

Multicore JIT یکی از قابلیت‌های کلیدی در دات نت 4.5 می‌باشد که در واقع راه حلی برای بهبود سرعت اجرای برنامه‌های دات نت است. قبل از معرفی این قابلیت ابتدا اجازه دهید نحوه کامپایل یک برنامه دات نت را بررسی کنیم.

انواع compilation

در حالت کلی دو نوع فرآیند کامپایل داریم:

#### Explicit

در این حالت دستورات قبل از اجرای برنامه به زبان ماشین تبدیل می‌شوند. به این نوع کامپایلرها AOT یا Ahead Of Time گفته می‌شود. این نوع از کامپایلرها برای اطمینان از اینکه CPU بتواند قبل از انجام تعاملی تمام خطوط کد را تشخیص دهد، طراحی شده اند.

#### Implicit

این نوع compilation به صورت دو مرحله ای صورت می‌گیرد. در اولین قدم سورس کد توسط یک کامپایلر به یک زبان سطح میانی (IL) تبدیل می‌شود. در مرحله بعدی کد IL به دستورات زبان ماشین تبدیل می‌شوند. در دات نت فریم ورک به این کامپایلر JIT یا Just-In-Time گفته می‌شود.

در حالت دوم قابلیت جابجایی برنامه به آسانی امکان پذیر است، زیرا اولین قدم از فرآیند به اصطلاح platform agnostic می‌باشد، یعنی قابلیت اجرا بر روی گستره وسیعی از پلت فرم‌ها را دارد.

#### کامپایلر JIT

JIT بخشی از Common Language Runtime یا CLR می‌باشد. CLR در واقع وظیفه مدیریت اجرای تمام برنامه‌های دات نت را برعهده دارد.



همانطور که در تصویر فوق مشاهده می‌کنید، سورس کد توسط کامپایلر دات نت به exe و یا dll کامپایل می‌شود. کامپایلر JIT تنها متدهایی را که در زمان اجرا (runtime) فراخوانی می‌شوند را کامپایل می‌کند. در دات نت فریم ورک سه نوع JIT داریم:

