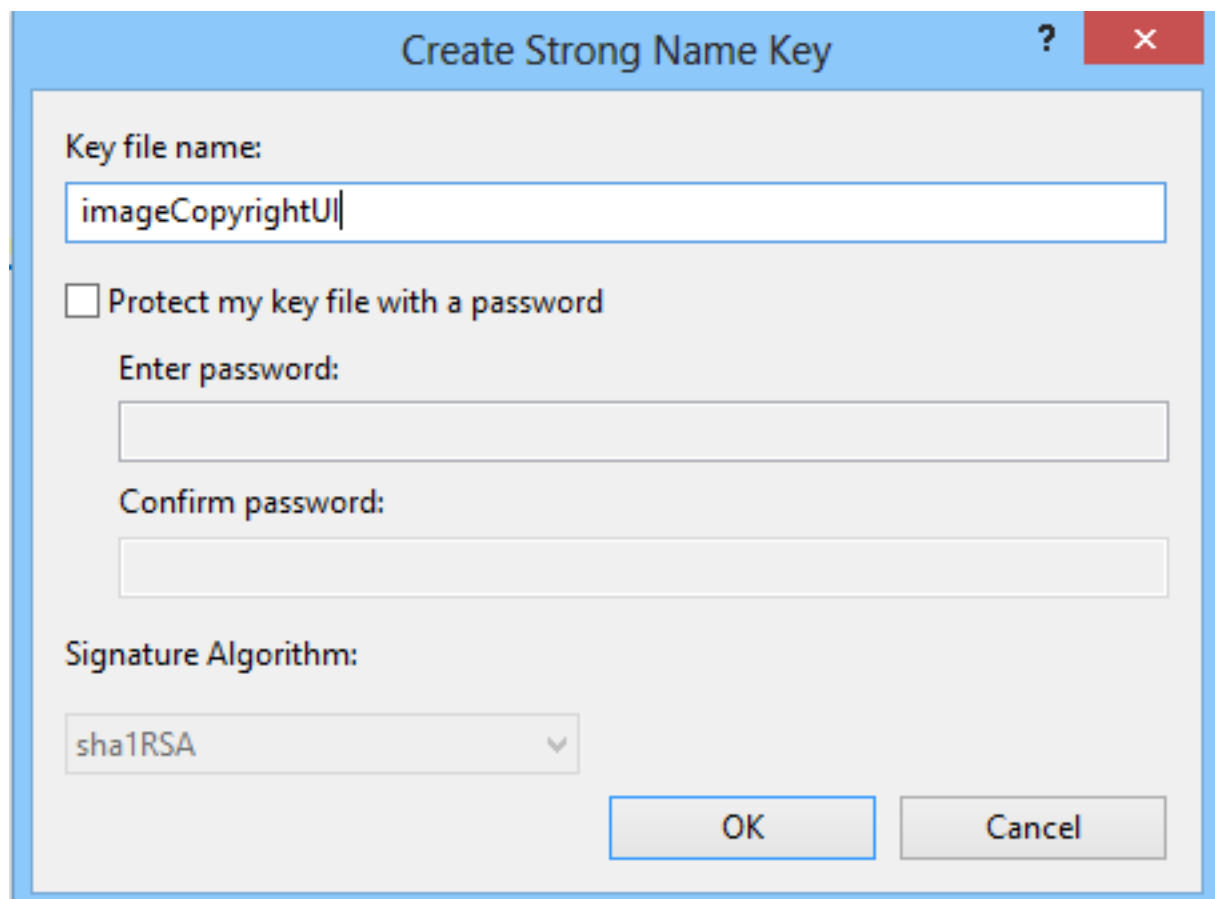
در قسمت قبلی ما یک هندلر ایجاد کردیم و درخواست‌هایی را که برای فایل jpg و به صورت GET ارسال میشد، هندل می‌کردیم و تگی را در گوشه‌ی تصویر درج و آن را در خروجی نمایش میدادیم. در این مقاله قصد داریم که کمی هندلر مورد نظر را توسعه دهیم و برای آن یک UI یا یک رابط کاربری ایجاد نماییم. برای توسعه دادن ماژولها و هندلرها ما یک dll نوشته و باید آن را در GAC که مخفف عبارت [Global Assembly Cache](http://www.dotnettips.info) ریجستر کنیم.



جهت اینکار یک پروژه از نوع class library ایجاد کنید. فایل class1.cs را که به طور پیش فرض ایجاد می‌شود، حذف کنید و رفرنس‌های Microsoft.Web.Administration.dll و Microsoft.Web.Management.dll را از مسیر زیر اضافه کنید:

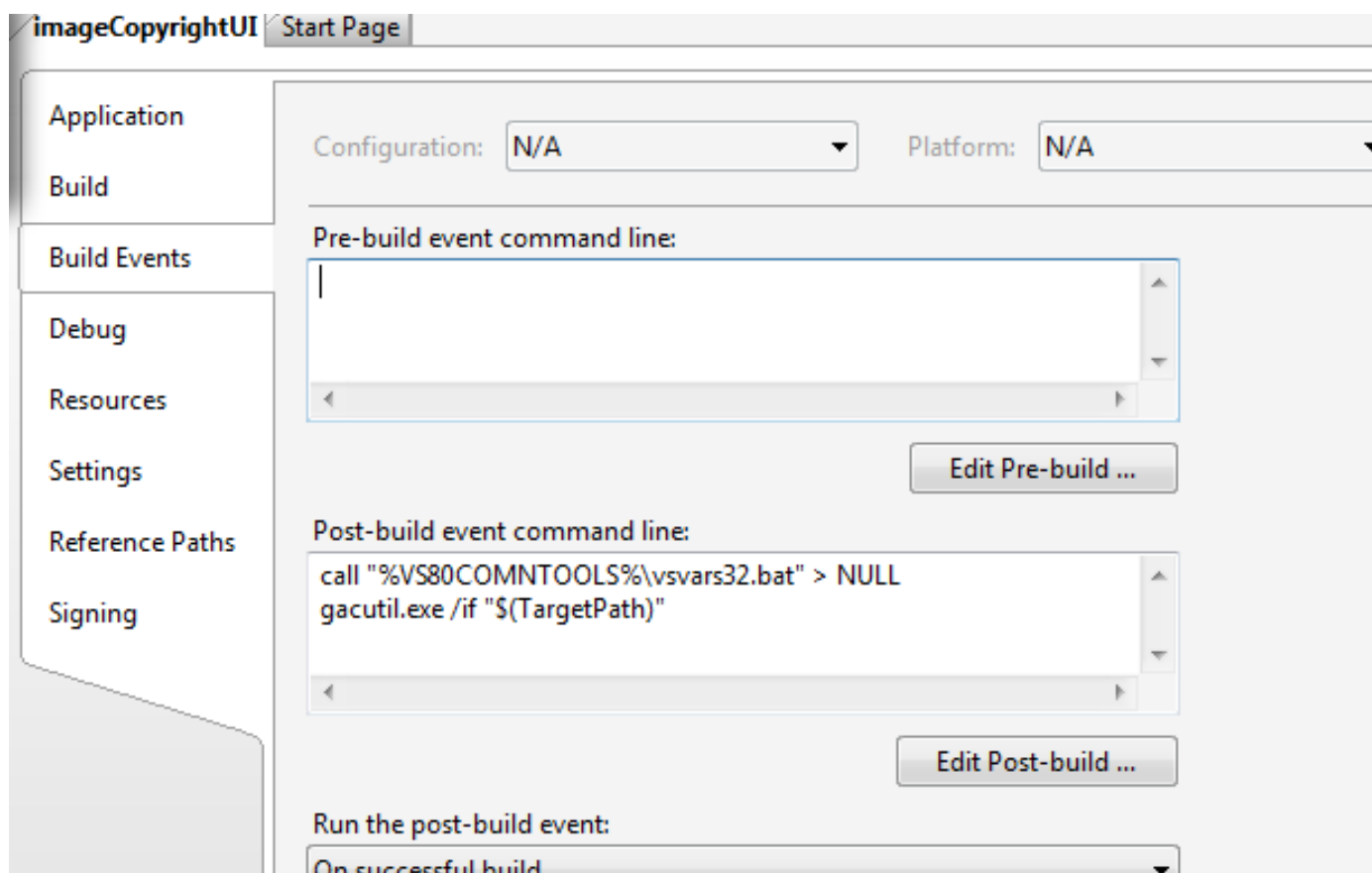
\Windows\system32\inetsrv

اولین رفرنس شامل کلاس‌هایی است که جهت ساخت ماژول‌ها برای کنسول IIS مورد نیاز است و دومی هم برای خواندن پیکربندی‌های نوشته شده مورد استفاده قرار می‌گیرد. برای طراحی UI بر پایه winform باید رفرنس‌های System.Windows.Forms.dll و System.Web.dll را از سری اسمبلی‌های دات نت نیز اضافه کنیم و در مرحله‌ی بعدی جهت ایجاد امضاء یا strong name ( [u](#) و [u](#) ) به خاطر ثبت در GAC پروژه را انتخاب و وارد Properties پروژه شوید. در تب signing گزینه sign the assembly را تیک زده و در لیست باز شده گزینه new را انتخاب نمایید و نام imageCopyrightUI را به آن نسبت داده و گزینه تعیین کلمه عبور را غیرفعال کنید و تایید و تمام. الان باید یک فایل snk مخفف strong name key ایجاد شده باشد تا بعداً با استفاده از این کلید dll ایجاد شده را در GAC ریجستر کنیم.



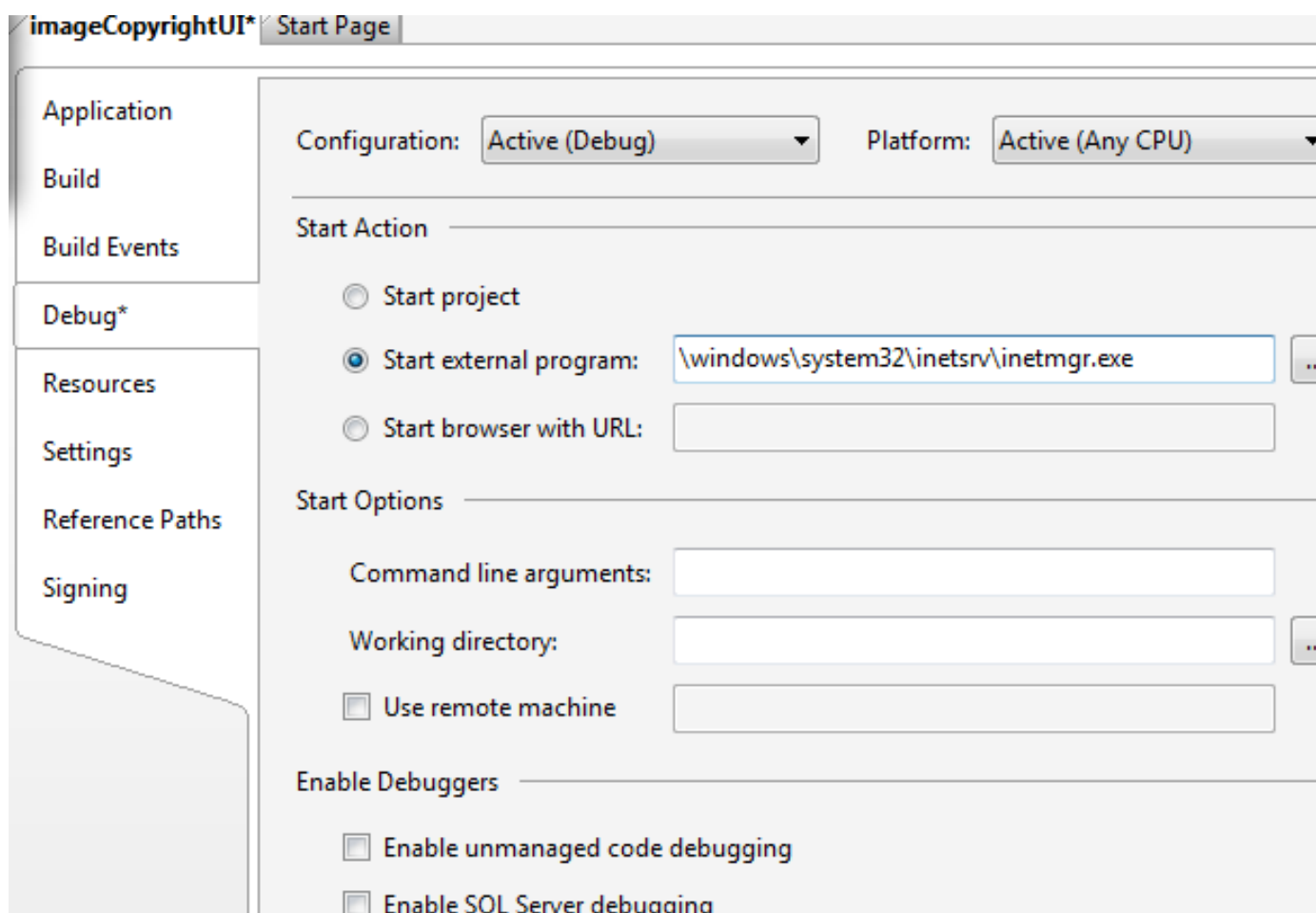
در مرحله بعدی در تب [Build Events](#) کد زیر را در بخش Post-build event command line اضافه کنید. این کد باعث می‌شود بعد از هر بار کامپایل پروژه، به طور خودکار در GAC ثبت شود:

```
call "%VS80COMNTOOLS%\vsvars32.bat" > NULL
gacutil.exe /if "$(TargetPath)"
```



نکته: در صورتی که از VS2005 استفاده می‌کنید در تب Debug در قسمت Start External Program مسیر زیر را قرار بدهید.  
اینکار برای تست و دیباگینگ پروژه به شما کمک خواهد کرد. این تنظیم شامل نسخه‌های اکسپرس نمی‌شود.

\windows\system32\inetsrv\inetmgr.exe



بعد از پایان اینکار پروژه را Rebuild کنید. با اینکار dll در GAC ثبت می‌شود. استفاده از سوییچ‌های if به طور همزمان در درستیور gacutil به معنی این هست که اگر اولین بار است نصب می‌شود، پس با سوییچ i نصب کن. ولی اگر قبلاً نصب شده است نسخه جدید را به هر صورتی هست جایگزین قبلی کن یا همان reinstall کن.

