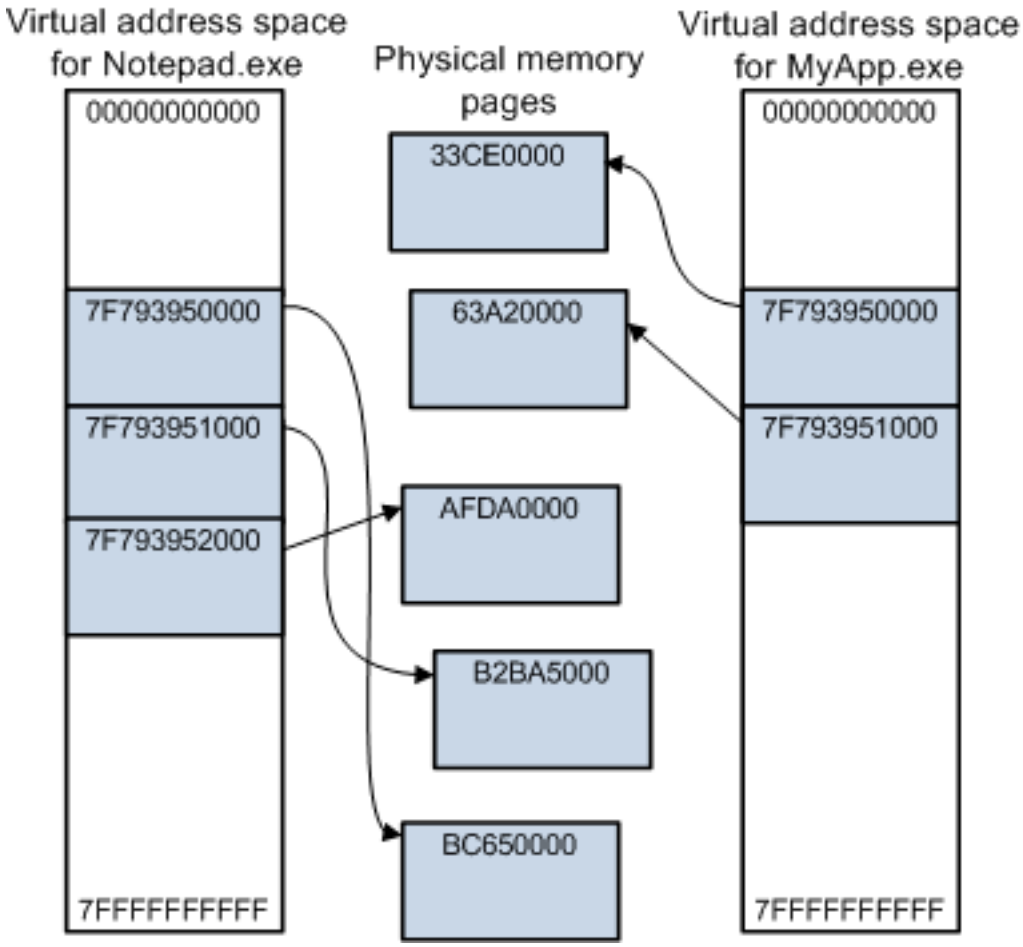
عنوان: **آشنایی با ذخیره سازی در حافظه**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۰۹/۳۰ ۱۰:۲۵
آدرس: www.dotnettips.info
گروه‌ها: IIS, Memory Manegement, Operating System

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 کیلوبیی) 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 می باشد 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 شده اند.

تفاوت user mode و kernel mode

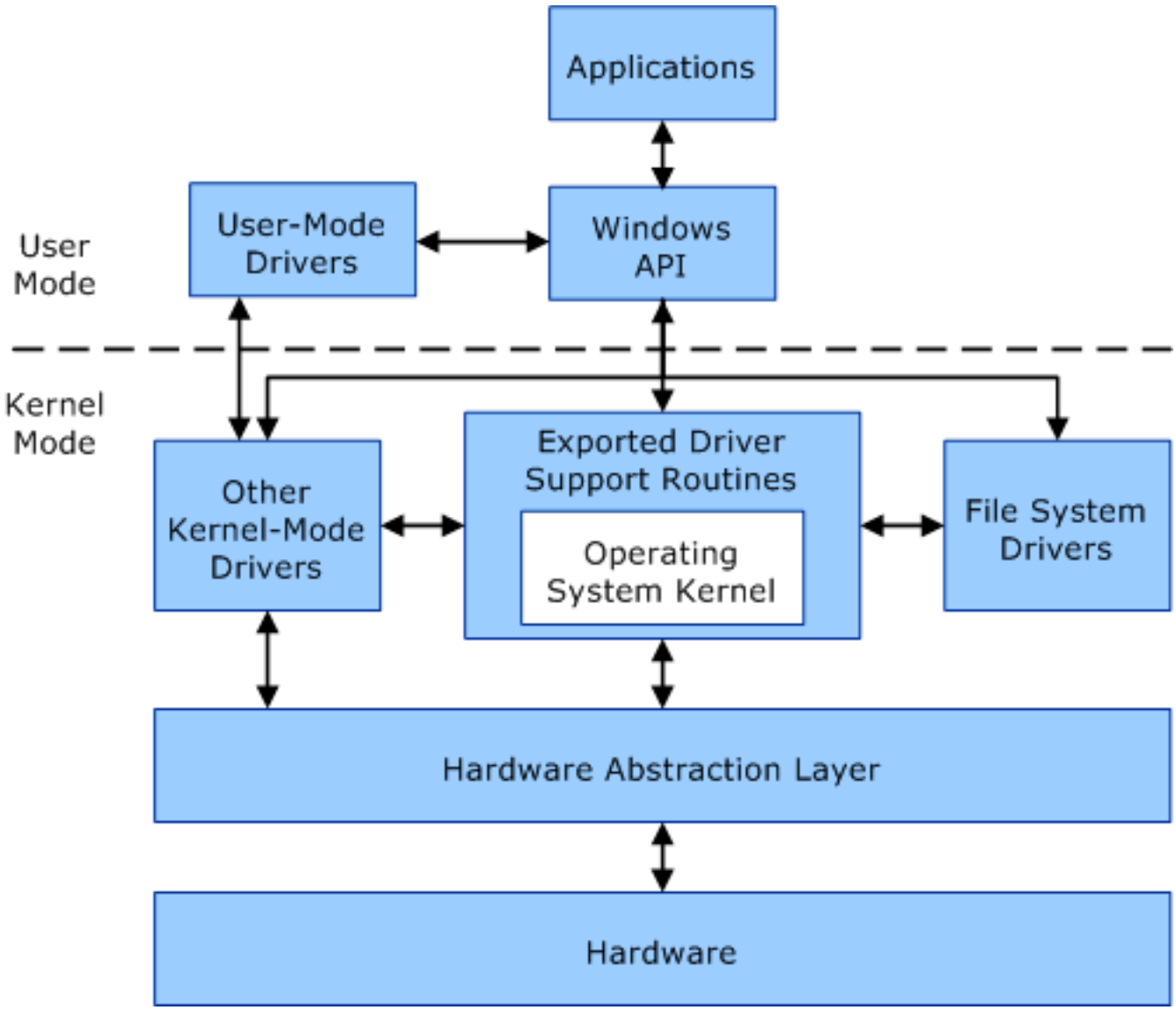
هر پردازش در سیستم بر اساس user mode مد کاربر یا kernel mode مد کرنل اجرا میشود. پردازش‌ها بر اساس هر نوع کد بین این دو بخش سوییچ می‌کنند. اپلیکیشن‌ها بر اساس مد کاربر و هسته سیستم عامل و اکثر درایورها بر اساس مد کرنل کار می‌کنند؛ ولی تعدادی از آن‌ها هم در مد کاربر.

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 محدودیت هایی هم دارد، چرا که بعضی از آن‌ها توسط سیستم عامل رزرو شده اند و برنامه نمی‌تواند به آن قسمت‌ها دسترسی داشته باشد و این باعث می‌شود که داده‌های برنامه از خسارت و آسیب دیدن حفظ شوند.

تمام برنامه هایی 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 تصادفا در یک آدرس مجازی اشتباه که میتواند متعلق به سیستم عامل یا یک درایور دیگر باشد بنویسد. یعنی اگر یک درایور کرنل کراش کند کل سیستم عامل کرش میکند.