#### Normal JIT Compilation

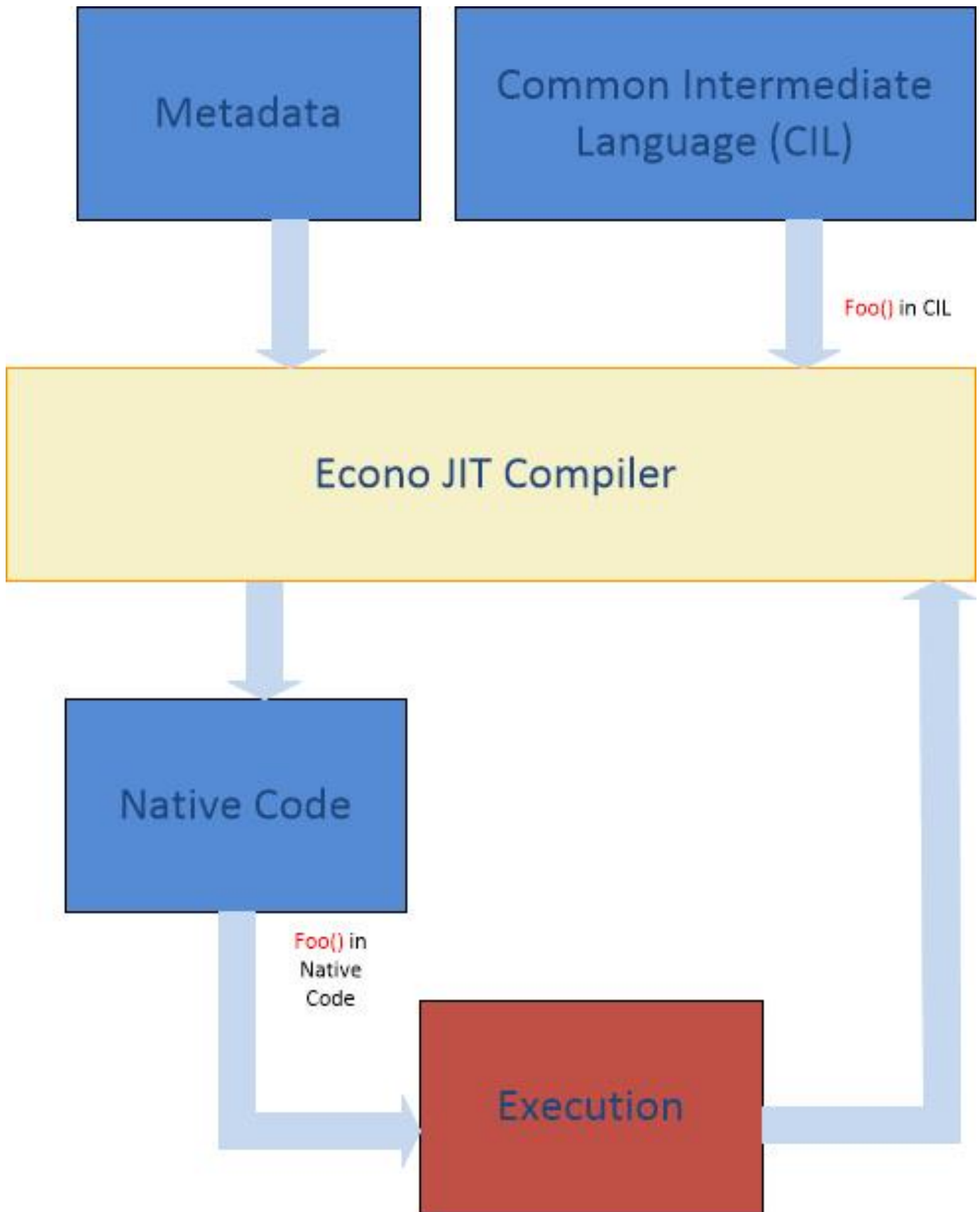
در این نوع کامپایل، متدها در زمان فراخوانی در زمان اجرا کامپایل می‌شوند. بعد از اجرا، متد داخل حافظه ذخیره می‌شود. به متدهای ذخیره شده در حافظه JITted گفته می‌شود. دیگر نیازی به کامپایل متد jit شده نیست. در فراخوانی بعدی، متد مستقیماً

از حافظه کش در دسترس خواهد بود.



#### Econo JIT Compilation

این نوع کامپایل شبیه به حالت Normal JIT است با این تفاوت که متدها بلافاصله بعد از اجرا از حافظه حذف می‌شوند.



Pre-JIT Compilation

یکی دیگر از حالت‌های کامپایل برنامه‌های دات نتی Pre-JIT Compilation می باشد. در این حالت به جای متدهای مورد استفاده،

کل اسمبلی کامپایل می‌شود. در دات نت می‌توان اینکار را توسط [Ngen.exe](#) یا (Native Image Generator) انجام داد. تمام دستورالعمل‌های CIL قبل از اجرا به کد محلی (Native Code) کامپایل می‌شوند. در این حالت runtime می‌تواند از native images به جای کامپایلر JIT استفاده کند. این نوع کامپایل عملیات تولید کد را در زمان اجرای برنامه به زمان Installation منتقل می‌کند، در اینصورت برنامه نیاز به یک Installer برای اینکار دارد.



همانطور که عنوان شد Ngen.exe برای در دسترس بودن نیاز به Installer برای برنامه دارد. توسط Multicore JIT متدها بر روی دو هسته به صورت موازی کامپایل می‌شوند، در اینصورت می‌توانید تا 50 درصد از JIT Time صرفه جویی کنید.

Multicore JIT همچنین می‌تواند باعث بهبود سرعت در برنامه‌های WPF شود. در نمودار زیر می‌توانید حالت‌های استفاده و عدم استفاده از Multicore JIT را در سه برنامه WPF نوشته شده مشاهده کنید.