### ساخت یک Module Provider

رابط‌های کاربری IIS همانند هسته و کل سیستمش، ماژولار و قابل خصوصی سازی است. رابط کاربری، مجموعه‌ای از ماژول‌هایی است که می‌توان آن‌ها را حذف یا جایگزین کرد. تگ ورودی یا معرفی برای هر UI یک module provider است. خیلی خودمانی، تگ ماژول پروایدر به معرفی یک UI در IIS می‌پردازد. لیستی از module provider ها را می‌توان در فایل زیر در تگ بخش <modules> پیدا کرد.

```
%windir%\system32\inetrv\Administration.config
```

در اولین گام یک کلاس را به اسم imageCopyrightUIModuleProvider.cs ایجاد کرده و سپس آن‌را به کد زیر، تغییر می‌دهیم. کد زیر با استفاده از ModuleDefinition یک نام به تگ Module Provider داده و کلاس imageCopyrightUI را که بعداً تعریف می‌کنیم، به عنوان مدخل entry رابط کاربری معرفی کرده:

```
using System;
using System.Security;
using Microsoft.Web.Management.Server;

namespace IIS7Demos
{
    class imageCopyrightUIProvider : ModuleProvider
    {
        public override Type ServiceType
```

```

    {
        get { return null; }
    }

    public override ModuleDefinition GetModuleDefinition(IManagementContext context)
    {
        return new ModuleDefinition(Name, typeof(imageCopyrightUI).AssemblyQualifiedName);
    }

    public override bool SupportsScope(ManagementScope scope)
    {
        return true;
    }
}

```

با ارث بری از کلاس module provider، سه متد بازنویسی می‌شوند که یکی از آن‌ها SupportsScope هست که میدان عمل پروایدر را مشخص می‌کند، مانند اینکه این پروایدر در چه میدانی باید کار کند که می‌تواند سه گزینه‌ی server,site,application باشد. در کد زیر مثلاً میدان عمل application انتخاب شده است ولی در کد بالا با برگشت مستقیم true، همه‌ی میدان را جهت پشتیبانی از این پروایدر اعلام کردیم.

```

public override bool SupportsScope(ManagementScope scope)
{
    return (scope == ManagementScope.Application) ;
}

```

حالا که پروایدر (معرف رابط کاربری به IIS) تامین شده، نیاز است قلب کار یعنی ماژول معرفی گردد. اصلی‌ترین متدی که باید از اینترفیس ماژول پیاده سازی شود متد initialize است. این متد جایی است که تمام عملیات در آن رخ می‌دهد. در کلاس زیر imageCopyrightUI ما به معرفی مدخل entry رابط کاربری می‌پردازیم. در سازنده‌های این متد، پارامترهای نام، صفحه رابط کاربری و توضیحی در مورد آن است. تصویر کوچک و بزرگ جهت آیکن سازی (در صورت عدم تعریف آیکن، چرخ دنده نمایش داده می‌شود) و توصیف‌های بلندتر را نیز شامل می‌شود.

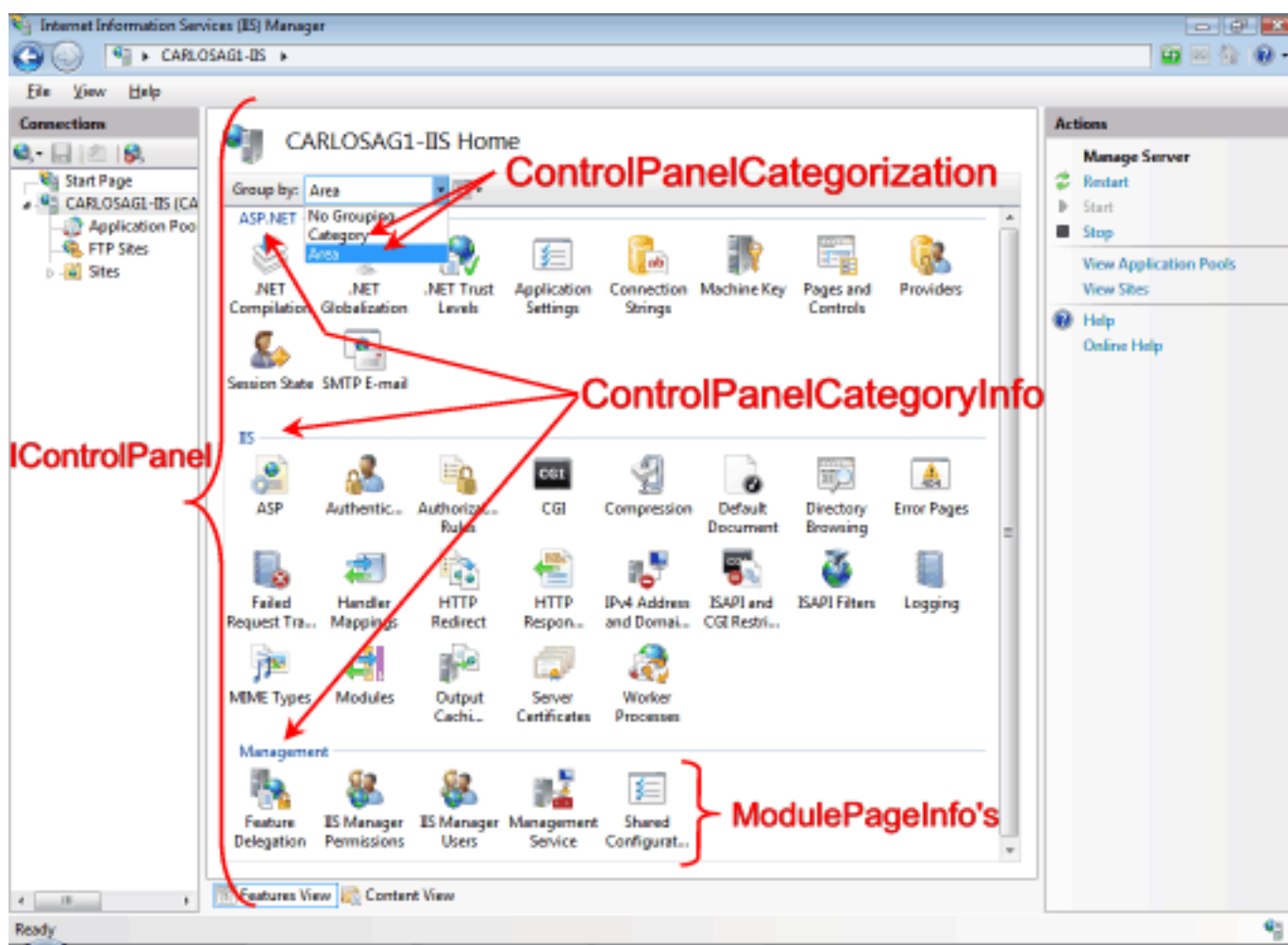
```

internal class imageCopyrightUI : Module
{
    protected override void Initialize(IServiceProvider serviceProvider, ModuleInfo moduleInfo)
    {
        base.Initialize(serviceProvider, moduleInfo);
        IControlPanel controlPanel = (IControlPanel)GetService(typeof(IControlPanel));
        ModulePageInfo modulePageInfo = new ModulePageInfo(this, typeof(imageCopyrightUIPage),
"Image Copyright", "Image Copyright",Resource1.Visual_Studio_2012,Resource1.Visual_Studio_2012);
        controlPanel.RegisterPage(modulePageInfo);
    }
}

```

شیء ControlPanel مکانی است که قرار است آیکن ماژول نمایش داده شود. شکل زیر به خوبی نام همه قسمت‌ها را بر اساس نام کلاس و اینترفیس آن‌ها دسته بندی کرده است:





پس با تعریف این کلاس جدید ما روی صفحه‌ی کنترل پنل IIS، یک آیکن ساخته و صفحه‌ی رابط کاربری را به نام `imageCopyrightUIPage`، در آن رجیستر می‌کنیم. این کلاس را پایینتر شرح داده‌ایم. ولی قبل از آن اجازه بدهید تا انواع کلاس‌هایی را که برای ساخت صفحه کاربرد دارند، بررسی نماییم. در این مثال ما با استفاده از پایه‌ای‌ترین کلاس، ساده‌ترین نوع صفحه ممکن را خواهیم ساخت. 4 کلاس برای ساخت یک صفحه وجود دارند که بسته به سناریوی کاری، شما یکی را انتخاب می‌کنید.

شامل اساسی‌ترین متدها و سورس‌ها شده و هیچگونه رابط کاری ویژه‌ای را در اختیار شما قرار نمی‌دهد. تنها یک صفحه‌ی خام به شما می‌دهد که می‌توانید از آن استفاده کرده یا حتی با ارث بری از آن، کلاس‌های جدیدتری را برای ساخت صفحات مختلف و ویژه‌تر بسازید. در حال حاضر که هیچ کدام از ویژگی‌های IIS فعلی از این کلاس برای ساخت رابط کاربری استفاده نکرده‌اند.	<b>ModulePage</b>
یک صفحه شبیه به دیالوگ را ایجاد می‌کند و شامل دکمه‌های <code>Cancel</code> و <code>Apply</code> میشود به همراه یک سری متدهای اضافی‌تر که اجازه‌ی <code>override</code> کردن آنها را دارید. همچنین یک سری از کارهایی چون <code>refresh</code> و از این دست عملیات خودکار را نیز انجام میدهد. از نمونه رابط‌هایی که از این صفحات استفاده می‌کنند میتوان <code>machine key</code> و <code>management service</code> را اسم برد.	<b>ModuleDialogPage</b>

<p>شامل اساسی‌ترین متدها و سورس‌ها شده و هیچگونه رابط کاری ویژه‌ای را در اختیار شما قرار نمی‌دهد. تنها یک صفحه‌ی خام به شما می‌دهد که می‌توانید از آن استفاده کرده یا حتی با ارث بری از آن، کلاس‌های جدیدتری را برای ساخت صفحات مختلف و ویژه‌تر بسازید. در حال حاضر که هیچ کدام از ویژگی‌های IIS فعلی از این کلاس برای ساخت رابط کاربری استفاده نکرده‌اند.</p>	<b>ModulePage</b>
<p>این صفحه یک رابط کاربری را شبیه پنجره property که در ویژوال استادیو وجود دارد، در دسترس شما قرار می‌دهد. تمام عناصر آن در یک حالت گرید grid لیست می‌شوند. از نمونه‌های موجود میتوان به CGI,ASP.Net Compilation اشاره کرد.</p>	<b>ModulePropertiesPage</b>
<p>این کلاس برای مواقعی کاربرد دارد که شما قرار است لیستی از آیتم‌ها را نشان دهید. در این صفحه شما یک ListView دارید که میتوانید عملیات جست و جو، گروه بندی و نحوه‌ی نمایش لیست را روی آن اعمال کنید.</p>	<b>ModuleListPage</b>

در این مثال ما از اولین کلاس نامبرده که پایه‌ی همه کلاس‌هاست استفاده می‌کنیم. کد زیر را در کلاسی به اسم imageCopyrightUIPage می‌نویسیم:

```
public sealed class imageCopyrightUIPage : ModulePage
{
    public string message;
    public bool featureenabled;
    public string color;

    ComboBox _colCombo = new ComboBox();
    TextBox _msgTB = new TextBox();
    CheckBox _enabledCB = new CheckBox();

    public imageCopyrightUIPage()
    {
        this.Initialize();
    }

    void Initialize()
    {
        Label crlabel = new Label();
        crlabel.Left = 50;
        crlabel.Top = 100;
        crlabel.AutoSize = true;
        crlabel.Text = "Enable Image Copyright:";
        _enabledCB.Text = "";
        _enabledCB.Left = 200;
        _enabledCB.Top = 100;
        _enabledCB.AutoSize = true;

        Label msglabel = new Label();
        msglabel.Left = 150;
        msglabel.Top = 130;
        msglabel.AutoSize = true;
        msglabel.Text = "Message:";
        _msgTB.Left = 200;
        _msgTB.Top = 130;
        _msgTB.Width = 200;
        _msgTB.Height = 50;

        Label collabel = new Label();
        collabel.Left = 160;
        collabel.Top = 160;
        collabel.AutoSize = true;
        collabel.Text = "Color:";
        _colCombo.Left = 200;
```



```

        _colCombo.Top = 160;
        _colCombo.Width = 50;
        _colCombo.Height = 90;
        _colCombo.Items.Add((object)"Yellow");
        _colCombo.Items.Add((object)"Blue");
        _colCombo.Items.Add((object)"Red");
        _colCombo.Items.Add((object)"White");

        Button apply = new Button();
        apply.Text = "Apply";
        apply.Click += new EventHandler(this.applyClick);
        apply.Left = 200;
        apply.AutoSize = true;
        apply.Top = 250;

        Controls.Add(crlabel);
        Controls.Add(_enabledCB);
        Controls.Add(collabel);
        Controls.Add(_colCombo);
        Controls.Add(msglabel);
        Controls.Add(_msgTB);
        Controls.Add(apply);
    }

    public void ReadConfig()
    {
        try
        {
            ServerManager mgr;
            ConfigurationSection section;
            mgr = new ServerManager();
            Configuration config =
                mgr.GetWebConfiguration(
                    Connection.ConfigurationPath.SiteName,
                    Connection.ConfigurationPath.ApplicationPath +
                    Connection.ConfigurationPath.FolderPath);

            section = config.GetSection("system.webServer/imageCopyright");
            color = (string)section.GetAttribute("color").Value;
            message = (string)section.GetAttribute("message").Value;
            featureenabled = (bool)section.GetAttribute("enabled").Value;

        }

        catch
        { }
    }

    void UpdateUI()
    {
        _enabledCB.Checked = featureenabled;
        int n = _colCombo.FindString(color, 0);
        _colCombo.SelectedIndex = n;
        _msgTB.Text = message;
    }

    protected override void OnActivated(bool initialActivation)
    {
        base.OnActivated(initialActivation);
        if (initialActivation)
        {
            ReadConfig();
            UpdateUI();
        }
    }

    private void applyClick(Object sender, EventArgs e)
    {
        try
        {
            UpdateVariables();
            ServerManager mgr;
            ConfigurationSection section;
            mgr = new ServerManager();
            Configuration config =
                mgr.GetWebConfiguration
            (

```

```

        Connection.ConfigurationPath.SiteName,
        Connection.ConfigurationPath.ApplicationPath +
        Connection.ConfigurationPath.FolderPath
    );

    section = config.GetSection("system.webServer/imageCopyright");
    section.GetAttribute("color").Value = (object)color;
    section.GetAttribute("message").Value = (object)message;
    section.GetAttribute("enabled").Value = (object)featureenabled;

    mgr.CommitChanges();
}

catch
{ }
}

public void UpdateVariables()
{
    featureenabled = _enabledCB.Checked;
    color = _colCombo.Text;
    message = _msgTB.Text;
}
}

```

اولین چیزی که در کلاس بالا صدا زده می‌شود، سازنده‌ی کلاس هست که ما در آن یک تابع تعریف کردیم به اسم **initialize** که به آماده سازی اینترفیس یا رابط کاربری می‌پردازد و کنترل‌ها را روی صفحه می‌چیند. این سه کنترل، یکی *ComboBox* برای تعیین رنگ، یک *Checkbox* برای فعال بودن ماژول و دیگری هم یک *textbox* جهت نوشتن متن است. مابقی هم که سه *label* برای نامگذاری اشیاست. بعد از اینکه کنترل‌ها روی صفحه درج شدند، لازم است که تنظیمات پیش فرض یا قبلی روی کنترل‌ها نمایش یابند که اینکار را به وسیله تابع **readConfig** انجام می‌دهیم و تنظیمات خوانده شده را در متغیرهای عمومی قرار داده و با استفاده از تابع **UpdateUI** این اطلاعات را روی کنترل‌ها ست می‌کنیم و به این ترتیب *UI* به روز می‌شود. این دو تابع را به ترتیب پشت سر هم در یک متد به اسم **OnActivated** که **override** کرده‌ایم صدا می‌زنیم. در واقع این متد یک جورایی همانند رویداد *Load* می‌باشد؛ اگر **true** برگرداند اولین فعال سازی رابط کاربری بعد از باز شدن IIS است و در غیر این صورت **false** بر میگرداند.