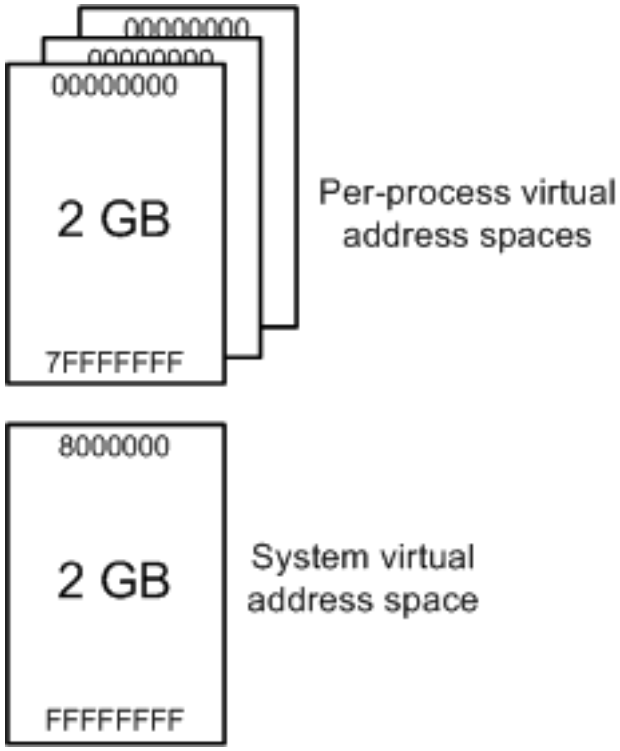
تصویر زیر به خوبی ارتباط بین مد کاربری و مد کرنل را نشان می‌دهد:



فضای کاربری و فضای سیستمی User space and system space

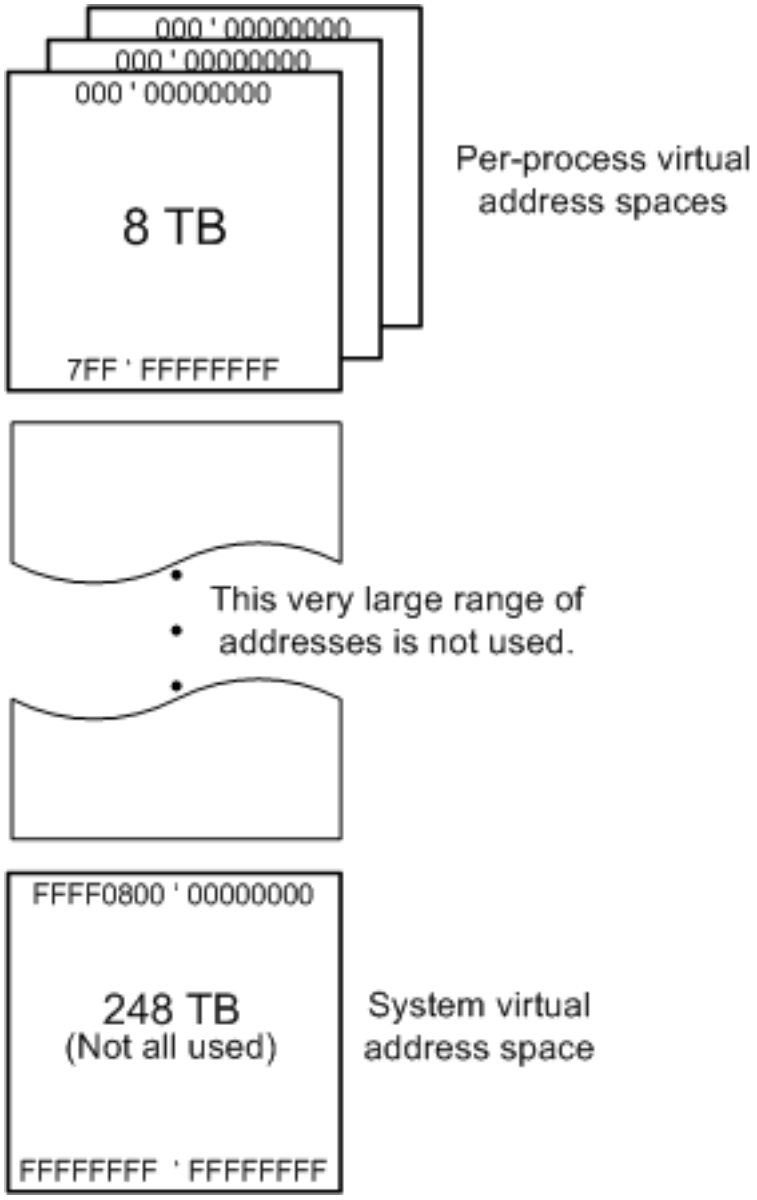
گفتیم بسیاری از پروسه‌ها در حالت user mode و پروسه‌های هسته سیستم عامل و درایورها در حالت kernel mode اجرا می‌شوند. هر پروسه مد کاربر از فضای آدرس دهی مجازی خودش استفاده می‌کند ولی در حالت کرنل همه از یک فضای آدرس دهی استفاده می‌کنند که به آن فضای سیستمی می‌گویند و برای مد کاربری می‌گویند فضای کاربری.

در سیستم‌های 32 بیتی نهایتا 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 `bcedit` 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.

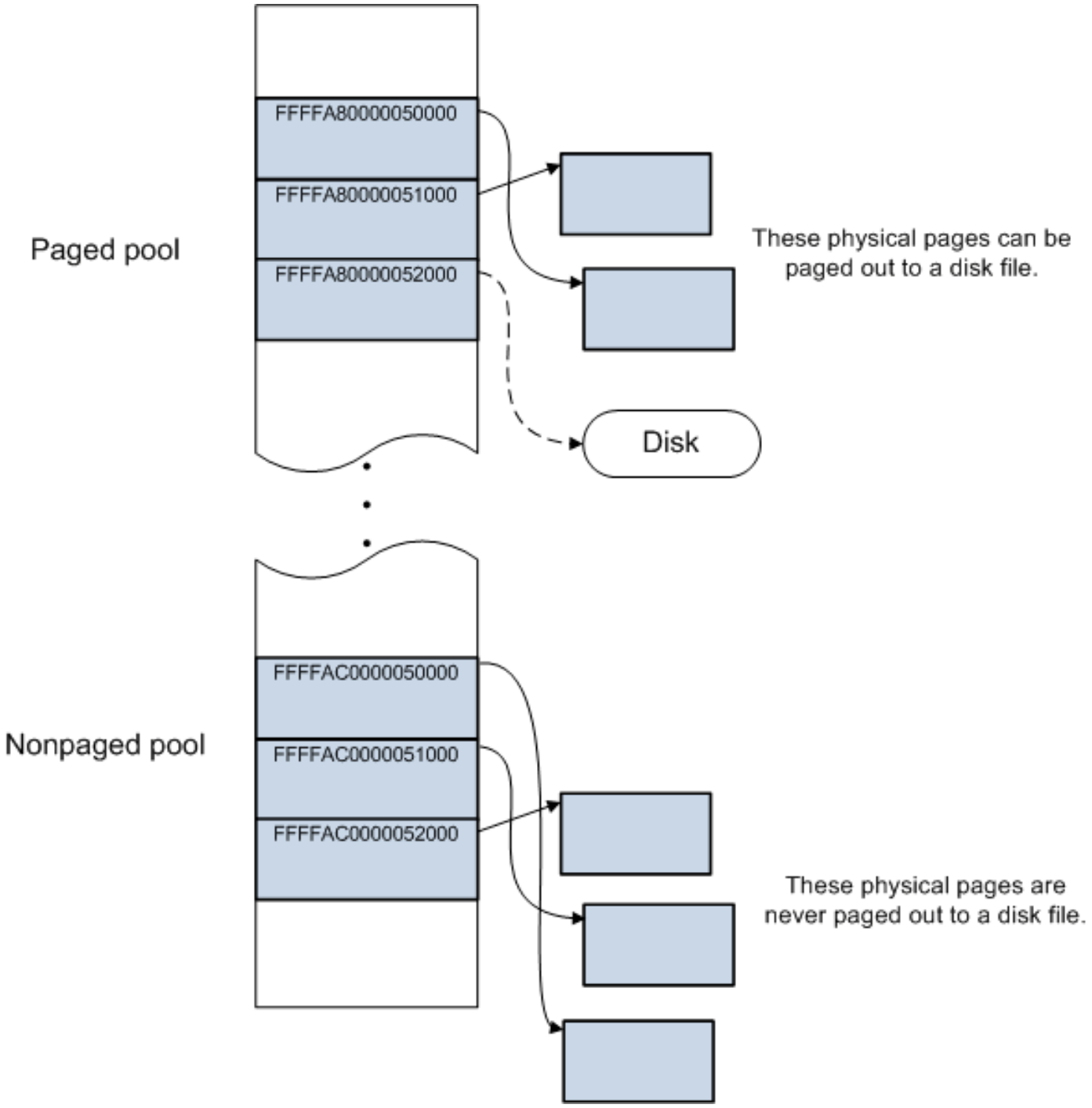


کدهایی که در user mode اجرا می‌شوند فقط به فضای کاربری دسترسی دارند و دسترسی آن‌ها به فضای سیستمی به منظور جلوگیری از تخریب داده ممکن نیست. ولی در حالت کرنل می‌توان به دو فضای سیستمی و کاربری دسترسی داشت. درایورهایی که در مدکرنل نوشته شده اند باید تمام دقت خود را در زمینه نوشتن و خواندن از فضای سیستمی 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

