

ویژوال استودیو به توسعه دهندگان این امکان را می‌دهد تا کدهایی را که تکراری بوده و به دفعات در متن برنامه مورد استفاده هستند به شکل یک قطعه کد آماده (در صورت نیاز با مقادیر پیش فرض) ذخیره کنند، سپس در مواقع نیاز بدون اینکه مجبور باشند آن را دوباره و دوباره بنویسند، تنها با تایپ کردن نام قطعه کد ذخیره شده و دو بار فشردن کلید Tab، کد تعریف شده توسط ویژوال استودیو در محل تعیین شده اضافه می‌گردد. به این قطعه کدهای آماده [code snippet](#) گفته می‌شود. خود ویژوال استودیو تعدادی [code snippet آماده](#) دارد که آشنایی با آنها می‌تواند سرعت کدنویسی را افزایش دهد. برای دیدن لیست کامل و مدیریت آنها به مسیر

Tools -> Code Snippets Manager (Ctrl+K, Ctrl+B)

بروید.

در ویژوال استودیو 2010 دو نوع snippet وجود دارد:

1- Expansion snippets: که در محل کرسر (Cursor) اضافه می‌شوند. مثل cw و enum که به ترتیب دستور writeLine و ساختار یک enum را ایجاد می‌کنند.

2- SurroundsWith snippets: که می‌توانند یک تکه کد انتخاب شده را در بر بگیرند مثل for و do که کد انتخاب شده را در یک حلقه for و do-while محصور می‌کنند.

نکته ای که باید توجه داشت این است که یک snippet می‌تواند از هر دو نوع باشد. برای مثال for و do و یا if، در صورتی که کدی انتخاب شده باشد آن را محصور می‌کنند و گرنه ساختار خالی مرتبط را در محل cursor اضافه می‌کنند.

همانطور که در ابتدا هم ذکر شد، علاوه بر snippet‌های آماده‌ی موجود، توسعه دهنده می‌تواند قطعه کدهایی را خود ایجاد کرده و مورد استفاده قرار دهد.

در اینجا یک expansion snippet خواهیم ساخت تا کار اضافه کردن بلاک try-catch-finally را برای ما انجام دهد.

- ابتدا یک فایل xml به پروژه اضافه می‌کنیم و آنرا TryCatchFinally.snippet می‌نامیم. توجه کنید که نام فایل باید به snippet. ختم شود.

- فایل را باز و درون آن راست کلیک کرده و گزینه Insert snippet > Snippet را انتخاب می‌کنیم. با اینکار یک [قالب پایه snippet](#) (که یک ساختار xml) است به فایل اضافه می‌شود. هر فایل snippet از دو بخش اصلی [header](#) و [snippet](#) تشکیل شده که بخش header اطلاعاتی کلی درباره قطعه کد را نگهداری می‌کند و بخش snippet مربوط به تعریف محتوای قطعه کد است.

```
<codesnippet format="1.0.0" xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <header>
    <title>title</title>
    <author>author</author>
    <shortcut>shortcut</shortcut>
    <description>description</description>
    <snippettypes>
      <snippettype>SurroundsWith</snippettype>
      <snippettype>Expansion</snippettype>
    </snippets>
  </header>
  <code></code>
</codesnippet>
```

```
</snippettypes>
</header>
<snippet>
  <declarations>
    <literal>
      <id>name</id>
      <default>value</default>
    </literal>
  </declarations>
  <code language="XML">
    <!--[CDATA[<test-->
      <name>$name$</name>
      $selected$ $end$]]>
  </code>
</snippet>
</codesnippet>
```

- قالب پیش فرض شامل هر دو نوع snippet است .

```
<codesnippet format="1.0.0" xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <header>
    ...
    <snippettypes>
      <snippettype>SurroundsWith</snippettype>
      <snippettype>Expansion</snippettype>
    </snippettypes>
  </header>
  ...
</codesnippet>
```

از آنجا که قصد داریم یک Expansion snippet بسازیم پس تگ SurroundsWith را حذف می‌کنیم .

```
<snippettypes>
  <snippettype>Expansion</snippettype>
</snippettypes>
```

- در بخش header مقدار تگ [Title](#) را به "Try Catch Finally" و مقدار تگ [Shortcut](#) را به "trycf" و [Description](#) را به "Adds a try-catch-finally block" تغییر می‌دهیم . Title عنوان snippet است و وجود آن ضروری است . اضافه کردن shortcut اختیاری است و به عنوان یک متن میانبر برای اضافه کردن snippet استفاده می‌شود .

```
<Header>
  <Title>Try Catch Finally</Title>
  <Author>mohsen.d</Author>
  <Shortcut>trycf</Shortcut>
  <Description>Adds a try-catch-finally block</Description>
```

- تگ [Literal](#) برای تعریف جایگزین برای بخشی از کد درون snippet که احتمال دارد پس از اضافه شدن ، توسط برنامه نویس و یا در صورت استفاده از [function](#) توسط خود ویژوال استودیو تغییر کند استفاده می‌شود . در قطعه کد try-catch-finally ، ما می‌خواهیم به کاربر اجازه بدهیم که نوع استثنائی را که catch می‌شود تغییر دهد .
تگ [id](#) نامی برای بخش قابل ویرایش تعریف می‌کند (که از آن در ادامه در تعریف خود قطعه کد استفاده می‌کنیم) . آنرا به "ExceptionName" تغییر می‌دهیم . تگ [default](#) هم مقدار پیش فرضی را برای آن بخش مشخص می‌کند . ما می‌خواهیم تمام استثناها را Catch کنیم پس مقدار پیش فرض را برابر "Exception" قرار می‌دهیم .

```
.....
<declarations>
  <literal>
```

```
<id>ExceptionName</id>
<default>Exception</default>
</literal>
</declarations>
...
```

- و در تگ [Code](#) ، خود قطعه کدی که ویژوال استودیو باید آن را اضافه کند ، تعریف می‌شود . مقدار مشخصه Language آن را به CSharp تغییر می‌دهیم و محتویات داخل آنرا به شکل زیر اضافه می‌کنیم .

```
<code language="CSharp">
<!--[CDATA[
try
{
    $end$
}
catch($ExceptionName$)
{

}
finally
{

}
}]-->
</code>
```

به نحوه استفاده از ExceptionName که در قسمت Literal تعریف کردیم توجه کنید . عبارت \$end\$ هم یک کلمه رزرو شده است که محل قرار گرفتن cursor را بعد از اضافه شدن قطعه کد مشخص می‌کند .

- در نهایت snippet ما به شکل زیر خواهد بود :

```
<codesnippet format="1.0.0" xmlns="http://schemas.microsoft.com/VisualStudio/2005/CodeSnippet">
  <header>
    <title>Try Catch Finally</title>
    <author>mohsen.d</author>
    <shortcut>trycf</shortcut>
    <description>Adds a try-catch-finally block</description>
    <snippettypes>
      <snippettype>Expansion</snippettype>
    </snippettypes>
  </header>
  <snippet>
    <declarations>
      <literal>
        <id>ExceptionName</id>
        <default>Exception</default>
      </literal>
    </declarations>
    <code language="CSharp">
      <!--[CDATA[
try
{
    $end$
}
catch($ExceptionName$)
{

}
finally
{

}
}]-->
      </code>
    </snippet>
  </codesnippet>
```

اضافه کردن snippet ساخته شده به visual studio

دو راه برای اضافه کردن snippet تعریف شده به ویژوال استودیو وجود دارد :
روش اول قرار دادن فایل snippet در پوشه code snippets ویژوال استودیو است که مسیر پیش فرض آن

```
C:\Users\<UserName>\Documents\Visual Studio 2010\Code Snippets\
```

است . این پوشه به ازای هر زبان دارای یک زیر پوشه است . این snippet را باید در پوشه #C قرار دهیم . همین که فایل را در پوشه مناسب قرار دهیم ویژوال استودیو بدون نیاز به restart شدن آن را خواهد شناخت .

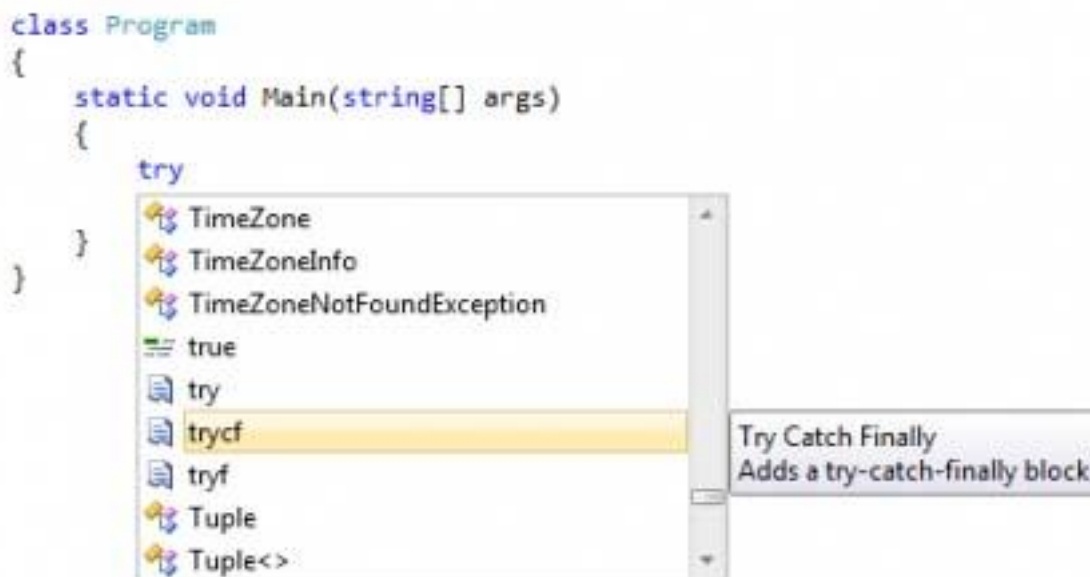
گزینه دوم import کردن فایل snippet به داخل ویژوال استودیو است . در ویژوال استودیو به مسیر

```
Tools -> Code Snippets Manager (Ctrl+K, Ctrl+B)
```

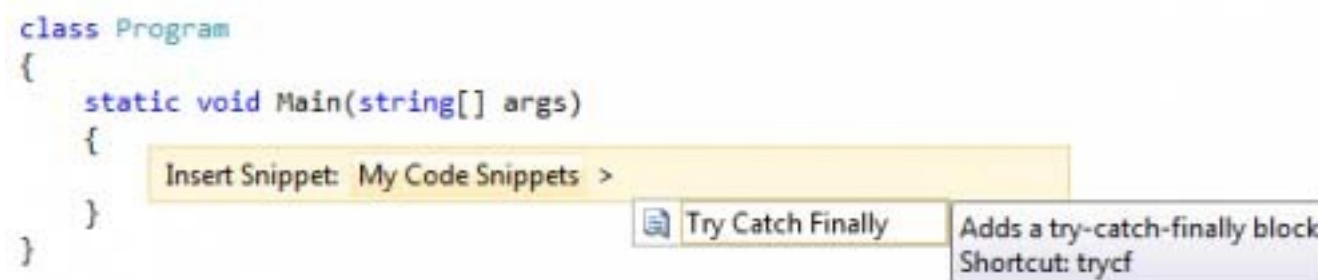
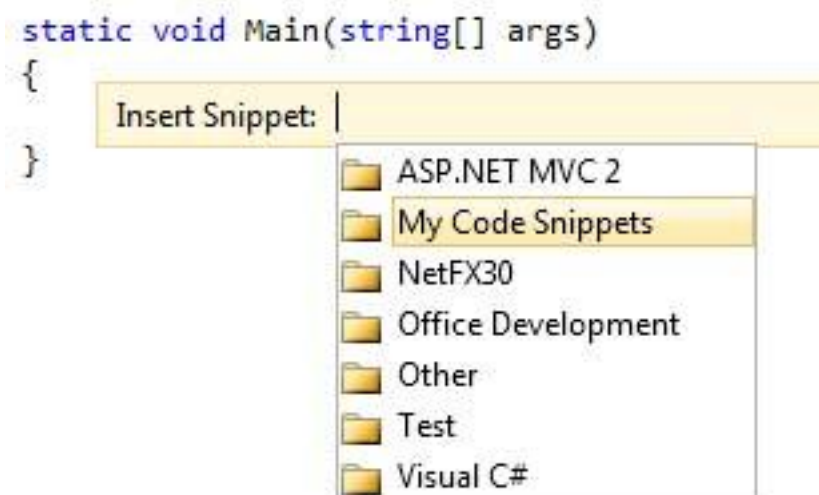
می‌رویم . در پنجره Code Snippets Manager ، بر روی کلید import کلیک و فایل موردنظر را یافته و انتخاب کرده و پوشه‌ی محل ذخیره شدن آن را تعیین می‌کنیم .

استفاده از snippet ساخته شده

برای استفاده از snippet می‌توانیم متن میانبر تعریف شده را تایپ کنیم و با دو بار فشردن کلید tab قطعه کد تعریف شده به محل کرسر اضافه می‌شود



همینطور با فشردن کلیدهای Ctrl+K و Ctrl+X به صورت پشت سر هم منوی “Insert Snippet” ظاهر می‌شود که از طریق آن می‌توانیم Snippet موردنظر را یافته (بدنبال Title تعریف شده برای snippet در پوشه ای که آنرا ذخیره کرده اید بگردید) و با انتخاب آن کد تعریف شده اضافه خواهد شد .



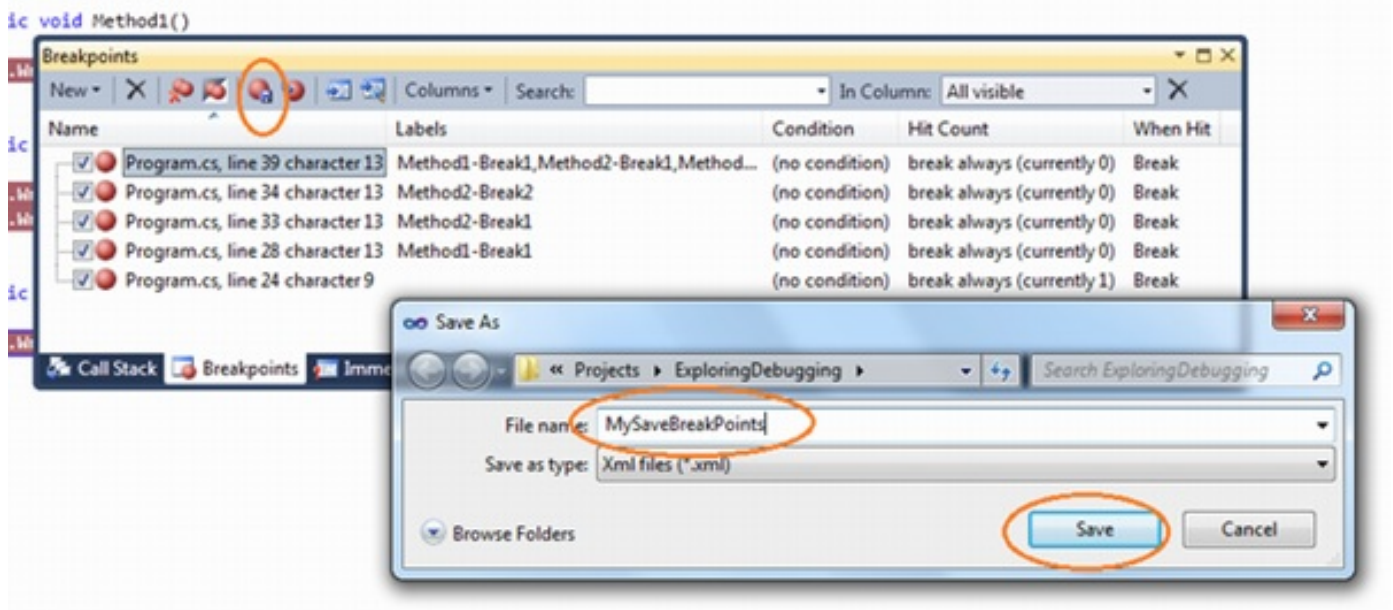
برای آشنایی با روش‌های مختلف دسترسی به snippet ها [اینجا را](#) بررسی کنید .

ابزارها

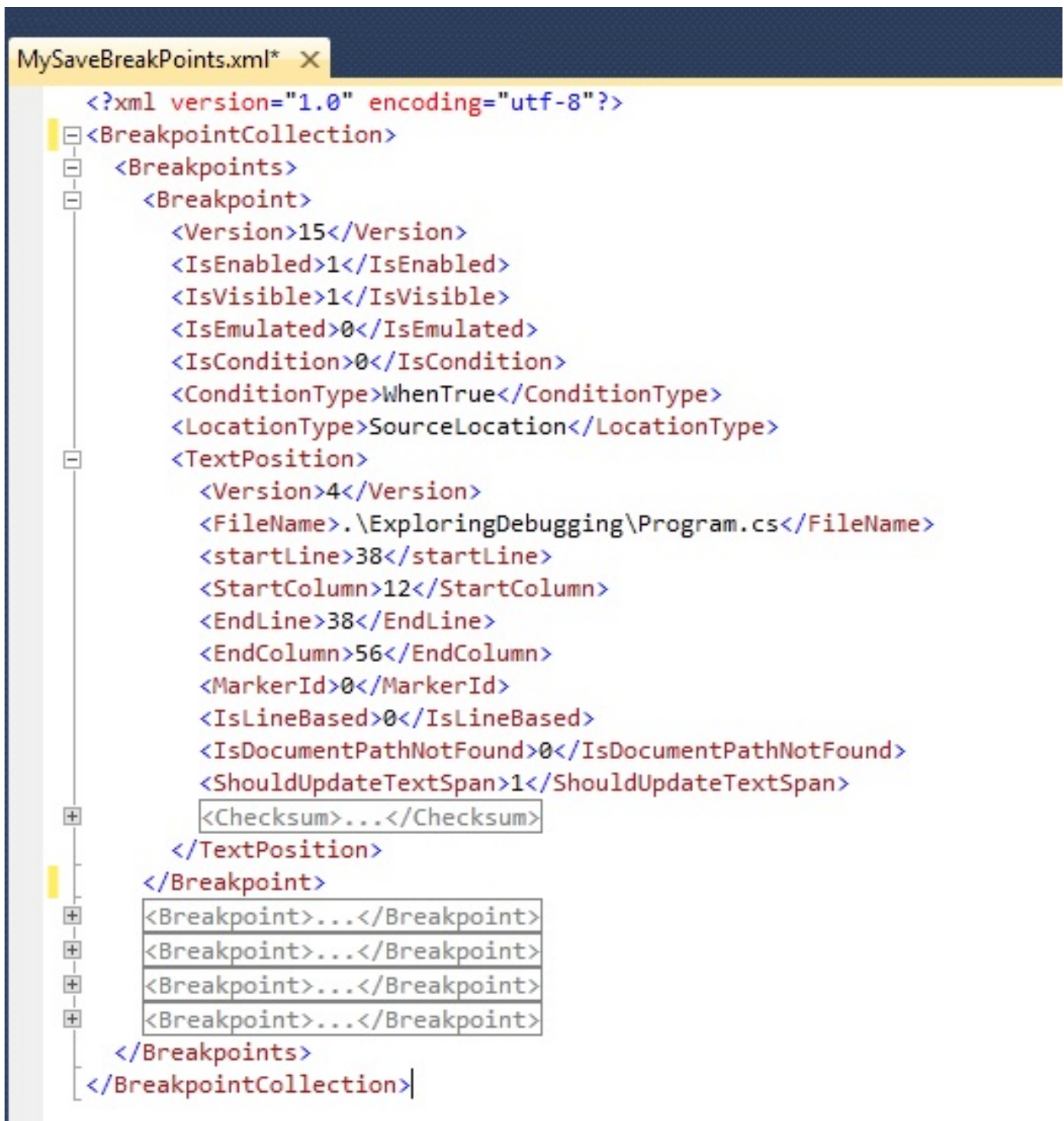
دستکاری خود فایل xml چندان جالب و خالی از خطا نیست . روش‌های بهتری برای ساخت و ویرایش snippet ها وجود دارد . [Snippet Editor](#) ابزاری برای ویرایش و ساخت snippet هاست و [Snippet Designer](#) هم یک پلاگین برای ویژوال استودیو است که کار مشابهی را انجام می‌دهد . یکی از کارهای جالبی که با این ابزار می‌توانید انجام دهید انتخاب یک قطعه از کد (مثل یک تابع) و سپس ساختن یک snippet از روی آن است .

[در این پروژه](#) هم مجموعه snippet های موجود در ویژوال استودیو 2010 برای زبان سی شارپ ، جهت سازگاری با [stylecop](#) ویرایش و refactor شده اند (در کنار تعریف snippet های دیگر) .

ویژوال استدیو Breakpoint ها را در یک فایل XML ذخیره میکند. برای ذخیره Breakpoint ها فقط کافی است بر روی دکمه Export در پنجره Breakpoint که در شکل زیر نمایش داده شده است کلیک کنید.



