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.

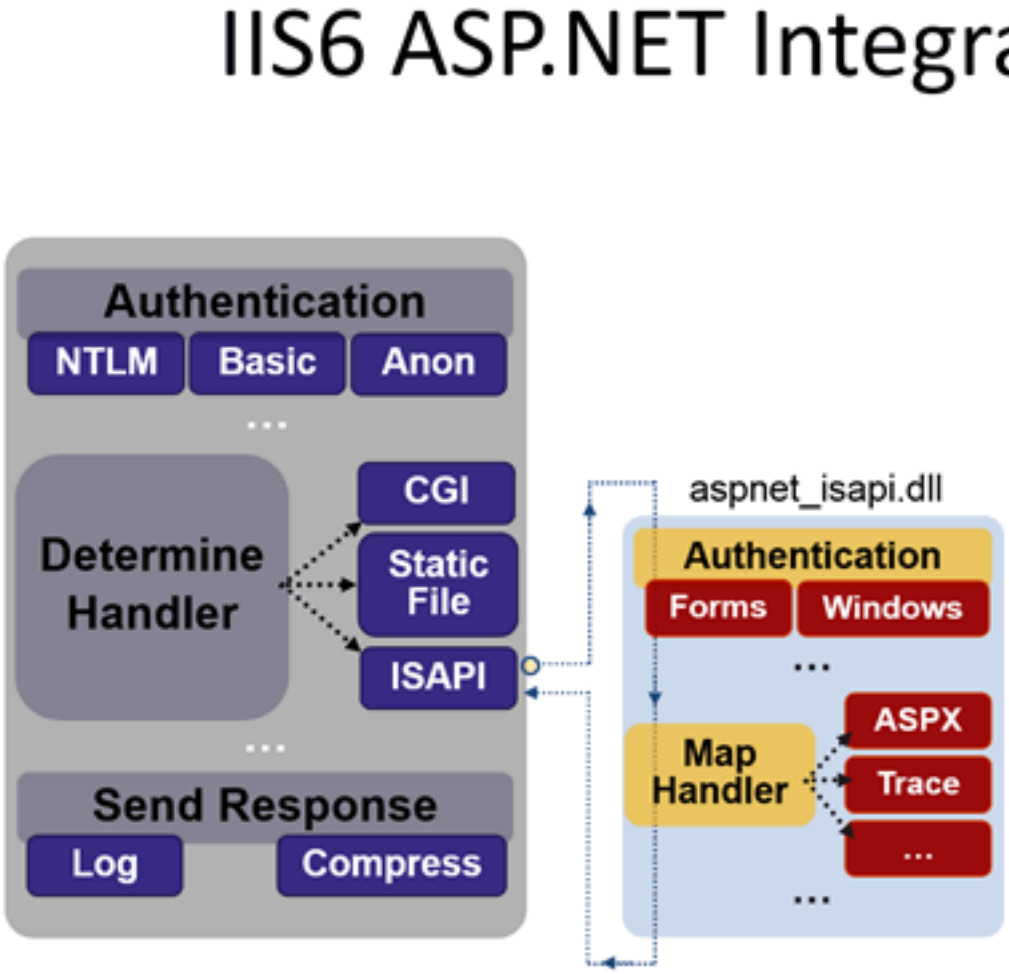
در نسخه‌های قبلی IIS یک روش قدیمی برای کدهای مدیریت شده وجود داشت که از طریق اینترفیسی به نام ISAPI صورت می‌گرفت. در این حالت ASPNET_ISAPI.DLL مسئول این کار بود و اگر هنوز هم میخواهید از این dll در نسخه‌های جدیدتر IIS کمک بگیرید باید به جای معرفی integration mode ، classic mode را معرفی کنید.

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.

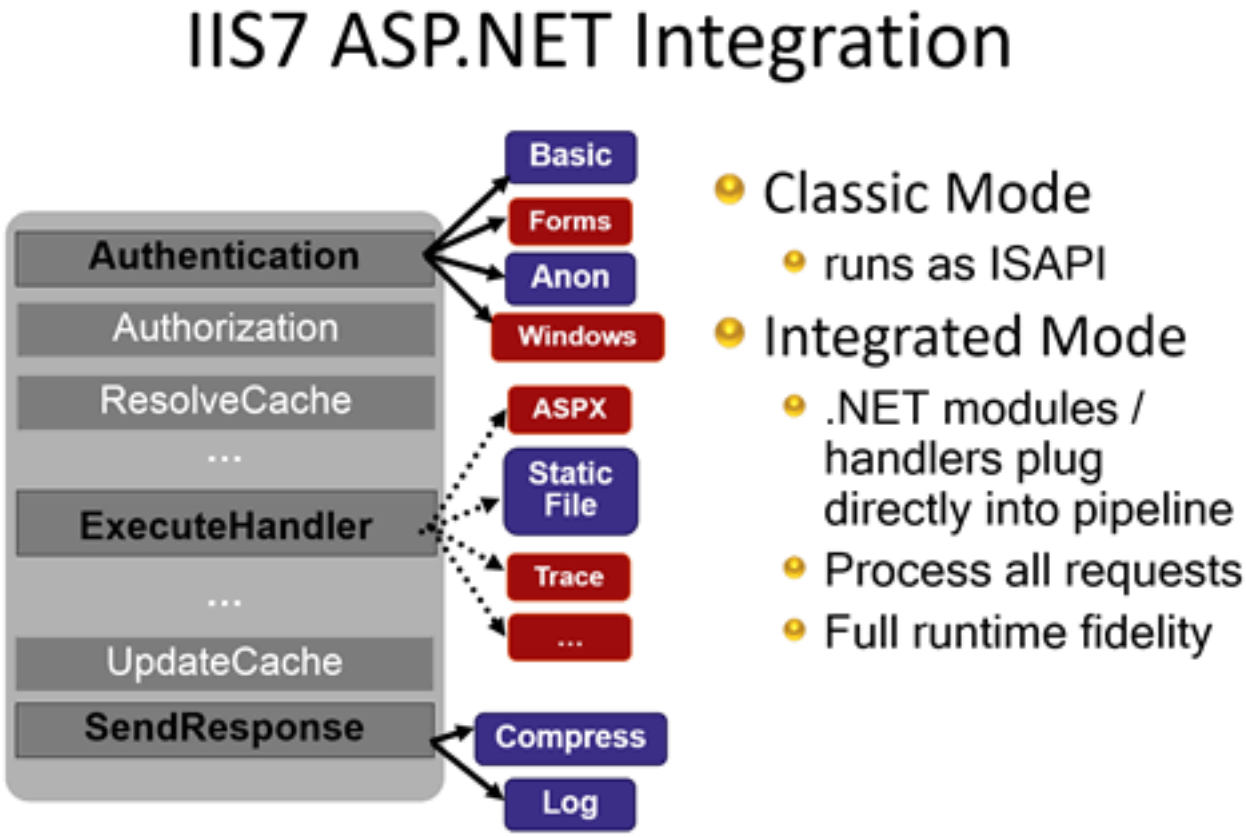
تفاوت بین دو روش کلاسیک و مجتمع integrated بر سر این هست که در روش جدید، ماژول شما به عنوان یک پلاگین برای IIS

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.