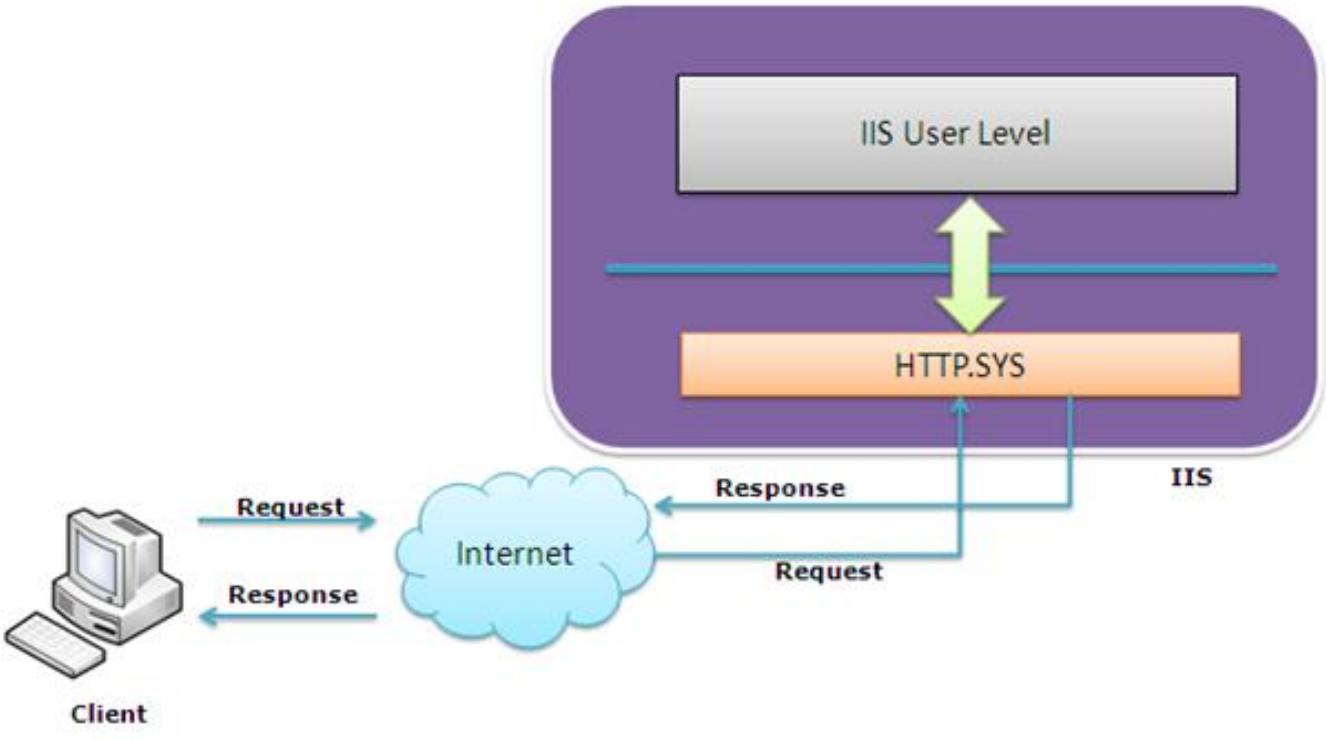
در فضای کاربری تمام صفحات در صورت نیاز توانایی انتقال به دیسک سخت را دارند ولی در فضای سیستمی همه بدین صورت نیستند. فضای سیستمی دو ناحیه حافظه تخصیصی پویا دارد که به نام‌های *paged pool* و *nonpaged pool* شناخته می‌شوند.
در سیستم‌های 32 بیتی *Pagedpool* توانایی 128 گیگ فضای آدرس دهی مجازی را از آدرس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

0xFFFFAC1F'FFFFFFFF و در سیستم‌های 64 بیتی توانایی 128 گیگ فضای آدرس دهی مجازی را از 0xFFFFA800'00000000 تا 0xFFFFA81F'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.



عنوان: **آشنایی با ساختار IIS قسمت اول**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۱ ۱:۲۰
آدرس: www.dotnettips.info
گروه‌ها: IIS, Web Capacity Analysis Tool -WCAT

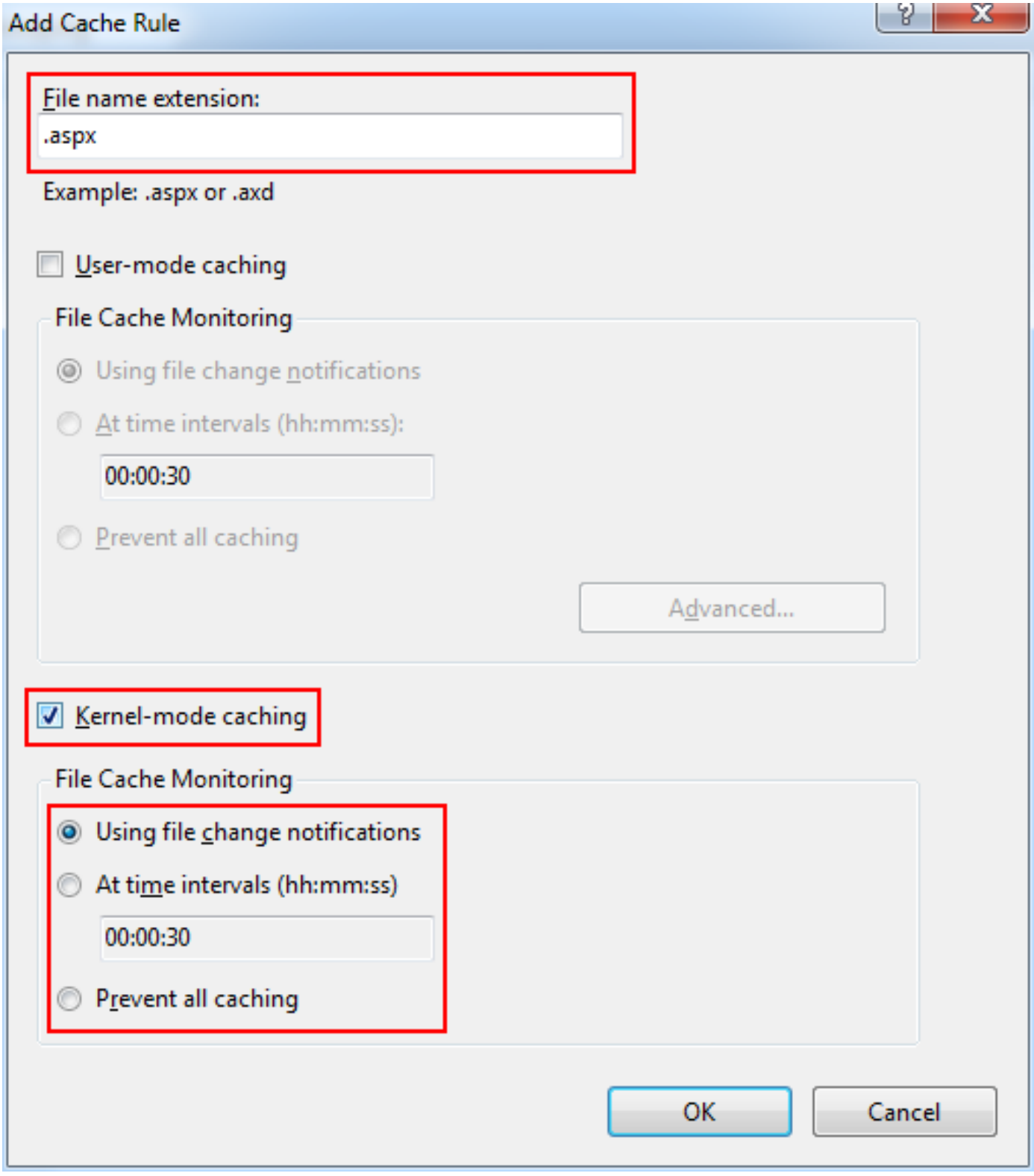
در مقاله قبل در مورد نحوه ذخیره سازی در حافظه نوشتیم و به user mode و kernel mode اشاراتی کردیم که می‌توانید به آن رجوع کنید.
در این سری مقالات قصد داریم به بررسی اجزا و روند کاری موجود در IIS بپردازیم که چگونه IIS کار می‌کند و شامل چه بخش هایی می‌شود. مطمئنا آشنایی با این بخش‌ها در روند شناسایی رفتارهای 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 ازآن‌ها 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.