شما می‌توانید فایل XML ذخیره شده را بعداً استفاده کنید و یا می‌توانید آن را به برنامه نویسان دیگر هم بدهید. اجازه دهید نگاهی داشته باشیم بر محتویات داخل فایل XML. فایل XML کلکسیونی از تگ BreakPoints داخل BreakpointCollection است. هر تگ Breakpoint حاوی اطلاعاتی در مورد یک Breakpoint خاص است.



```

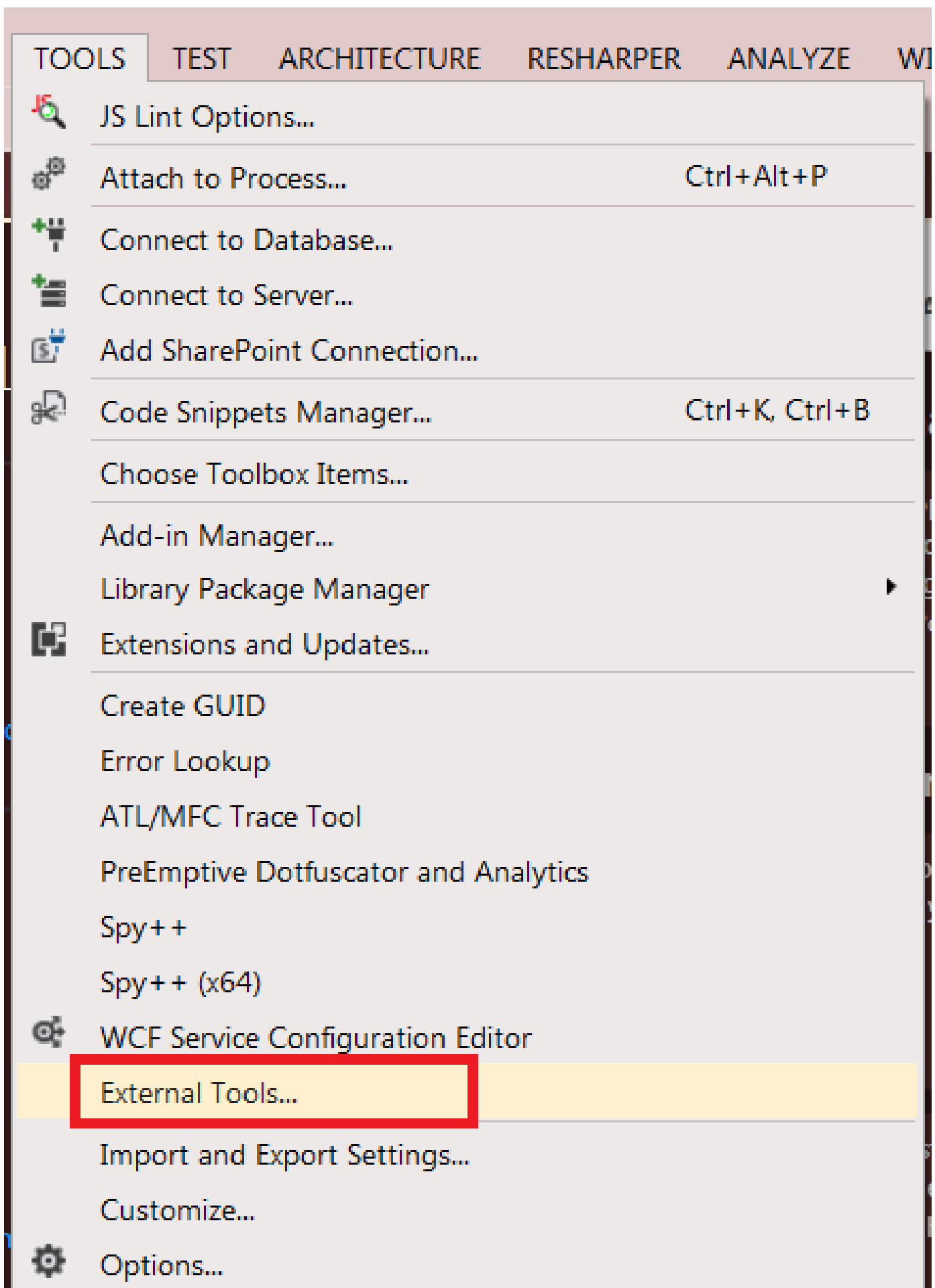
MySaveBreakPoints.xml* X
<?xml version="1.0" encoding="utf-8"?>
<BreakpointCollection>
  <Breakpoints>
    <Breakpoint>
      <Version>15</Version>
      <IsEnabled>1</IsEnabled>
      <IsVisible>1</IsVisible>
      <IsEmulated>0</IsEmulated>
      <IsCondition>0</IsCondition>
      <ConditionType>WhenTrue</ConditionType>
      <LocationType>SourceLocation</LocationType>
      <TextPosition>
        <Version>4</Version>
        <FileName>.\ExploringDebugging\Program.cs</FileName>
        <startLine>38</startLine>
        <StartColumn>12</StartColumn>
        <EndLine>38</EndLine>
        <EndColumn>56</EndColumn>
        <MarkerId>0</MarkerId>
        <IsLineBased>0</IsLineBased>
        <IsDocumentPathNotFound>0</IsDocumentPathNotFound>
        <ShouldUpdateTextSpan>1</ShouldUpdateTextSpan>
        <Checksum>...</Checksum>
      </TextPosition>
    </Breakpoint>
    <Breakpoint>...</Breakpoint>
    <Breakpoint>...</Breakpoint>
    <Breakpoint>...</Breakpoint>
  </Breakpoints>
</BreakpointCollection>

```

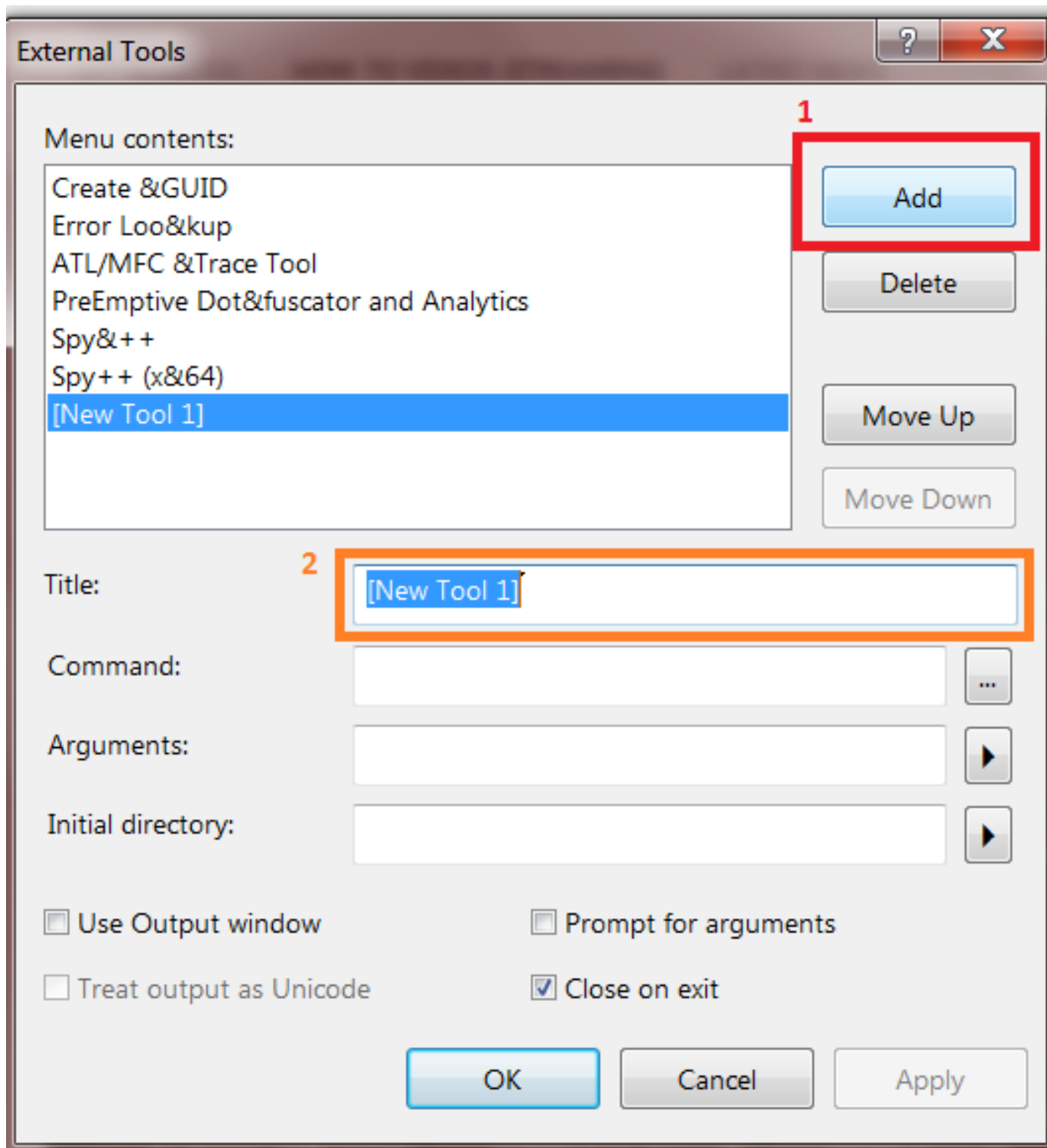
اگر شما هر زمانی همه Breakpoint ها را از کدتان حذف کردید به راحتی می‌توانید آن را تنها با کلیک بر روی Import وارد کدتان بکنید و تمام Breakpoint های ذخیره شده را بازآوری کنید.

نکته: Import کردن Breakpoint براساس شماره خط کد شما می‌باشد یعنی همان خطی که شما Breakpoint را گذاشته اید پس اگر شماره خط کد شما تغییر کند Breakpoint بروی خط قبلی گذاشته میشود.

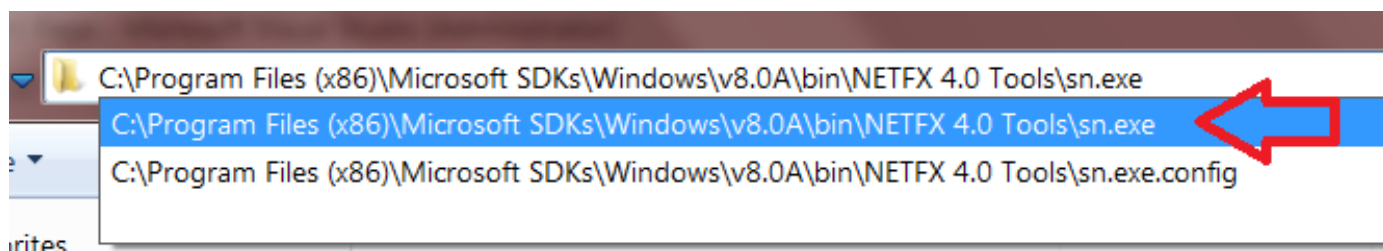
ایجاد Strong Name به اسمبلی برای داشتن یک هویت منحصر به فرد برای آن اسمبلی کمک می‌کند و یکی از پارامترهای آن داشتن Public Key Token برای اسمبلی است ([بیشتر](#)). در این پست قصد دارم به کمک ابزارهای جانبی Visual Studio 2012 که البته در 2010 نیز امکان پذیر است روشی برای تهیه آسان‌تر این Key ارائه کنم .
برای آغاز نرم افزار VS2012 را باز می‌کنیم و به منوی Tools رفته و گزینه External Tools را انتخاب می‌کنیم :



در پنجره‌ی پیش رو روی دکمه Add کلیک کنید و نامی برای Tools انتخاب کنید :



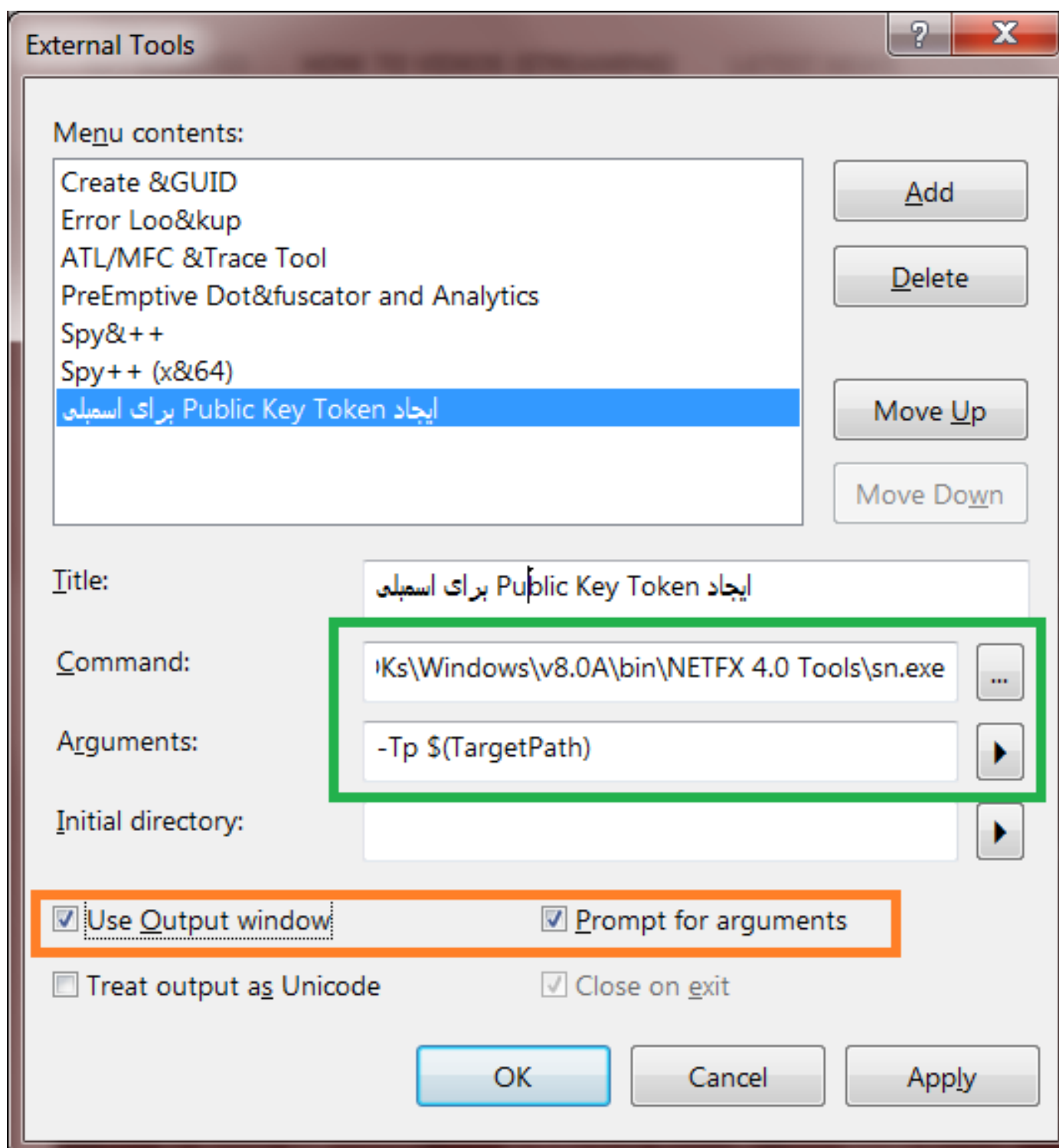
سپس مسیر فایل sn.exe را کپی کرده و در فیلد Command قرار دهید .



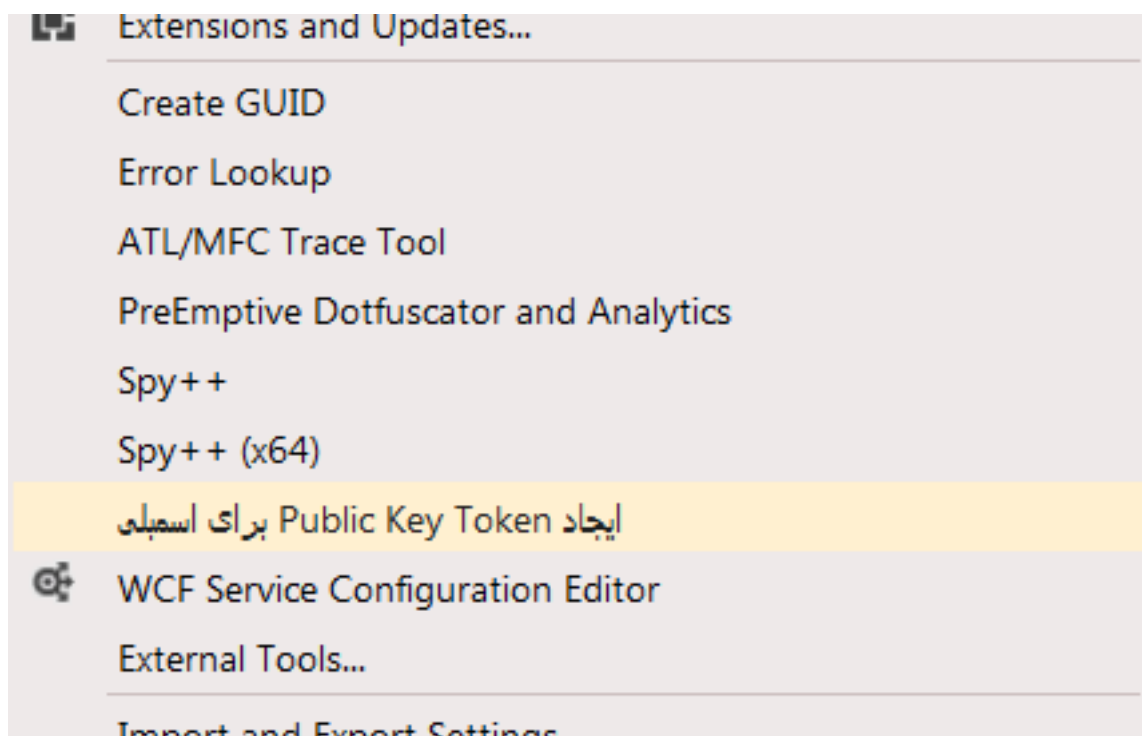
برای پارامتر از عبارت زیراستفاده کرده تا Public Key اسمبلی جاری را به شما بدهد . برای اطلاعات بیشتر در مورد آرگومانها به [اینجا](#) مراجعه کنید

```
-Tp $(TargetPath)
```

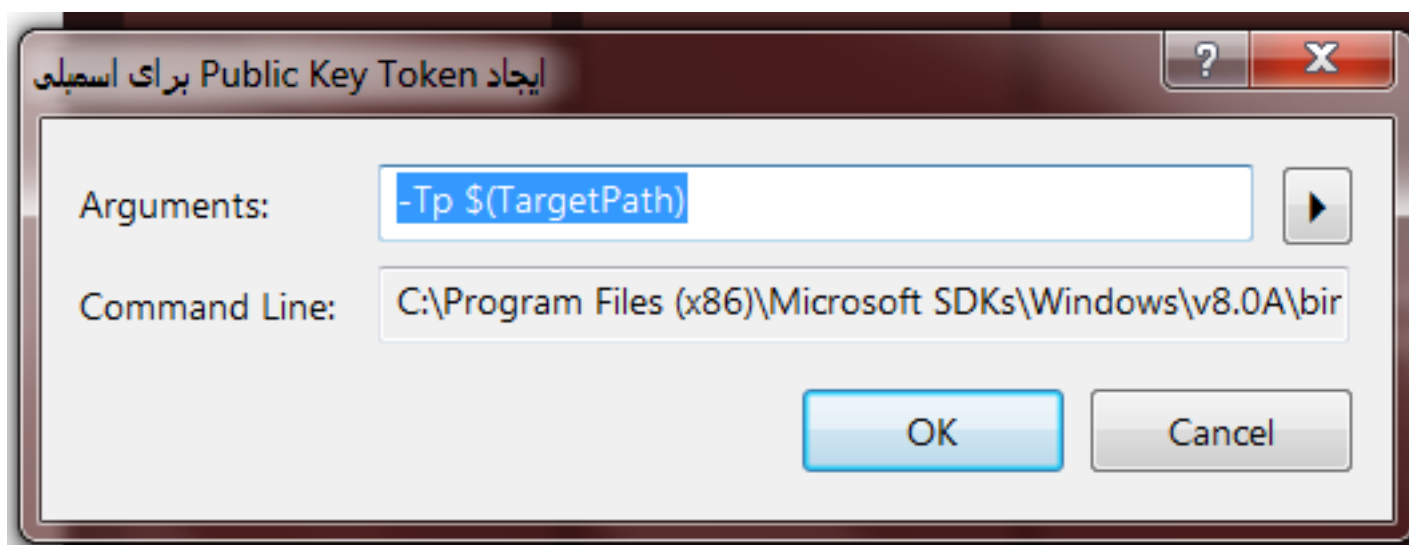
همچنین گزینه‌های Prompt for Arguments را برای دریافت آرگومان دلخواه شما (مثلا مواردی که مایلید برای یک اسمبلی دیگر key استخراج کنید) و Use Output window برای نمایش خروجی را علامت بزنید



روی OK کلیک کنید و به منوی Tools بازگردید :



حال روی نام پروژه خود در Solution Explorer کلیک کنید و روی Tools ساخته شده کلیک کنید :



و خروجی:

```
Output
Show output from: ایجاد Public Key Token برای اسمبلی
Microsoft (R) .NET Framework Strong Name Utility  Version 4.0.30319.17929
Copyright (c) Microsoft Corporation.  All rights reserved.

Public key (hash algorithm: sha1):
002400000480000009400000000602000000024000005253413100040000010001002dad9517360f66
c7e3968137a71d1c0fcfeb7264034b0ccb57528b66e557050568c78cce59087bfe25f4a012209a
d85c69657e823170306c46c2f99e2678e16f131dc865072165df730d2380b87a62e72dd3b2dde9
18c173785ebf08782cd457cf867822c5bb607bdf7c708ed6c1aab317262b951d54ea02167fc162
e4708aa8

Public key token is 25ce05e4457e7848
```

[موفق باشید](#)