در صورتی که کاربر مقادیر را تغییر دهد و روی گزینه *apply* کلیک کند تابع *applyClick* اجرا شده و ابتدا به تابع *UpdateVariables* ارجاع داده می‌شود که در آن مقادیر خوانده شده و در متغیرهای *Global* قرار می‌گیرند و سپس با استفاده از دو شیء از نوع *serverManger* و *ConfigSection* جایگذاری یا ذخیره می‌شوند. استفاده از دو کلاس *Servermanager* و *Configsection* در دو قسمت خواندن و نوشتن مقادیر به کار رفته‌اند. کلاس *servermanager* به ما اجازه دسترسی به تنظیمات IIS و قابلیت‌های آن را می‌دهد. در تابع *ReadConfig* مسیر وب سایتی را که در لیست IIS انتخاب شده است، دریافت کرده و به وب کانفیگ آن وب سایت رجوع نموده و تگ *imageCopyright* آن را که در تگ *system.webserver* قرار گرفته است، می‌خواند (در صورتی که این تگ در آن وب کانفیگ موجود نباشد، خواندن و سپس ذخیره مجدد آن روی تگ داخل فایل *applicationHost.config* اتفاق می‌افتد که نیجتا برای همه‌ی وب سایت‌هایی که این تگ را ندارند یا مقدارهای پیش فرض آن را تغییر نداده‌اند رخ می‌دهد) عملیات نوشتن هم مشابه خواندن است. تنها باید خط زیر را در آخر برای اعمال تغییرات نوشت؛ مثل EF با گزینه *Context.SaveChanges*:

```
mgr.CommitChanges();
```

وقت آن است که رابط کاربری را به IIS اضافه کنیم؛ پروژه را *Rebuild* کنید. بعد از آن با خطوطی که قبلاً در *Post-Build Command* نوشتیم باید *dll* ما در *GAC* رجیستر شود. برای همین آدرس زیر را در *cmd* تایپ کنید:

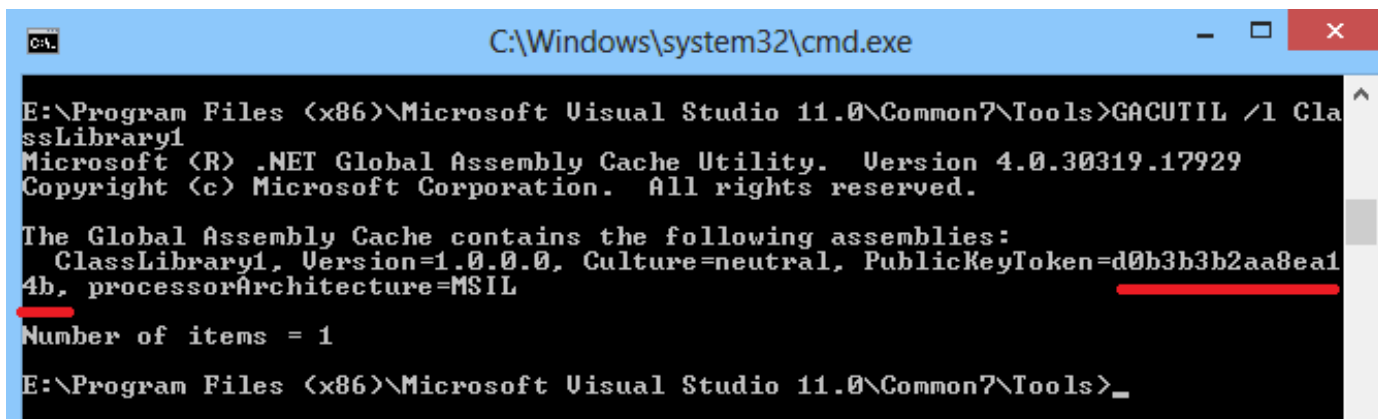
```
%vs110comntools%\vsvars32.bat
```

عبارت اول که **مسیر ویژوال استودیوی** شماست و عدد 110 یعنی نسخه‌ی 11. هر نسخه‌ای را که استفاده می‌کنید، یک صفر جلوش بگذارید و جایگزین عدد بالا کنید. مثلاً نسخه 8 می‌شود 80 و فایل *bj* بالا هم دستورات *visual studio* را برای شما آزاد می‌کند.

سپس دستور زیر را وارد کنید:

```
GACUTIL /1 ClassLibrary1
```

کلمه classLibrary1 نام پروژه‌ای ما بود که در GAC رجیستر شده است. با سوییچ 1 تمامی اطلاعات اسمبلی‌هایی که در GAC رجیستر شده‌اند، نمایش می‌یابند. ولی اگر اسم آن اسمبلی را جلویش بنویسید، فقط اطلاعات آن اسمبلی نمایش میابد. با اجرای خط فوق می‌توانیم کلید عمومی public key خود را بدانیم که در شکل زیر مشخص شده است:



```
C:\Windows\system32\cmd.exe

E:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\Tools>GACUTIL /1 ClassLibrary1
Microsoft (R) .NET Global Assembly Cache Utility. Version 4.0.30319.17929
Copyright (c) Microsoft Corporation. All rights reserved.

The Global Assembly Cache contains the following assemblies:
  ClassLibrary1, Version=1.0.0.0, Culture=neutral, PublicKeyToken=d0b3b3b2aa8ea14b, processorArchitecture=MSIL
Number of items = 1
E:\Program Files (x86)\Microsoft Visual Studio 11.0\Common7\Tools>
```

پس اگر کلید را دریافت کرده‌اید، خط زیر را به فایل administration.config در تگ <ModuleProviders> اضافه کنید:

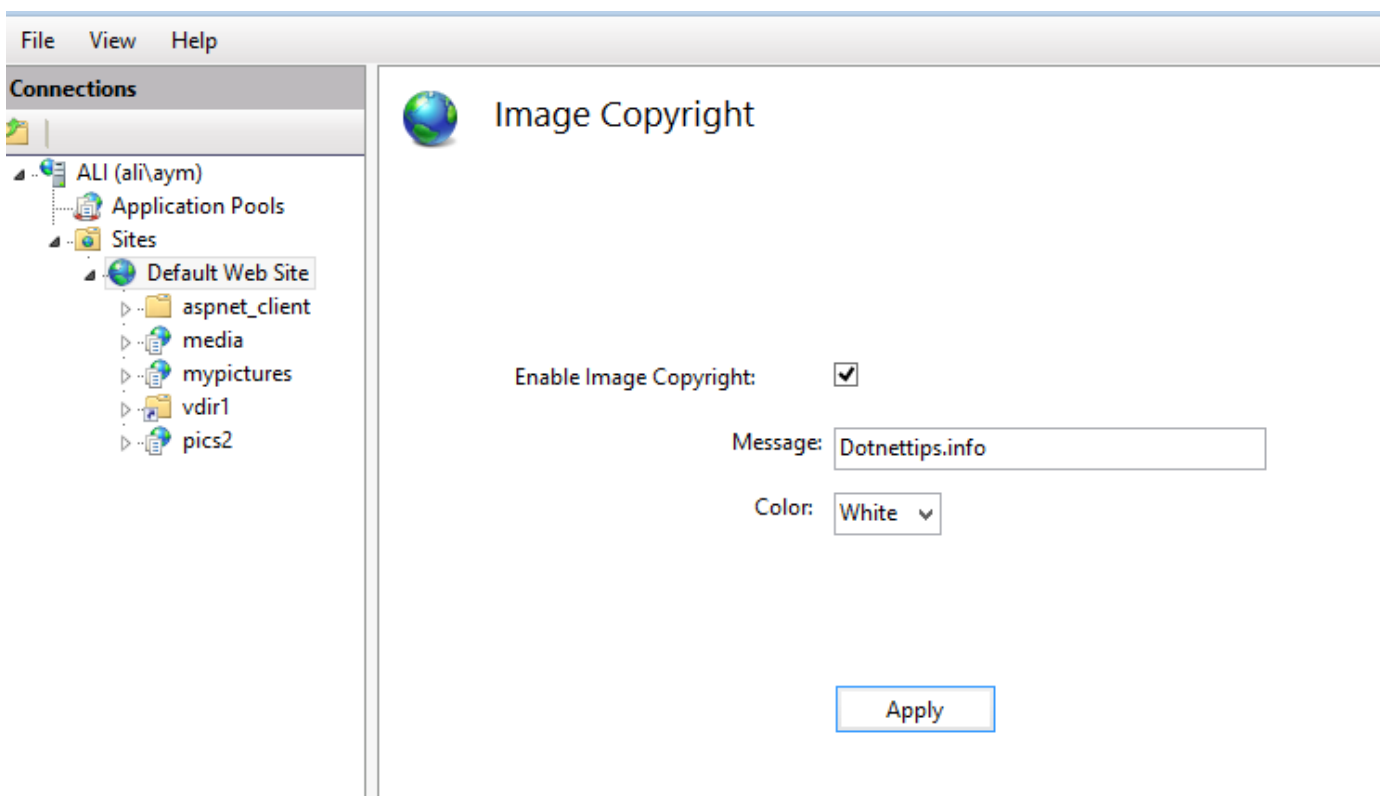
```
<add name="imageCopyrightUI" type="ClassLibrary1.imageCopyrightUIProvider, ClassLibrary1, Version=1.0.0.0, Culture=neutral, PublicKeyToken=d0b3b3b2aa8ea14b"/>
```

عبارت ClassLibrary1.imageCopyrightUIProvider به کلاس imageCopyrightUIProvider اشاره می‌کند که در این کلاس UI معرفی می‌شود. مابقی عبارت هم کاملاً مشخص است و در لینک‌های بالا در مورد Strong name توضیح داده شده‌اند.

فایل administration.config در مسیر زیر قرار دارد:

```
%windir%\system32\inetssrv\config\administration.config
```

حالا تنها کاری که نیاز است، باز کردن IIS است. به بخش وب سایت‌ها رفته و اپلیکیشنی که قبلاً با نام mypictures را ایجاد کرده بودیم، انتخاب کنید. در سمت راست، آخر لیست، بخش others باید ماژول ما دیده شود. بازش کنید و تنظیمات آن را تغییر دهید و حالا یک تصویر را از اپلیکیشن mypictures، روی مرورگر درخواست کنید تا تغییرات را روی تگ مشاهده کنید:



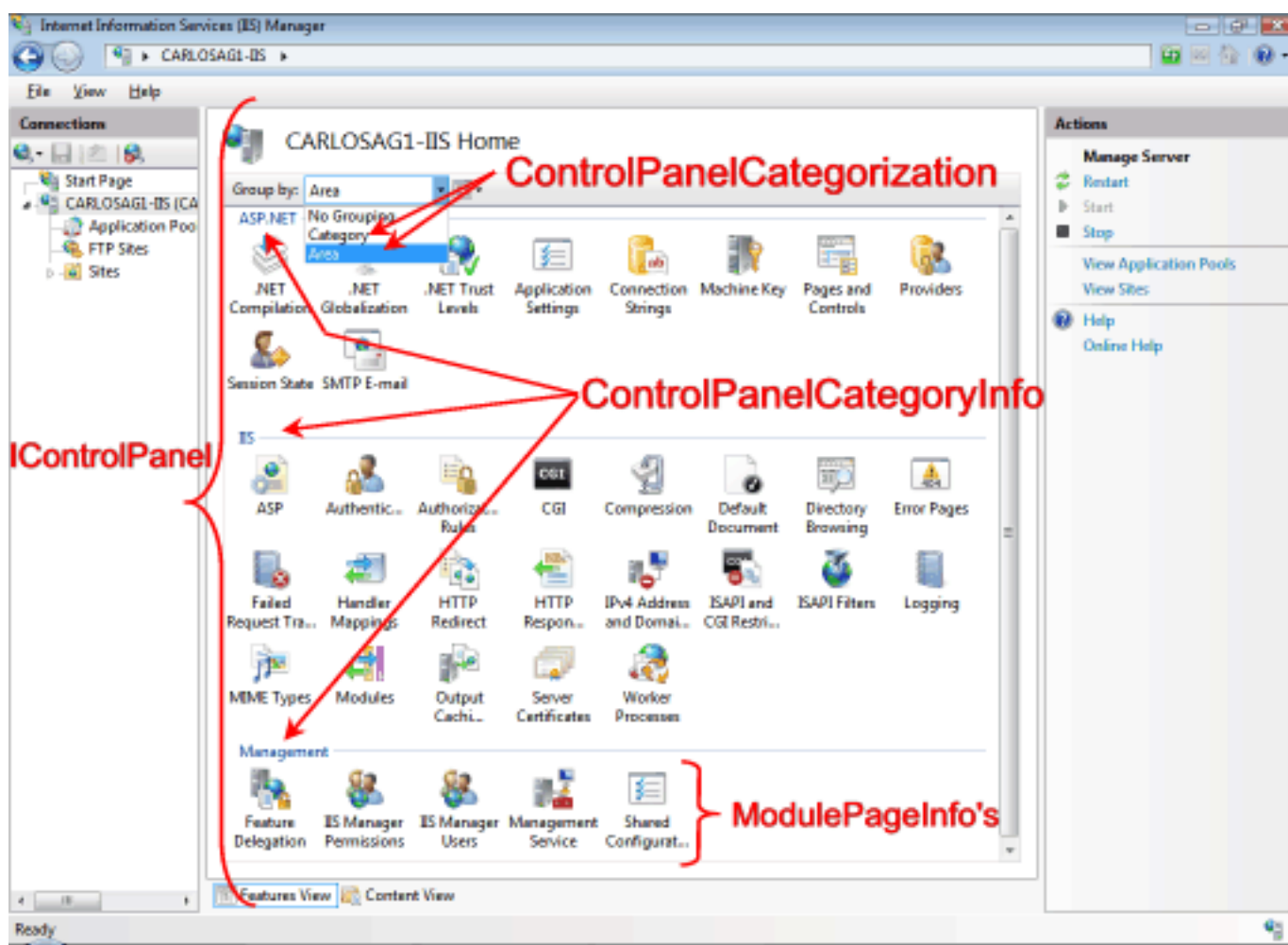
حالا دیگر باید ماژول نویسی برای IIS را فراگرفته باشیم. این ماژول‌ها می‌توانند از یک مورد ساده تا یک کلاس مهم و امنیتی باشند که روی سرور شما برای همه یا بعضی از وب سایت‌ها در حال اجرا هستند و در صورت لزوم و اجازه شما، برنامه نویسی‌ها می‌توانند مثل همه‌ی تگ‌های موجود در وب کانفیگ سایتی را که مینویسند، تگ ماژول شما و تنظیمات آن را با استفاده از attribute یا خصوصیت‌های تعریف شده، بر اساس سلايق و نیازهایشان تغییر دهند و روی سرور شما آپلود کنند. الان شما یک سرور خصوصی سازی شده دارید.