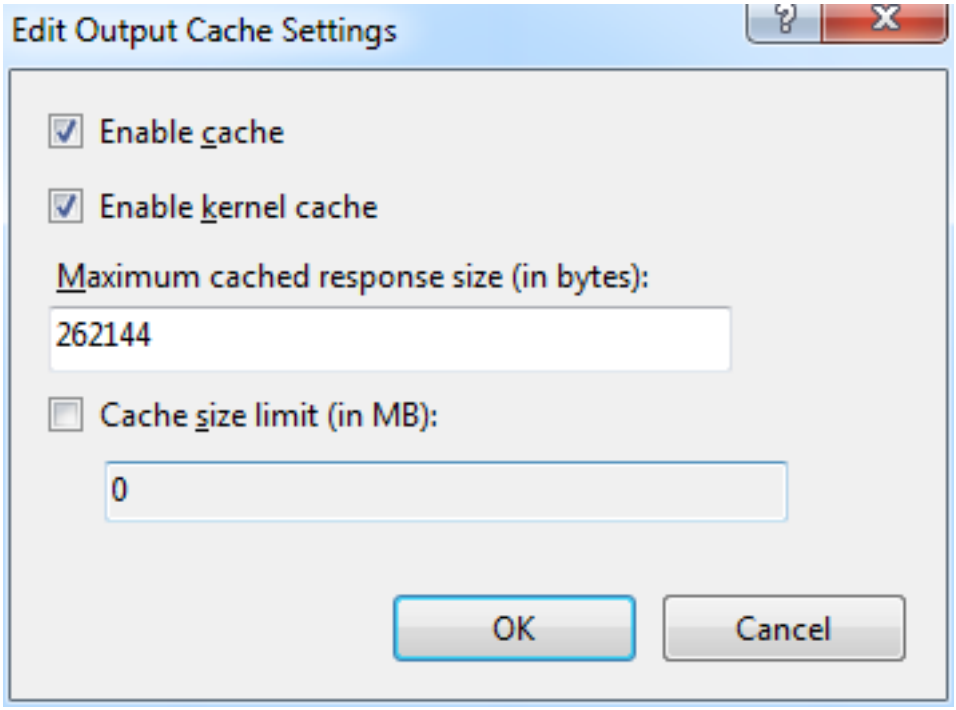
Http.sys مزایای زیر را به همراه دارد:

صف درخواست مد کرنل: به خاطر اینکه کرنل مستقیما درخواست‌ها را به پروسه‌های مربوطه میفرستد و اگر پروسه موجود نباشد، درخواست را در صف گذاشته تا بعدا پروسه مورد نظر آن را از صف بیرون بکشد.
برای درخواست‌ها یک پیش پردازش و همچنین اعمال فیلترهای امنیتی اعمال می‌گردد.
عملیات کش کردن تماما در محیط کرنل مد صورت می‌گیرد؛ بدون اینکه به حالت یوزرمد سوییچ 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 مستقیما وارد عمل می‌شود و برداشته شدن واسط در سر راه، موجب افزایش عمل caching می‌شود. همچنین دسترسی به کش باعث می‌شود که مستقیما پاسخ از کش به کاربر برسد و توابع پردازشی در حافظه بارگذاری نشوند. البته این کش کردن محدودیت هایی 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 ازمکانیزم‌های 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 باشه *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 ریجستری UriMaxUriBytes قرار دارد. اطلاعات بیشتر
اندازه هدر بیشتر از اندازه تعیین شده باشد که عموما 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 (مثلا aspx) از این قوانین پیروری کنند و مکانیزم کش کردن به سه روش جلوگیری از کش کردن، کش زمان دار و کش بر اساس آخرین تغییر فایل انجام گردد.



برای تعیین مقدار سایز 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 می‌بینید 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 بگردید و از صحت پارامترهای server و url اطمینان پیدا کنید. در گام بعدی 5 پنجره خط فرمان باز کرده و در یکی از آن‌ها دستور netsh http show cachestate را بنویسید تا تمامی وروردی‌های 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 از 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 را ارسال کنید باید کش کرنل دارای ورودی باشد.

برای تغییرات در سطح http.sys می‌توانید از ریجستری کمک بگیرید. در اینجا تعداد زیادی از تنظیمات ذخیره شده در ریجستری برای http.sys لیست شده است.

عنوان: **آشنایی با ساختار IIS قسمت دوم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۲ ۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: IIS, Performance, Application pool

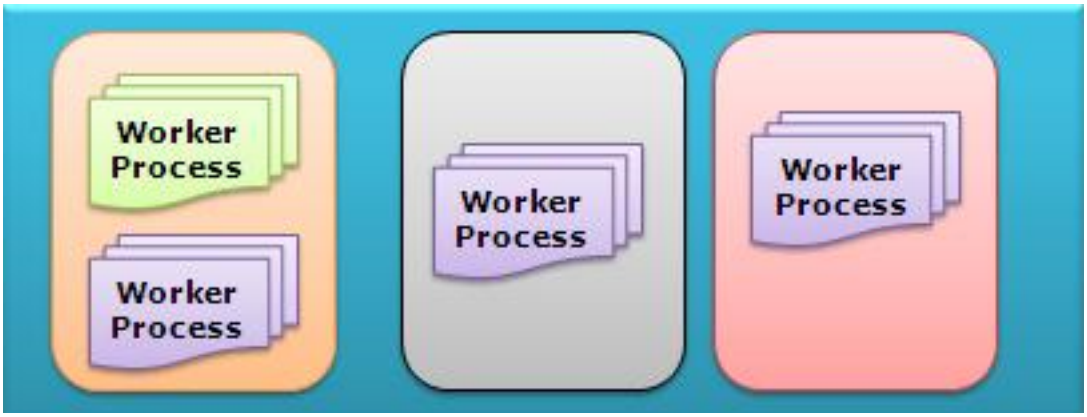
در قسمت قبلی گفتیم که IIS از تعدادی کامپوننت تشکیل شده است و به یکی از آن‌ها به نام Http.sys پرداختیم. در این قسمت قصد داریم به WWW Services بپردازیم.
اجازه بدهید قبل از هر چیزی به دو مفهوم اصلی در IIS بپردزیم :

1. Worker Process

2. Application Pool

پروسه‌های کارگر w3wp.exe وظیفه‌ی اجرای برنامه‌های asp.net را در IIS ، به عهده دارند. این پروسه‌ها 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 پروسه‌ها قلب برنامه‌های ASP.Net بر روی IIS هستند .

Application Pool:این پول‌ها در واقع ظرفی یا در برگیرنده ای برای پروسه‌های کارگر به حساب می‌آیند. این پول‌ها پروسه‌های کارگر را از هم جدا و دسته بندی می‌کنند تا قابلیت اعتماد، امنیت و در دسترس بودن بدهند. موقعی که یک پروسه یا حتی یک پول دچار مشکل می‌شود، این اطمینان داده می‌شود که تاثیری بر دیگر پول‌ها یا پروسه‌های 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 پروسه کارگر web garden می گویند.



World Wide WebPublishing 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 پروسه 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 کارآیی وب سایت‌ها و کش IIS نظارت می‌کند و البته یک شمارنده کارآیی performance counter هم ایجاد می‌کند. کار شمارنده کارآیی 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 کارآیی 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 نویس‌ها 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.diagnostic 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.

عنوان: **آشنایی با ساختار IIS قسمت سوم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۳ ۱:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, IHttpModule

همانطور که در مطلب قبلی گفتم، در این مطلب قرار است به WAS بپردازیم؛ در دنباله متن قبلی گفتیم که دومین وظیفه WWW Service این است: موقعی‌که 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 درخواست هایی 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 lsitener adapterها ارتباط بین WAS و listenerهای مختلف را در IIS، برقرار می‌کنند.
آداپتورها اطلاعات لازم را از WAS می‌گیرند و به listenerهای مربوطه انتقال می‌دهند تا listenerها بر اساس آن تنظیمات یا پیکربرندی‌ها، 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 درخواست هایی که از طریق پروتوکل 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 تغییری کند، 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 درخواست هاست، 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 پروتوکل‌های Http و net.tcp بهره بگیرید.

ماژول‌ها در IIS

قبلا مقاله ای در مورد moduleها با نام " کمی در مورد httpmodulehaها " 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 دقیقا 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 پروتوکل‌ها 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 پروتوکل 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 ، توع 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 به منظور ایجاد خروجی response	Inetsrv\Cgi.dll
DefaultDocumentModule	تلاش برای ساخت یک سند پیش فرض برای درخواست هایی که دایرکتوری والد ارسال می‌شود	Inetsrv\Defdoc.dll
DirectoryListingModule	لیست کردن محتوای یک دایرکتوری	Inetsrv\dirlist.dll
IsapiModule	میزبانی فایل های ISAPI	Inetsrv\Isapi.dll
IsapiFilterModule	پشتیبانی از فیلتر های ISAPI	Inetsrv\Filter.dll
ServerSideIncludeModule	پردازش کدهای include شده سمت سرور	Inetsrv\Iis_ssi.dll
StaticFileModule	ارائه فایل‌های ایستا	Inetsrv\Static.dll
FastCgiModule	پشتبانی 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 عیب یابی و لاگ کردن

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

ماژول‌های مدیریتی و نظارتی بر کل ماژول‌ها

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

از IIS6 به بعد در حالت integrated و ماقبل، در حالت کلاسیک می‌باشند. اگر مقاله ماژول ها 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 می‌آورند 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 کاملا آشنا باشید؛ برای تایید هویت کاربر	System.Web.Security.FormsAuthentica

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

عنوان: **آشنایی با ساختار IIS قسمت چهارم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۴ ۱:۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS

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.