عنوان: Build Events

نویسنده: یوسف نژاد

تاریخ: ۱۳۹۱/۱۱/۱۱ ۲۳:۴۵

آدرس: www.dotnettips.info

برچسب‌ها: Visual Studio, MSBuild, Build Events

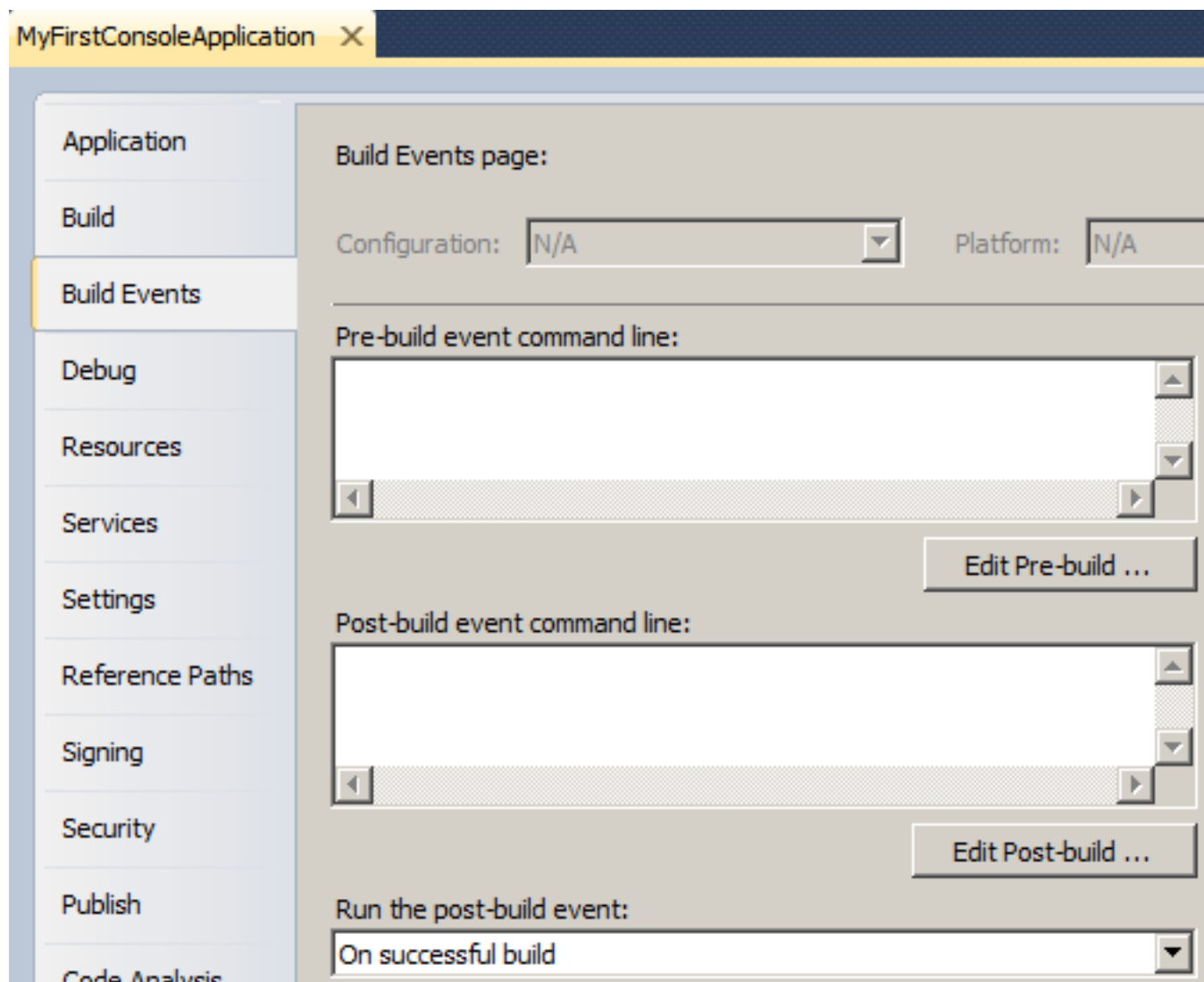
در ویژوال استودیو یک ویژگی جالب با عنوان **Pre/Post-Build Event** وجود دارد. این ویژگی به رویدادهای «قبل از بیلد» و «بعد از بیلد» اشاره دارد. از این ویژگی برای اجرای یکسری دستورات، قبل (Pre-build) یا بعد (Post-build) از عملیات بیلد استفاده میشود. دستوراتی که در این قسمت قابل اجرا هستند دقیقاً همانند دستورات موجود در یک batch فایل میباشند. حتی میتوان یک فایل bat. را در این قسمت فراخوانی کرد. بطور خلاصه هرگونه دستوری که درون Command Prompt ویندوز یا در یک bat. فایل قابل اجرا باشد در این قسمت نیز قابل استفاده است. درنهایت تمام این دستورات توسط برنامه Cmd.exe اجرا میشوند.

نکته: قبل از ادامه بهتر است به این نکته اشاره کنم که مجموعه این دستورات چیزی فراتر از فراخوانی ساده یکسری فایل exe. هستند. درواقع کدی که در این قسمت به آن اشاره میشود، دارای ساختاری به صورت یک زبان برنامه نویسی ساده است. یعنی متنی نهایی‌ای که برای اجرا به cmd.exe ارسال میشود میتواند شامل دستورات ساده و اولیه برنامه نویسی چون `if .. then .. else` و حلقه `for` و از این قبیل نیز باشد. برای آشنایی بیشتر با زبان این نوع دستورات به منابع زیر مراجعه کنید: [Batch file](#)

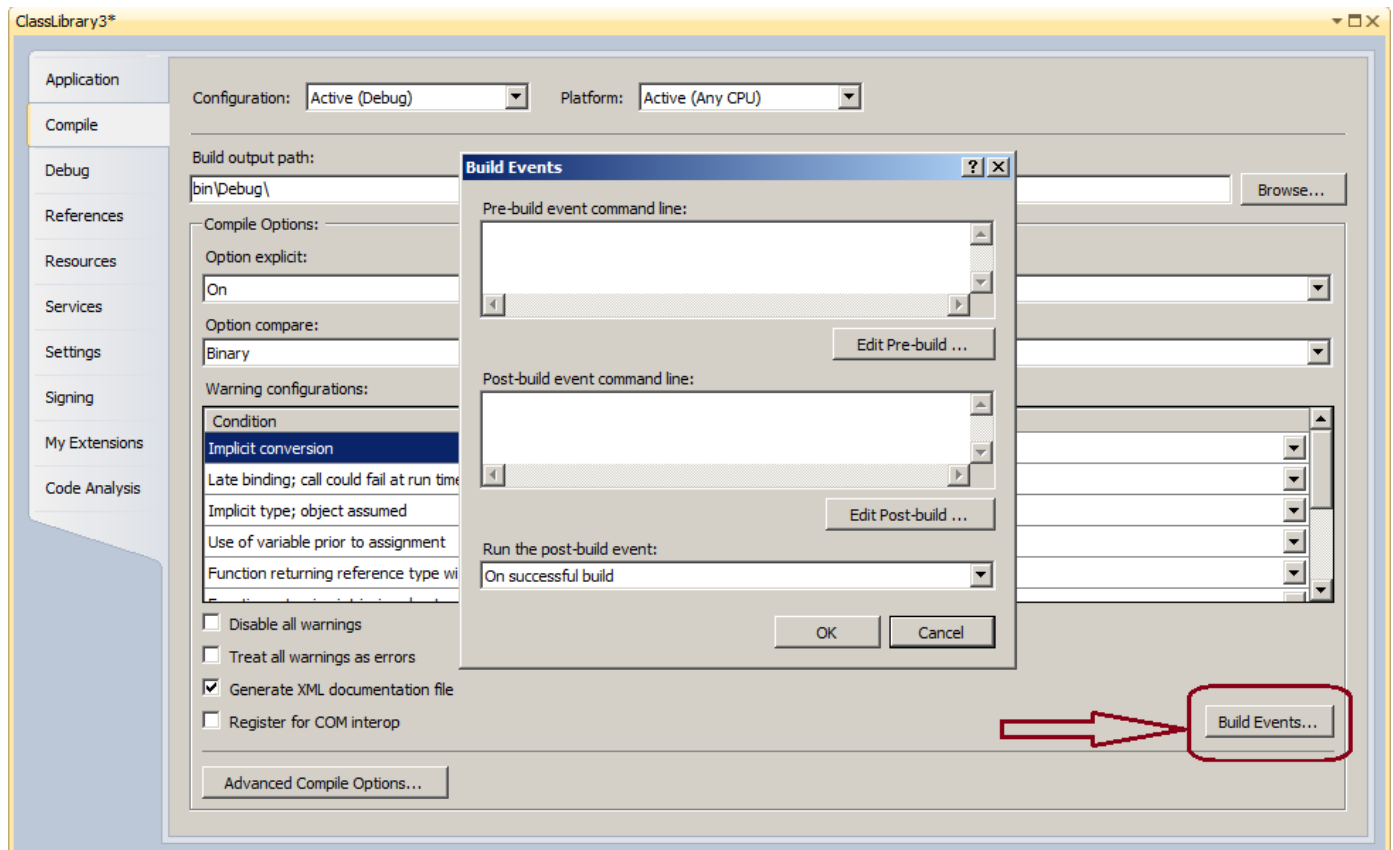
[Using batch files](#)

تنظیم رویدادهای بیلد (Build Events)

برای تنظیم این رویدادها باید به تب Build Events در صفحه پراپرتی‌های پروژه موردنظر مراجعه کنید. همانند تصویر زیر در یک پروژه کنسول C#:



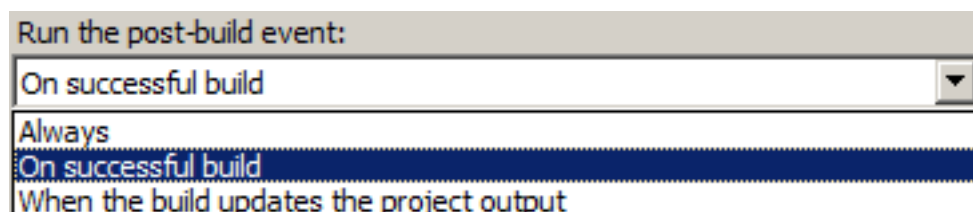
البته در پروژه‌های VB.NET مسیر منتهی به این قسمت کمی فرق میکند که در تصویر زیر نشان داده شده است:



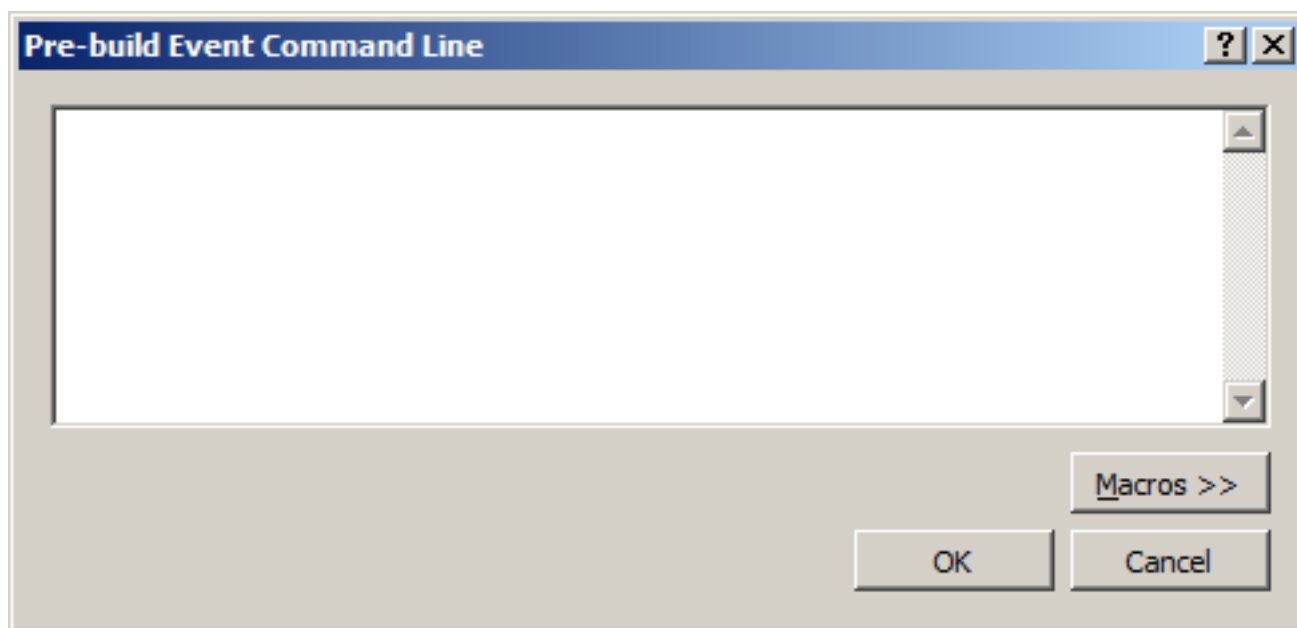
در پروژه‌های مربوط به زبانهای دیگر هم مسیر رسیدن به این رویدادها کمی متفاوت است. برای کسب اطلاعات بیشتر به [اینجا](#) مراجعه کنید.

در این قسمت میتوان همانند یک فایل batch دستورات موردنظر را در خطوط مجزا برای اجرا اضافه کرد. از این دستورات معمولاً برای مدیریت عملیات بیلد، کپی فایل‌های موردنیاز قبل یا بعد از بیلد، پاک کردن فولدرها، تغییر برخی تنظیمات با توجه به نوع کانفیگ بیلد (Debug یا Release)، ثبت یک اسمبلی در GAC و یا حتی اجرای برخی آزمونهای واحد و ... استفاده میشود.

نکته: در صورتیکه پروژه به روز باشد (یعنی ویژوال استودیو نیازی به تولید فایل اسمبلی نهایی پروژه به دلیل عدم وجود تغییری در کد برنامه نبیند) بدلیل عدم اجرای عملیات بیلد، دستورات قسمت Pre-build اجرا نمیشوند. اجرای دستورات قسمت Post-build نیز بستگی به تنظیمات قسمت Run the post-build events: همانند تصویر زیر دارد:



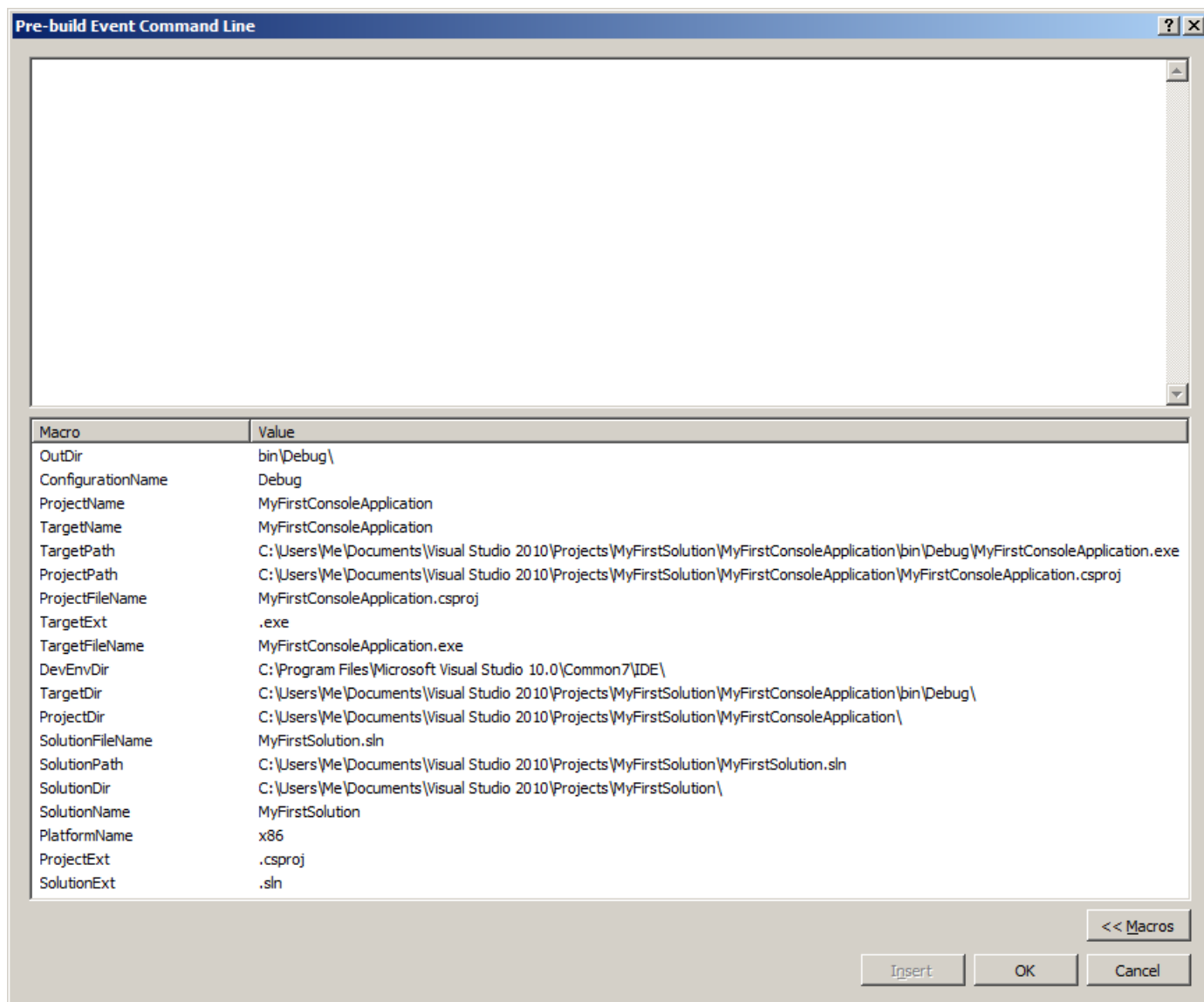
برای استفاده راحتتر از این ویژگی فرمی مخصوص وارد کردن این دستورات در ویژوال استودیو وجود دارد. برای دیدن این فرم بر روی دکمه Edit Pre-build... یا Edit Post-build... کلیک کنید. پنجره زیر نمایش داده میشود:



در این پنجره میتوان دستورات مورد نظر را وارد کرد. با اینکه هیچ امکان خاصی برای کمک به اضافه و ویرایش دستورات در این پنجره وجود ندارد! اما تنها ویژگی موجود در این فرم کمک بسیاری برای تکمیل دستورات موردنظر میکند. قبل از توضیح این ویژگی بهتر است با مفهوم Macro در این قسمت آشنا شویم.

Macro

در Build Events ویژوال استودیو یکسری متغیرهای ازقبل تعریف شده وجود دارد که به آنها Macro گفته میشود. برای مشاهده لیست این ماکروها روی دکمه Macro >> کلیک کنید. پنجره مربوطه به صورت زیر گسترش می‌یابد تا جدولی به نام Macro Table را نمایش دهد:



همانطور که مشاهده میکنید تعداد 19 ماکرو به همراه مقادیرشان در این قسمت به نمایش گذاشته شده است. برای استفاده از این ماکروها کافی است تا روی یکی از آنها دابل کلیک کنید یا پس از انتخاب ماکروی موردنظر روی دکمه Insert کلیک کنید. دقت کنید که نحوه نمایش این ماکروها در متن دستورات به صورت زیر است:

`$(<Macro_Name>)`

که به جای عبارت `<Macro_Name>` عنوان ماکرو قرار میگیرد. مثلاً:

`$(OutDir)` یا `$(ProjectName)`

نکته: نام این ماکروها case-sensitive نیست .

نحوه اجرای دستورات توسط ویژوال استودیو

ویژوال استودیو برای اجرای دستورات کار خاصی به صورت مستقیم انجام نمیدهد! وظیفه اصلی برعهده MSBuild ([^](#)) است. این ابزار پس از جایگزین کردن مقادیر ماکروها، محتوای کل دستورات موجود در هر یک از رویدادها را در یک فایل batch ذخیره میکند و فایل مربوط به هر رویداد را در زمان خودش به اجرا میگذارد. مثلاً دستور زیر را درنظر بگیرید:

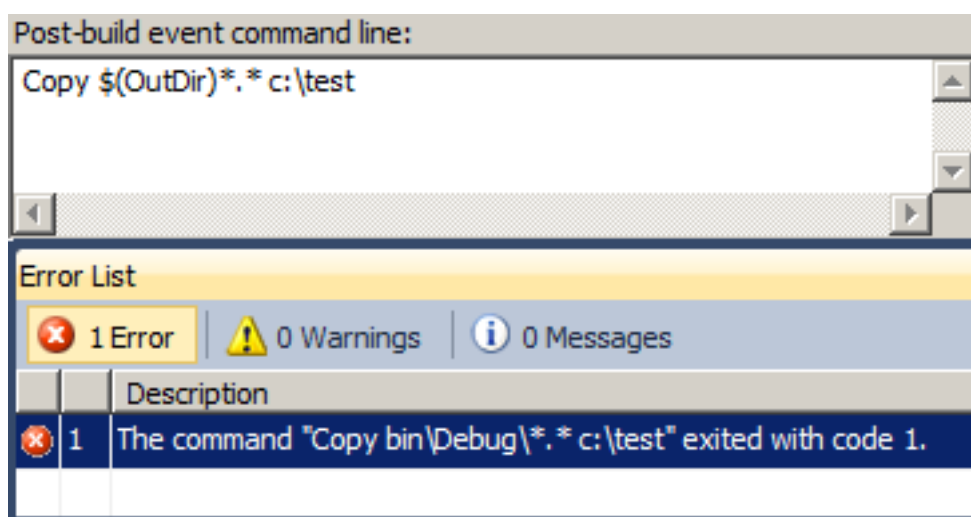
```
Copy $(OutDir)*.* %WinDir%
```

پس از ذخیره در فایل batch نهایی به صورت زیر در خواهد آمد:

```
Copy bin\Debug\*.* %WinDir%
```

نکته: در این زبان برنامه نویسی، عبارتی چون %WinDir% معرف یک متغیر است. در این مورد خاص این عبارت یک متغیر محیطی (Environment Variable) است. اطلاعات بیشتر در [اینجا](#).