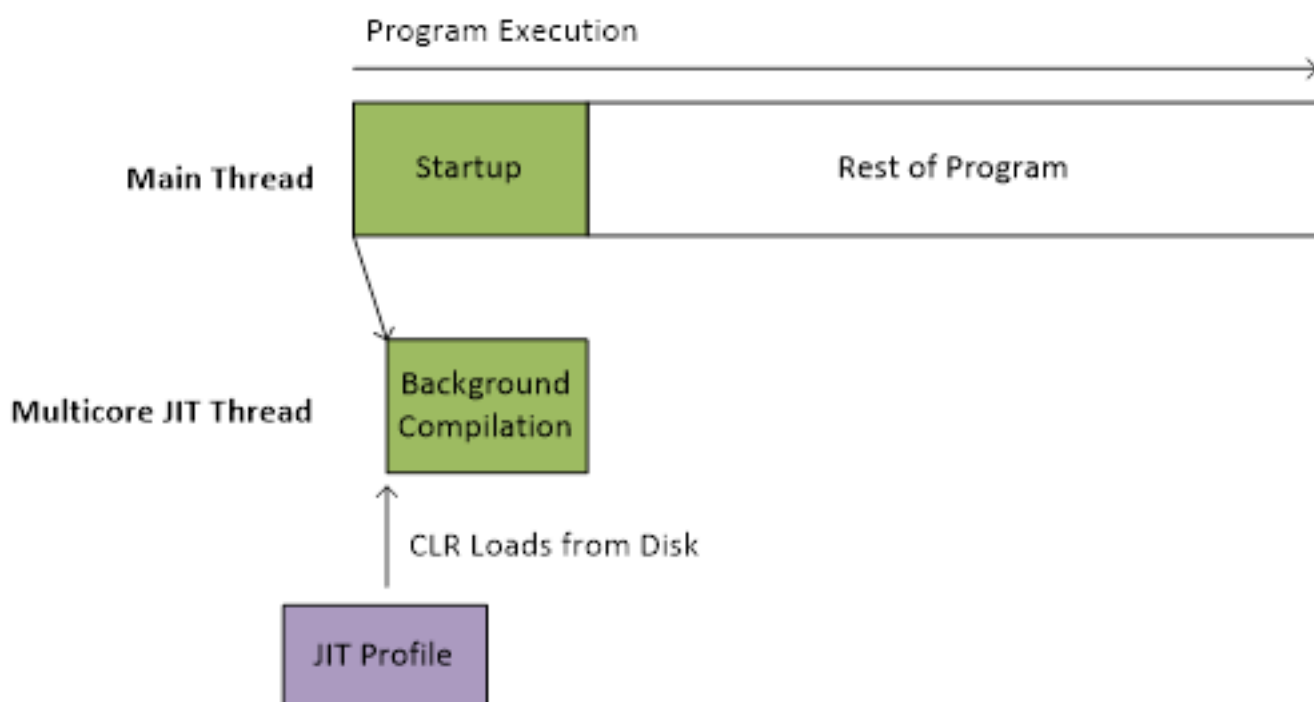
### Multicore JIT در عمل

Multicore JIT از دو مد عملیاتی استفاده می‌کند: مد ثبت (Recording mode)، مد بازپخش (Playback mode)

در حالت ثبت کامپایلر JIT هر متدی که نیاز به کامپایل داشته باشد را رکورد می‌کند. بعد از اینکه CLR تعیین کند که اجرای برنامه به اتمام رسیده است، تمام متدهایی که اجرا شده اند را به صورت یک پروفایل بر روی دیسک ذخیره می‌کند.



هنگامیکه Multicore JIT فعال می‌شود، با اولین اجرای برنامه، حالت ثبت مورد استفاده قرار می‌گیرد. در اجراهای بعدی، از حالت بازپخش استفاده می‌شود. حالت بازپخش پروفایل را از طریق دیسک بارگیری کرده، و قبل از اینکه این اطلاعات توسط ترد اصلی مورد استفاده قرار گیرد، از آنها برای تفسیر (کامپایل) متدها در پیش‌زمینه استفاده می‌کند.