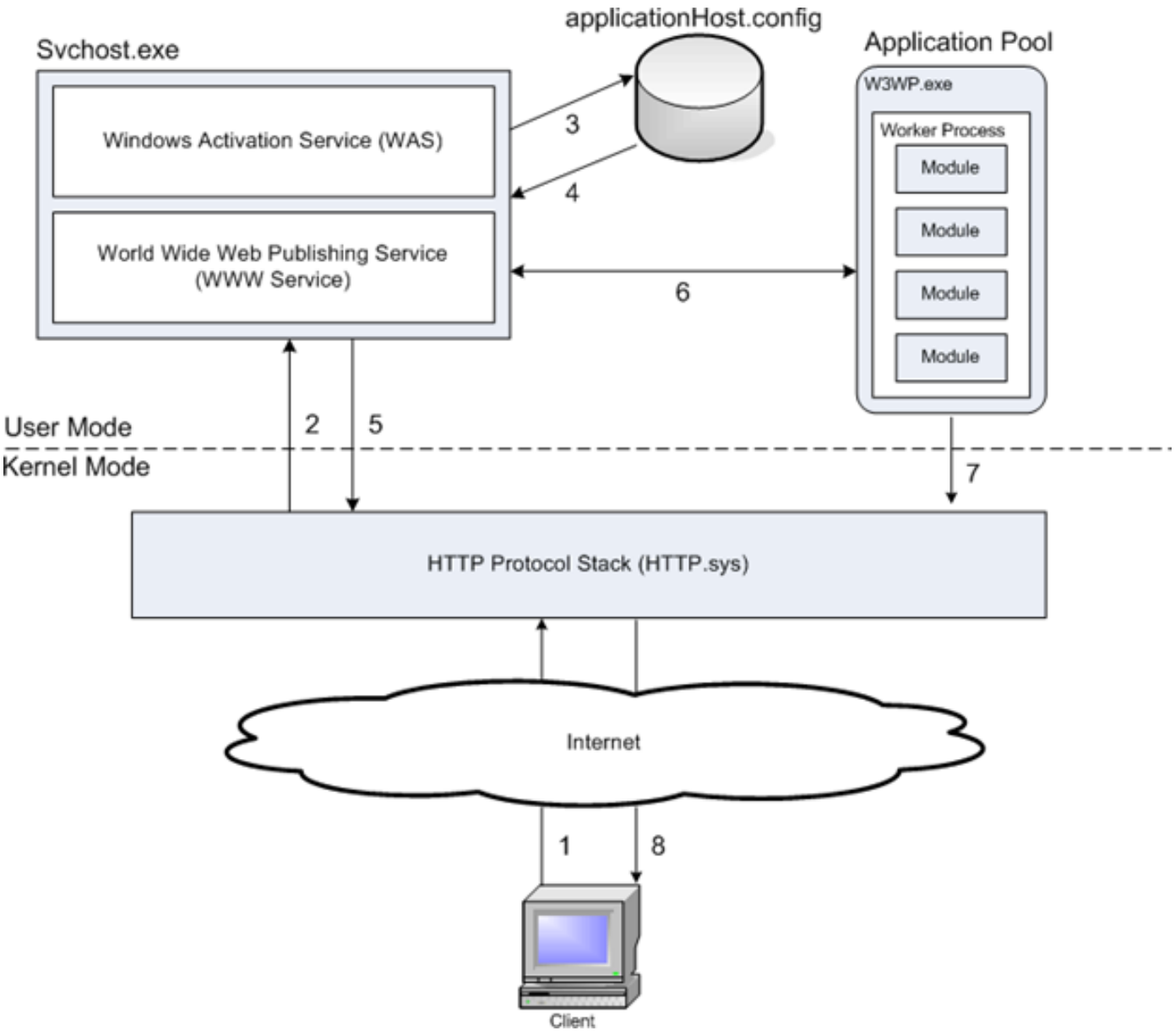
WWW Service که یک آداپتور برای Http.sys هست، اطلاعات را از WAS دریافت می‌کند. این اطلاعات شامل پیکربندی application pool و سایت می‌باشد.

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.

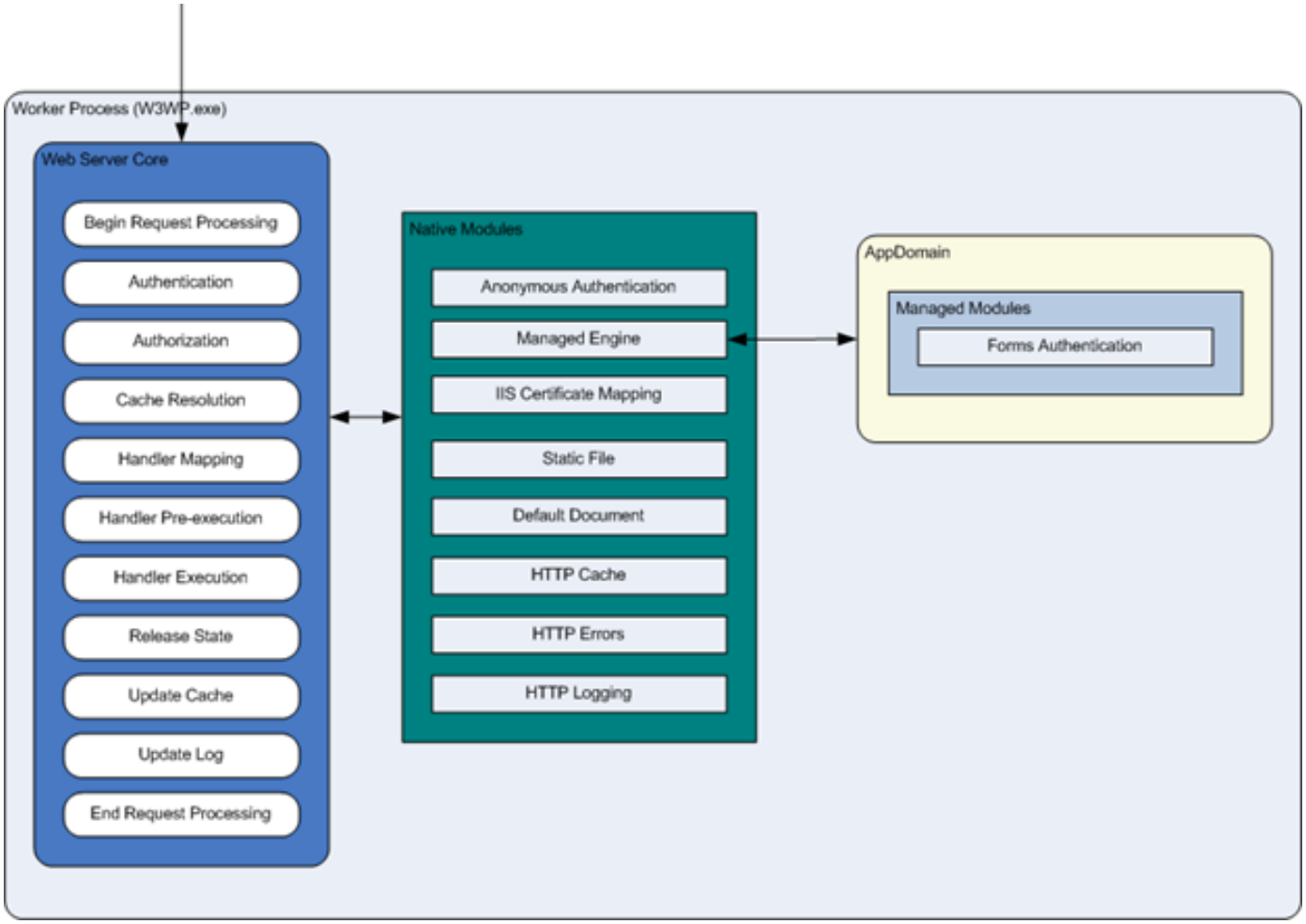
Http.sys نتیجه را دریافت و برای کلاینت می‌فرستد.



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

در پروسه‌های کارگر، یک درخواست از مراحل لیست شده ای 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 managedEngine native 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 گشترش

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 وروردی‌های 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 کارآیی 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 partyها را به توسعه دهنده می‌دهد.

Extensibility

با توجه به موارد بالا، خصوصی سازی باعث گسترش امکانات IIS می‌شود که می‌تواند به دلایل زیر اتفاق بیفتد:
قدرت بخشی به برنامه‌های وب. امکانات و قدرتی که می‌تواند در این حالت به برنامه‌های در حال اجرا داد به مراتب بیشتر از استفاده از لایه‌های داخلی خود برنامه هست. برای اینکار شما می‌توانید کدهای خود را با ASP.Net نوشته یا از کدهای native چون C++ استفاده کنید.
تجربه‌ای از توسعه پذیری ساده‌تر و راحت تر
استفاده از قدرت و تمامی امکانات را به شما می‌دهد و می‌توانید تمام دستورات را برای همه منابع حتی فایل‌های ایستا، ASP ، CGI و دیگر منابع اجرا کنید.