MSBuild عملیات اجرای این batch فایل‌های تولیدی را زیر نظر دارد و هرگونه خطای موجود در این دستورات را به عنوان خطای زمان بیلد گزارش می‌دهد. اما از آنجاکه کل دستورات مربوط به هر رویداد درون یک فایل batch اجرا می‌شود، امکان گزارش محل دقیق خطای رخ داده وجود ندارد. یعنی در صورتیکه مثلاً تنها یکی از صدها خط دستور نوشته شده در این قسمت خطا بدهد تنها یک خطا و برای تمام دستورات نمایش داده می‌شود. البته همانطور که حدس می‌توان حدس زد اجرای این دستورات ترنزشال نیست و اجرای تمامی دستورات تا قبل از وقوع خطا برگشت ناپذیر خواهند بود. برای نمونه به تصویر زیر و خطای نمایش داده شده دقت کنید:



نمونه اصلاح شده دستور فوق به صورت زیر است:

```
Copy "$(ProjectDir)$(OutDir)*.*" c:\test
```

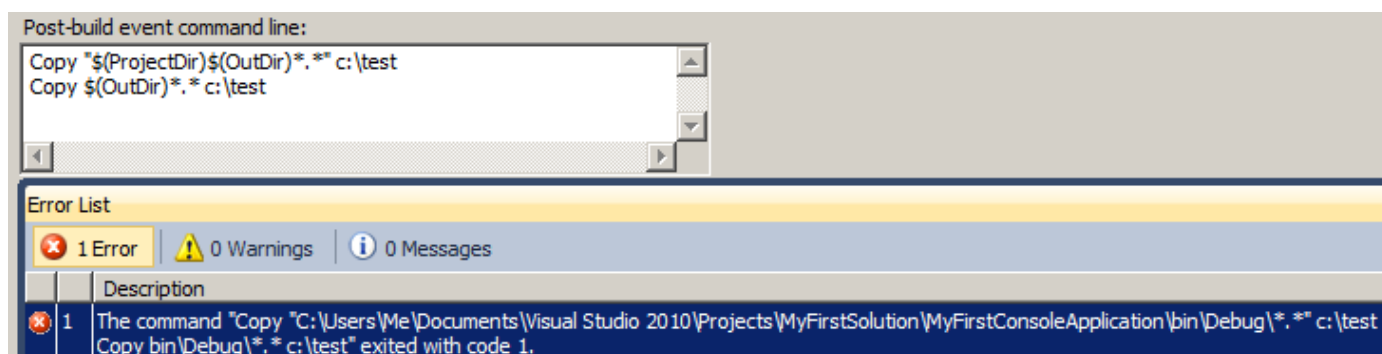
نکته: به دلیل استفاده از کاراکتر فاصله به عنوان جداکننده آرگومانها در دستورات DOS، وجود فاصله در مسیرهای مورد استفاده در این دستورات عملیات را دچار خطا خواهد کرد. راه حل استفاده از کاراکتر " در ابتدا و انتهای رشته‌های مربوط به مسیرها همانند دستور بالا است.

نکته: در صورت استفاده از یک فایل bat. برای ذخیره دستورات، امکان استفاده مستقیم از ماکروهای ویژوال استودیو درون آن وجود نخواهد داشت! یکی از راه‌حلها پاس کردن این متغیرها به صورت پارامتر در زمان فراخوانی فایل bat. است. مثلاً:

```
"$(ProjectDir)postBuild.bat" "$(SolutionPath)"
```

برای دریافت این پارامترهای پاس شده درون batch فایل باید از عبارات 1% برای پارامتر اول و 2% برای پارامتر دوم و ... تا 9%

برای پارامتر نهم است. برای کسب اطلاعات بیشتر به منابع معرفی شده در ابتدای مطلب مخصوصا قسمت [Using batch parameters](#) مراجعه کنید.
حال مجموعه دستورات زیر و خطای رخ داده را در نظر بگیرید:



با بررسی مطلب متوجه میشویم با اینکه خط اول مجموعه دستورات فوق درست بوده و کاملا صحیح اجرا میشود اما خطای رخ داده به کل دستورات اشاره دارد و مشخص نشده است که کدام دستور مشکل دارد. دقت کنید که دستور اول کاملا اجرا میشود! راه حل ساده ای در [اینجا](#) برای حل این مشکل ارائه شده است. در این راه حل با استفاده از قابلیت‌های این زبان، کل عملیات و مخصوصا خطاهای رخ داده در این مجموعه دستورات هندل میشود تا کنترل بهتری در این مورد بر روی فرایند وجود داشته باشد. نمونه این راه حل به صورت زیر است:

```
echo -----
echo Copy "$(ProjectDir)$(OutDir)*.*" c:\test --Starting...
Copy "$(ProjectDir)$(OutDir)*.*" c:\test
if errorlevel 1 goto error
echo Copy "$(ProjectDir)$(OutDir)*.*" c:\test --DONE!
echo -----
echo -----
echo Copy $(OutDir)*.* c:\test --Starting...
Copy $(OutDir)*.* c:\test
if errorlevel 1 goto error
echo Copy $(OutDir)*.* c:\test --DONE!
echo -----
goto ok
:error
echo POSTBUILDSTEP for $(ProjectName) FAILED
notepad.exe
exit 1
:ok
echo POSTBUILDSTEP for $(ProjectName) COMPLETED OK
```

با استفاده از مجموعه دستوراتی شبیه دستورات بالا میتوان لحظه به لحظه اجرای عملیات را بررسی کرد.
نکته: خروجی تمام این دستورات و نیز خروجی دستورات echo در پنجره Output ویژوال استودیو به همراه سایر پیغامهای بیلد نمایش داده میشود.
نکته: در اسکرپیت فوق برای درک بیشتر مسئله با استفاده از دستور notepad.exe در قسمت error: از وقوع خطا اطمینان حاصل میشود. دقت کنید تا زمانیکه برنامه اجرا شده Notepad بسته نشود فوکس به ویژوال استودیو برنمیگردد و عملیات بیلد تمام نمیشود.
نکته: در صورت استفاده از دستور exit 0 در انتهای قسمت error: (به جای دستور exit 1 موجود) به دلیل اعلام خروج موفق از عملیات، ویژوال استودیو خطایی نمایش نخواهد داد و عملیات بیلد بدون نمایش خطا و با موفقیت به پایان خواهد رسید. درواقع استفاده از هر عددی غیر از صفر به معنی خروج با خطا است که این عدد غیر صفر کد خطا یا error level را مشخص میکند ([^](#) و [^](#)).

یکی از دستورات جالبی که میتوان در این رویدادها از آن استفاده کرد، دستور نصب نسخه ریلیز برنامه در GAC است. نحوه

استفاده از آن میتواند به صورت زیر باشد:

```
if $(ConfigurationName) == Release (
gacutil.exe /i "$(SolutionDir)$(OutDir)$(TargetFileName)"
)
```

نکته: در صورتیکه در دستورات مربوط به رویداد قبل از بیلد یعنی Pre-build خطایی رخ بدهد عملیات بیلد متوقف خواهد شد و برای پروژه فایلی تولید نمیشود. اما اگر این خطا در رویداد بعد از بیلد یعنی Post-build رخ دهد با اینکه ویژوال استودیو وقوع یک خطا را گزارش میدهد اما فایل‌های خروجی پروژه حاصله از عملیات بیلد تولید خواهند شد.

نکته: توجه داشته باشید که در استفاه از این ویژگی زیاده‌روی نباید کرد. استفاده زیاد و بیش از حد (و با تعداد زیاد دستورات) از این رویدادها ممکن است عملیات بیلد را دچار مشکلاتی پیچیده کند. دیباگ این رویدادها و دستورات موجود در آنها بسیار مشکل خواهد بود. اگر تعداد خطوط دستورات موردنظر زیاد باشد بهتر است کل دستورات را درون یک فایل bat. ذخیره کنید و این فایل را بطور جداگانه مدیریت کنید که کار راحتتری است.

نکته: بهتر است قبل از وارد کردن دستورات درون این رویدادها، ابتدا تمام دستورات را در یک پنجره cmd آزمایش کنید تا از درستی ساختار و نتیجه آن‌ها مطمئن شوید.

رویدادهای بیلد و MSBuild

همانطور که در [اینجا](#) توضیح داده شده است، ویژوال استودیو از ابزار MSBuild برای تولید اپلیکیشن‌ها استفاده میکند. عملیات مدیریت رویدادهای بیلد نیز توسط این ابزار انجام میشود. اگر به فایل پروژه مربوط به مثال قبل مراجعه کنید به محتوایی شبیه خطوط زیر میرسید:

```
...
<PropertyGroup>
<PostBuildEvent>echo -----
echo Copy "$(ProjectDir)$(OutDir)*.*" c:\test --Starting...
Copy "$(ProjectDir)$(OutDir)*.*" c:\test
if errorlevel 1 goto error
echo Copy "$(ProjectDir)$(OutDir)*.*" c:\test --DONE!
echo -----
echo -----
echo Copy $(OutDir)*.* c:\test --Starting...
Copy $(OutDir)*.* c:\test
if errorlevel 1 goto error
echo Copy $(OutDir)*.* c:\test --DONE!
echo -----
goto ok
:error
echo POSTBUILDSTEP for $(ProjectName) FAILED
notepad.exe
exit 1
:ok
echo POSTBUILDSTEP for $(ProjectName) COMPLETED OK</PostBuildEvent>
</PropertyGroup>
...
```

همانطور که میبینید در ویژوال استودیو تنها ذخیره این تنظیمات در فایل پروژه انجام میشود و کلیه عملیات توسط ابزار MSBuild مدیریت میگردد. امکان بهره‌برداری از این رویدادها با استفاده مستقیم از ابزار MSBuild نیز وجود دارد اما به دلیل مفصل بودن بحث، جستجوی بیشتر به خوانندگان واگذار میشود.

منابع برای مطالعه بیشتر: [\(#How to: Specify Build Events \(C](#)

[Specifying Custom Build Events in Visual Studio](#)

[Pre-build Event/Post-build Event Command Line Dialog Box](#)

[Customize Your Project Build Process](#)

نظرات خوانندگان

نویسنده: سعید

تاریخ: ۱۸:۹ ۱۳۹۱/۱۱/۱۵

با تشکر از مطلب مفیدتان. برای کاربردهای معمولی تاجایی که دیدم بیشتر مثلا برای obfuscating خودکار اسمبلی پس از بیلد ازش استفاده میشه. اما در کارهای تیمی در continuous integration به نظر می‌رسه خیلی کاربرد داره. بررسی کیفیت کد، اجرای آزمون‌های واحد، اجرای آنالیزهای خودکار و مثل این‌ها

نویسنده: صابر فتح الهی

تاریخ: ۱:۵۹ ۱۳۹۲/۰۳/۰۷

سلام با تشکر از پست شما
من می‌خوام اندازه پشته توی ویژوال استودیو تغییر بدم در Post Build Event کد زیر نوشتم

```
editbin.exe /STACK:1000000 $(TargetFileName)
```

اما در زمان کامپایل پروژه با این خطا مواجه می‌شم

```
Error 7 The command "editbin.exe /STACK:10000 MS-AUV.exe" exited with code 9009.MS-AUV
```

ممنون میشم راهنماییم کنین

نویسنده: یوسف نژاد

تاریخ: ۹:۴۹ ۱۳۹۲/۰۳/۰۷

با سلام در پاسخ به مشکل شما چند نکته باید اشاره بشه.

نکته اول: ماکروی TargetFileName فقط اسم فایل خروجی پروژه رو برمی‌گردونه، در صورتیکه برای کارکردن دستور فوق مسیر کامل فایل نیازه. چون برنامه editbin.exe درون مسیر خروجی پروژه شما اجرا نمی‌شه. شما می‌تونین از ماکروی TargetPath استفاده کنید که مسیر کامل فایل خروجی پروژه رو برمی‌گردونه.

نکته دوم: کد خطای 9009 مربوط به پیدا نکردن فایل هست. البته فایلی که در اینجا پیدا نشده خروجی پروژه شما نیست بلکه خود ابزار editbin هستش. مسیر درستش در سیستم 32 بیتی برای ویژوال استودیو 2010 اینه:

```
C:\Program Files\Microsoft Visual Studio 10.0\VC\bin\editbin.exe
```

اما چون این مسیرها معمولا حاوی **فاصله** هستند نیاز به استفاده از **دابل کوتیشن** در ابتدا و انتها وجود داره. بنابراین دستور کامل باید به صورت زیر باشه:

```
"C:\Program Files\Microsoft Visual Studio 10.0\VC\bin\editbin.exe" /STACK:1000000 "$(TargetPath)"
```

اما با اجرای دستور فوق باز هم خطایی صادر میشه که کمی خطرناکتر از قبلیه. و اما دلیلش:

نکته سوم: متن زیر از [msdn](#) گرفته شده:

```
You can start this tool only from the Visual Studio command prompt. You cannot start it from a system
```

command prompt or from Windows Explorer.

البته منظور دقیق‌تر این جمله اینه که ابزار editbin نیاز به یکسری تنظیمات و متغیرهای ازپیش تعیین شده داره که در Visual Studio command prompt انجام شده. اما نگران نباشید برای تنظیم این تنظیمات و تبدیل خط فرمان Build Events در ویژوال استودیو به یک Visual Studio command prompt کافیست که خط زیر رو در ابتدای مجموعه دستورات build events خودتون قرار بدین:

```
call "$(DevEnvDir)..\Tools\vsvars32.bat"
```

این بچ فایل حاوی دستوراتی نسبتاً مفصل برای تنظیم تنظیمات موردنیاز است. درواقع با اجرای این بچ فایل هر خط فرمانی تقریباً تبدیل به Visual Studio command prompt خواهد شد. با توجه به ماکروی \$(DevEnvDir) مسیر کامل این فایل در سیستم 32 بیتی و برای ویژوال استودیوی 2010 به صورت زیر است:

```
C:\Program Files\Microsoft Visual Studio 10.0\Common7\Tools\vsvars32.bat
```

بنابراین برای کار کردن دستور موردنظر شما کافیست که این دو دستور به صورت زیر در Post Build Event اضافه بشه:

```
call "$(DevEnvDir)..\Tools\vsvars32.bat"
"C:\Program Files\Microsoft Visual Studio 10.0\VC\bin\editbin.exe" /STACK:1000000 "$(TargetPath)"
```

نکته چهارم: با توجه به اشاره‌ای که در نکته قبلی شد ("با اجرای این فایل هر خط فرمانی تقریباً تبدیل به Visual Studio command prompt خواهد شد.") بنابراین دستور فوق را میتوان به صورت زیر خلاصه کرد:

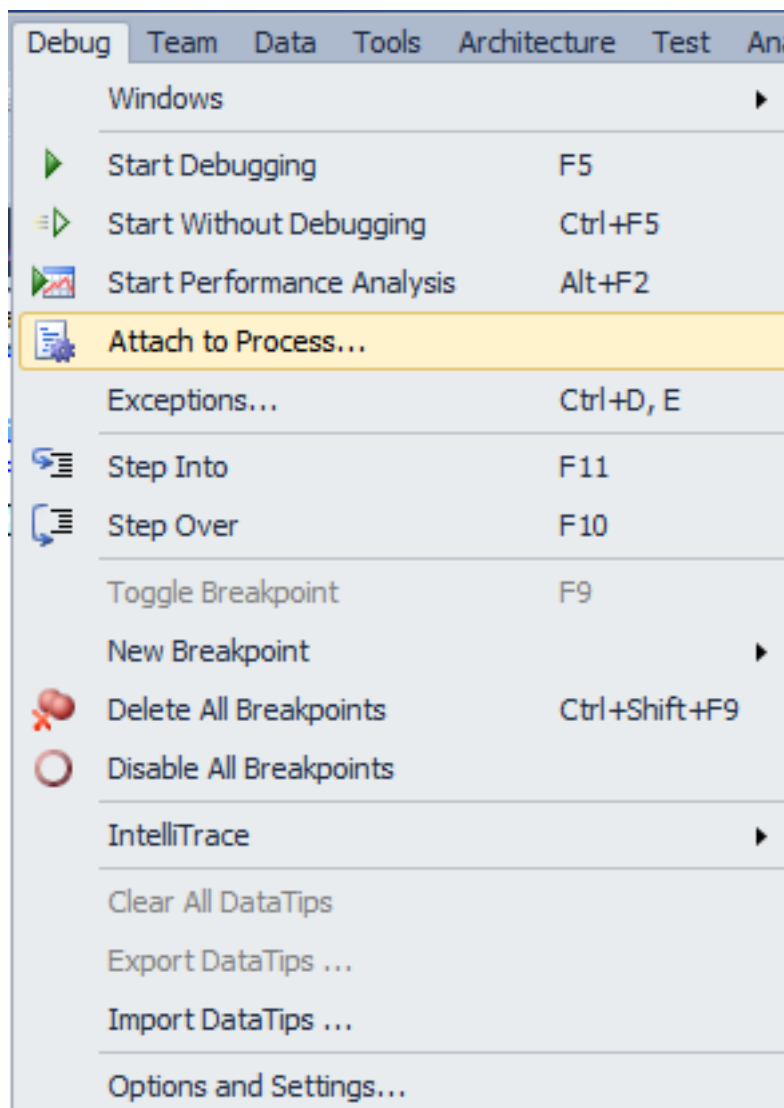
```
call "$(DevEnvDir)..\Tools\vsvars32.bat"
"editbin.exe" /STACK:1000000 "$(TargetPath)"
```

موفق باشید.

نویسنده: یوسف نژاد
تاریخ: ۹۵۳ ۱۳۹۲/۰۳/۰۷

نکته پنجم: پس از بررسی معلوم شد که اگر دستورات فوق ازطریق خط فرمان Build Events اجرا شوند استفاده از همان ماکروی \$(TargetFileName) نیز کفایت می‌کند.

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



کدام w3wp.exe مرتبط با Application جاری من است ؟



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

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

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



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



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

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

اکنون شناسه مشخص شده و می‌توانید به Debug کردن پردازش

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

[موفق باشید](#)

نظرات خوانندگان

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

سلام.

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

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

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

موفق باشید.

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

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

نویسنده: مهدی
تاریخ: ۱۳:۵۱ ۱۳۹۲/۰۱/۲۴

راه حلی دیگر

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

appcmd list wp
```

نویسنده: حمید
تاریخ: ۱۸:۳۱ ۱۳۹۲/۰۱/۲۹

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

ما در شرکت برای Source Control از SVN استفاده می‌کنیم، مزایای سورس کنترل آنقدر واضح است که دیگه من اینجا چیزی ازش نمیگم

اما برای استفاده از سورس کنترل یک مشکلی وجود دارد، اگر شما تعدادی پروژه را به کاربران خاصی بدین و تعدادی رو ندین، اون کاربر وقتی پروژه‌ها را می‌گیره با مشکل ارجاعات پروژه‌ها مواجهه است. چرا که برخی از پروژه‌های ارجاعی، روی کامپیوتر برنامه نویس 1 وجود نداره. برعکسش هم همین طوره، چون اون کاربر، پروژه‌های ارجاعی رو نداره، باید به جاش به اسمبلی نهایی اون پروژه ارجاع بده. بنابراین وقتی مدیر پروژه‌ها رو می‌گیره، باز ارجاعات اشتباه هستند!

ما اینجا برای رفع این مشکل ابزاری درست کردیم، به اسم SolutionExplorer.

این ابزار فایل solution رو به همراه پوشه حاوی فایل‌های اسمبلی می‌گیره. اگر پروژه ای به اسمبلی ای ارجاع داده باشه که پروژه اش توی solution باشه، ارجاع به اسمبلی رو تبدیل میکنه به ارجاع به پروژه و برعکسش، اگر پروژه ای به پروژه دیگه ای ارجاع داده باشه که توی solution وجود نداشته باشه، توی پوشه اسمبلی ها، دنبال اسمبلی ای می‌گرده که اسمش شبیه اسم پروژه ارجاعی باشه و اگر پیدا کنه، ارجاع رو عوض می‌کنه

البته برای جلوگیری از به هم ریختگی، نرم افزار از فایل‌های پروژه ای که دستکاری می‌کنه، پشتیبان می‌گیره [دانلود پروژه](#)

توجه:

- * این برنامه از تمامی جهات تست نشده است، با ریسک خودتون ازش استفاده کنید (ما تو شرکت دیگه ریسکی نداریم :)
- * سیستم نامگذاری اسمبلی‌ها و پروژه‌های ما ممکنه فرق کنه
- * اگر به مشکلی برخوردید، لطفا زیر همین مطلب برام بنویسید
- * انتخاب پوشه اسمبلی ها، الزامی نیست

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۱:۴۰ ۱۳۹۲/۰۲/۰۲

ممنون. ایده خوبی هست.

یک روش دیگر هم استفاده از نیوگت هست برای مدیریت لوکال وابستگی‌ها

[Creating and then using a NuGet local repository](#)

[How to access NuGet when NuGet.org is down](#)

نویسنده: بهزاد
تاریخ: ۱۱:۴۲ ۱۳۹۲/۰۲/۰۲

در مورد وابستگی‌های نوگت کاری نکردیم، فقط در مورد پروژه‌ها و اسمبلی‌هاست

نویسنده: محسن خان
تاریخ: ۱۲:۳۰ ۱۳۹۲/۰۲/۰۲

نیوگت لوکال روی شبکه می‌تونید تعریف کنید بر اساس وابستگی‌های داخلی خودتون. یعنی کاملاً مستقل از نیوگت روی اینترنت.

نویسنده: یوسف نژاد
تاریخ: ۸:۲۹ ۱۳۹۲/۰۲/۰۳

شما میتونین خروجی تمام پروژه‌های ریفرنس داده شده در پروژه‌های دیگه رو به یک مسیر مشخص و مشترک تنظیم کنید. تمام پروژه‌ها هم ریفرنس خودشون رو از اون مسیر مشخص بگیرن. سپس فایل‌های dll یا exe موردنظر رو بصورت multi-check out تنظیم کنید. بعدش هرکسی که آخرین نسخه از اون کتابخونه رو داره توسعه میده هر روز چکین کنه و بقیه هم هر روز get latest کنن. کاری که ما داریم به راحتی در شرکت خودمون انجام میدیم.

نویسنده: سیروس
تاریخ: ۱۳:۳۱ ۱۳۹۲/۰۲/۰۳

ما هم از این روش استفاده می‌کنیم.

نویسنده: منیژه محمدی
تاریخ: ۲۱:۱۳ ۱۳۹۲/۰۸/۲۶

در مورد TFS چطور ؟ در مورد ان پیشنهادی ندارید؟

در مطلب قبلی [Web.config File Transformation #1](#) با مفهوم انتقال وب کانفیگ و برخی از روش‌های آن آشنا شدید در ادامه به موارد دیگری خواهیم پرداخت.

قواعد انتقال وب کانفیگ

در کل دو ویژگی اصلی در انتقال وب کانفیگ وجود دارد که یک xdt:Transform و دیگری xdt:Locator می باشد. این دو در واقع چگونگی تغییر فایل انتقالی در زمان deploy آن را تعیین می‌کنند. این ویژگی‌ها از نوع xml می‌باشد که در فضای نام XML-Document- Transform تعریف شده و با پسوند xdt شروع می‌شود.

قاعده ویژگی Locator

این ویژگی برای جستجو در فایل وب کانفیگ استفاده می‌شود

:Condition

عبارت condition برای تعیین شرط قاعده Locator استفاده می‌شود

```
Locator="Condition(XPath expression)"
```

مثال:

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
      providerName="newprovider"
      xdt:Transform="Replace"
      xdt:Locator="Condition(@name='oldname'
        or @providerName='oldprovider')"/>
  </connectionStrings>
</configuration>
```

مثال بالا نشان می‌دهد که چگونه دنبال connectionstring ی بگردیم که نام آن oldname یا نام ارائه دهنده آن oldprovider باشد.