از آنجا که این مقاله طولانی شده است، باقی موارد ویرایشی روی این UI را در مقاله بعدی بررسی خواهیم کرد.

در دو مقاله پیشین ( [۱](#) و [۲](#) ) در مورد اینکه چگونه یک httpmodule یا httpHandler را به عنوان یک ماژول جدید به IIS اضافه کنیم صحبت کردیم و الان قصد داریم که در این بخش این مبحث را ببندیم. آخرین بار توانستیم که یک UI را به IIS نسبت داده و از آن استفاده کنیم و الان قصد داریم آن را تکمیل‌تر کرده و کمی آن را شکل‌تر کنیم. اولین نکته‌ای که توجه ما را جلب میکند این است که ماژول ما یک آیکن پیش فرض (چرخ دنده) دارد که بهتر است به آیکن دلخواه ما تغییر کند. به این منظور در پروژه، یک فایل Resource ایجاد کنید و یک تصویر را به این فایل Resource اضافه کنید و در کلاس imageCopyrightUI کد را به صورت زیر تغییر دهید:

```
internal class imageCopyrightUI : Module
{
    protected override void Initialize(IServiceProvider serviceProvider, ModuleInfo moduleInfo)
    {
        base.Initialize(serviceProvider, moduleInfo);
        IControlPanel controlPanel = (IControlPanel)GetService(typeof(IControlPanel));
        ModulePageInfo modulePageInfo = new ModulePageInfo(this, typeof(imageCopyrightUIPage),
        "Image Copyright", "Image Copyright", Resource1.Visual_Studio_2012, Resource1.Visual_Studio_2012);
        controlPanel.RegisterPage(modulePageInfo);
    }
}
```

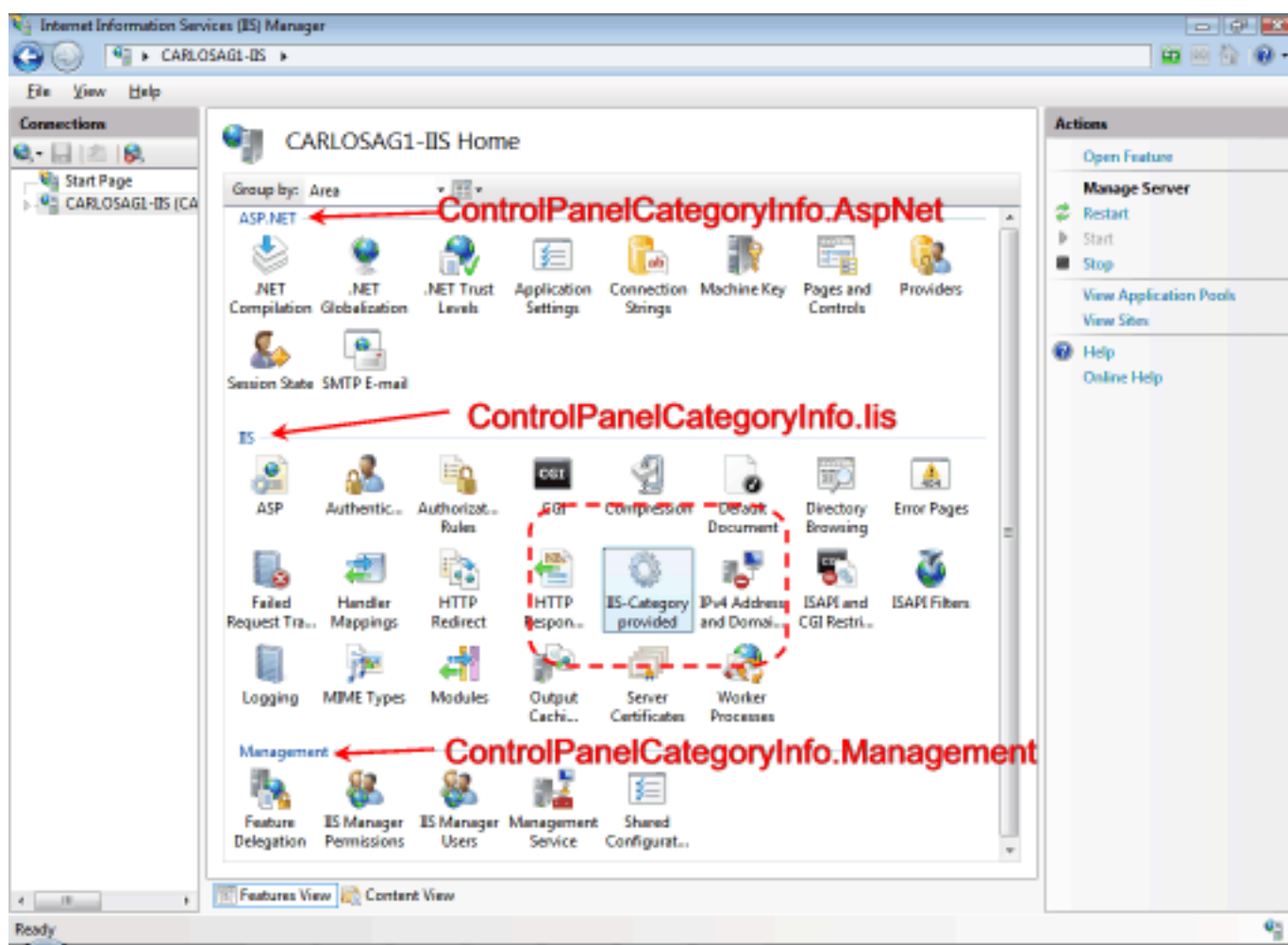
حال پروژه را Rebuild کنید و IIS را مجدداً باز کنید تا ببینید که کامپوننت جدید شما آیکن جدید را دارد. دومین نکته‌ای که به چشم می‌آید این است که اگر دقت کنید سایر ماژول‌ها یا کامپوننت‌ها هر کدام در گروهی جداگانه جای گرفته‌اند و افزونه‌ی شما در یک گروه به اسم others، که زیاد جالب نیست و شاید دوست داشته باشید که ماژول شما هم در یکی از همین دسته‌ها جای بگیرد یا حتی اینکه خودتان یک گروه جدید بسازید. برای اینکه خوب قسمت‌ها و نام کلاس‌های آن‌ها را یاد بگیرید به شکلی که در قسمت قبلی هم گذاشته بودیم خوب دقت کنید.



نکته ای که باید توجه داشته باشید این است که گروه بندی در IIS به سه شیوه صورت می‌گیرد: یکی به شیوه ناحیه **Area** مثل **ASP.Net** و **management** و **IIS**، دسته بندی **Category** مثل **Application Development**، **ServerFeatures** و بدون گروه بندی که می‌توانید از طریق لیست **Group By** در بالای پنجره IIS یکی از این سه حالت را انتخاب نمایید برای اینکه ماژول در یک ناحیه یا دسته‌ای مشخص قرار بگیرد کد زیر را بنویسید:

```
internal class imageCopyrightUI : Module
{
    protected override void Initialize(IServiceProvider serviceProvider, ModuleInfo moduleInfo)
    {
        base.Initialize(serviceProvider, moduleInfo);
        IControlPanel controlPanel = (IControlPanel)GetService(typeof(IControlPanel));
        ModulePageInfo modulePageInfo = new ModulePageInfo(this, typeof(imageCopyrightUIPage),
        "Image Copyright", "Image Copyright", Resource1.Visual_Studio_2012, Resource1.Visual_Studio_2012);
        controlPanel.RegisterPage(ControlPanelCategoryInfo.AspNet, modulePageInfo);
    }
}
```

اگر خوب دقت کنید می‌بینید که تنها تغییر کدها در متد **regiterpage** بوده است که با استفاده از کلاس **ControlPanelCategoryInfo** ناحیه مورد نظر را مشخص کردیم و اگر حالا پروژه را **Rebuild** کنید می‌بینید که در ناحیه **ASP.Net** قرار گرفته است. گروه‌های مختلفی دیگری چون **Application Development**، **Server**، **Performance** و ... هم در این کلاس وجود دارند که جمعا 9 گروه می‌شوند.



حال این سوال پیش می‌آید اگر بخواهیم گروه اختصاصی ایجاد کنیم، چه کاری باید انجام شود. پس کد را به شکل زیر تغییر دهید:

```
internal class imageCopyrightUI : Module
{
    protected override void Initialize(IServiceProvider serviceProvider, ModuleInfo moduleInfo)
    {
        base.Initialize(serviceProvider, moduleInfo);
        IControlPanel controlPanel = (IControlPanel)GetService(typeof(IControlPanel));
        ModulePageInfo modulePageInfo = new ModulePageInfo(this, typeof(imageCopyrightUIPage),
        "Image Copyright", "Image Copyright", Resource1.Visual_Studio_2012, Resource1.Visual_Studio_2012);

        ControlPanelCategorization areaCategorization = null;
        foreach (ControlPanelCategorization categorization in controlPanel.Categorizations) {
            if (categorization.Key == ControlPanelCategorization.AreaCategorization) {
                areaCategorization = categorization;
                break;
            }
        }
        ControlPanelCategoryInfo cat = new ControlPanelCategoryInfo("dotnettips", "Dot Net Tips", "This
is a Tutorial", areaCategorization);
        controlPanel.RegisterCategory(cat);
        controlPanel.RegisterPage(cat.Name, modulePageInfo);
    }
}
```

در خطوط foreach ما به دنبال نوع گروهی که قرار است ماژول ما در آن قرار بگیرد می‌گردیم و علاقمندیم که گروه بندی ما در نوع Area باشد. برای همین این نوع را یافته و در متغیری از نوع ControlPanelCategorization قرار می‌دهیم. سپس توسط کلاس ControlPanelCategoryInfo یک نوع گروه بندی جدید ایجاد کرده ایم که به ترتیب نام، عنوان، توضیح و نهایتاً نوع گروه بندی را در آن لحاظ می‌کنیم و سپس با دستور controlPanel.RegisterCategory گروه جدید خود را که در area قرار دارد، در IIS ثبت می‌کنیم و موقع افزودن صفحه مستقیماً نام گروه را در آن ذکر می‌کنیم. پروژه را Rebuild و IIS را مجدداً اجرا کنید. اگر Group by

شما روی Area باشد که به طور پیش فرض چنین است شما باید گروه Dot Net Tips را ببینید؛ حال از طریق لیست Group By گزینه‌ی Category را انتخاب نمایید. همانطور که مشاهده می‌کنید دوباره ماژول شما در دسته‌ی Others قرار می‌گیرد؛ چرا که ما برای Area گروه بندی را لحاظ کرده بودیم. برای اینکه بتوانیم در دو حالت، دسته بندی داشته باشیم، کد را به شکل زیر تغییر می‌دهیم:

```
ControlPanelCategorization areaCategorization = null;
ControlPanelCategorization CategoryCategorization = null;
foreach (ControlPanelCategorization categorization in controlPanel.Categorizations) {
    if (categorization.Key == ControlPanelCategorization.AreaCategorization) {
        areaCategorization = categorization;
    }
    if (categorization.Key == ControlPanelCategorization.CategoryCategorization)
    {
        CategoryCategorization = categorization;
    }
}
ControlPanelCategoryInfo cat1=new ControlPanelCategoryInfo("dotnettipsarea","Dot Net Tips
Area","This is a Tutorial",areaCategorization);
controlPanel.RegisterCategory(cat1);
controlPanel.RegisterPage(cat1.Name,modulePageInfo);

ControlPanelCategoryInfo cat2 = new ControlPanelCategoryInfo("dotnettipsat", "Dot Net Tips
Category", "This is a Tutorial", CategoryCategorization);
controlPanel.RegisterCategory(cat2);
controlPanel.RegisterPage(cat2.Name, modulePageInfo);
```

فکر کنم اتفاق بالا با توجه به مواردی که قبلا یاد گرفتید با وضوح کامل مشخص باشد که اینبار دو حالت گروه بندی را ذخیره و هر کدام را جداگانه در کنترل پنل IIS ثبت کردیم.