ASP.NET Integration

تمامی موارد گفته شده بالا در این گزینه خلاصه می‌شود : محیط ASP.Net Integration به شما امکان استفاده از تمامی امکانات و منابع را به طور کامل می‌دهد. دانلود ماژول‌های مدیریت شده
دانلود ماژول‌های native

عنوان: **آشنایی با ساختار IIS قسمت پنجم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۵ ۹:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS

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 برنخورید، IIS را به صورت کامل یعنی full feature نصب نمایید. از بخش control panel>programs & features>Turn Windows features on or off 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 اپلیکشن‌ها باشد باید از طریق فایل کانفیگ سایت از مسیر تگ‌های <system.webserver><modules> 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 framwork. نیاز دارد، این ویژگی ذکر می‌شود. در صورتی که ماژولی قصد اجرای بر روی فریم ورک اشتباهی داشته باشد سبب خطا خواهد شد.

Mana gedHandler

با معرفی IIS7 ما با یک مدل توسعه پذیر روبرو شدیم و می‌توانستیم ماژول‌ها و هندلرهای خود را بنویسیم و مستقیما 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 و... شامل این ماژول نخواهند شد. واضح‌ترین مثال در این زمینه Forms Authentication می‌باشد و موقعی 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.

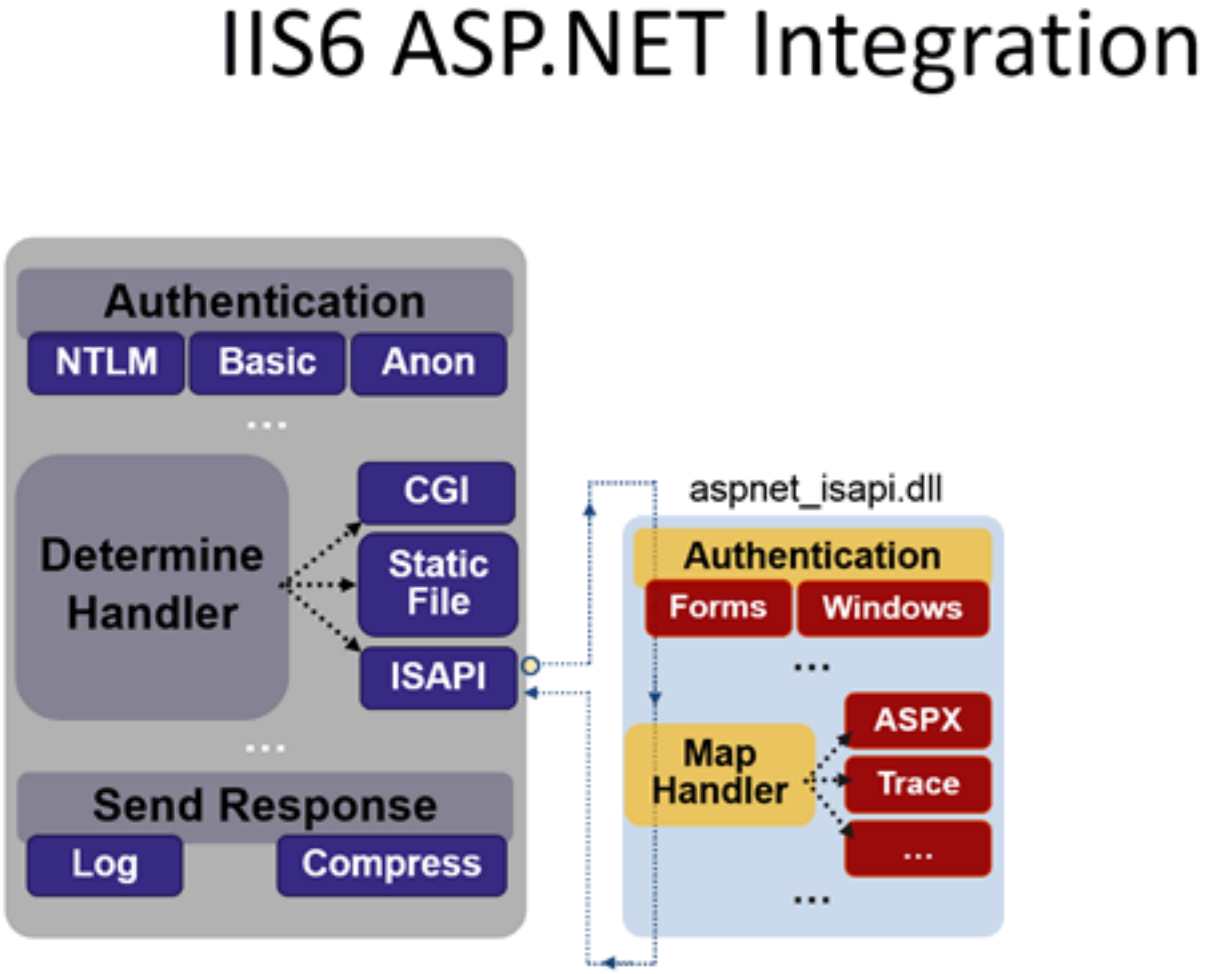
در نسخه‌های قبلی IIS یک روش قدیمی برای کدهای مدیریت شده وجود داشت که از طریق اینترفیسی به نام ISAPI صورت می‌گرفت. در این حالت ASPNET_ISAPI.DLL مسئول این کار بود و اگر هنوز هم میخواهید از این dll در نسخه‌های جدیدتر IIS کمک بگیرید باید به جای معرفی integration mode ، classic mode را معرفی کنید.

با برابر کردن precondtion به مقدار "intgretedmode" 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.

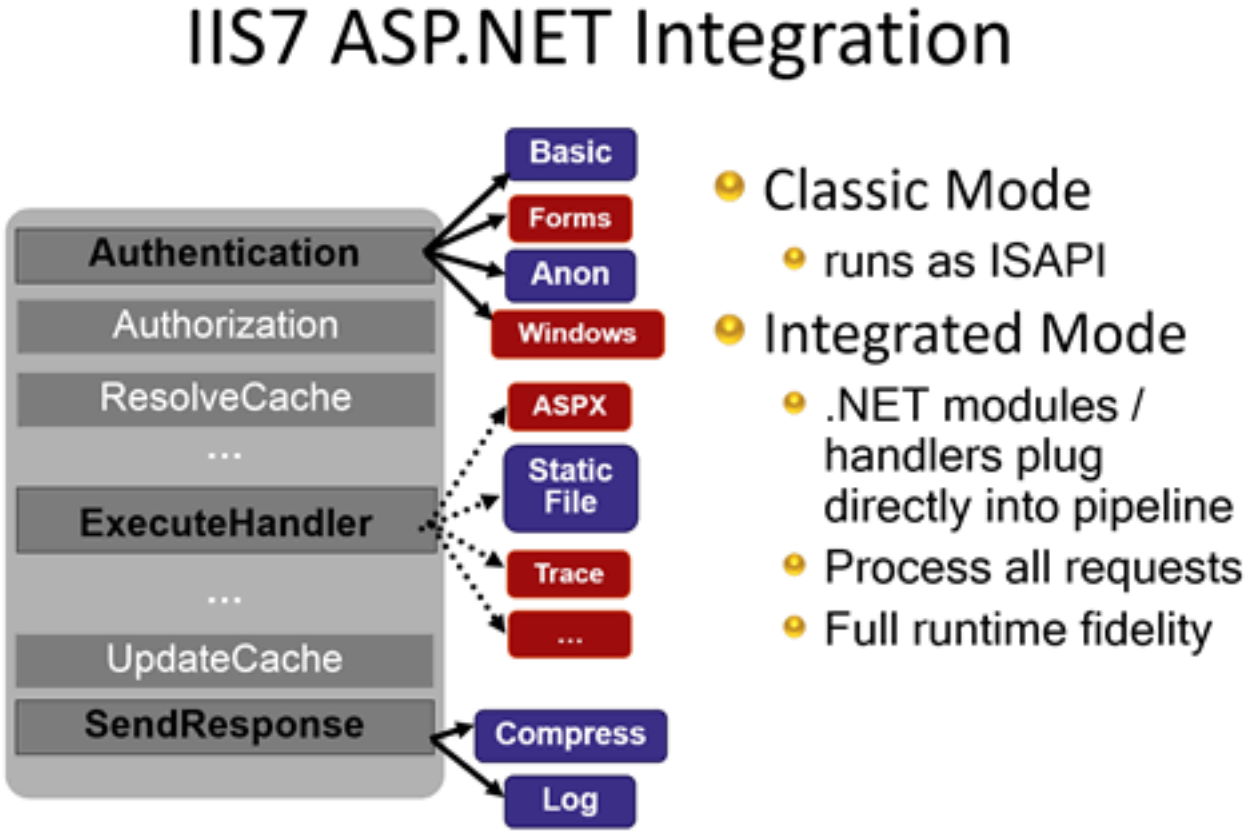
تفاوت بین دو روش کلاسیک و مجتمع integrated بر سر این هست که در روش جدید، ماژول شما به عنوان یک پلاگین برای IIS

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؛ مثلا 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 می‌داند، 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.