:Match

برای انتخاب عنصر یا عناصری که مقدار آن برابر ویژگی یا ویژگی‌های تعیین شده باشد. در صورتی که چندین ویژگی یافت شود، فقط عناصری که همه ویژگی‌های تعیین شده آن‌ها برابر باشد انتخاب می‌شود. نام ویژگی‌ها را می‌بایست با کما از هم جدا نمود.

```
Locator="Match(comma-delimited list of one or more attribute names)"
```

مثال:

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
      providerName="newprovider"
      xdt:Transform="Replace"
      xdt:Locator="Match(name)"/>
  </connectionStrings>
</configuration>
```

مثال فوق دنبال عناصری که ویژگی نام آنها برابر AWLT باشد می‌گردد.

:XPath

عبارتی از مسیرهای عناصر xml را جستجو می‌کند.

```
Locator="XPath(XPath expression)"
```

مثال:

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
      providerName="newprovider"
      xdt:Transform="Replace"
      xdt:Locator="XPath(configuration/connectionStrings[@name='AWLT'
        or @providerName='System.Data.SqlClient'])" />
  </connectionStrings>
</configuration>
```

در این مثال همانند مثال Condition همان عناصر را با استفاده از XPath جستجو می‌نماید.

قاعده ویژگی Transform

این ویژگی نحوه انتقال عنصر در فایل وب کانفیگ را مشخص می‌سازد.

:Replace

عناصر تعیین شده با عناصر انتقالی جایگزین می‌شود

```
Transform="Replace"
```

:Insert

عنصر انتقالی به عنوان فرزند عنصر تعیین شده اضافه می‌گردد.

```
Transform="Insert"
```

مثال:

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add name="AWLT" connectionString="newstring"
      providerName="newprovider"
      xdt:Transform="Insert" />
  </connectionStrings>
</configuration>
```

connectionstring انتقالی را به لیست رشته‌های اتصال فایل انتقالی اضافه می‌نماید.

:InsertBefore

عنصر انتقالی را قبل از مسیر تعیین شده با XPath قرار می‌دهد.

```
Transform="InsertBefore(XPath expression)"
```

مثال:

```
<configuration xmlns:xdt="...">
  <authorization>
    <allow roles="Admins"
      xdt:Transform="InsertBefore(/configuration/system.web/authorization/deny[@users='*'])" />
  </authorization>
</configuration>
```

عنصر انتقالی که همان allow می‌باشد که قبل از عنصر مسیر configuration/system.web/authorization/deny برای همه کاربر

است قرار می‌دهد.

:InsertAfter

عنصر تعیین شده را بعد از مسیر تعیین شده با XPath قرار می‌دهد.

```
Transform="InsertAfter(XPath expression)"
```

مثال:

```
<configuration xmlns:xdt="...">
  <authorization>
    <deny users="UserName"
      xdt:Transform="InsertAfter
        (/configuration/system.web/authorization/allow[@roles='Admins'])" />
    </authorization>
  </configuration>
```

:Remove

عنصر تعیین شده را از فایل انتقالی حذف می‌نماید. اگر چندین عنصر یافت شود اولین عنصر حذف خواهد شد.

```
Transform="Remove"
```

مثال

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add xdt:Transform="Remove" />
  </connectionStrings>
</configuration>
```

همه عناصر add عنصر connectionstring را انتخاب و اولین add را حذف می‌نماید.

:RemoveAll

عنصر یا عناصر تعیین شده را از فایل انتقالی حذف می‌نماید.

```
Transform="RemoveAll"
```

مثال:

```
<configuration xmlns:xdt="...">
  <connectionStrings>
    <add xdt:Transform="RemoveAll" />
  </connectionStrings>
</configuration>
```

همه عناصر add را از فایل انتقالی حذف می‌نماید.

:RemoveAttribute

ویژگی تعیین شده را از عناصر انتخاب شده حذف می‌نماید.

```
Transform="RemoveAttributes(comma-delimited list of one or more attribute names)"
```

مثال:

```
<configuration xmlns:xdt="...">
```

```
<compilation
  xdt:Transform="RemoveAttributes(debug,batch)">
</compilation>
</configuration>
```

ویژگی debug و batch را از عنصر compilation وب کانفیگ انتقالی حذف می‌نماید.

:SetAttribute

ویژگی تعیین شده را با مقادیر انتقالی مقدار دهی می‌کند.

```
Transform="SetAttributes(comma-delimited list of one or more attribute names)"
```

مثال:

```
<configuration xmlns:xdt="...">
  <compilation
    batch="false"
    xdt:Transform="SetAttributes(batch)">
  </compilation>
</configuration>
```

ویژگی batch وب کانفیگ انتقالی را با مقدار false مقدار دهی می‌کند.

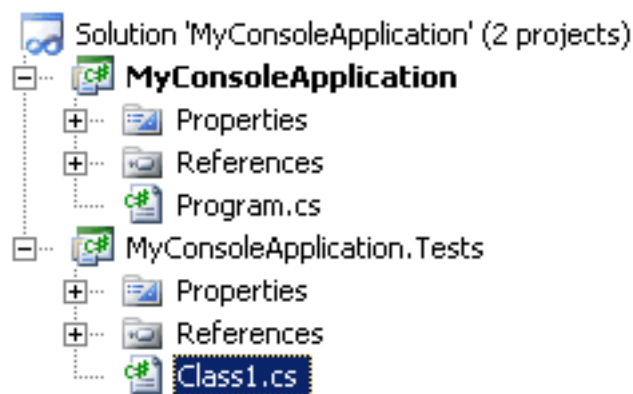
ادامه دارد...

در ویژوال استودیو ذیل منوی File، گزینه‌ای وجود دارد به نام Export template که کار آن تهیه یک قالب، بر اساس ساختار پروژه جاری است. این قابلیت جهت تهیه قالب‌های سفارشی، برای کاهش زمان تهیه پروژه‌ها بسیار مفید است. به این ترتیب می‌توان بسیاری از نکات مدنظر را، در یک قالب ویژه لحاظ کرد و به دفعات بدون نیاز به copy/paste مداوم فایل‌ها و تنظیمات اولیه، بسیار سریع یک پروژه جدید دلخواه را ایجاد نمود.

اما ... این قالب تهیه شده، صرفاً بر اساس یکی از چندین پروژه Solution جاری تهیه می‌شود و همچنین نصب و توزیع آن نیز دستی است. در ادامه قصد داریم با نحوه تهیه یک قالب جدید پروژه متشکل از چندین پروژه، به همراه تهیه فایل VSI نصاب آن، آشنا شویم.

تهیه یک ساختار نمونه

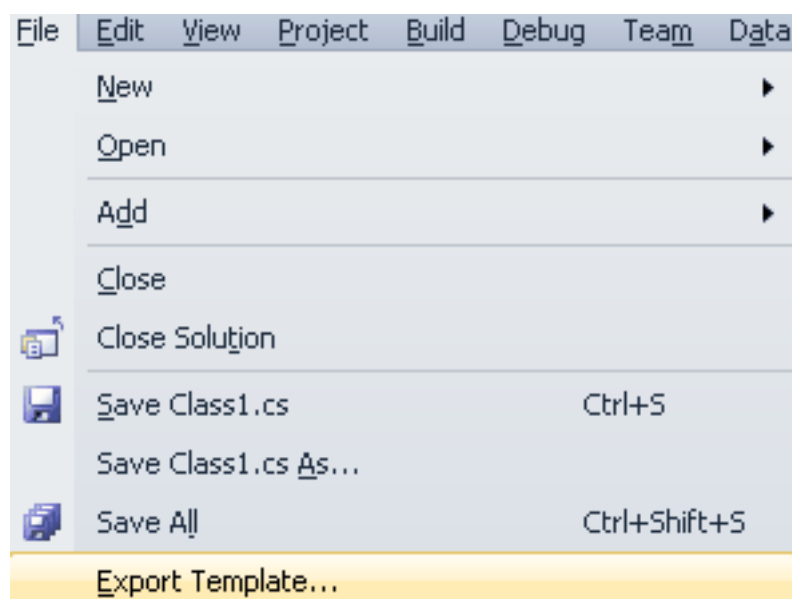
یک پروژه جدید کنسول را به نام MyConsoleApplication ایجاد کنید. سپس به Solution جاری، یک Class library جدید را به نام مثلاً MyConsoleApplication.Tests اضافه نمائید. تا اینجا به شکل زیر خواهیم رسید:



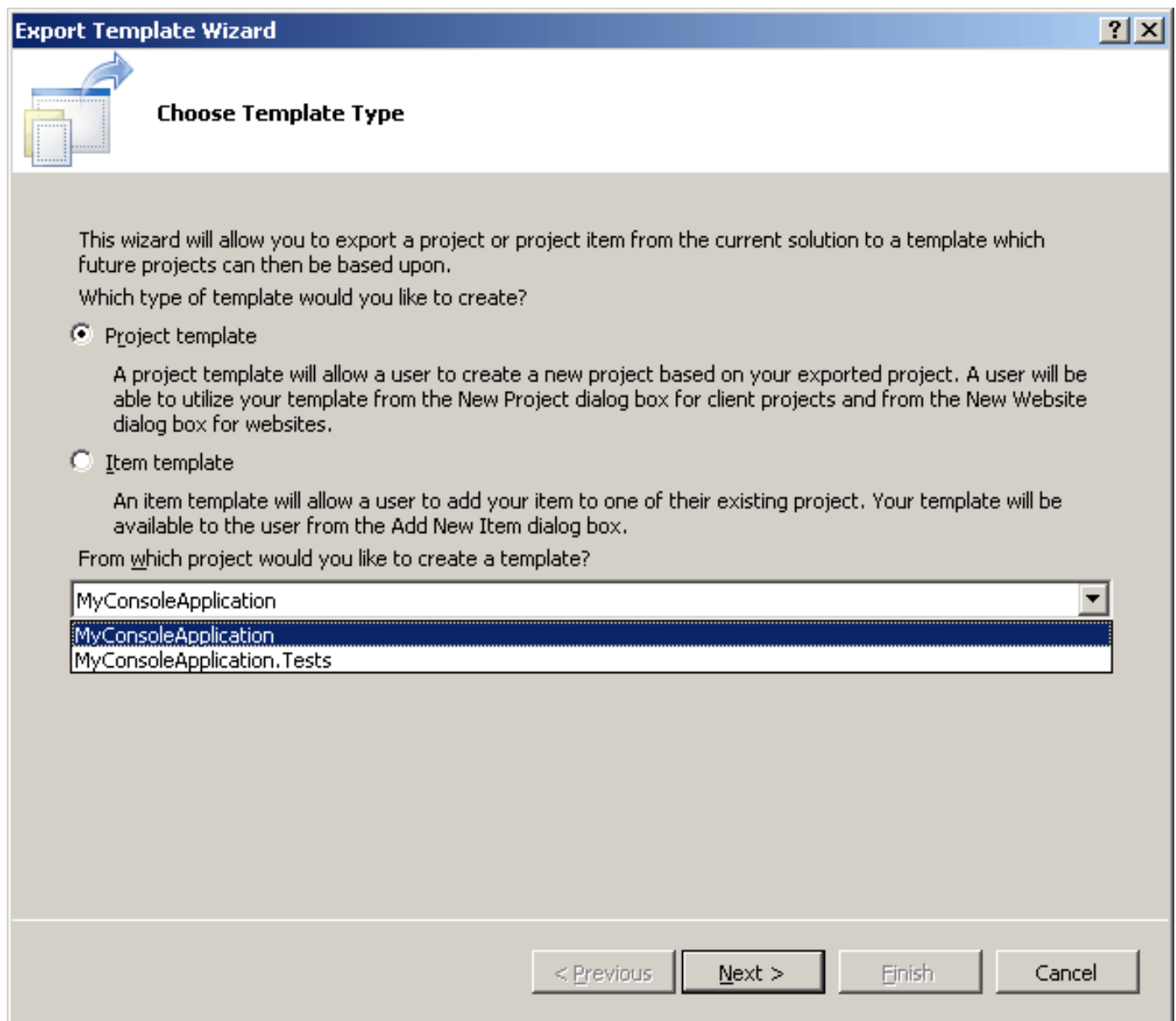
اکنون قصد داریم از این پروژه خاص، یک قالب تهیه کنیم؛ تا هر بار نخواهیم یک چنین مرحله‌ای را تکرار کنیم.

تهیه قالب به ازای هر پروژه در Solution

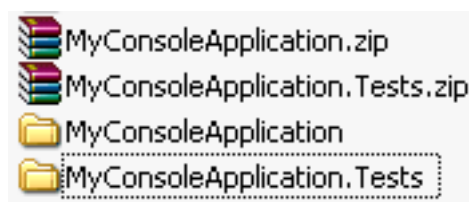
در همین حال که Solution باز است، به منوی File و گزینه Export template مراجعه کنید.



در اینجا تنها امکان انتخاب یک پروژه وجود دارد. به همین جهت این مرحله را باید به ازای هر تعداد پروژه موجود در Solution یکبار تکرار کرد.



اکنون در پوشه My Documents\Visual Studio 2010\My Exported Templates دو فایل zip به نام‌های MyConsoleApplication.Tests.zip و MyConsoleApplication.zip وجود دارند. هر دو فایل را توسط برنامه‌های مخصوص گشودن فایل‌های Zip گشوده و تبدیل به دو پوشه باز شده MyConsoleApplication.Tests و MyConsoleApplication کنید.



افزودن فایل MyTemplate.vstemplate چند پروژه‌ای

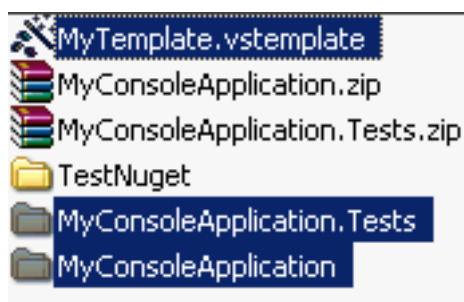
در همین پوشه جاری که اکنون حاوی دو پوشه باز شده است، یک فایل متنی جدید را با محتوای ذیل به نام

[MyTemplate.vstemplate](#) ایجاد کنید:

```
<VSTemplate Version="3.0.0" Type="ProjectGroup"
xmlns="http://schemas.microsoft.com/developer/vstemplate/2005">
  <TemplateData>
    <Name>MyConsoleApplication</Name>
    <Description>MyConsoleApplication Desc</Description>
    <ProjectType>CSharp</ProjectType>
  </TemplateData>
  <TemplateContent>
    <ProjectCollection>
      <ProjectTemplateLink ProjectName="MyConsoleApplication">
        MyConsoleApplication\MyTemplate.vstemplate</ProjectTemplateLink>
      <ProjectTemplateLink ProjectName="MyConsoleApplication.Tests">
        MyConsoleApplication.Tests\MyTemplate.vstemplate</ProjectTemplateLink>
      </ProjectCollection>
    </TemplateContent>
  </VSTemplate>
```

در اینجا به ازای هر پروژه، یک ProjectTemplateLink ایجاد خواهد شد که به فایل MyTemplate.vstemplate موجود در قالب آن اشاره می‌کند.

در ادامه این دو پوشه باز شده و فایل MyTemplate.vstemplate فوق را انتخاب کرده:



و همگی را تبدیل به یک فایل zip جدید کنید؛ مثلاً به نام MyConsoleApplicationTemplates.zip.

تهیه فایل نصاب از قالب پروژه جدید

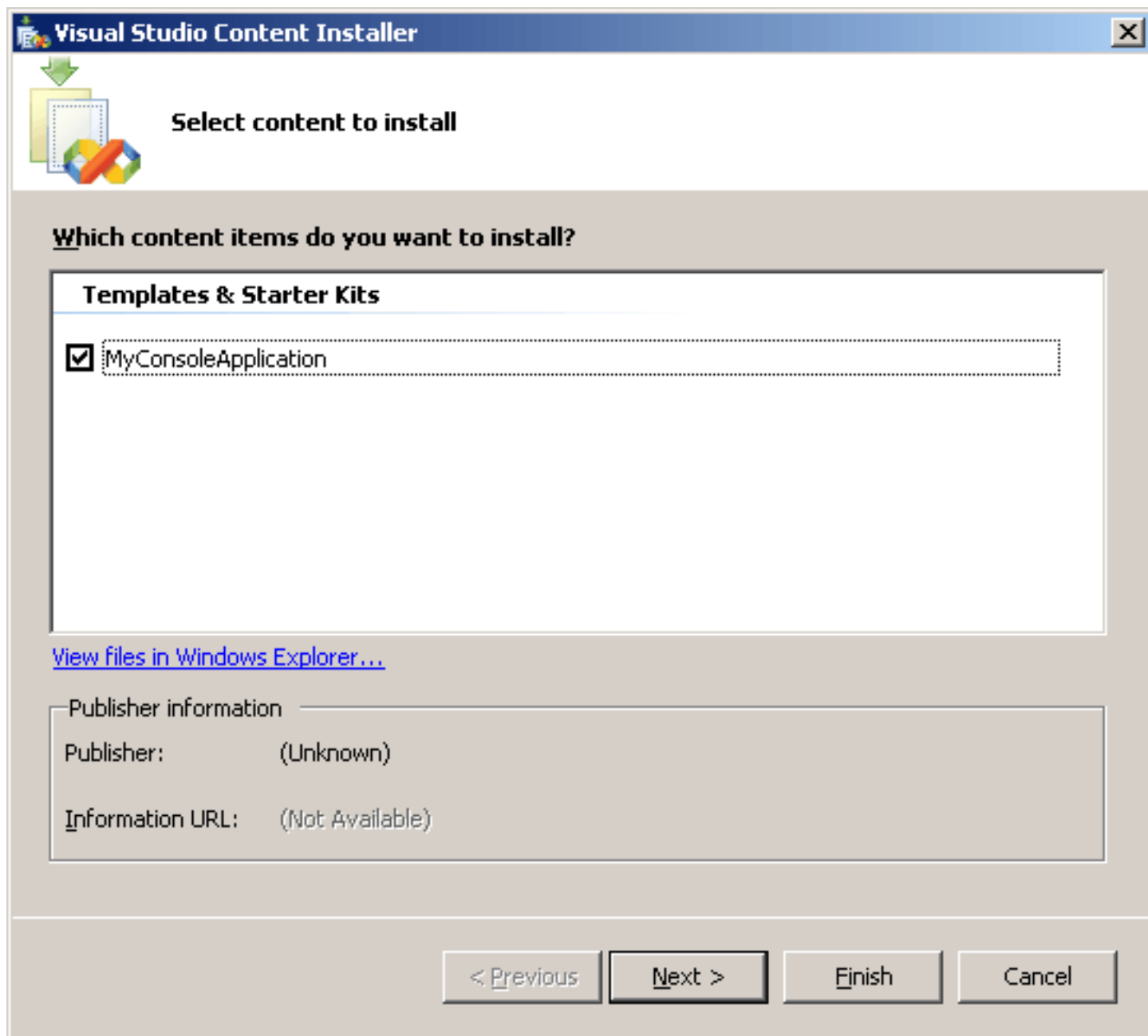
تا اینجا موفق شدیم، چندین قالب پروژه تهیه شده را به هم متصل کرده و تبدیل به یک فایل zip نهایی کنیم. مرحله بعد ایجاد فایلی است متنی به نام [MyConsoleApplicationTemplates.vscontent](#) با محتویات زیر:

```
<VSContent xmlns="http://schemas.microsoft.com/developer/vscontent/2005">
  <Content>
    <FileName>MyConsoleApplicationTemplates.zip</FileName>
    <DisplayName>MyConsoleApplication</DisplayName>
    <Description>A C# project that ...</Description>
    <FileContentType>VSTemplate</FileContentType>
    <ContentVersion>1.0</ContentVersion>
    <Attributes>
      <Attribute name="ProjectType" value="Visual C#" />
      <Attribute name="ProjectSubType" value="Web" />
      <Attribute name="TemplateType" value="Project" />
    </Attributes>
  </Content>
</VSContent>
```

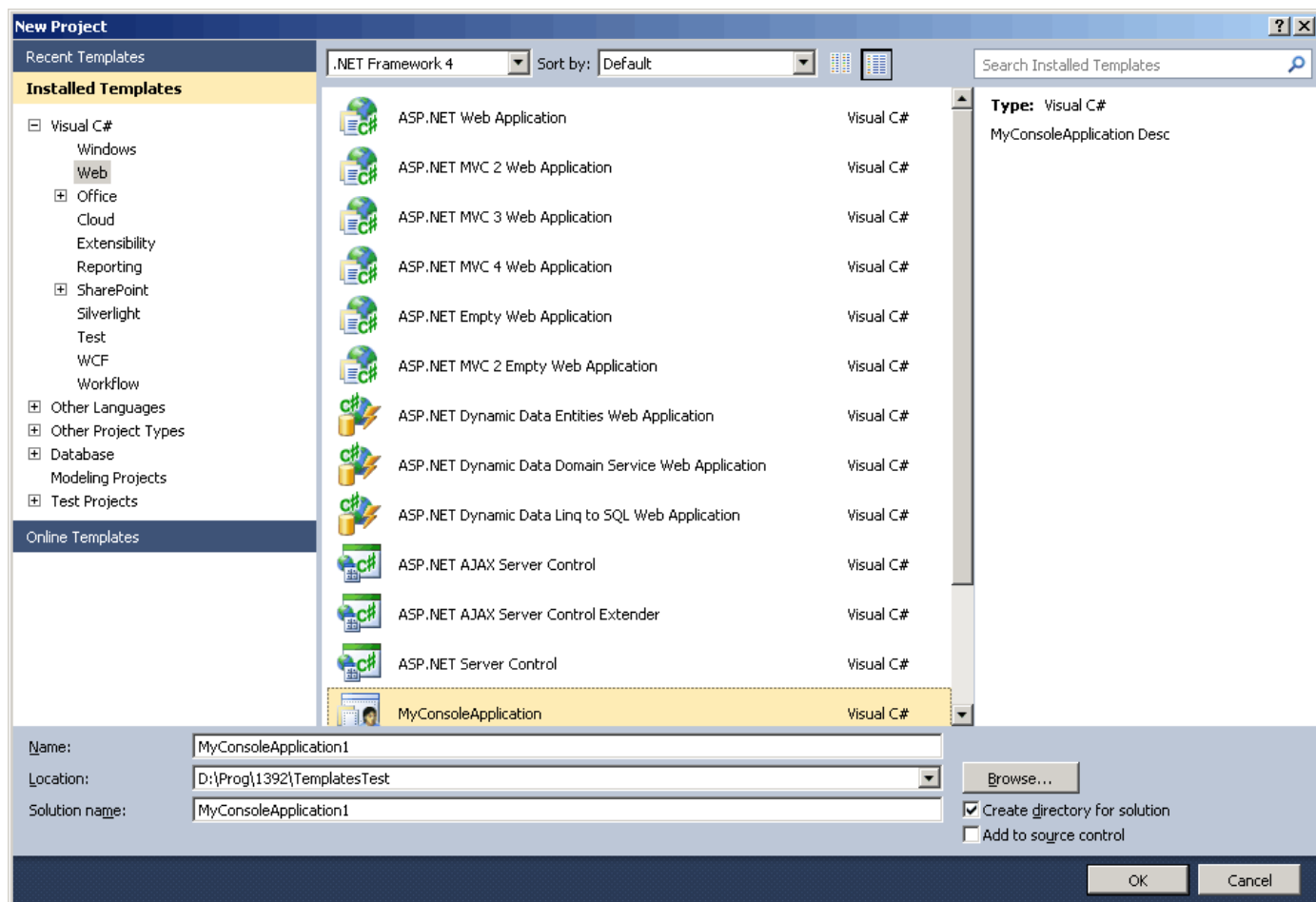
در اینجا توسط قسمت Attributes مشخص می‌کنیم که قالب پروژه جدید باید در صفحه new project، در کدام مدخل قرار گیرد. بنابراین مطابق تنظیمات فوق، قالب جدید ذیل پروژه‌های وب سی‌شارپ قرار خواهد گرفت. مقدار FileName آن دقیقاً معادل نام

فایل zip ایی است که در مرحله قبل ایجاد کردیم.