بین سایر گزینه‌های کنترل پنل گزینه controlpanel.regiterhomepage هم به چشم می‌خورد که اگر رابط کاربری را با این گزینه رجیستر کنیم صفحه اصلی IIS پریده و رابط کاربری ما جایگزینش می‌شود که البته این قسمت را باید با احتیاط با آن برخورد کرد وگرنه اگر بخواهیم همین رابط کاربری را به عنوان صفحه‌ی خانگی رجیستر کنیم، دسترسی ما به دیگر ماژول‌ها قطع خواهد شد.

[سورس پروژه](#)

تا به اینجا این مبحث از سری آموزشی ما بسته می‌شود. در مقالات آینده موارد دیگری از IIS را مورد بررسی قرار خواهیم داد.



در این مطلب و همینطور مطلب بعدی قرار است به مبحث لاگ فایل‌ها Logfile بپردازیم. همانطور که می‌دانید سیستم IIS مثل هر سیستم دیگری لاگ‌هایی دارد که به مرور زمان این لاگ‌ها می‌توانند مقدار زیادی از ظرفیت دیسک سخت را به خود اختصاص بدهند و این عمل می‌تواند موجب بروز مشکلاتی در سرور شود. به خوبی به یاد دارم که برای یکی از مشتریانم VPS تهیه نموده بودیم و بعد از یک سال با من تماس گرفت که سایت بالا نمی‌آید و وقتی بررسی شد، دیدم که از فضای دیسک سخت چند گیگابایتی، تنها چند مگابایت به طور ناچیز فضا برایش باقی مانده است و باعث شده سرور از کار بیفتد. پس از بررسی متوجه شدیم تمام این فضاها توسط لاگ فایل‌ها پر شده است و از آنجا که سرویس دهنده تا مبلغی را به عنوان مدیریت سرور، ماهانه دریافت نکند، مدیریت این سرور مجازی را به عهده نداشته اند که البته بعدها با انتقال به یک سرور دیگر از یک سرویس دهنده دیگر مشکلات ما در مورد سرور برای همیشه حل شد.

پس این داستان به خوبی روشن می‌کند که مدیریت این لاگ‌ها چقدر می‌تواند مهم و حیاتی باشد. آقای تیم وَن از تیم تحریریه مایکروسافت در بخش IIS موارد زیر را برای مدیریت لاگ‌ها بر می‌شمارد:

فعال سازی فشرده سازی

انتقال لاگ‌ها به یک سیستم راه دور

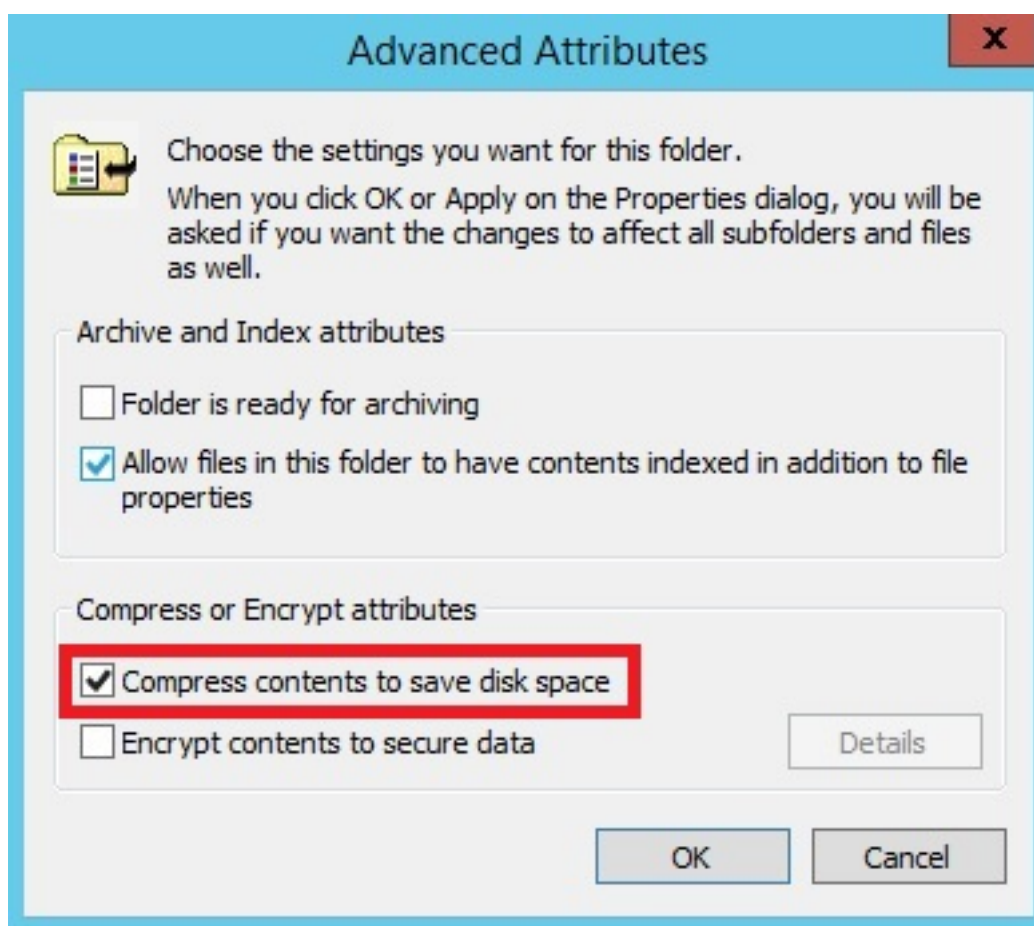
حذف لاگ فایل‌های قدیمی از طریق اسکریپت نویسی

حذف لاگ فایل‌های قدیمی توسط IIS Log File Cleaner

### فشرده سازی دایرکتوری لاگ فایل‌ها

مسیر ذخیره لاگ فایل‌ها در آدرس زیر می‌باشد. به این آدرس رفته و Properties دایرکتوری مورد نظر را باز کنید و در برگه‌ی General آن، گزینه‌ی Advanced را زده تا در کادر جدیدی که باز می‌شود، گزینه‌ی Compress contents to save disk space را انتخاب کنید تا محتویات در هنگام ذخیره روی دیسک سخت فشرده شوند.

```
%SystemDrive%\inetpub\logs\LogFiles
```



این روش ساده‌ترین روش موجود برای مدیریت لاگ هاست ولی روش نهایی نیست و باز به مرور زمان این روش هم کارایی خودش را از دست خواهد داد. این روش بیشتر شبیه خرید زمان می‌باشد تا اینکه یک راه حل نهایی برای حل مشکل باشد. البته این را هم باید مدنظر داشت که موقع تیک زدن گزینه بالا عملیات فشرده سازی باعث کند شدن سرعت کامپیوتر در حین آغاز عمل ذخیره سازی لاگ فایل‌ها هم خواهد شد. پس اگر قصد چنین کاری را دارید در ساعاتی که سرور کمترین فشار از طرف کاربران را دارد یا اصطلاحاً پیک کاری آن پایین است انجامش دهید.

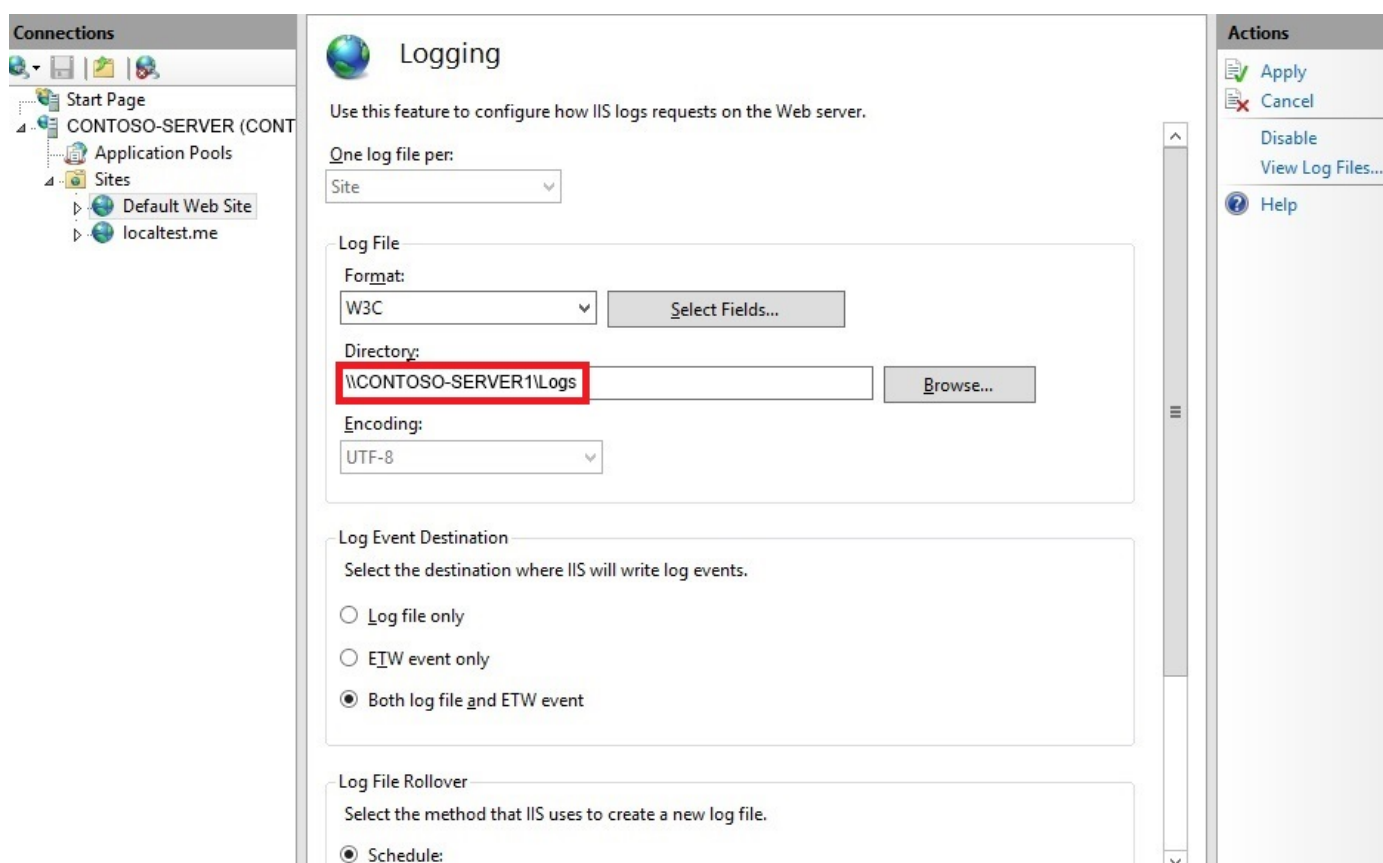
#### انتقال لاگ فایل‌ها به یک سیستم راه دور

همانطور که در بالا اشاره کردیم محل پیش فرض ذخیره سازی لاگ‌ها در مسیر

```
%SystemDrive%\inetpub\logs\LogFiles
```

قرار دارد و این محل ذخیره سازی برای هر سرور یا حتی یک وب سایت خاص در صفحه تنظیمات Logging مشخص شده است و شما در می‌توانید این لاگ‌ها را حتی برای کل سرور یا مربوط به یک سایت خاص، به سروری دیگر انتقال دهید. این امکان می‌تواند به امنیت سیستم هم کمک فراوانی کند تا اگر دیسک محلی Local Disk هم دچار مشکل شد، باز خواندن لاگ فایل‌ها میسر باشد و با استفاده از [ابزارهای تحلیل لاگ فایل‌ها](#)، آن‌ها را مورد بررسی قرار دهیم. برای تغییر محل ذخیره سازی لاگ‌ها به یک سیستم راه دور، راه حل زیر را طی کنید.

در IIS وب سایتی را که می‌خواهید لاگ آن انتقال یابد، انتخاب کنید؛ یا اگر لاگ کل سیستم IIS را می‌خواهید انتقال بدهید نام سرور را در لیست درختی انتخاب کنید و از ماژول‌های سمت راست، ماژول Logging را انتخاب کنید و در قسمت Directory که محل ذخیره سازی فعلی لاگ‌ها را نوشته شده است، به صورت [UNC](#) آدرس دهی کنید. در آدرس زیر اولی نام سرور است -Contoso-\\server1 و دومی هم Logs نام پوشه‌ای که به اشتراک گذاشته شده است.



### حذف لاگ فایل‌های قدیمی با استفاده از اسکریپت

با این روش می‌توانید لاگ فایل‌هایی را که بعد از مدتی معین که دلخواه شما هست، از سیستم حذف نمایید و اگر این اسکریپت را زمان بندی خودکار نمایید، می‌توانید از مراقبت مداوم و ثابت این کار نیز رها شوید.

با استفاده از VBScript بررسی می‌کنیم که اگر مثلاً عمر لاگ فایل به 30 روز رسیده است، باید حذف شوند. خط دوم کد زیر نهایت عمر یک لاگ فایل را مشخص می‌کند:

```
sLogFolder = "c:\inetpub\logs\LogFiles"
iMaxAge = 30 'in days
Set objFSO = CreateObject("Scripting.FileSystemObject")
set colFolder = objFSO.GetFolder(sLogFolder)
For Each colSubfolder in colFolder.SubFolders
    Set objFolder = objFSO.GetFolder(colSubfolder.Path)
    Set colFiles = objFolder.Files
    For Each objFile in colFiles
        iFileAge = now-objFile.DateCreated
        if iFileAge > (iMaxAge+1) then
            objFSO.deletefile objFile, True
        end if
    Next
Next
```

اسکریپت بالا تمامی subfolderها را برای همه سایت‌ها بررسی کرده و لاگ‌های آنان را حذف می‌کند. ولی اگر دوست دارید این عملیات را تنها به یک وب سایت محدود کنید، باید مسیر را در خط اول دقیق‌تر مشخص کنید.