در نتیجه، ترد اصلی به کامپایل دیگری نیاز ندارد، در این حالت سرعت اجرای برنامه بیشتر می‌شود. حالت‌های ثبت و بازپخش تنها برای کامپیوترهایی با چندین هسته فعال می‌باشند.

#### استفاده از Multicore JIT

در برنامه‌های ASP.NET 4.5 و Silverlight 5 به صورت پیش فرض این ویژگی فعال می‌باشد. از آنجائیکه این برنامه‌ها hosted application هستند؛ در نتیجه فضای مناسبی برای ذخیره سازی پروفایل در این نوع برنامه‌ها موجود می‌باشد. اما برای برنامه‌های



Desktop این ویژگی باید فعال شود. برای اینکار کافی است دو خط زیر را به نقطه شروع برنامه تان اضافه کنید:

```
public App()
{
    ProfileOptimization.SetProfileRoot(@"C:\MyAppFolder");
    ProfileOptimization.StartProfile("Startup.Profile");
}
```

توسط متد [SetProfileRoot](#) می‌توانیم مسیر ذخیره سازی پروفایل JIT را مشخص کنیم. در خط بعدی نیز توسط متد StartProfile نام پروفایل را برای فعال سازی Multicore JIT تعیین می‌کنیم. در این حالت در اولین اجرای برنامه پروفایلی وجود ندارد، Multicore JIT در حالت ثبت عمل می‌کند و پروفایل را در مسیر تعیین شده ایجاد می‌کند. در دومین بار اجرای برنامه CRL پروفایل را از اجرای قبلی برنامه بارگذاری می‌کند؛ در این حالت Multicore JIT به صورت بازپخش عمل می‌کند.

همانطور که عنوان شد در برنامه‌های ASP.NET 4.5 و Silverlight 5 قابلیت Multicore JIT به صورت پیش فرض فعال می‌باشد. برای غیر فعال سازی آن می‌توانید با تغییر فلگ profileGuidedOptimizations به None اینکار را انجام دهید:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
  <!-- ... -->
  <system.web>
    <compilation profileGuidedOptimizations="None" />
  <!-- ... -->
  </system.web>
</configuration>
```

عنوان: کامپایل خودکار یک پروژه برای دو فریم ورک

نویسنده: وحید نصیری

تاریخ: ۱۸:۵ ۱۳۹۳/۰۶/۲۵

آدرس: [www.dotnettips.info](http://www.dotnettips.info)

گروه‌ها: MSBuild, .NET 4.5, Compile

فرض کنید می‌خواهید زمانیکه دکمه‌ی build در VS.NET فشرده شد، دو نسخه‌ی دات نت 4 و دات نت 4.5، از پروژه‌ی شما در پوشه‌های مجزایی کامپایل شده و قرار گیرند. در ادامه نحوه‌ی انجام این‌کار را بررسی خواهیم کرد.

### پروژه نمونه

تنظیمات ذیل را بر روی یک پروژه از نوع class library دات نت 4 در VS 2013 اعمال خواهیم کرد.

### ویرایش فایل پروژه برنامه

برای اینکه تنظیمات کامپایل خودکار مخصوص دات نت 4.5 را نیز به این پروژه دات نت 4 اضافه کنیم، نیاز است فایل csproj آن‌را مستقیماً ویرایش نمائیم. این تغییرات شامل مراحل ذیل هستند:

#### الف) تعریف متغیر Framework

```
<PropertyGroup>
  <!-- ...-->
  <Framework Condition=" '$(Framework)' == '' ">NET40</Framework>
</PropertyGroup>
```

به ابتدای فایل csproj در قسمت PropertyGroup آن یک متغیر جدید را به نام Framework اضافه کنید. از این متغیر در شرط‌های کامپایل استفاده خواهد شد.

#### ب) ویرایش مسیر خروجی تنظیمات کامپایل فعلی