مرحله بعد انتخاب دو فایل `MyConsoleApplicationTemplates.zip` و `MyConsoleApplicationTemplates.vscontent` و تبدیل ایندو به یک فایل zip جدید است. پس از ایجاد فایل جدید، پسوند آنرا به VSI تغییر دهید؛ برای مثال نام آنرا به `MyConsoleApplicationTemplates.vsi` تغییر دهید. اکنون این فایل نهایی با دوبار کلیک بر روی آن قابلیت اجرا و نصب خودکار را پیدا می‌کند.



پس از نصب، بلافاصله ذیل قسمت پروژه‌های وب قابل دسترسی و استفاده خواهد بود:



بنابراین به صورت خلاصه:

- (1) به ازای هر پروژه، یک فایل قالب zip معادل آن باید تهیه شود.
- (2) تمام این فایل‌های zip را گشوده و تبدیل به پوشه‌های متناظری کنید.
- (3) یک فایل MyTemplate.vstemplate را در پوشه ریشه مرحله 2 جهت تعریف ProjectTemplateLink ها اضافه کنید.
- (4) فایل جدید MyTemplate.vstemplate مرحله 3 و تمام پوشه‌های قالب‌های باز شده مرحله 2 را zip کنید.
- (5) سپس یک فایل vscontent نصاب را تهیه و آنرا با فایل zip مرحله 4 مجددا zip کرده و پسوند آنرا به VSI تغییر دهید. اکنون می‌توان این فایل VSI را توزیع کرد.

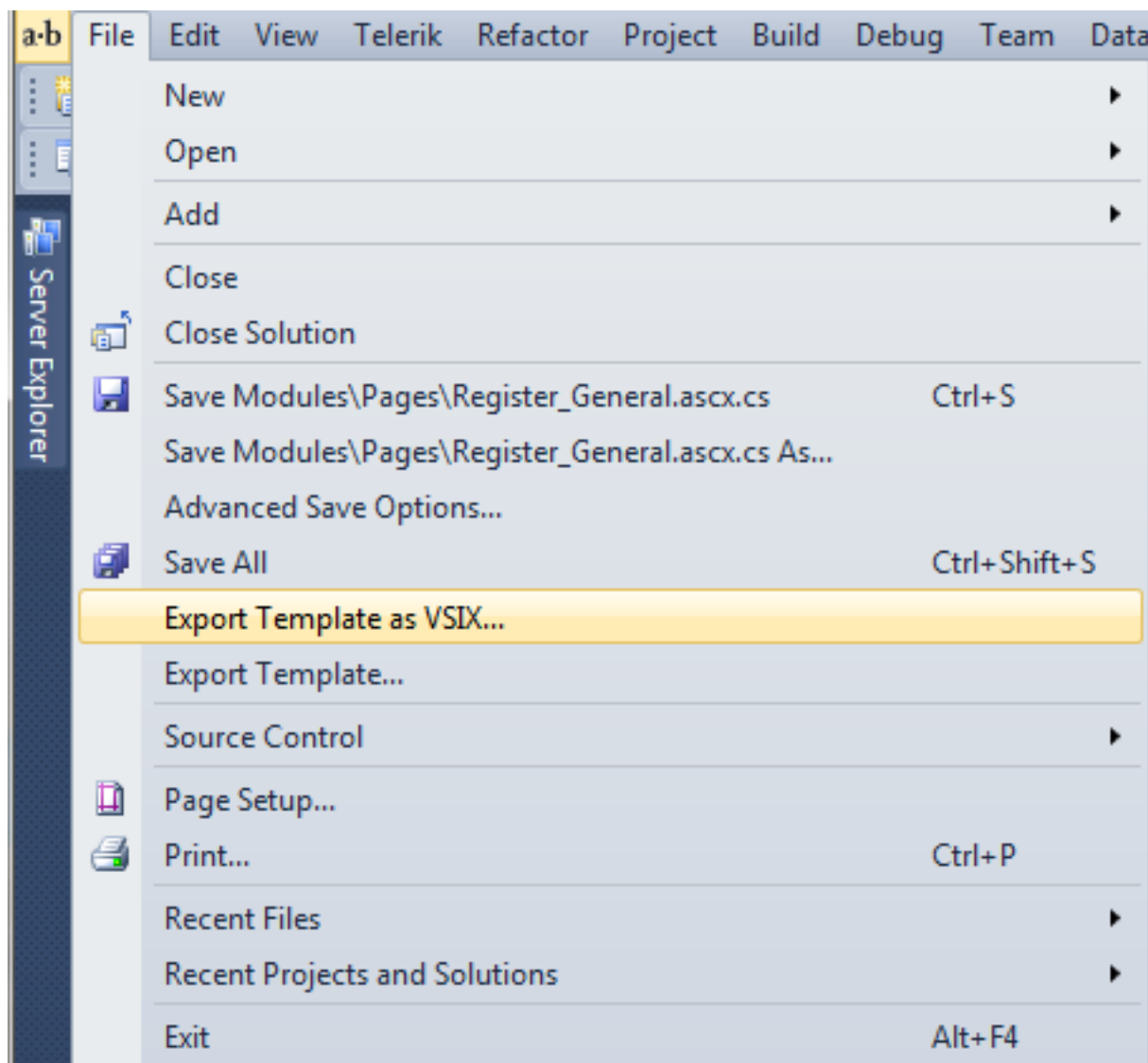
نظرات خوانندگان

نویسنده: مجتبی صحرائی
تاریخ: ۱۱:۵۷ ۱۳۹۲/۰۲/۲۹

سلام و ممنون

بنده از این روش استفاده کرده بودم و نهایتاً برای خودکار سازی این اعمال از افزونه [ExportTemplate\(vsix\).vsix](#) ویژوال استودیو استفاده کردم

طریقه استفاده اون هم به این صورت هستش که پس از نصب گزینه Export Template as VSIX... در منوی فایل ظاهر میشه و با کلیک بر روی اون تمامی پروژه‌های موجود در Solution جاری رو لیست می‌کنه و می‌تونید انتخاب کنید و Export کنید



نویسنده: مجتبی صحرائی
تاریخ: ۱۲:۲ ۱۳۹۲/۰۲/۲۹

نکته تکمیلی اینکه این امکان وجود دارد که پس از انتخاب پروژه(ها)، میتونید برای نصاب خودتون آیکون قرار بدید، لایسنس گذاری کنید و در ضمن ساخت نصاب عمل import شدن به VS هم به طور خودکار انجام بشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۹ ۱۳:۱۰

- ممنون. افزونه خیلی کاربردی و مفیدی است.
- البته در حالت دستی عنوان شده امکان تعریف آیکون و غیره هم هست. در متن، لینک داده شده به مراجع تولید فایل‌های vscontent و vstemplate که برای نمونه یک مدخل اضافه‌تر برای آیکون پیدا می‌کند :

<Icon>__Template_small.png</Icon>

در کل بد نیست یک برنامه نویس بدون پشت صحنه این اعمال به چه صورتی هست.

نویسنده: مجتبی صحرایی
تاریخ: ۱۳۹۲/۰۲/۲۹ ۱۳:۴۱

بله دقیقا با نظر شما موافقم

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۲/۰۲/۲۹ ۱۷:۵۳

سلام ... خیلی ممنون بابت این افزونه ...
گویا این افزونه روی VS 2012 کار نمیکنه! ... راهی هست که بشه کار کنه؟!

نویسنده: بهمن خلفی
تاریخ: ۱۳۹۲/۰۲/۲۹ ۱۸:۰۰

با سلام و عرض تشکر

یک سوال : اینکه بخواهیم از روشی که شما ارائه دادید استفاده کنیم ولی اگر بخواهیم بدین گونه باشد که :
پس از درست کردن قالب مد نظر پروژه بخواهیم یک ساختاری مانند پروژه MVC یا یک ساختار دلخواه داشته باشیم بصورتی که همواره می‌خواهیم در قالب پروژه ، 3 فولدر وجود داشته باشند که فولدر اول همانام پروژه + Module باشد : اگر نام پروژه هنگام ایجاد یا Add کردن Factor باشد یک فولدر در داخل همان پروژه جدید بنام FactorModule ایجاد شود و دو فولدر دیگه FactorAdmin و FactorUser در آن ایجاد شود و به همین ترتیب...؟!

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۹ ۱۸:۰۱

فایل extension.vsixmanifest [افزونه رو](#) باید ویرایش کنید (فایل vsix در اصل یک فایل zip است). مثلا VisualStudio
Version آن الان 10 است که باید بشود 11. بعد در SupportedFrameworkRuntimeEdition آن باید MaxVersion به 4.5 تنظیم شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۹ ۲۰:۰۴

امکان سفارشی سازی قالب ساز با کدنویسی هم میسر است. نیاز است اینترفیس IWizard پیاده سازی شود. در اینجا هر نوع کدی رو که لازم بود می‌شود در متد ProjectFinishedGenerating آن تدارک دید. مثلا پوشه درست کند، تنظیمات پروژه را تغییر دهد و امثال آن.

- یک مثال از پیاده سازی اینترفیس IWizard:

[Creating custom project template with wizard for Visual Studio](#)

- مثلاً پروژه sharp-architecture از [همین روش](#) استفاده می‌کند.

نویسنده: وحید نصیری
تاریخ: ۲۲:۴۳ ۱۳۹۲/۰۲/۲۹

یک نکته: روش دیگر ساخت قالب، استفاده از برنامه [Templify](#) است.

نویسنده: وحید نصیری
تاریخ: ۰:۲۱ ۱۳۹۲/۰۲/۳۰

یک نکته تکمیلی دیگر:

با نصب SDK ویژوال استودیو ([^](#) و [^](#)) یک قالب جدید تولید فایل‌های VSIX به مجموعه قالب‌های پروژه‌ها اضافه می‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۱:۴۰ ۱۳۹۲/۰۲/۳۰

یک نکته مهم!

اگر روش فوق را امتحان کنید (چه استفاده از افزونه یاد شده یا حتی روش دستی مقدماتی فوق)، هر نامی را که در ابتدای کار ایجاد Solution جدید وارد کنید، به زیر پروژه‌های اضافه شده اعمال نمی‌شود. یعنی همان نام ابتدایی خودشان را خواهند داشت که این مورد اصلاً جالب نیست.

برای رفع آن نیاز است از متغیری به نام \$safeprojectname\$ استفاده شود (هرجایی که نام پروژه به صورت مستقیم استفاده شده، حتی نام پوشه‌ها یا فایل‌ها) به همراه ReplaceParameters=true. یک مثال را در این مورد در پیوست ذیل می‌توانید دریافت کنید:

[MyConsoleApplicationTemplates.zip](#)

روش نصب دستی این قالب با کپی کردن آن در پوشه My Documents\Visual Studio xyz\Templates\ProjectTemplates است.

نویسنده: وحید نصیری
تاریخ: ۱۱:۴۴ ۱۳۹۲/۰۲/۳۰

و یا از \$safeprojectname\$ باید استفاده شود به روشی که [در اینجا توضیح دادم](#).

نویسنده: وحید نصیری
تاریخ: ۱۵:۲ ۱۳۹۲/۰۳/۰۴

نکات مطرح شده در این مطلب، تبدیل به یک پروژه شد: « [Solution template generator](#) »

نویسنده: ایلیا اکبری فرد
تاریخ: ۸:۴۵ ۱۳۹۲/۰۳/۳۱

با سلام.

یک پروژه class library در solution خود دارم. چگونه می‌شود فضای نام پیش فرض این پروژه و فایل‌های درون آنرا براساس \$safeprojectname\$ تغییر داد.

نویسنده: وحید نصیری
تاریخ: ۸:۵۶ ۱۳۹۲/۰۳/۳۱

- اینکار باید جداگانه (جدای از پروژه در حال کار) و به صورت دستی انجام شود (یک search و replace است).

- یا پروژه « [Solution template generator](#) » این کارها رو به صورت خودکار انجام می‌ده.

نویسنده: جواد جوادی
تاریخ: ۱۰:۴۹ ۱۳۹۴/۰۴/۰۲

سلام؛ در بخش آخر که Visual Studio Content Installer می باشد [طبق لینک](#) ارجاعی شما برای vs 2013 پشتیبانی ندارد در قسمت نسخه‌های پشتیبانی 2005 و 2008 و 2010 و 2012 است ولی 2013 نیست و در صورت اجرای فایل vsi. با خطای زیر مواجه می‌شویم :

```
Installation stoped because the directory for projectType value did not exsist .. the projectType is invalid for your installation of Visual Studio
```

در صورت حذف ProjectType/Attribute نصب انجام می‌شود ولی در لیست پروژه‌ها نمایش داده نمی‌شود.
با تشکر از شما

نویسنده: وحید نصیری
تاریخ: ۱۱:۰۰ ۱۳۹۴/۰۴/۰۲

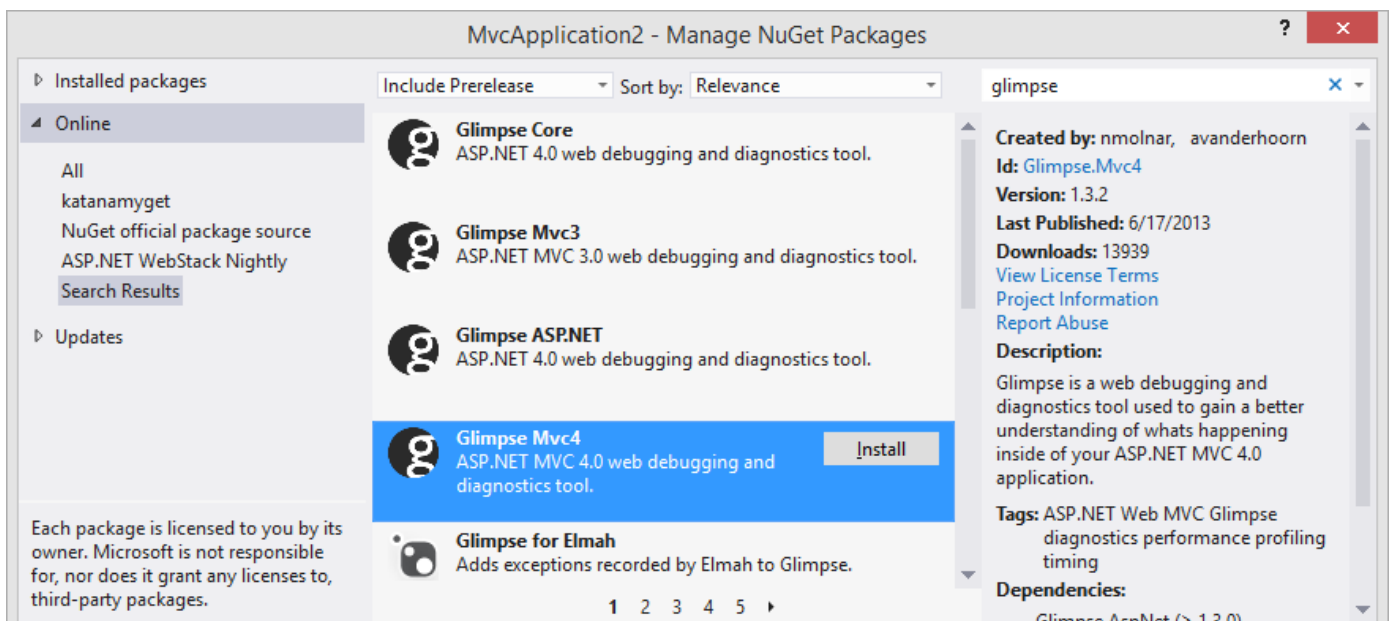
نیازی نیست تا این کارها را دستی انجام دهید. نکات مطرح شده در این مطلب، تبدیل به یک پروژه شد: « [Solution template generator](#) »

در مطلب [MiniProfiler](#) ابزار مانیتور کارایی وب سایت‌ها را بررسی کردیم. اما ابزار [Glimpse](#) هم جزو ابزارهای حرفه‌ای است که در مطلب آقای هانسلن در سایت خود به آن پرداخته بودند. اما دیدم جای یک مطلب فارسی در این رابطه خالی است.

Glimpse چیست؟

glimpse یک ابزار حرفه‌ای برای نمایش زمان اجرای کدها، پیکربندی سرور، درخواست‌های وب، اشکال زدایی و بررسی کارایی وب سایت‌های MVC و Web Forms می‌باشد. البته بدون آنکه در کدهای پروژه شما تغییری ایجاد نماید.

ابتدا در پنجره Nuget عبارت glimpse را جستجو و آن را نصب نمایید:



کتابخانه‌های زیادی برای این ابزار آماده شده‌اند:

کتابخانه Glimpse Core

که هسته اصلی ابزار است، حتما باید نصب شود.

کتابخانه Glimpse ASP.NET

برای بررسی وب سایت‌های نوشته شده با ASP.NET Web Forms استفاده می‌شود. البته برای MVC هم لازم است.

کتابخانه Glimpse Mvc2, Glimpse Mvc3, Glimpse Mvc4

برای بررسی وب سایت‌های نوشته شده با ASP.NET Mvc

کتابخانه Glimpse Ado

برای بررسی و نمایش زمان کوئری بر روی پایگاه داده

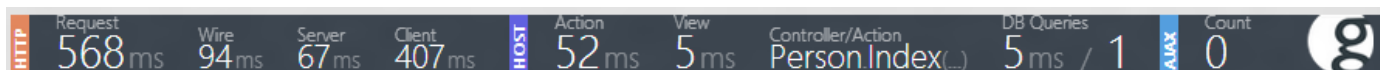
کتابخانه Glimpse EF4.3, Glimpse EF5, Glimpse EF6

برای زمانیکه از نگارش‌های مختلف Entity Framework استفاده می‌نماییم

پس از نصب کتابخانه‌های مورد نیاز، پروژه را rebuild و سپس اجرا نمایید. برای فعال کردن glimpse آدرس <http://your-site/Glimpse.axd> را اجرا کنید تا صفحه تنظیمات آن فعال شوند و سپس بر روی گزینه Turn Glimpse on، کلیک کنید. همچنین با گزینه Turn Glimpse off می‌توانید آن را غیر فعال نمایید.



علاوه بر این، تنظیمات استاندارد این ابزار قابل تغییر است. به صفحه اصلی سایت برگشته و صفحه را بروز رسانی کنید. ابزار glimpse در پایین مرورگر نمایش داده می‌شود.