شکل زیر هم نمونه ای حالت مجتمع هست:



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 خصوصت precondition استفاده می‌کنند به شرح زیر هستند:

`ISAPI filters`

`globalModules`

`handlers`

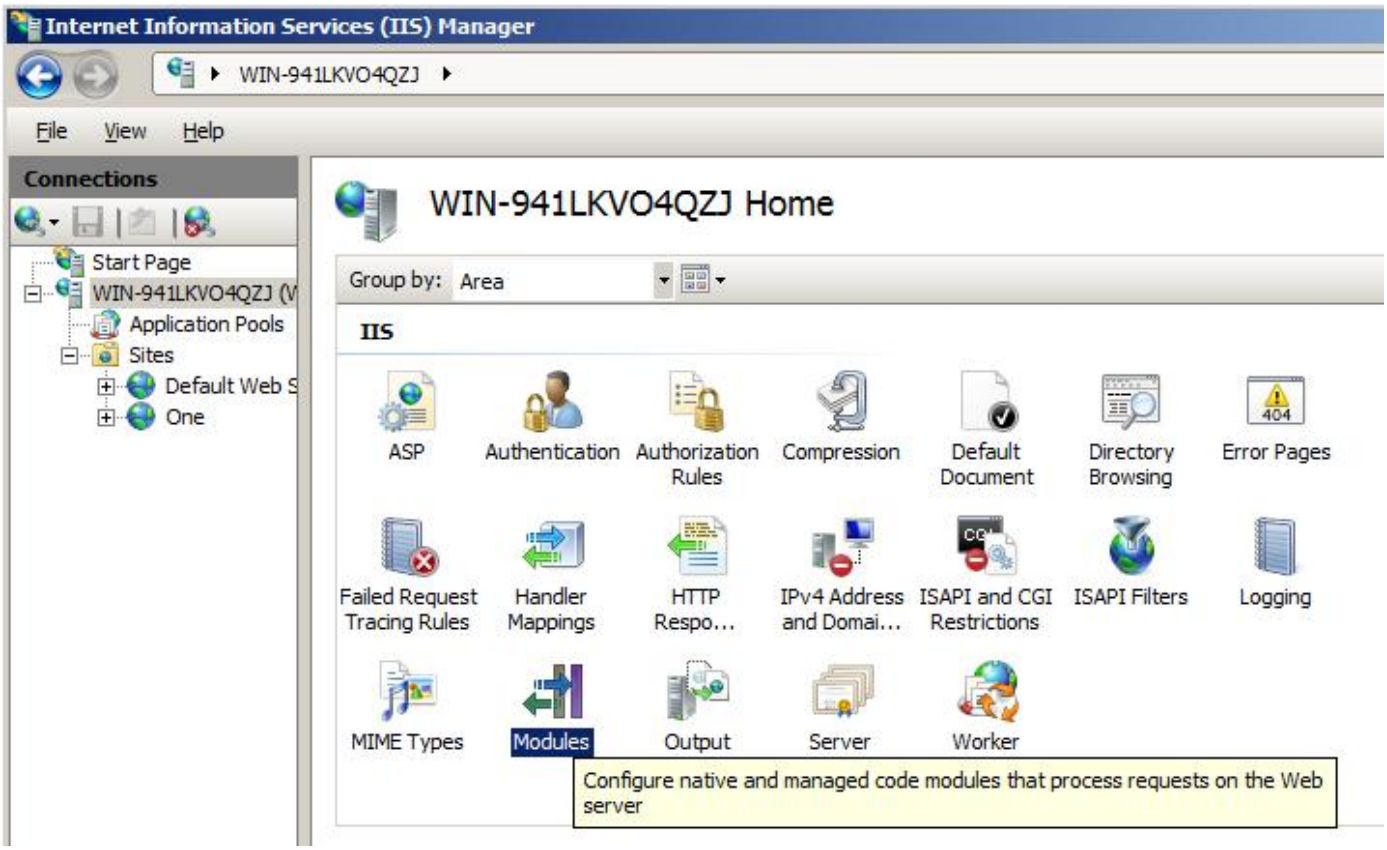
`modules`

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.

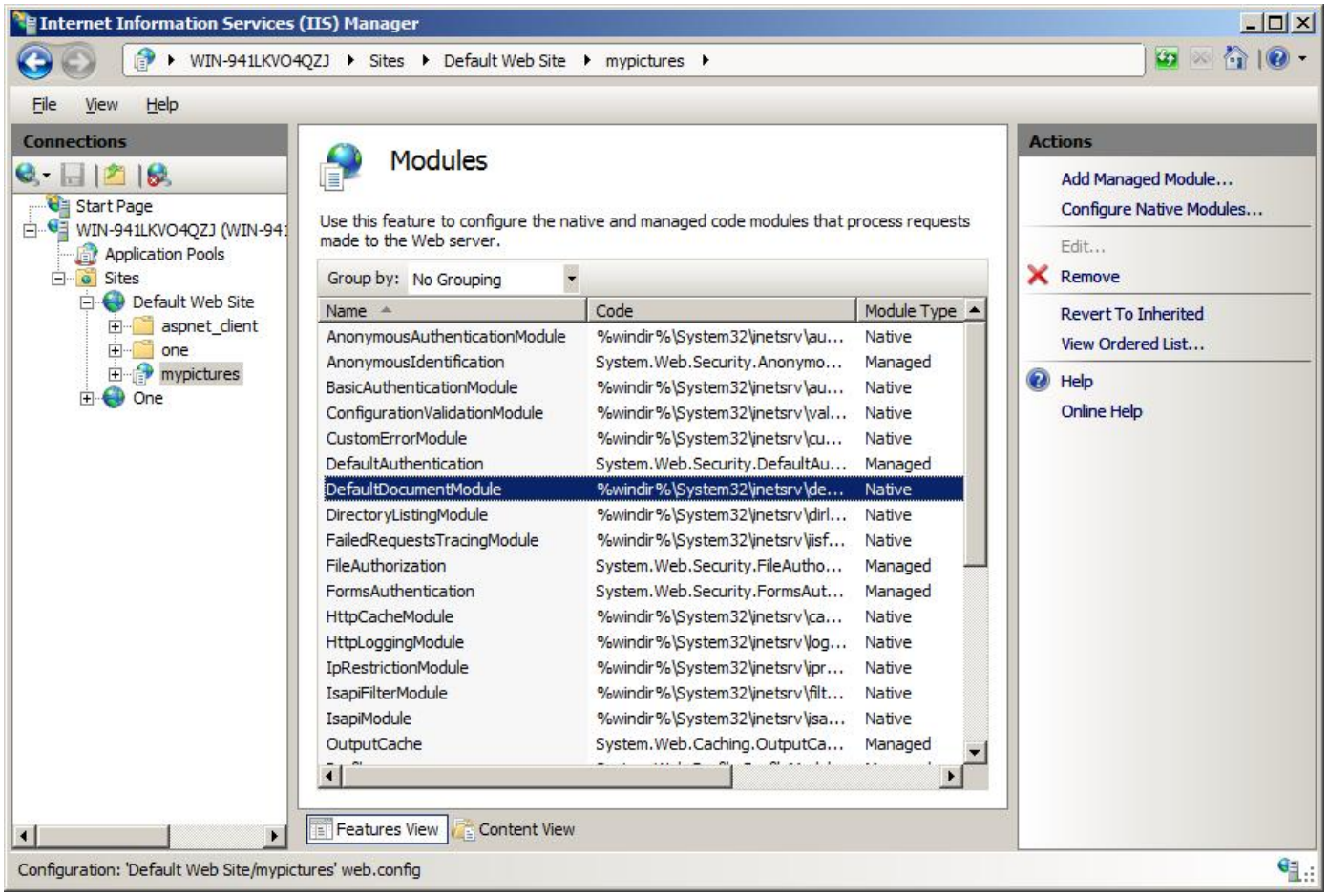
عنوان: **آشنایی با ساختار IIS قسمت ششم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۷ ۰:۴۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS

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]
```

سوپیچ aap.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 سوییچ 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 /?
```

عنوان: **آشنایی با ساختار IIS قسمت هفتم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۸ ۱۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, Security

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 هندلِ 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 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 مثلا 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 :	مشخصا 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 htttp.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 احتمالا بر کارآیی و استفاده از منابع هم اثر می‌گذارد.
نام ماژول :	**DynamicCompressionModule**
توضیح :	پیاده سازی in-memory compression در محتوای پویا
تگ قابل پیکربندی :	system.webServer/httpCompression and system.webServer/urlCompression.
وابستگی :	وابستگی ندارد چرا که به طور پیش فرض غیرفعال است.