```
<PropertyGroup Condition=" '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <!-- ...-->
  <OutputPath>bin\$(Configuration)\$(Framework)\</OutputPath>
</PropertyGroup>
```

در حال حاضر حداقل تنظیمات کامپایل حالت debug، در فایل پروژه موجود است. مقدار OutputPath آن‌را به نحو فوق تغییر دهید تا خروجی نهایی را در پوشه‌ای مانند bin\Debug\NET40 ایجاد کند. بدیهی است اگر حالت release هم وجود دارد، نیاز است مقدار OutputPath آن‌را نیز به همین ترتیب ویرایش کرد.

#### ج) افزودن تنظیمات کامپایل دات نت 4.5 به پروژه جاری

```
<PropertyGroup Condition=" '$(Framework)' == 'NET45' And '$(Configuration)|$(Platform)' == 'Debug|AnyCPU' ">
  <TargetFrameworkVersion>v4.5</TargetFrameworkVersion>
  <PlatformTarget>AnyCPU</PlatformTarget>
  <DebugSymbols>true</DebugSymbols>
  <DebugType>full</DebugType>
  <Optimize>>false</Optimize>
  <OutputPath>bin\$(Configuration)\$(Framework)\</OutputPath>
  <DefineConstants>DEBUG;TRACE;NET45</DefineConstants>
  <ErrorReport>prompt</ErrorReport>
  <WarningLevel>4</WarningLevel>
</PropertyGroup>

<PropertyGroup Condition=" '$(Framework)' == 'NET45' And '$(Configuration)|$(Platform)' == 'Release|AnyCPU' ">
  <TargetFrameworkVersion>v4.5</TargetFrameworkVersion>
  <PlatformTarget>AnyCPU</PlatformTarget>
  <DebugType>pdbonly</DebugType>
  <Optimize>true</Optimize>
  <OutputPath>bin\$(Configuration)\$(Framework)\</OutputPath>
  <DefineConstants>TRACE;NET45</DefineConstants>
```

```
<ErrorReport>prompt</ErrorReport>
<WarningLevel>4</WarningLevel>
</PropertyGroup>
```

در اینجا تنظیمات حالت debug و release مخصوص دات نت 4.5 را مشاهده می‌کنید. برای نگارش‌های دیگر، تنها کافی است مقدار TargetFrameworkVersion را ویرایش کنید.

همچنین اگر به DefineConstants آن دقت کنید، مقدار NET45 نیز به آن اضافه شده‌است. این مورد سبب می‌شود که بتوانید در پروژه‌ی جاری، شرطی‌هایی را ایجاد کنید که کدهای آن فقط در حین کامپایل برای دات نت 4.5 به خروجی اسمبلی نهایی اضافه شوند:

```
#if NET45
public class ExtensionAttribute : Attribute { }
#endif
```

#### د) افزودن تنظیمات پس از build

در انتهای فایل csproj قسمت AfterBuild به صورت کامنت شده موجود است. آن را به نحو ذیل تغییر دهید:

```
<Target Name="AfterBuild">
  <Message Text="Enter After Build TargetFrameworkVersion:${TargetFrameworkVersion}
Framework:${Framework}" Importance="high" />
  <MSBuild Condition="'${Framework}' != 'NET45'" Projects="$(MSBuildProjectFile)"
Properties="Framework=NET45" RunEachTargetSeparately="true" />
  <Message Text="Exiting After Build TargetFrameworkVersion:${TargetFrameworkVersion}
Framework:${Framework}" Importance="high" />
</Target>
```

این تنظیم سبب می‌شود تا کامپایل مخصوص دات نت 4.5 نیز به صورت خودکار فعال گردد و خروجی آن در مسیر bin\Debug\NET45 به صورت جداگانه‌ای قرار گیرد.