این ابزار شامل سه قسمت است:

HTTP

اطلاعات Request و زمان پاسخ و اطلاعات سرور نمایش داده می‌شود

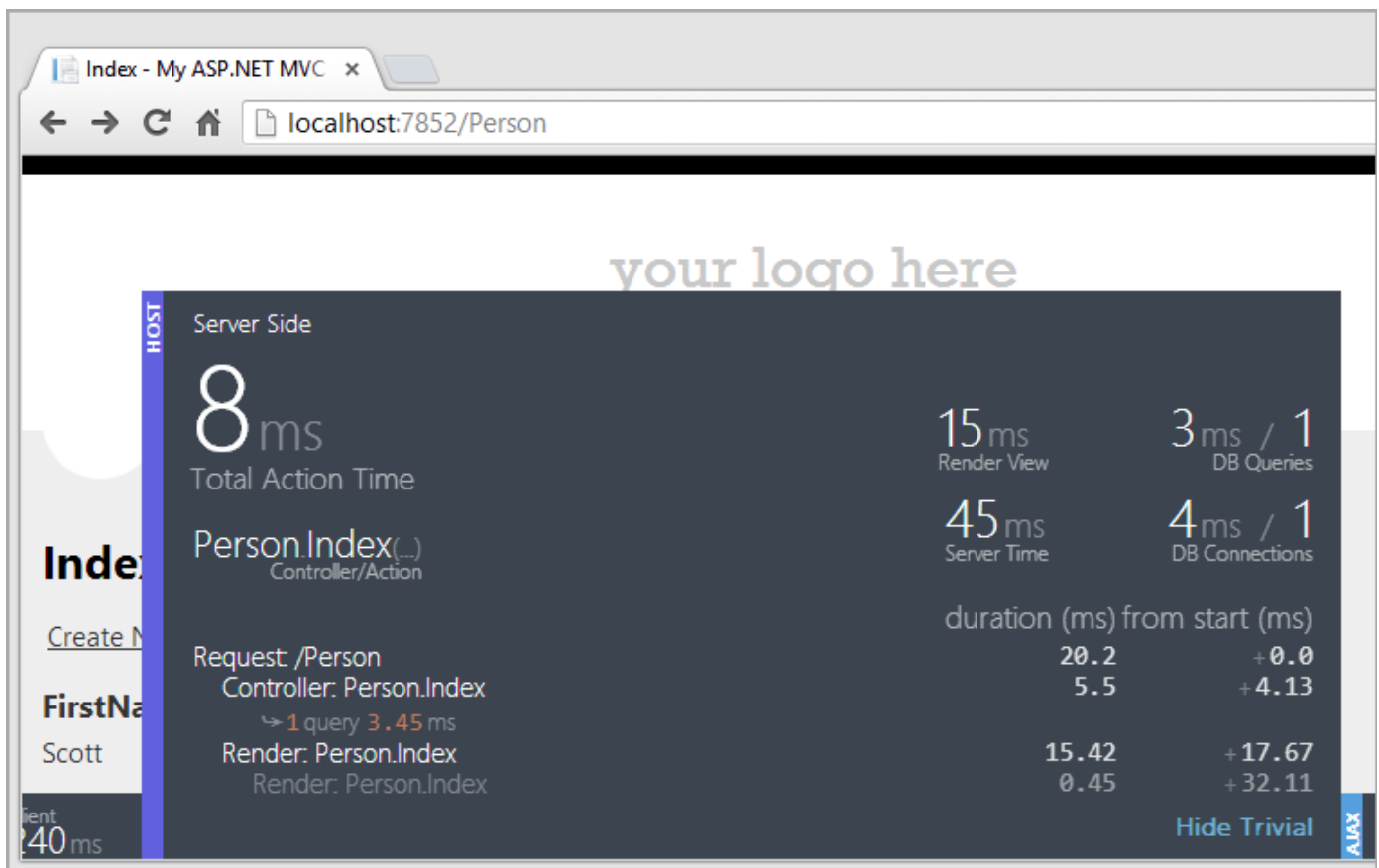
HOST

اطلاعات صفحه اجرا شده، زمان پاسخ و تعداد کوئری‌های اجرا شده و زمان آن نمایش داده می‌شوند

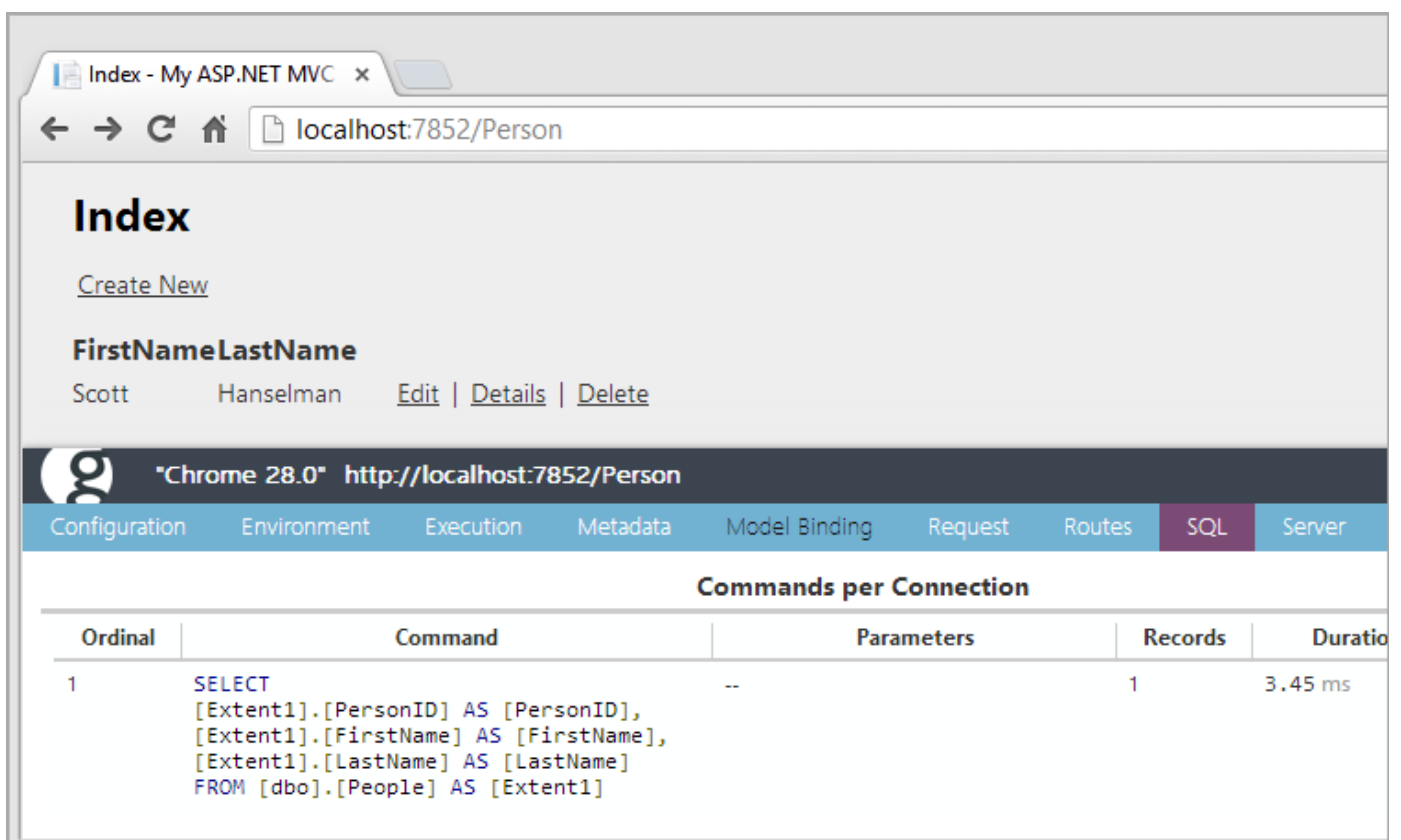
AJAX

اطلاعات درخواست‌های اجکسی این صفحه و تعداد آن نمایش داده می‌شوند

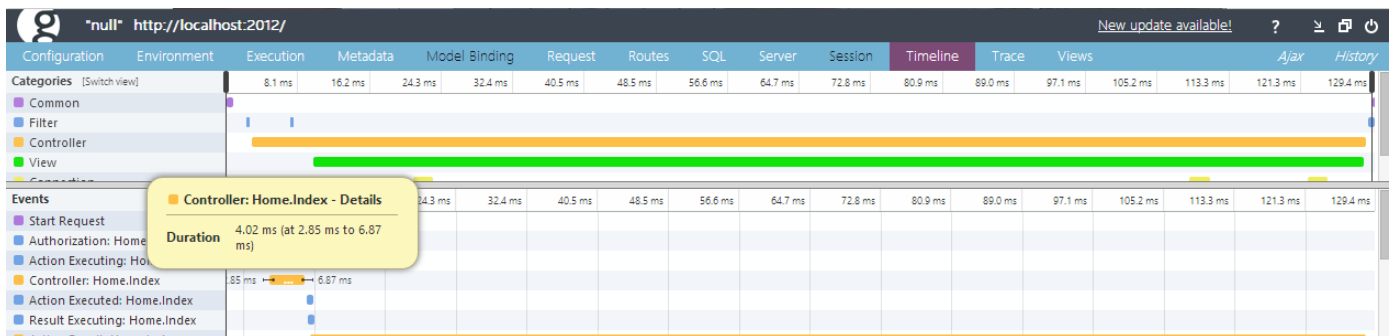
بر روی هر یک از این قسمت‌ها با حرکت ماوس، جزئیات آن قسمت نمایش داده می‌شود.



اگر بر روی آیکون g ابزار کلیک کنید، همچون developer tools مرورگرها باز شده و دارای زبانه‌های متعددی می‌باشد. مثلاً اگر پلاگین ado و ef5 نصب باشند، در زبانه SQL می‌توانید کوئری‌های اجرا شده و زمان مصرف شده آن‌ها را مشاهده نمایید



زبانه دیگر Timeline است که زمان انقباد اشیاء و رویدادها را بصورت گرافیکی نمایش می‌دهد.



در مطلب بعدی به جزئیات بیشتری از این ابزار می‌پردازم.

نظرات خوانندگان

نویسنده: ali

تاریخ: ۱۳۹۲/۰۵/۰۷ ۱۲:۵۶

ممنون بابت این مطلب مفید.

هنگام آپلود سایت اگر نخواهیم این ابزار کار کند باید قبلش uninstall کنیم؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۰۵/۰۸ ۱۰:۵۰

نه الزاما. میشه در وب کانفیگ [غیرفعالش کرد](#).

نویسنده: رضا گرمارودی

تاریخ: ۱۳۹۲/۱۲/۰۳ ۱۲:۰۹

glimpse و Miniprofiler هر دو با Ef6 مشکل دارند. گرچه در سایت‌های برنامه‌های فوق عنوان شده که Ef6 را پوشش میدهند اما هر کدام به نحوی باگی دارند. از اونجایی که در Ef6 با Rdbms اسکيوال CE کار می‌کنم و همانند Sql server پروفایلری نداره که دستورات ارسالی را بشه دید شما در Ef6 به غیر از دو پروفایل ذکر شده از چه پروفایلری استفاده می‌کنید؟

نویسنده: وحید نصیری

تاریخ: ۱۳۹۲/۱۲/۰۳ ۱۳:۰۸

روش « [نمایش خروجی SQL کدهای Entity framework 6 در کنسول دیباگ ویژوال استودیو](#) » نیاز به ابزار اضافی ندارد.

نویسنده: هیمین صادقی

تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۴:۵۵

سپاس از مطب شما

زمانی که با entity framework 6 استفاده می‌کنیم

خطا زیر رو می‌ده راه حل برایش مشکل وجود دارد

No Entity Framework provider found for the ADO.NET provider with invariant name 'System.Data.Odbc'. Make sure the provider is registered in the 'entityFramework' section of the application config file. See <http://go.microsoft.com/fwlink/?LinkId=260882> for more information

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۵:۰۴

برای EF 6 [بسته جداگانه](#) دارد:

PM> Install-Package Glimpse.EF6

نویسنده: هیمین صادقی

تاریخ: ۱۳۹۳/۰۴/۲۸ ۱۵:۱۲

با نصب این بسته بازم خطا رخ می‌ده

در سایت‌های مختلف جستجو کردم پاسخ مناسب پیدا نکردم

نویسنده: وحید نصیری
تاریخ: ۱۷:۱۴ ۱۳۹۳/۰۴/۲۸

- محل گزارش خطاهای این پروژه

+ در EF 6 فایل کانفیگ برنامه حتما باید ویرایش شود و تعریف پروایدر را داشته باشد ([_](#) و [_](#))؛ مثلا:

```
<entityFramework>
  <providers>
    <provider invariantName="System.Data.Odbc"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
    <provider invariantName="System.Data.SqlClient"
type="System.Data.Entity.SqlServer.SqlProviderServices, EntityFramework.SqlServer" />
  </providers>
</entityFramework>
```

نویسنده: علی یگانه مقدم
تاریخ: ۲۰:۴۰ ۱۳۹۴/۰۳/۱۰

با تشکر از مطالبی که ارئه کردید
شما زحمت دو مطلب در این زمینه رو کشیدید
خواستم بدونم که شما تجربه عملی کار با این دو ابزار را دارید
به نظر شما کدام ابزار برای انتخاب بهتر هست؟
ابزاری که در این مقاله معرفی کردید یا miniprofiler

نویسنده: مجتبی کاویانی
تاریخ: ۱۳:۴۳ ۱۳۹۴/۰۳/۱۱

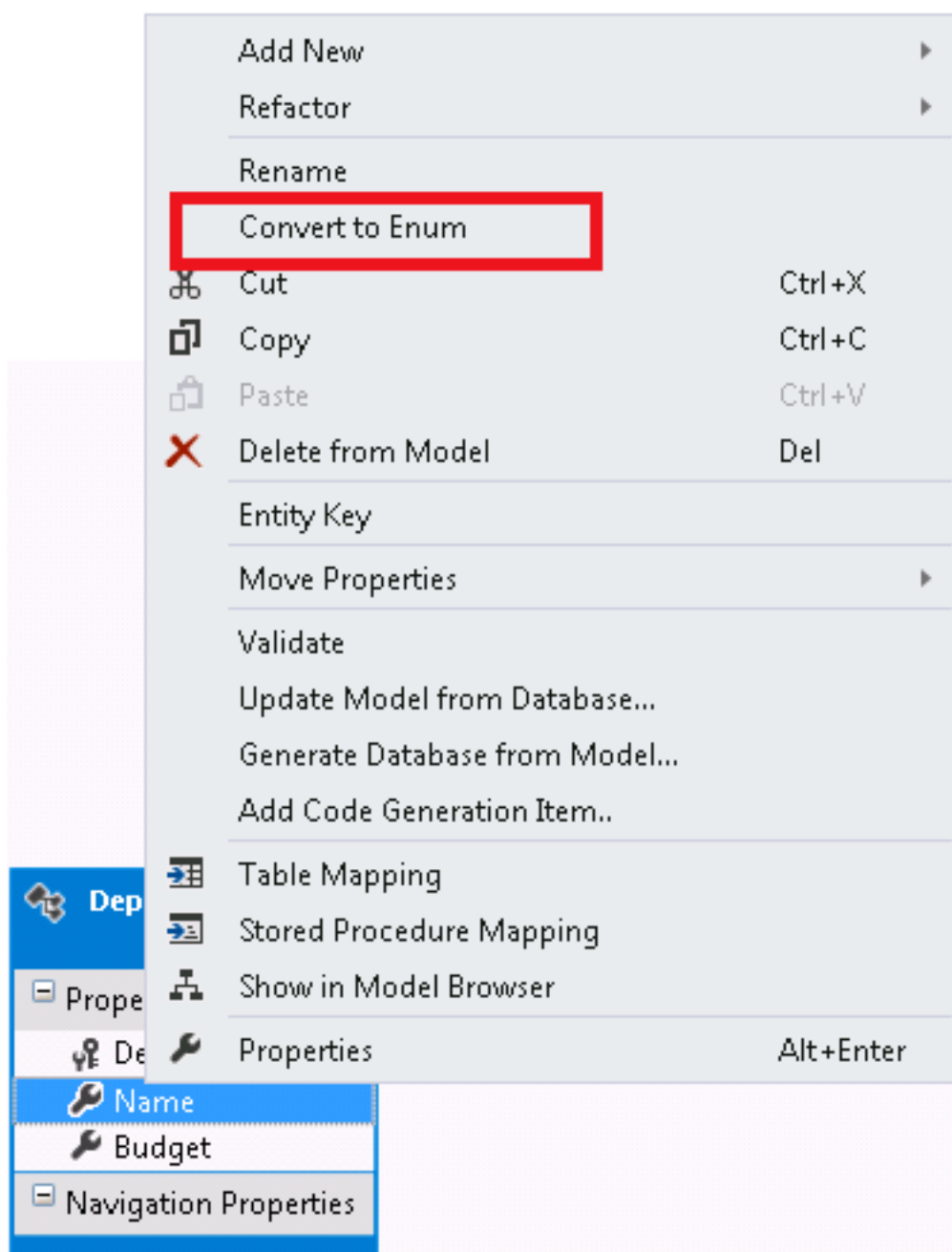
ابزار Glimpse خیلی حرفه ای تر است حتی امکان استفاده MiniProfiler بصورت پلاگین در آن نیز وجود دارد

Install-Package Glimpse.Miniprofiler

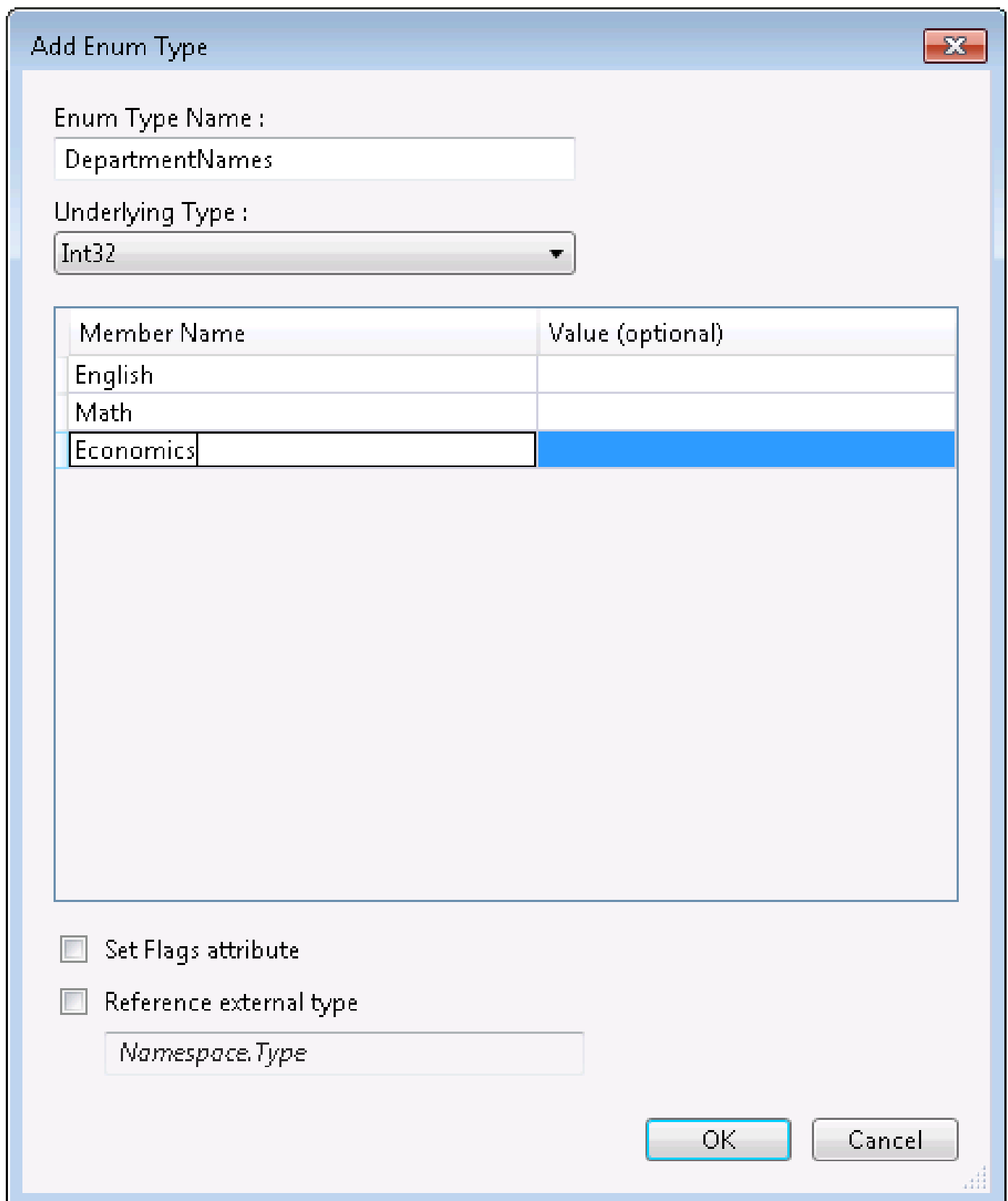
ویرایش 2012 ابزار Visual Studio جهت کار با EF امکانات جدیدی دارد که سعی دارم به طور خلاصه چند مورد آنرا توضیح دهم.

پشتیبانی از Enum

در نسخه‌های قبل از EF 5 پشتیبانی توکاری از Enumها وجود نداشت و برنامه نویس برای استفاده از آنها مجبور بود از روش‌های دیگری استفاده کند؛ مانند [^](#) استفاده کند. در نسخه 5 این امکان بر راحتی قابل اعمال است. بدین منظور کافی است: 1- در Designer بر روی خصوصییتی که قصد تبدیل آنرا به enum دارید راست کلیک کرده و گزینه Convert to enum را انتخاب کنید.



2- در پنجره Add enum ابتدا در قسمت Underlying type نوع Int32 را انتخاب کنید سپس می‌توان نام enum و اعضای آنرا و تعیین کرد.



Add Enum Type

Enum Type Name :
DepartmentNames

Underlying Type :
Int32

Member Name	Value (optional)
English	
Math	
Economics	

☐ Set Flags attribute

☐ Reference external type

Namespace.Type

OK Cancel

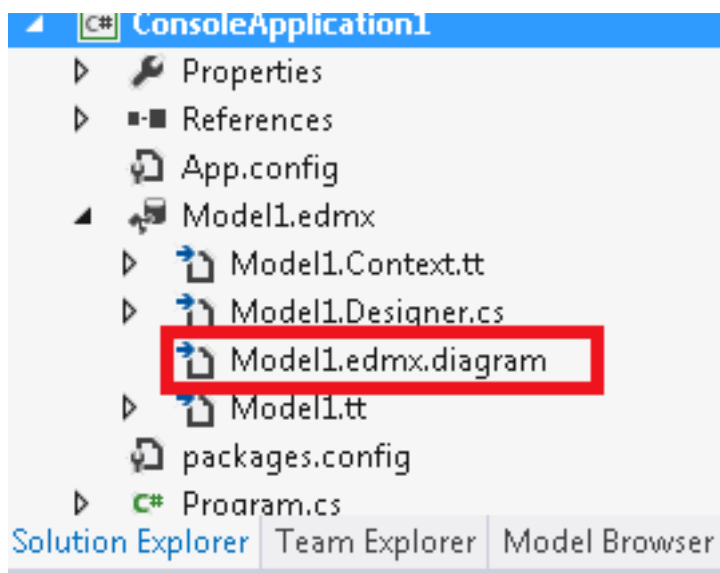
3- دکمه ok را کلیک کنید و سپس پروژه را Build کنید.