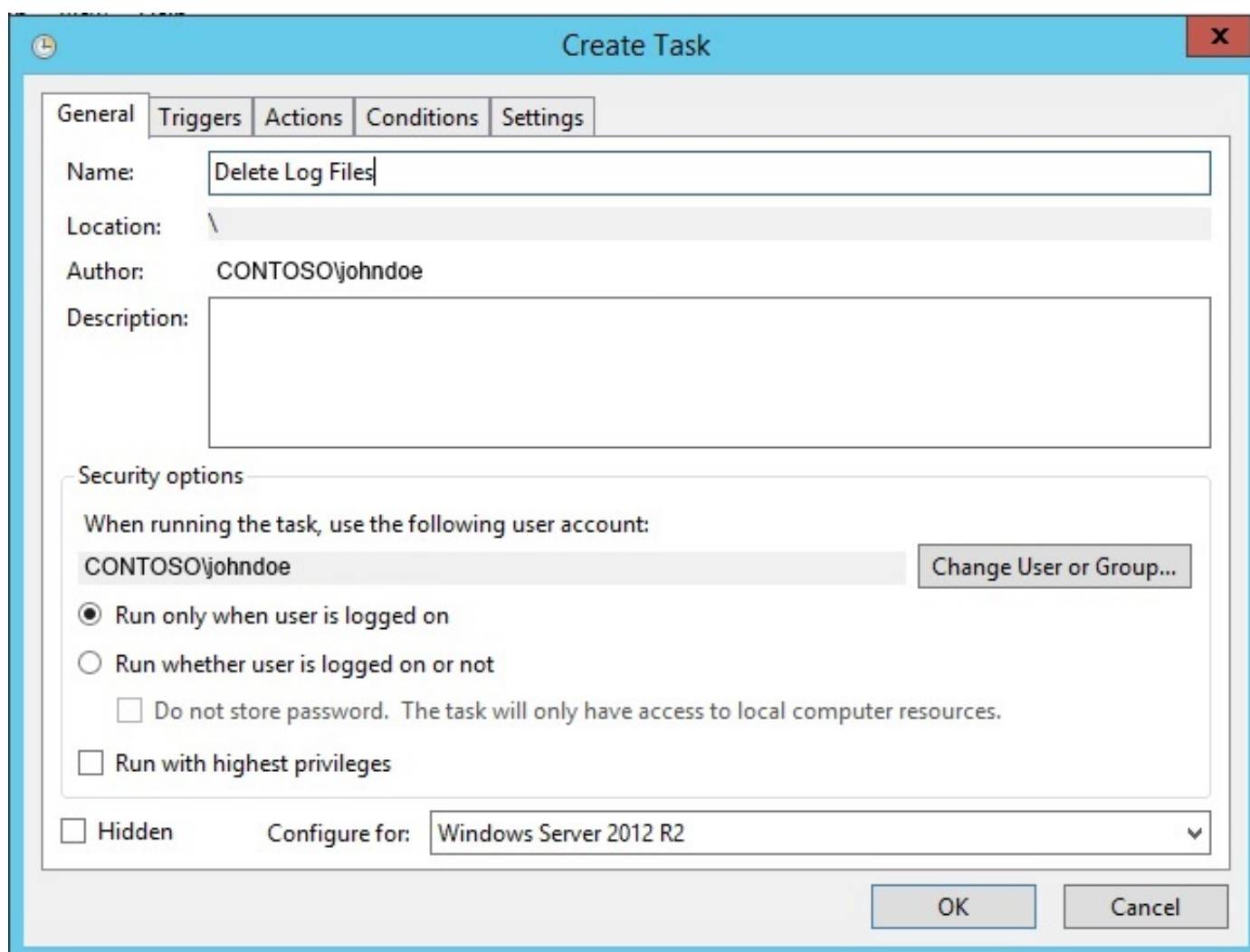
برای اجرای دستی اسکریپت در cmd تایپ کنید:

```
cscript.exe c:\scripts\retentionscript.vbs
```

ولی اگر میخواهید این اسکریپت در هر دوره‌ی زمانی خاص اجرا شود، یا زمان بندی Scheduling گردد، دیگر مجبور نیستید هر بار به فکر نگهداری از لاگ‌ها باشید.

### زمان بندی اجرای اسکریپت

server manager ( قابل تست در ویندوزهای سرور ) را باز کرده و از منوی Tools گزینه Task Scheduler را انتخاب کنید و در قسمت Actions گزینه Create Task را انتخاب نمایید. در کادر باز شده نام " Delete Log Files " را برای مثال برگزینید و در قسمت Security هم کاربری که اجازه اجرای اسکریپت را دارد مشخص کنید.



برگه Triggers را انتخاب کرده و گزینه New را انتخاب کنید و عملیات زمان بندی را تنظیم کنید و حتما بعد از زمان بندی مطمئن باشید که تیک Enabled فعال است.

**New Trigger**

Begin the task: **On a schedule**

**Settings**

☐ One time  
☒ **Daily**  
☐ Weekly  
☐ Monthly

Start: 5/ 5/2014 6:05:15 PM ☐ Synchronize across time zones

Recur every: 1 days

**Advanced settings**

☐ Delay task for up to (random delay): 1 hour  
☐ Repeat task every: 1 hour for a duration of: 1 day  
☐ Stop all running tasks at end of repetition duration  
☐ Stop task if it runs longer than: 3 days  
☐ Expire: 5/ 6/2015 9:19:02 AM ☐ Synchronize across time zones

☒ **Enabled**

OK Cancel

در برگه Actions هم گزینه New را انتخاب کنید؛ در کادر باز شده از لیست Start a program را انتخاب کرده و در قسمت Program\script، دستور cscript را ذکر نمایید و به عنوان آرگومان ورودی Add arguments هم مسیر اسکریپت خود را ذکر نمایید و کادر را تایید کنید.

X

## New Action

You must specify what action this task will perform.

---

Action:

Start a program

▼

Settings

Program/script:

cscript

Browse...

Add arguments (optional):

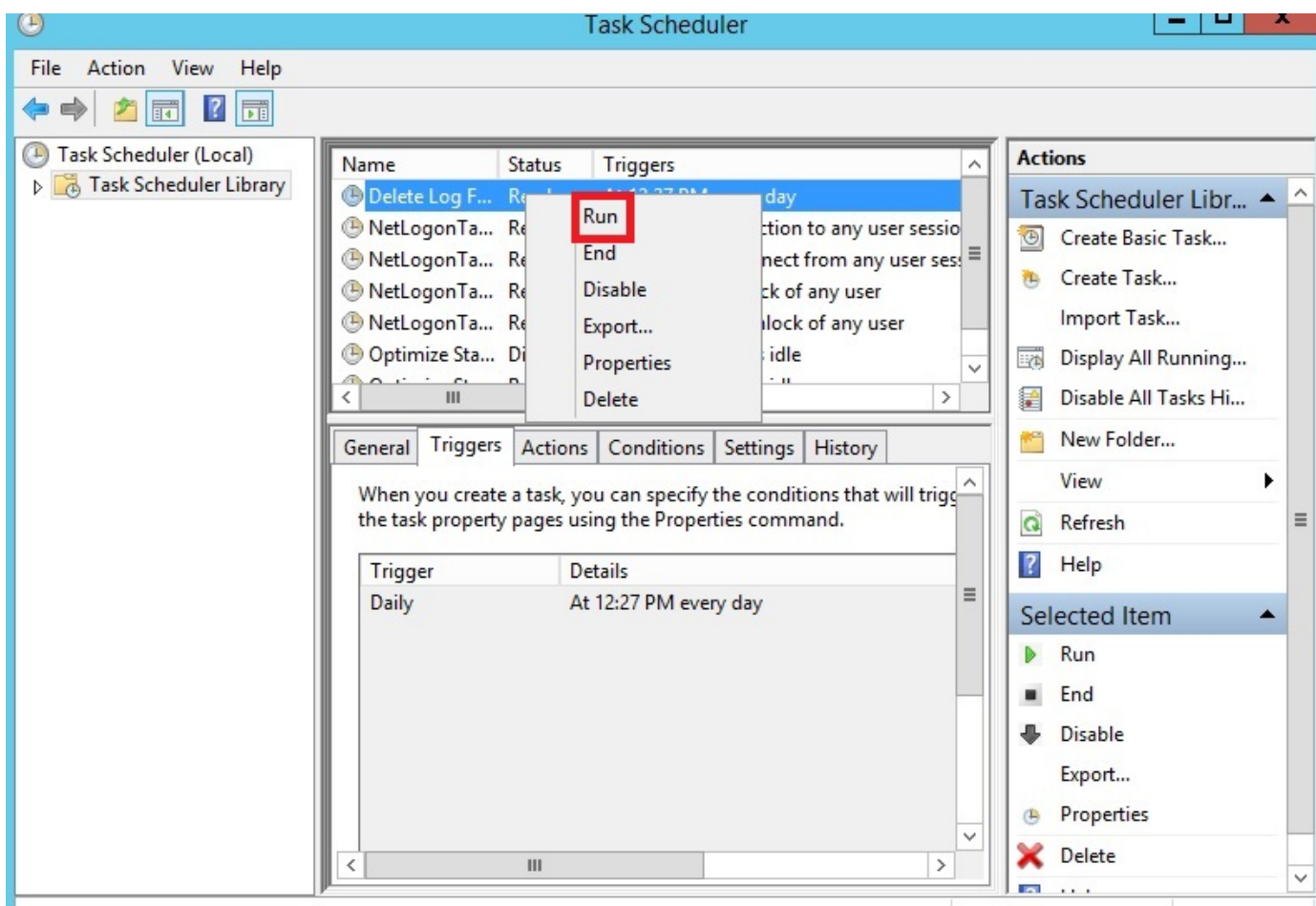
C:\iis\Log\_File\_Deletion.v

Start in (optional):

OK

Cancel

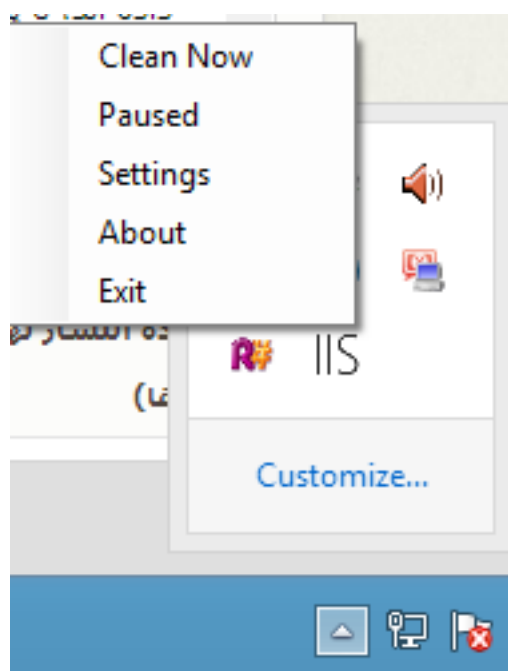
برای آغاز زمان بندی در لیست وظیفه‌های فعال active task pane، وظیفه ای که الان ساخته اید را اجرا کرده و به مسیر ذخیره لاگ‌ها رفته و می‌بینید که لاگ‌های مورد نظر حذف شده‌اند؛ پس از صحت اجرای اسکریپت مطمئن می‌شویم. دوباره به لیست وظایف رفته و گزینه End را بزنید تا وظیفه، در حالت Ready قرار گیرد تا از همین الان فرایند زمان بندی اجرای اسکریپت آغاز شود.



### حذف لاگ فایل‌ها با استفاده از IIS Log Cleaner Tools

ساده‌ترین ابزار برای مدیریت حذف لاگ فایل‌هاست که هر یک ساعت یکبار اجرا شده و لاگ فایل‌های تاریخ گذشته را که زمانش را شما تعیین می‌کنید، به سمت سطل زباله که البته درستش بازیافت است Recycle Bin انتقال می‌دهد تا از ضرر از دست دادن لاگ‌ها جلوگیری کند که بعداً شما می‌توانید آن‌ها را به صورت دستی حذف کنید. همچنین عملیات خودکار حذف را نیز می‌توان متوقف نمود.

ابتدا برنامه را از [اینجا](#) دانلود کنید. موقعیکه برنامه را اجرا کنید، در نوتیفیکیشن taskbar می‌نشیند و برنامه با یک پیغام به شما اعلام می‌کند، این اولین بار است که برنامه را باز کرده‌اید. پس یک سر به setting آن بزنید؛ با انتخاب گزینه‌ی settings برنامه بسته شده و فایل Settings.txt برای شما باز می‌شود که مدت زمان عمر لاگ فایل و مسیر ذخیره آن‌ها، از شما پرسیده می‌شود که مقدار عمر هر لاگ فایل به طور پیش فرض 30 روز و مسیر ذخیره‌ی لاگ‌ها همان مسیر پیش فرض IIS است که اگر شما دستی آن را تغییر داده اید، با پرسیدن آن، از محل لاگ‌ها اطمینان کسب می‌کند. در صورتی که قصد تغییری را در فایل، دارید آن را تغییر داده و ذخیره کنید و برنامه را مجدداً اجرا کنید.



نکات نهایی در مورد این برنامه :

اگر از ابزار IIS Cleaner Tool استفاده می‌کنید باید دستی سطل بازیافت را هم پاک کنید و هم اینکه می‌توانید یک محدودیت حجمی برای Recycle Bin قرار دهید که اگر به یک حدی رسید، خودکار پاک کند تا مشکلی برای سیستم عامل ایجاد نشود که البته به طور پیش فرض چنین است.

برنامه بالا به طور پیش فرض ریشه‌ی لاگ‌ها را حذف می‌کند. پس اگر می‌خواهید فقط سایت خاصی را مد نظر داشته باشد، آدرس دایرکتوری آن را اضافه کنید. البته چون این برنامه فقط روی یک دایرکتوری کار می‌کند و شما چند وب سایت دارید و مثلاً می‌خواهید سه تایی آن‌ها را پاکسازی کنید، چاره‌ی جز استفاده از اسکریپت‌های با زمان بندی ندارید.

برنامه‌ی بالا فقط فایل‌هایی با پسوند log را به سطل بازیافت انتقال می‌دهد.

برنامه‌ی بالا یک سرویس نیست و باید به طور دستی توسط کاربر اجرا گردد. پس اگر ریست هم شد باید دستی اجرا شود یا آن را به داخل پوشه startup بکشید.

برنامه برای اجرایش نیاز به لاگین کاربر و مجوز نوشتن در آن پوشه را دارد تا به درستی کار کند.

در قسمت بعدی مبحث لاگ‌ها را ادامه خواهیم داد و با مازول Logging در IIS و تنظیماتش آشنا خواهیم شد.



پیکربندی قسمت لاگ‌ها، میتواند برای یک سرور و یا وب سایت خاص از طریق فایل کانفیگ یا از طریق خود IIS انجام گیرد. برای اینکه به بیشتر این قابلیت‌ها در IIS دسترسی داشت، باید یکی از نسخه‌های ویندوز سرور 2012 و ویندوز 8 را نصب کرده باشید. لاگ‌ها به ثبت خطاها و درخواست‌های HTTP می‌پردازند و با تحلیل آن‌ها میتوان عملیات بهینه سازی را بر روی سرو اجرا کرد. تمامی ثبت لاگ‌ها توسط Http.sys انجام می‌گیرد.