نام ماژول :	StaticCompressionModule
توضیح :	پیادسازی 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. مثلا 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

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

نام ماژول :	**StaticFileModule**
توضیح :	فایل‌های ایستا را به همراه پسوند ارسال می‌کند. مثل jpg,html و نوع محتوا را بر اساس staticContent/mimeMap پیکربندی می‌کند.
تگ قابل پیکربندی :	system.webServer/staticContent
وابستگی :	ندارد.
اثرات حذف آن :	فایل‌های ایستا دیگر ارائه نشده و به جای آن خطای 404 بازگشت داده می‌شود.
نام ماژول :	**AnonymousAuthenticationModule**
توضیح :	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.
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 می‌باشد و در صورت نبودن هیچ سیستم شناسایی وجود نداشته و در نبود شیء httpuser سیستم خطای 401.2 را تولید می‌کند.

نام ماژول :	**CertificateMappingAuthenticationModule**
توضیح :	مجوز SSL را به Active Directory نگاشت می‌کند.
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 :	درخواست‌ها، نرمال رسیدگی میشوند انگار SSL وجود ندارد.
نام ماژول :	**BasicAuthenticationModule**
توضیح :	پیاده سازی پایه‌ای و روتین شناسایی کاربران بر اساس آن چیزی که در استاندارد زیر آمده است RFC 2617 .
تگ قابل پیکربندی :	system.webServer/security/authentication/basicAuthentication
وابستگی :	None.
اثرات حذف آن :	حداقل باید یک سیستم امنیتی برای شناساسایی یا authenticate وجود داشته باشد. httpuser یک ساختار

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

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

نام ماژول :	**IISCertificateMappingAuthenticationModule**
توضیح :	پیاده سازی نگاشت مجوزهای IIS، نگهداری و ذخیره اطلاعات همه نگاشت‌ها 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 می‌باشد و در صورت نبود، هیچ سیستم شناسایی یافت نشده و نبود شیء httpuser در سیستم، خطای 401.2 را تولید می‌کند.
نام ماژول :	**UrlAuthorizationModule**
توضیح :	پیاده سازی authorization بر اساس قوانین پیکربندی شده
تگ قابل پیکربندی :	system.webServer/security/authorization
وابستگی :	ندارد.
اثرات حذف آن :	محتواهای محافظت شده توسط authorization دیگر محافظت نمی‌شوند.

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

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

نام ماژول :	**IpRestrictionModule**
توضیح :	یک سیستم تشخیص دسترسی بر اساس آی پی‌های ورژن4
تگ قابل پیکربندی :	system.webServer/security/ipSecurity
وابستگی :	IPv4 stack باید نصب شود.
اثرات حذف آن :	کلاینت هایی که IP هایشان در IPsecurity لیست شده‌اند ندید گرفته میشوند
نام ماژول :	**RequestFilteringModule**
توضیح :	پیاده سازی یک مجموعه قدرتمند از قوانین امنیتی که درخواست‌های مشکوک را پس می‌زند.
تگ قابل پیکربندی :	system.webServer/security/requestFiltering
وابستگی :	ندارد.
اثرات حذف آن :	دیگر قوانین امنیتی اجرا نخواهند شد و سبب وجود مشکلات امنیتی میشود.

نام ماژول :	**CustomLoggingModule**
توضیح :	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. Implements the ILogPlugin interface on top of IIS. ILogPlugin is a previous COM implementation that allowed customers to extend IIS logging. We do not not recommend extending IIS using this interface. Instead, customers should write a module and subscribe to the RQ_LOG_REQUEST notification.
تگ قابل پیکربندی :	system.webServer/httpLogging and system.applicationhost/sites/site/logFile/customLogPluginClsid
وابستگی :	ندارد.
اثرات حذف آن :	مسلما پلاگین‌های‌های این اینترفیس از کار می‌افتند که سیستم ODBC Logging هم جز آن است.
نام ماژول :	**CustomErrorModule**
توضیح :	پیاده سازی مدیریت خطاهای ویژه
تگ قابل پیکبرندی :	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 درحال 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 پیکبرندی :	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

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

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

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

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

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

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

عنوان: **آشنایی با ساختار IIS قسمت هشتم**
نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۰/۰۹ ۲۳:۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, IIS, IHandler

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 httpandler پرداخته‌ایم، ولی بهتر هست در این مثال کمی حالت پیشرفته‌تر آن‌را 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 برخواهید خورد 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 اسکیما را 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 هد.
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، شمای خود را بر اساس نام فایل مشخص می‌کنیم و خصوصیت overrideModeDefault، یک قفل گذار امنیتی برای تغییر محتواست. 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 مقدادهی 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" />
 </location>
                </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 معرفی شود؛ 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، در صورتیکه 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 شی 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 نجام 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 هندل 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 احیانا 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.

عنوان: **مروری بر تاریخچه محدودیت حافظه مصرفی برنامه‌های ASP.NET در IIS**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۵/۰۵ ۱۱:۲۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: IIS

زمانیکه اولین نگارش ASP.NET حدود 10 سال قبل منتشر شد، تنها سیستم عاملی که از آن پشتیبانی می‌کرد، ویندوز سرور 2000 بود، تنها پروسه‌ی اجرایی آن aspnet_wp نام داشت و تنها معماری پشتیبانی شده هم X86 بود. به پروسه‌ی 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 نشتی 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 قابلیتی 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، مثلا 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 گاها 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). متاسفانه 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. خصوصا 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 RAM 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 مجددا «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 دقیقا پیدا کنید : (+)
- ضمنا یکی از تنظیمات App pool ، مرتبط است با تعیین دقیقا 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 planهای زیادی داخل هست و خیلی مسایل دیگر (یعنی این مصرف صحیح حافظه هست نه نشتی حافظه).
در کل می‌شود برای اس کیوال سرور محدودیت حافظه گذاشت؛ در موردش قبلا مطلب نوشتم در سایت هست : (+)
ضمنا 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 حاظر 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. می‌شود آن‌را 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 می‌شود. 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 تکنولوژی‌های زیر ایجاد شده:
ASP.Net MVC 4, Entity Framework 4 , SQL CE
آن را بر روی یک ویندوز سرور 2012 نسخه دیتاسنتر نصب کردم
سرور : 2GB Ram و CPU Dual Core 1.8
غیر از این سایت هیچ سایت دیگری بر روی این سرور میزبانی نشده است.
صفحات با سرعت نسبتاً خوبی باز میشوند، اما به هر شکلی iis را تنظیم میکنم، اگر پس از 2 یا 3 دقیقه درخواستی به سمت سرور ارسال نگردد، برنامه از حافظه خارج میشود. اگر درخواستی برای مشاهده صفحه به سرور ارسال شود 15 تا 20 ثانیه طول میکشد تا دوباره کامپایل انجام شود و صفحه درخواستی نمایش یابد.
تصویر تنظیمات Application Pool

(General)	
.NET Framework Version	**v4.0**
Enable 32-Bit Applications	False
Managed Pipeline Mode	Integrated
Name	Hand
Queue Length	1000
Start Automatically	True
Start Mode	OnDemand
CPU	
Limit (1/1000 of %)	**1000**
Limit Action	NoAction
Limit Interval (minutes)	5
Processor Affinity Enabled	**True**
Processor Affinity Mask	4294967295
Processor Affinity Mask (64-bit option)	4294967295
Process Model	
Generate Process Model Event Log Entry	
Idle Time-out Reached	**False**
Identity	**LocalSystem**
Idle Time-out (minutes)	**0**
Load User Profile	**False**
Maximum Worker Processes	**4**
Ping Enabled	True
Ping Maximum Response Time (seconds)	90
Ping Period (seconds)	30
Shutdown Time Limit (seconds)	90
Startup Time Limit (seconds)	90
Process Orphaning	
Enabled	False
Executable	
Executable Parameters	
Rapid-Fail Protection	
"Service Unavailable" Response Type	HttpLevel
Enabled	True
Failure Interval (minutes)	5
Maximum Failures	5
Shutdown Executable	
Shutdown Executable Parameters	
Recycling	
Disable Overlapped Recycle	False
Disable Recycling for Configuration Change:	False
Generate Recycle Event Log Entry	
Private Memory Limit (KB)	0
Regular Time Interval (minutes)	**0**
Request Limit	0
Specific Times	**TimeSpan[] Array**
Virtual Memory Limit (KB)	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: ۱۳۹۱/۰۶/۲۱ ۱۳:۵۷

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

نویسنده: داود
تاریخ: ۱۳۹۲/۰۸/۱۴ ۱۳:۱۹

سلام؛ ما یه سایت داریم که در روز حدود 1000 تا کاربر داره. در قسمت admin، آپلود فایل هم زیاد داریم.
RAM وب سرور هم 8GB هست.
WinServer 2008 32bit - CPU: Xeon 5160 3GHz
IIS 7
تقریبا روزی یکی دوبار شاید هم هر دو سه روز یه بار نیاز به recycle داشته باشیم.
پیشنهاد می‌کنید Virtual Memory Usage و Private Memory Usage چند باشه تا کمترین نیاز رو برای recycle داشته باشیم؟

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

- 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 مهیای 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 راسا مدیریت کنه؛ یا از یک کش سرور مجزا استفاده کنید.