```
Test.cs
DualTargetFrameworks
DualTargetFrameworks.Test
5
6 namespace DualTargetFrameworks
7 {
8
9 #if NET45
10 public class ExtensionAttribute : Attribute { }
11 #endif
12
13 public class Test
14 {
15 }
16 }
```

```
Output
Show output from: Build
1>----- Build started: Project: DualTargetFrameworks, Configuration: Debug Any CPU -----
1> DualTargetFrameworks -> D:\Prog\1393\DualTargetFrameworks\DualTargetFrameworks\bin\Debug\NET40\DualTargetFrameworks.dll
1> Enter After Build TargetFrameworkVersion:v4.0 Framework:NET40
1> DualTargetFrameworks -> D:\Prog\1393\DualTargetFrameworks\DualTargetFrameworks\bin\Debug\NET45\DualTargetFrameworks.dll
1> Enter After Build TargetFrameworkVersion:v4.5 Framework:NET45
1> Exiting After Build TargetFrameworkVersion:v4.5 Framework:NET45
1> Exiting After Build TargetFrameworkVersion:v4.0 Framework:NET40
===== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped =====
```

برای آزمایش بیشتر، فایل csproj نهایی را از اینجا می‌توانید دریافت کنید:



## نظرات خوانندگان

نویسنده: مهدی نقدی  
تاریخ: ۱۳۹۳/۰۶/۳۰ ۲۰:۵۶

خیلی موضوع خوبی بود. واقعا خسته نباشید.  
اگر از کتابخانه ای استفاده کنیم که DLL دات نت ۴ و ۴.۵ جداگانه ای رو ارائه داده باشه، چطور میشه این موضوع را پوشش داد؟

نویسنده: وحید نصیری  
تاریخ: ۱۳۹۳/۰۶/۳۰ ۲۲:۱

ویژگی‌های Condition ذکر شده در متن، در مورد ارجاعات هم قابل تنظیم است:

```
<Reference Include="MyAssembly" Condition=".....">  
  <SpecificVersion>False</SpecificVersion>  
  <HintPath>path\to\MyAssembly.dll</HintPath>  
</Reference>
```

همچنین در مورد فایل‌های سورس:

```
<Compile Include="Class20.cs" Condition="....." />
```

در این مطلب یک ترند ساده و سریع برای دوستانی که می‌خواهند از ویژوال استودیو 2010 برای ساختن برنامه‌ی Setup پروژه‌های خود استفاده کنند، آورده می‌شود.

اگر برای ساخت برنامه‌های نصب خود بخواهید از ویژوال استودیو 2010 استفاده کنید و ورژن دات نت برنامه شما بالاتر از 4 باشد، متوجه خواهید شد که در قسمت prerequisites، ورژن دات نت مورد نظر شما وجود ندارد. برای اضافه کردن .net 4.5 و بالاتر به برنامه‌ی نصب خود باید یک Bootstrapper ایجاد کرده و به Bootstrapper Package های موجود در ویندوز اضافه نمایید. در [اینجا](#) نحوه ایجاد و استفاده از Bootstrapper توضیح داده شده است. اما برای اینکه نخواهید درگیر ساخت XML manifests برای .Net 4.5 شوید، یک راه حل ساده‌تر وجود دارد و آن استفاده از Bootstrapper های ساخته شده هنگام نصب ورژن‌های بالاتر ویژوال استودیو که شامل ورژن‌های مورد نیاز از دات نت نیز هستند می‌باشد. برای این کار کافی است به مسیر زیر بر روی سیستم مراجعه نمایید:

C:\Program Files (x86)\Microsoft SDKs\Windows\v8.1A\Bootstrapper\Packages

در مسیر فوق، فولدرهای DotNetFX45، DotNetFX451 و DotNetFX452 را مشاهده می‌کنید که شامل فایل‌های مورد نیاز برای اضافه کردن Bootstrapper به ویژوال استادیو 2010 است. برای اینکار فولدر مربوطه را کپی نمایید و در مسیر زیر قرار دهید:

C:\Program Files (x86)\Microsoft SDKs\Windows\v7.0A\Bootstrapper\Packages

فقط توجه داشته باشید که باید فایل نصب دات نت (مثلا برای دات نت 4.5 فایل dotNetFx45\_Full\_x86\_x64.exe) را به آن اضافه نمایید.

حال اگر ویژوال استادیو 2010 را باز کنید و یک پروژه ستاپ ایجاد نمایید می‌بینید که ورژن مورد نظر به قسمت prerequisites اضافه شده است.