### نحوه‌ی ذخیره سازی لاگ‌ها

در این بخش نحوه‌ی ذخیره سازی و فرمت ذخیره‌ی لاگ‌ها را در دو سطح سایت و سرور به طور جداگانه بررسی می‌کنیم. در IIS ماژول Logging را باز کنید و در لیست One log file per می‌توانید مشخص کنید که لاگ‌ها در چه سطحی اجرا شوند. اگر گزینه‌ی server باشد، تمامی خطاها و درخواست‌های رسیده به سرور در یک فایل لاگ ثبت می‌شوند. ولی اگر سطح سایت باشد، برای هر سایت بر روی IIS لاگ‌ها، جداگانه بررسی می‌شوند. به طور پیش فرض سطح سایت انتخاب شده است.

### سطح سایت

موقعی که در لیست، سایت را انتخاب کنید، در لیست format می‌توانید تعیین کنید که لاگ‌ها به چه صورتی باید ذخیره شوند. مواردی که در این حالت لیست می‌شوند گزینه‌های W3C, IIS, NCSA, Custom می‌باشند که در زیر یکایک آن‌ها را بررسی می‌کنیم:

**فرمت IIS:** این فرمت توسط مایکروسافت ارائه شده و در این حالت لاگ‌های همه‌ی وب سایت‌ها ذخیره می‌شوند. به این فرمت Fixed ASCII Based Text نیز می‌گویند؛ چرا که اجازه‌ی خصوصی سازی ندارد و نمی‌توانید بگویید چه فیلدهایی در لاگ قرار داشته باشند. لاگ فایل‌های این فرمت با ، (کاما) از هم جدا می‌شوند و مقدار زمانی که برای هر فیلد ثبت می‌شود، به صورت محلی local Time می‌باشد.

فیلدهایی که در لاگ این نوع فرمت خواهند آمد، به شرح زیر است:

Client IP address

User name

Date

Time

Service and instance

Server name

Server IP address

Time taken

Client bytes sent

Server bytes sent

Service status code (A value of 200 indicates that the request was fulfilled successfully)

.Windows status code (A value of 0 indicates that the request was fulfilled successfully)

Request type

Target of operation

Parameters (the parameters that are passed to a script)

احتمال این وجود دارد که بعضی از فیلدها در بعضی رکوردها، شامل اطلاعاتی نباشند که به جای مقدار آن علامت - ثبت می‌گردد و برای کاراکترهایی که قابل نمایش نیستند یا کاراکتر نمایشی ندارند، از علامت + استفاده می‌شود. دلیل اینکار هم این است که ممکن است یک کاربر مهاجم، به ارسال اطلاعات کلیدهای کنترلی چون [Carriage return](#) اختصارا CR یا [Line Feed](#) به اختصار LF کند، که باعث شکسته شدن خط لاگ فایل می‌شود و در نتیجه از استاندارد خارج خواهد شد و هنگام خواندن آن هم با خطا روبرو می‌شویم؛ در نتیجه با جایگزینی چنین کاراکترهایی با + از این اتفاق جلوگیری می‌شود. شکل زیر نمونه ای از یک خط لاگ در این فرمت است:

```
192.168.114.201, -, 03/20/01, 7:55:20, W3SVC2, SERVER, 172.21.13.45, 4502, 163, 3223, 200, 0, GET, /DeptLogo.gif, -,
```

نام فیلد	نوع حالت مقداردهی	توضیح اتفاقات افتاده
Client IP address	192.168.114.201	آی پی کلاینت
User name	-	کاربر ناشناس است
Date	03/20/01	تاریخ فعالیت
Time	7:55:20	ساعت فعالیت
	Service and instance	W3SVC2
لاگی که مربوط به سایت خاصی می‌شود به صورت W3SVC# نمایش داده می‌شود که علامت # شماره سایت می‌باشد که در اینجا این لاگ مربوط به سایت شماره 2 است	Server name	SERVER
نام سرور	Server IP	172.21.13.45
آی پی سرور	Time taken	4502
چقدر انجام عملیات این درخواست به طول انجامیده است که بر حسب میلی ثانیه است.	Client bytes sent	163
تعداد بایت هایی که از طرف کلاینت به سرور ارسال شده است	Server bytes sent	3223
تعداد بایت هایی که از طرف سرور به سمت کلاینت ارسال شده است	Service status code	200
درخواست کاملاً موفقیت آمیز بوده است	Windows status code	0
درخواست کاملاً موفقیت آمیز بوده است	Request type	GET
نوع درخواست کاربر	Target of operation	/DeptLogo.gif
کاربر قصد دانلود یک فایل تصویری GIF	Parameters	-

نام فیلد	نوع حالت مقداردهی	توضیح اتفاقات افتاده
داشته است که نامش Deptlogo است		

**فرمت NCSA:** این فرمت توسط مرکز علمی کاربردهای ابرمحاسباتی [National Center for Supercomputing Applications](http://www.nsl.gov/research/resources/publications/nscsa/) ایجاد شده و دقیقاً مانند قبلی نمیتوان در آن نوع فیلدها را مشخص کرد و برای جدا سازی، از فاصله space استفاده می‌کند و ثبت مقدار زمان در آن هم به صورت محلی و هم UTC می‌باشد.

این فیلدها در لاگ آن نمایش داده می‌شوند:

Remote host address

Remote log name (This value is always a hyphen)

User name

Date, time, and Greenwich mean time (GMT) offset

Request and protocol version

Service status code (A value of 200 indicates that the request was fulfilled successfully)

Bytes sent

نمونه ای از یک لاگ ثبت شده:

```
172.21.13.45 - Microsoft\JohnDoe [08/Apr/2001:17:39:04 -0800] "GET /scripts/iisadmin/ism.dll?http/serv
HTTP/1.0" 200 3401
```

نام فیلد	مقدار ثبت شده	توضیح اتفاق افتاده
Remote host address	172.21.13.45	آی پی کلاینت
Remote log name	-	نامی وجود ندارد
User name	Microsoft\JohnDoe	نام کاربری
Date, time, and GMT offset	[08/Apr/2001:17:39:04 -0800]	تاریخ و ساعت فعالیت به صورت محلی که 8 ساعت از مبدا گرینویچ بیشتر است
Request and protocol version	GET /scripts/iisadmin/ism.dll?http/serv HTTP/1.0	کاربر با متد GET و Http نسخه‌ی یک، درخواست فایل ism.dll را کرده است.
Service status code	200	عملیات کاملاً موفقیت آمیز بود.
Bytes sent	3401	تعداد بایت‌های ارسال شده به سمت کاربر

امنیت در برابر کاربران مهاجم مانند همان فرمت قبلی صورت گرفته است.

**فرمت W3C:** توسط W3C توسط کنسرسیوم جهانی وب ارائه شده است و یک فرمت customizable ASCII text-based است. به این معنی که میتوان فیلدهایی که در گزارش نهایی می‌آید را خودتان مشخص کنید، که برای اینکار در کنار لیست، دکمه‌ی Select

وجود دارد که میتوانید هر کدام از فیلدهایی را که خواستید، انتخاب کنید تا به ترتیب در خط لاگ ظاهر شوند. تاریخ ثبت به صورت UTC است.

نام فیلد	توضیح	به طور پیش فرض انتخاب شده است
Date	تاریخ رخ دادن فعالیت	بله
Time	ساعت رخ دادن فعالیت بر اساس UTC	بله
Client IP Address	آی پی کلاینت	بله
User Name	نام کاربری که هویت آن تایید شده و در صورتی که هویت تایید شده نباشد و کاربر ناشناس باشد، جای آن - قرار می گیرد	بله
Service Name and Instance Number	نام و شماره سایتی که درخواست در آن صورت گرفته است	خیر
Server Name	نام سروری که لاگ روی آن ثبت می شود	خیر
Server IP Address	آی پی سرور که لاگ روی آن ثبت می شود	بله
Server Port	شماره پورتی که سرویس مورد نظر روی آن پورت اعمال می شود.	بله
Method	متد درخواست مثل GET	بله
URI Stem	هدف درخواست یا Target مثل index.htm	بله
URI Query	کوئری ارسال شده برای صفحات داینامیک	بله
HTTP Status	کد وضعیتی HTTP status	بله
Win32 Status	کد وضعیتی ویندوز	خیر
Bytes Sent	تعداد بایت های ارسال شده به سمت کلاینت	خیر
Bytes Received	تعداد بایت های دریافت شده از سمت کلاینت	خیر
Time Taken	زمان به طول انجامیدن درخواست بر حسب میلی ثانیه	خیر
Protocol Version	درخواست با چه نسخه ای از پروتکل http یا ftp ارسال شده است	خیر
Host	اگر در هدر درخواست ارسالی این گزینه بوده باشد، نوشته خواهد شد.	خیر
User Agent	اطلاعات را از هدر درخواست می گیرد.	بله
Cookie	اگر کوکی رد و بدل شده باشد، محتویات کوکی ارسالی یا دریافت شده	خیر
Referrer	کاربر از چه سایتی به سمت سایت ما آمده است.	خیر
Protocol Substatus		بله

نام فیلد	توضیح	به طور پیش فرض انتخاب شده است
	<p>در صورت رخ دادن خطا در IIS ، کد خطا بازگردانده میشود. در IIS به منظور امنیت بیشتر و کاهش حملات، محتوای خطاهای رخ داده در IIS به صورت متنی نمایش داده نمی‌شوند و شامل کد خطایی به اسم Substatus Code هستند تا مدیران شبکه با ردیابی لاگ‌ها پی به دلیل خطا و درخواست‌های ناموفق ببرند. برای مثال Error 404.2 به این معنی است که فایل درخواستی به دلیل قوانین محدود کننده، قفل شده و قابل ارائه نیست. ولی هکر تنها با خطای 404 یعنی وجود نداشتن فایل روبرو می‌شود. در حالت substatus code، کد شماره 2 را هم خواهید داشت که در لاگ ثبت می‌شود.</p> <p>هر شخصی که در سرور توانایی دسترسی به لاگ‌ها را داشته باشد، می‌تواند کد دوم خطا را نیز مشاهده کند. برای مثال مدیر سرور متوجه میشود که یکی از فایل‌های مورد نظر به کاربران، خطای 404 نمایش میدهد و با بررسی لاگ‌ها متوجه می‌شود که کد خطا 404.9 هست. از آنجا که ما همه‌ی کدها را حفظ نیستیم به این <a href="#">صفحه</a> رجوع می‌کنیم و متوجه میشویم تعداد کاربرانی که برای این فایل، اتصال connection ایجاد کرده‌اند بیش از مقدار مجاز است و مدیر میتواند این وضع را کنترل کند. برای مثال تعداد اتصالات مجاز را نامحدود unlimited تعیین کند.</p>	

حروف - و + برای موارد بالا هم صدق می‌کند. در ضمن گزینه‌های زیر در حالتی که درخواست از پروتکل FTP باشد مقداری نخواهند گرفت:

uri-query

host

(User-Agent)

Cookie

Referrer

substatus

**گزینه Custom:** موقعی که شما این گزینه را انتخاب کنید مازول logging غیرفعال خواهد شد. زیرا این امکان در IIS قابل پیکر بندی نیست و نوشتن مازول آن بر عهده شما خواهد بود؛ با استفاده از اینترفیس های ILogPluginEx ، ILogPlugin و ILogUIPlugin آن را پیاده سازی کنید.

#### ذخیره اطلاعات به انکدینگ UTF-8 و موضوع امنیت

در صورتی که شما از سایتی با زبانی غیر از انگلیسی و لاتین و فراتر از ANSI استفاده می کنید، این گزینه حتما باید انتخاب شده باشد تا درخواست را بهتر لاگ کند. حتی برای وب سایت های انگلیسی زبان هم انتخاب این گزینه بسیار خوب است؛ چرا که اگر به سمت سرور کاراکترهای خاصی در URL ارسال شوند، نمی تواند با کدپیچ موجود آن ها را درست تبدیل کند.

#### ادامه ی تنظیمات

موارد بعدی که در تنظیمات لاگ ها کاملا مشخص و واضح است، عملیات زمان بندی است که برای ساخت یک فایل لاگ جدید به کار می رود؛ برای مثال هر ساعت یک لاگ فایل جدید بسازد و فعالیت های موجود در هر ساعت در یک لاگ ذخیره می شوند. گزینه ی بعدی حداکثر حجم هر فایل لاگ است که به صورت بایت مشخص می شود. اگر مقداری که تعیین میکنید کمتر از 1048576 بایت باشد، خودش به طور پیش فرض همان 1048576 بایت را در نظر خواهد گرفت. گزینه بعدی do not create a new logfile بدین معناست که همه ی لاگ ها در یک فایل ذخیره می شوند و فایل جدیدی برای لاگ ها ایجاد نمی شود.

گزینه آخری به اسم use local time for filenames and rollover است که اگر انتخاب شود، نامگذاری هر فایل لاگ بر اساس زمان محلی ساخت فایل لاگ خواهد بود. در صورتیکه انتخاب نشود، نامگذاری با زمان UTC درج خواهد شد.

#### سطح سرور

لاگ ها فقط در سمت سرور انجام می گیرد و لاگ هر سایت در یک فایل لاگ ثبت می شود. اگر بخواهید لاگ ها را در سطح سرور انجام دهید، گزینه ی binary هم اضافه خواهد شد.

**Binary:** در این گزینه دیگر از قالب بندی یا فرمت بندی لاگ ها خبری نیست و لاگ هر وب سایت به صورت اختصاصی صورت نمی گیرد. عملیات ذخیره سازی و ثبت هر لاگ می تواند از منابع یک سرور از قبیل حافظه و CPU و ... استفاده کند و اگر تعداد این وب سایت ها بالا باشد، باقی روش ها باعث فشار به سرور می شوند. برای همین ایجاد یک فایل خام از لاگ ها در این مواقع می تواند راهگشا باشد. برای همه یک فایل لاگ ایجاد شده و بدون قالب بندی ذخیره می کند. پسوند این نوع لاگ ها ibl است که مخفف Internet Binary Log می باشد. دلیل این تغییر پسوند این است که اطمینان کسب شود کاربر، با برنامه های متنی چون notepad یا امثال آن که به Text Utilities معروفند فایل را باز نمی کند. برای خواندن این فایل های میتوان از برنامه ی [Log parser](#) استفاده کرد. پروتکل های FTP, NNTP و SMTP در این حالت لاگشان ثبت نمی شود.

در مبحث قبلی گفتیم که ویرایش تنظیمات لاگ‌ها از طریق IIS با ویرایش مستقیم فایل‌های کانفیگ میسر است. در این مقاله که قسمت پایانی مبحث لاگ هاست، در مورد ویرایش فایل‌های کانفیگ صحبت می‌کنیم؛ همچنین استفاده از دستورات appcmd برای ویرایش و نهایتاً کد نویسی در زبان سی شارپ و جاوااسکریپت. تنظیمات لاگ سایت‌ها در فایل applicationhost در آدرس زیر قرار دارد:

C:\Windows\System32\inetsrv\config\applicationHost.config

برای هر تگ سایت، یک تگ <logfile> وجود دارد که ویژگی‌های Attributes آن، نوع ثبت لاگ را مشخص می‌کنند و می‌توانید مستقیماً در اینجا به ویرایش بپردازید. البته ویرایش فایل کانفیگ از طریق IIS به طور مستقیم هم امکان پذیر است. برای این منظور در IIS سرور را انتخاب و از بین ماژول‌های قسمت management گزینه‌ی Configuration Editor را انتخاب کنید. در قسمت Section گزینه‌ی System.applicationhost را باز کرده و از زیر مجموعه‌های آن گزینه‌ی Site را برگزینید. در تنظیمات باز شده، گزینه collection را انتخاب کنید تا در انتهای سطر، دکمه‌ی ... پدیدار گردد. روی آن کلیک کنید تا محیطی ویرایشی باز گردد که به شما اجازه‌ی افزودن و ویرایش خصوصیت‌ها را می‌دهد. برای ویرایش لاگ‌ها باید خصوصیت logfile را باز کنید. اگر قسمت قبلی را مطالعه کرده باشید، باید بسیاری از این خصوصیت‌ها و مقادیر را بشناسید. خصوصیات دیگری را هم مشاهده خواهید کرد که شاید قبلاً ندیده‌اید که البته بستگی به ورژن IIS شما دارد؛ مثلاً خصوصیت‌های flushByEntryCountW3Clog و maxLogLineLength از IIS8.5 اضافه شده اند.

#### جدول خصوصیت ها

خصوصیت	توضیح
customLogPluginClsid	یک پارامتر رشته‌ای اختیاری که در آن، آی دی کلاس یا کلاس‌هایی نوشته می‌شود که برای custom logging نوشته شده‌اند و این گزینه ترتیب اجرای آن‌ها را تعیین می‌کند.
directory	اختیاری است. محل ذخیره‌ی لاگ فایل‌ها را مشخص می‌کند و در صورتیکه ذکر نشود، همان مسیر پیش فرض است.
enabled	اختیاری است. فعال بودن سیستم لاگ برای آن سایت را مشخص می‌کند. مقدار پیش فرض آن true است.
flushByEntryCountW3Clog	این مقدار مشخص می‌کند چند رخداد باید اتفاق بیفتد تا عمل ذخیره سازی لاگ صورت گیرد. اگر بعد از هر رخداد عمل ثبت لاگ انجام شود، سرعت ثبت لاگ‌ها بالا می‌رود؛ ولی باعث استفاده‌ی مداوم از منابع و همچنین درخواست ثبت اطلاعات را روی دیسک خواهد داد و تاوان آن با زیاد شدن عملیات روی دیسک، پرداخته خواهد شد. ولی در حالتیکه چند رخداد را نگهداری سپس دسته‌ای ثبت کند، باعث افزایش کارایی و راندمان سرور خواهد شد. در صورتیکه سرور به مشکلات لحظه‌ای برخورد می‌کند مقدار آن را کاهش دهید. مقدار پیش فرض 0 است. یعنی اینکه ثبت، بعد از 64000 لاگ خواهد بود.
localTimeRollover	نحوه‌ی نامگذاری فایل‌های لاگ را مشخص می‌کند که مقدار بولین گرفته و اختیاری است. به طور پیش فرض مقدار false دارد.

خصوصیت	توضیح
<b>logExtFileFlags</b>	این گزینه در حالتی به کارتان می‌آید که فرمت W3C را برای ثبت لاگ‌ها انتخاب کرده باشید و در اینجا مشخص می‌کنید که چه فیلدهایی باید در لاگ باشند و اگر بیش از یکی بود میتوان با ، (کاما) از هم جدایشان کرد.
<b>logFormat</b>	نوع فرمت ذخیره سازی لاگ‌ها
<b>logSiteId</b>	اختیاری است و مقدار پیش فرض آن true است. بدین معنا که کد یا شماره‌ی سایت هم در لاگ خواهد بود و این در حالتی است که گزارش در سطح سرور باشد. در غیر این صورت اگر هر سایت، جداگانه لاگی برای خود داشته باشد، ذکر نمی‌گردد.
<b>logTargetW3C</b>	اختیاری است و مقدار file و * ETW را می‌گیرد که به طور پیش فرض روی File تنظیم است. در این حالت فایل لاگ‌ها در یک فایل متنی توسط http.sys ذخیره می‌شود. ولی موقعیکه از ETW استفاده می‌شود، http.sys با استفاده از iislogprovider داده‌ها را به سمت ETW ارسال میکند که منجر به اجرای سرویس Logsvc شده که از داده‌ها کوئری گرفته و آن‌ها را مستقیماً از پروسه‌های کارگر جمع‌آوری و به سمت فایل لاگ ارسال می‌کند. همچنین انتخاب این دو گزینه نیز ممکن است.
<b>maxLogLineLength</b>	حداکثر تعداد خطی که یک لاگ میتواند داشته باشد تا اینکه بتوانید در مصرف دیسک سخت صرفه جویی کنید و بیشتر کاربرد آن برای لاگ‌های کاستوم است. این عدد باید از نوع Uint باشد و اختیاری است و از 2 تا 65536 مقدار میپذیرد که مقدار پیش فرض آن 65536 می‌باشد.
<b>period</b>	همان مبحث زمان بندی در مورد ایجاد فایل‌های لاگ است که در مقاله‌ی پیشین بررسی کردیم و مقادیر Dially, Hourly, monthly و weekly را می‌پذیرد. همچنین maxsize هم هست؛ موقعی که لاگ به نهایت حجمی که برای آن تعیین کردیم میرسد.
<b>truncateSize</b>	اختیاری است و مقدار آن از نوع int64 است. حداکثر حجم یک فایل لاگ را مشخص می‌کند تا اگر period روی maxsize تنظیم شده بود، حداکثر حجم را میتوان از اینجا تعیین نمود. در مقاله پیشین در این باره صحبت کردیم؛ حداقل عدد برای آن 1,048,576 است و اگر کمتر از آن بنویسید، سیستم همین عدد 1,048,576 را در نظر خواهد گرفت. مقدار پیش فرض آن 20971520 می‌باشد.

\* ETW یا [Event Tracing Windows](#) ، سیستم و یا نرم افزاری برای عیب یابی و نظارت برای کامپوننت‌های ویندوزی است و یکی از استفاده کننده‌هایش IIS است که از ویندوز 2000 به بعد اضافه شده‌است. برای قطع کردن این ماژول در IIS هم میتوانید [قسمت هفتم](#) را بررسی نمایید و دنیال ماژول TracingModule بگردید. این ماژول به صورت Real time به ثبت رخدادهای IIS می‌پردازد.

به غیر از خصوصیات بالا، خصوصیت customFields نیز از IIS 8.5 (به بعد) در دسترس است. اگر قصد دارید به غیر از فیلدهای W3C فیلدهای اختصاصی دیگری نیز داشته باشید، میتوان از این گزینه استفاده کرد. این فیلدهای کاستوم می‌توانند اطلاعاتشان را از request header ، response header و server variables دریافت کنند. این ویژگی تنها در فرمت W3C و در سطح سایت قابل انجام است. موقعی که یک فایل لاگ شامل فیلدهای اختصاصی شود، به انتها نام فایل X\_ اضافه میگردد تا نشان دهد شامل یک فیلد



اختصاصی یا کاستوم است. نحوه تعریف آن در فایل applicationhost به شکل زیر است:

```
<system.applicationHost>
  <sites>
    <siteDefaults>
      <logFile logFormat="W3C"
        directory="%SystemDrive%\inetpub\logs\LogFiles"
        enabled="true">
      <customFields>
        <clear/>
        <add logFieldName="ContosoField" sourceName="ContosoSource"
          sourceType="ServerVariable" />
      </customFields>
    </logFile>
  </siteDefaults>
</sites>
</system.applicationHost>
```

### تغییر تنظیمات لاگ با Appcmd

```
appcmd.exe set config -section:system.applicationHost/sites /siteDefaults.logFile.enabled:"True"
/commit:apphost
appcmd.exe set config -section:system.applicationHost/sites /siteDefaults.logFile.logFormat:"W3C"
/commit:apphost
appcmd.exe set config -section:system.applicationHost/sites
/siteDefaults.logFile.directory:"%SystemDrive%\inetpub\logs\LogFiles" /commit:apphost
```

تنظیمات تگ لاگ با برنامه نویسی و اسکریپت نویسی همچنین با رفرنس Microsoft.web.administration در پروژه‌های دات نتی خود میتوانید امکان ویرایش تنظیمات را در برنامه‌های خود نیز داشته باشید:

```
using System;
using System.Text;
using Microsoft.Web.Administration;

internal static class Sample
{
    private static void Main()
    {
        using (ServerManager serverManager = new ServerManager())
        {
            Configuration config = serverManager.GetApplicationHostConfiguration();
            ConfigurationSection sitesSection = config.GetSection("system.applicationHost/sites");
            ConfigurationElement siteDefaultsElement = sitesSection.GetChildElement("siteDefaults");

            ConfigurationElement logFileElement = siteDefaultsElement.GetChildElement("logFile");
            logFileElement["logFormat"] = @"W3C";
            logFileElement["directory"] = @"%SystemDrive%\inetpub\logs\LogFiles";
            logFileElement["enabled"] = true;

            serverManager.CommitChanges();
        }
    }
}
```

با استفاده از اسکریپت نویسی توسط جاوا اسکریپت و وی بی اسکریپت هم نیز این امکان مهیاست:

```
var adminManager = new ActiveXObject('Microsoft.ApplicationHost.WritableAdminManager');
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";
var sitesSection = adminManager.GetAdminSection("system.applicationHost/sites",
"MACHINE/WEBROOT/APPHOST");
var siteDefaultsElement = sitesSection.ChildElements.Item("siteDefaults");

var logFileElement = siteDefaultsElement.ChildElements.Item("logFile");
logFileElement.Properties.Item("logFormat").Value = "W3C";
logFileElement.Properties.Item("directory").Value = "%SystemDrive%\\inetpub\\logs\\LogFiles";
logFileElement.Properties.Item("enabled").Value = true;

adminManager.CommitChanges();
```

## FTP Logging

برای اطمینان از نصب Ftp logging موقع نصب، باید از مورد زیر مطمئن باشید:

IIS را باز کنید و در لیست درختی، سرور را انتخاب کنید. در قسمت FTP میتوانید گزینهی Ftp logging را ببینید. تنظیمات این قسمت هم دقیقا همانند قسمت logging میباشد و همان موارد برای آن هم صدق میکند.

## بررسی تگ آن در applicationhost

تگ این نوع لاگ در فایل applicationhost در زیر مجموعهی تگ <site> به شکل زیر نوشته میشود:

```
<system.ftpServer>
  <log centralLogFileMode="Central">
    <centralLogFile enabled="true" />
  </log>
</system.ftpServer>
```

گزینه **centralLogFileMode** دو مقدار **central** و **site** را میپذیرد. اگر گزینهی **central** انتخاب شود، یعنی همهی لاگها را داخل یک فایل در سطح سرور ثبت کن ولی اگر گزینهی **site** انتخاب شده باشد، لاگ هر سایت در یک فایل ثبت خواهد شد.

گزینهی **logInUTF8** یک خصوصیت اختیاری است که مقدار پیش فرض آن **true** میباشد. در این حالت باید تمامی رشتهها به انکدینگ UTF-8 تبدیل شوند.

همانطور که میبینید تگ **log** در بالا یک تگ فرزند هم به اسم **centralLogFile** دارد که همان خصوصیات جدول بالا در آن مهبیاست.

## دسترسی به تنظیمات این قسمت توسط دستور Appcmd:

```
appcmd.exe set config -section:system.ftpServer/log /centralLogFileMode:"Central" /commit:apphost
appcmd.exe set config -section:system.ftpServer/log /centralLogFile.enabled:"True" /commit:apphost
```

## دسترسی به تنظیمات این قسمت توسط دات نت:

```
using System;
using System.Text;
using Microsoft.Web.Administration;

internal static class Sample
{
    private static void Main()
    {
        using (ServerManager serverManager = new ServerManager())
        {
            Configuration config = serverManager.GetApplicationHostConfiguration();

            ConfigurationSection logSection = config.GetSection("system.ftpServer/log");
            logSection["centralLogFileMode"] = @"Central";

            ConfigurationElement centralLogFileElement = logSection.GetChildElement("centralLogFile");
            centralLogFileElement["enabled"] = true;

            serverManager.CommitChanges();
        }
    }
}
```

### دسترسی به تنظیمات این قسمت توسط Javascript:

```
var adminManager = new ActiveXObject('Microsoft.ApplicationHost.WritableAdminManager');
adminManager.CommitPath = "MACHINE/WEBROOT/APPHOST";

var logSection = adminManager.GetAdminSection("system.ftpServer/log", "MACHINE/WEBROOT/APPHOST");
logSection.Properties.Item("centralLogFileMode").Value = "Central";

var centralLogFileElement = logSection.ChildElements.Item("centralLogFile");
centralLogFileElement.Properties.Item("enabled").Value = true;

adminManager.CommitChanges();
```

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 (شبیه 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

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

عنوان:	لیست تازه‌های IIS 7.5
نویسنده:	وحید نصیری
تاریخ:	۱۳۸۸/۰۱/۱۳ ۱۹:۴۴:۰۰
آدرس:	<a href="http://www.dotnettips.info">www.dotnettips.info</a>
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.