در ادامه به راحتی می‌توان از آن در برنامه به صورت زیر استفاده کرد:

```
var department = (from d in context.Departments
                  where d.Name == DepartmentNames.English
                  select d).FirstOrDefault();
```

تقسیم یک مدل در Entity Framework به چند دیاگرام

هنگامی که یک دیاگرام جدید ایجاد می‌کنید این دیاگرام به طور پیش فرض با نام Diagram1 به پوشه Diagrams اضافه می‌شود. اطلاعات مربوط به ظاهر موجودیت مانند رنگ و شکل و روابط آنها نیز در فایل با پسوند edmx.diagram قرار می‌گیرند. شما بصورت دستی نمی‌توانید این فایل را تغییر دهید چون اطلاعات آن دوباره توسط جنریتور رونویسی می‌شود. لذا تغییر در دیاگرام به روش دستی مورد اطمینان نیست!



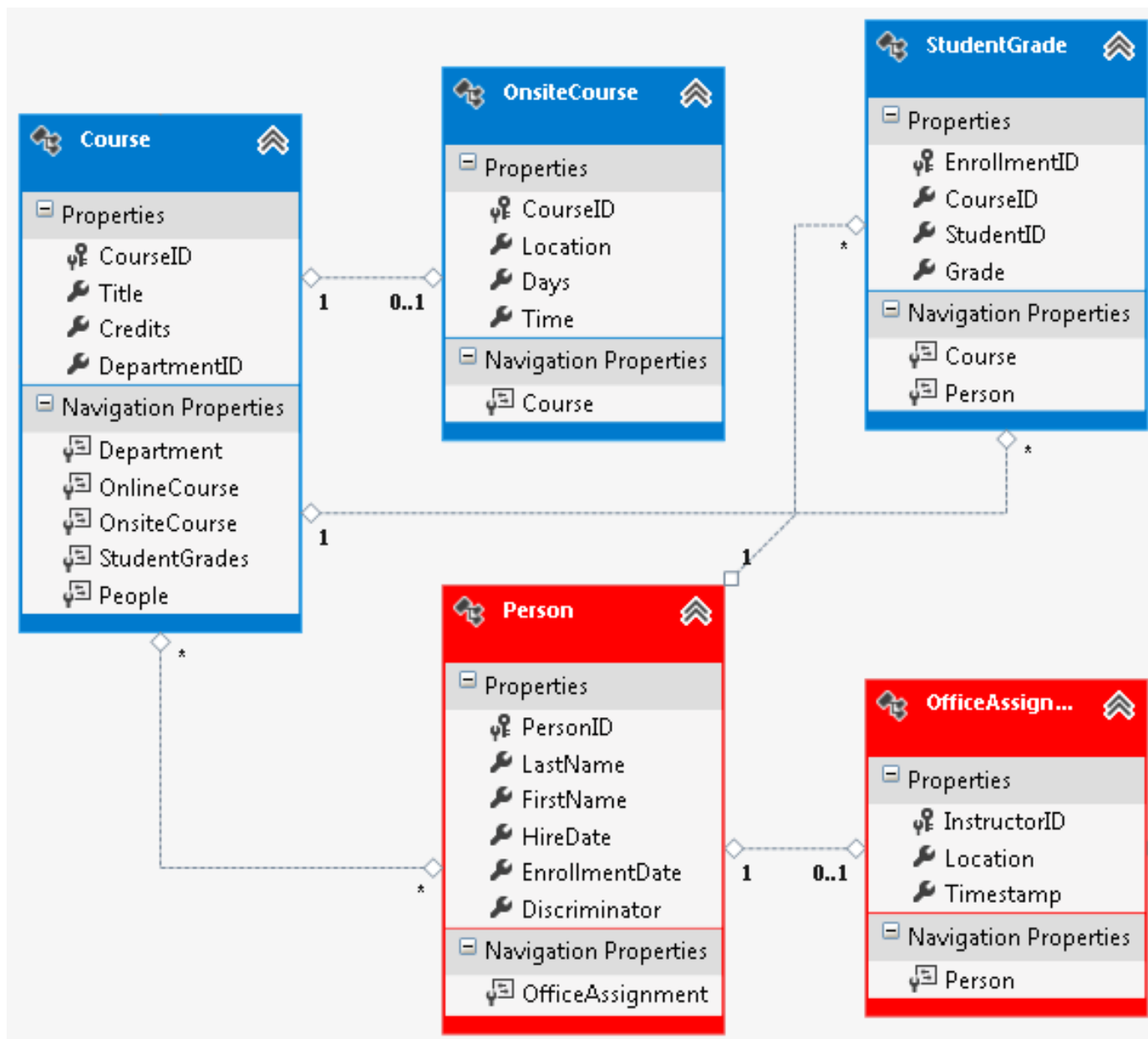
حتما برای کسانی که از EF Designer استفاده می‌کنند پیش آمده که بخواهند مدل موجودیت هایشان را بجای یک فایل در چند فایل قرار دهند. اینکار مخصوصا زمانی که تعداد موجودیت‌ها زیاد است لازم به نظر می‌رسد بعلاوه اینکه مدیریت و مرور موجودیت‌ها را در پروژه‌های بزرگ آسانتر می‌کند. خوشبختانه این امکان در Visual Studio 2012 ایجاد شده است.

بدین منظور در دیاگرام برنامه موجودیت هایی را که می‌خواهید به دیاگرام دیگری انتقال دهید با کلیک و شیفت انتخاب کنید. سپس راست کلیک کرده و گزینه Move to new Diagram را انتخاب کنید. دیاگرام جدیدی ایجاد شده و موجودیت انتخاب شده به آنجا انتقال داده می‌شود. بهتر است موجودیت هایی که برای انتقال انتخاب کرده اید به صورت یک گروه مستقل باشند یعنی با موجودیت‌های دیگر رابطه نداشته باشند.

کار انتقال به یک دیاگرام جدید را می‌توان به کمک کلیدهای Ctrl+C و Ctrl+X نیز انجام داد، باید توجه داشت در حالتی که موجودیت را کپی می‌کنید، نام موجودیت جدید با اضافه شدن یک عدد از موجودیت موجود جدا می‌شود.

تغییر رنگ موجودیت

روش دیگری که جهت متمایز و جدا کردن موجودیت‌ها می‌توان از آن استفاده کرد، تغییر رنگ آنهاست. بدین منظور پس از انتخاب موجودیت‌ها می‌توانید با تغییر مقدار Fill Color در پنجره Properties رنگ موجودیت‌های انتخاب شده را تغییر داد.



نظرات خوانندگان

نویسنده:

میثم

تاریخ:

۱۷:۵۵ ۱۳۹۲/۰۵/۱۴

پشتیبانی از Enum در linq2sql وجود ندارد؟

نویسنده:

سیروس

تاریخ:

۲۳:۱۸ ۱۳۹۲/۰۵/۱۴

به شکل مطرح شده خیر، اما بصورت دستی می‌توان اینکار را انجام داد. به [^](#) و [^](#) مراجعه کنید.

عنوان: نحوه نمایش منوهای Visual studio 2012 با حروف کوچک

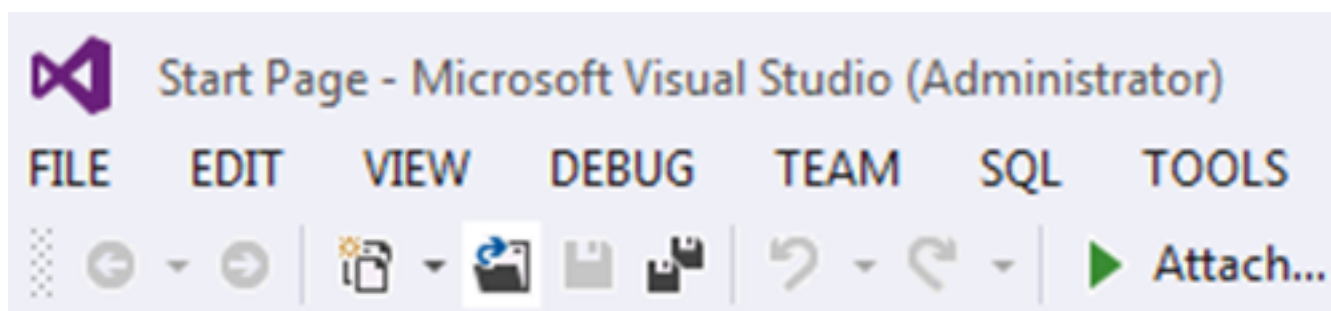
نویسنده: بهمن خلفی

تاریخ: ۱۶:۵۵ ۱۳۹۲/۰۵/۱۵

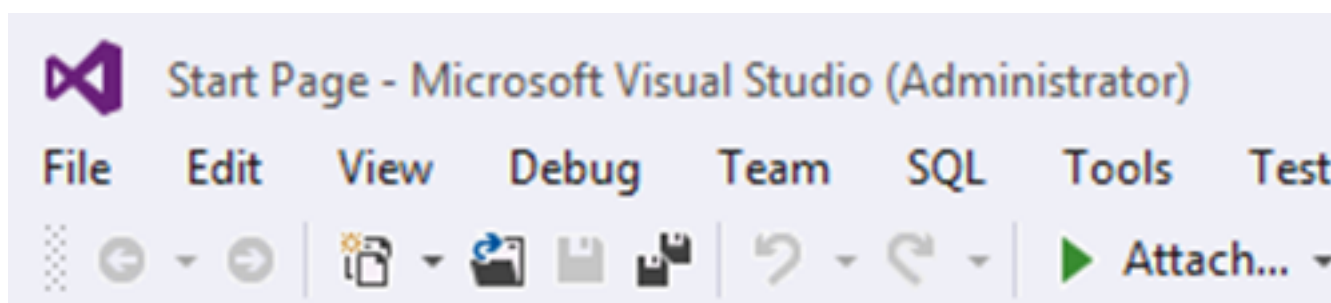
آدرس: www.dotnettips.info

برچسب‌ها: Tips, Visual Studio

چند روز پیش بصورت اتفاقی به این فکر افتادم که چرا منوهای visual studio 2012 برخلاف ظاهر زیبای خود محیط، اینقدر زمخت و با حروف بزرگ نوشته است.



و اینکه به چه صورت میتوانم آنها را بصورت حروف کوچک نمایش دهم و حس کنجکاوی اونم از نوع مخصوص گل کرد.



برای اینکار دو روش وجود دارد :

روش 1 - تغییر مقدار در رجیستری سیستم عامل ویندوز

بدین صورت که شما باید به این مسیر مراجعه نموده

در ویندوز 7 :

```
HKEY_CURRENT_USER\Software\Microsoft\VisualStudio\11.0\General\
```

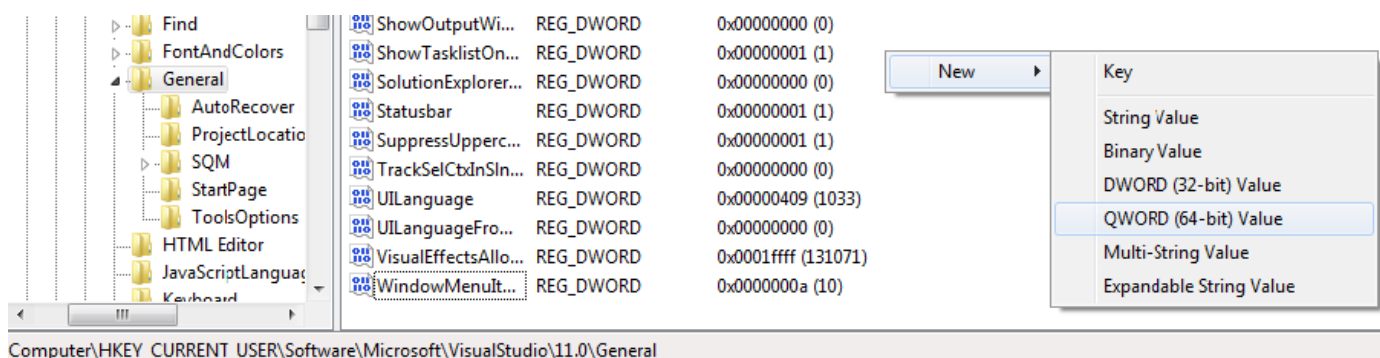
در ویندوز 8 :

```
HKEY_CURRENT_USER\Software\Microsoft\VSWinExpress\11.0\General\
```

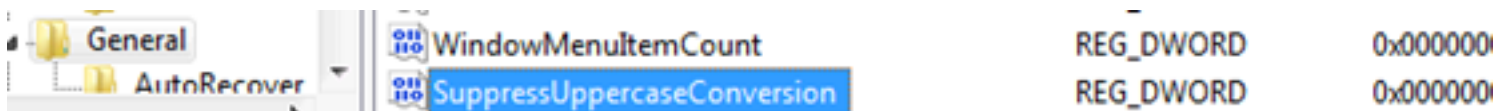
در نسخه web express :

```
HKEY_CURRENT_USER\Software\Microsoft\VSWebExpress\11.0\General\
```

ایجاد یک کلید از نوع DWORD :



و با نام SuppressUppercaseConversion و با مقدار 1 در مسیر یاد شده تنظیم نمائید.

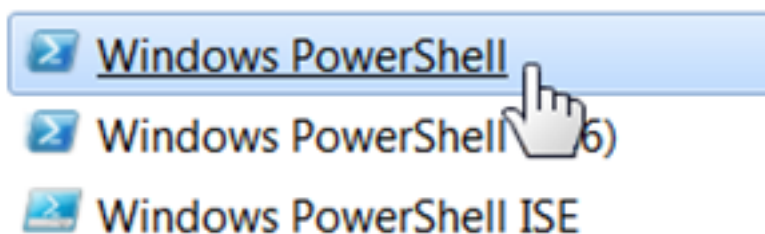


سپس راه اندازی مجدد visual studio و مشاهده منوهای تغییر یافته .

روش 2 - کسانی مثل من کمی تنبل هستند و از این کارهای فوق دوست ندارند راه آسانتر را میتوانند تجربه کنند بصورت ذیل:

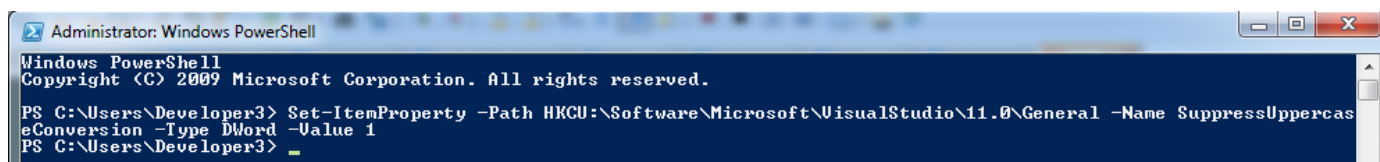
در منوی start ویندوز با تایپ کلمه powershell و انتخاب Windows PowerShell به صفحه‌ای آبی رنگ (در ویندوز 7) وارد میشود .

Programs (5)



دستور ذیل را کپی و به پنجره powershell انتقال دهید :

```
Set-ItemProperty -Path HKCU:\Software\Microsoft\VisualStudio\11.0\General -Name SuppressUppercaseConversion -Type DWord -Value 1
```



```
Administrator: Windows PowerShell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Developer3> Set-ItemProperty -Path HKCU:\Software\Microsoft\VisualStudio\11.0\General -Name SuppressUppercas
eConversion -Type DWord -Value 1
PS C:\Users\Developer3>
```

سپس راه اندازی مجدد visual studio و مشاهده منوهای تغییر یافته

نظرات خوانندگان

نویسنده: مرتضی

تاریخ: ۱۸:۵۸ ۱۳۹۲/۰۵/۱۵

با استفاده از [VSCommands](#) هم میتونید

نویسنده: sysman

تاریخ: ۱۰:۳۲ ۱۳۹۲/۰۵/۱۶

می تونید از [All Caps Menu Option](#) هم استفاده کنید.

نویسنده: بهمن خلفی

تاریخ: ۱۱:۲۱ ۱۳۹۲/۰۵/۱۶

با تشکر از شما دوستان عزیز - مفید بود.

تنها مزیت کار فوق این است که نیازی به دانلود و یا نصب برنامه و یا پلاگین ندارد . البته هدف انجام این کار بود . از شما عزیزان هم تشکر و قدردانی میکنم .

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

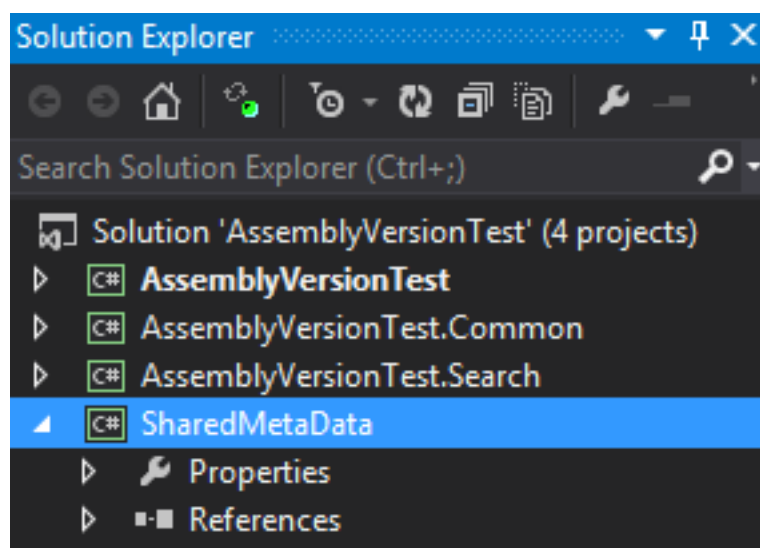
فقط توجه داشته باشید که باید فایل نصب دات نت (مثلا برای دات نت 4.5 فایل dotNetFx45_Full_x86_x64.exe) را به آن اضافه نمایید.

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.

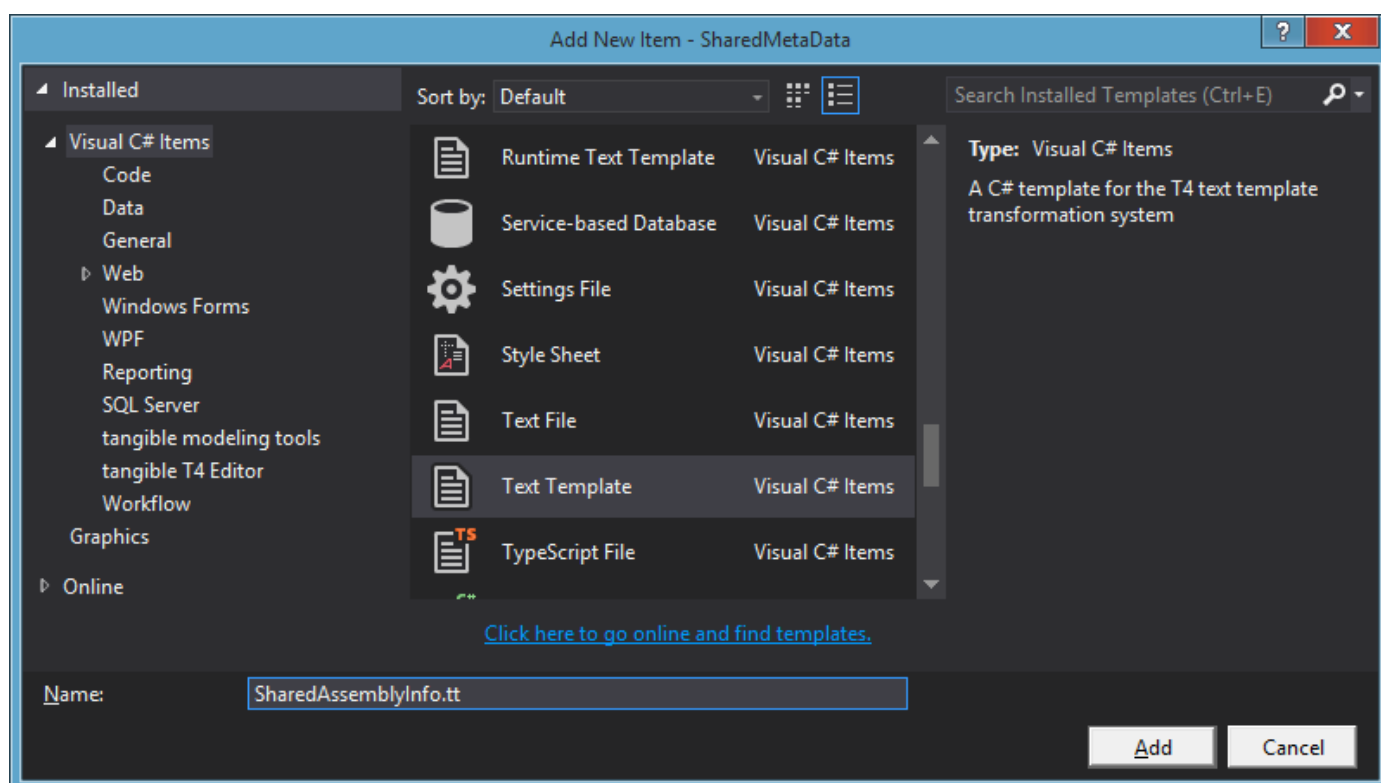
عموماً برای نگهداری ساده‌تر قسمت‌های مختلف یک پروژه، اجزای آن به اسمبلی‌های مختلفی تقسیم می‌شوند که هر کدام در یک پروژه‌ی مجزای ویژوال استودیو قرار خواهند گرفت. یکی از نیازهای مهم این نوع پروژه‌ها، داشتن شماره نگارش یکسانی بین اسمبلی‌های آن است. به این ترتیب توزیع نهایی ساده‌تر شده و همچنین پشتیبانی از آن‌ها در دراز مدت، بر اساس این شماره نگارش بهتر صورت خواهد گرفت. برای مثال در لاگ‌های خطای برنامه با بررسی شماره نگارش اسمبلی مرتبط، حداقل می‌توان متوجه شد که آیا کاربر از آخرین نسخه‌ی برنامه استفاده می‌کند یا خیر. روش معمول انجام این کار، به روز رسانی دستی تمام فایل‌های AssemblyInfo.cs یک Solution است و همچنین اطمینان حاصل کردن از همگام بودن آن‌ها. در ادامه قصد داریم با استفاده از فایل‌های T4، یک فایل SharedAssemblyInfo.tt را جهت تولید اطلاعات مشترک Build بین اسمبلی‌های مختلف یک پروژه، تولید کنیم.

ایجاد پروژه‌ی SharedMetadata

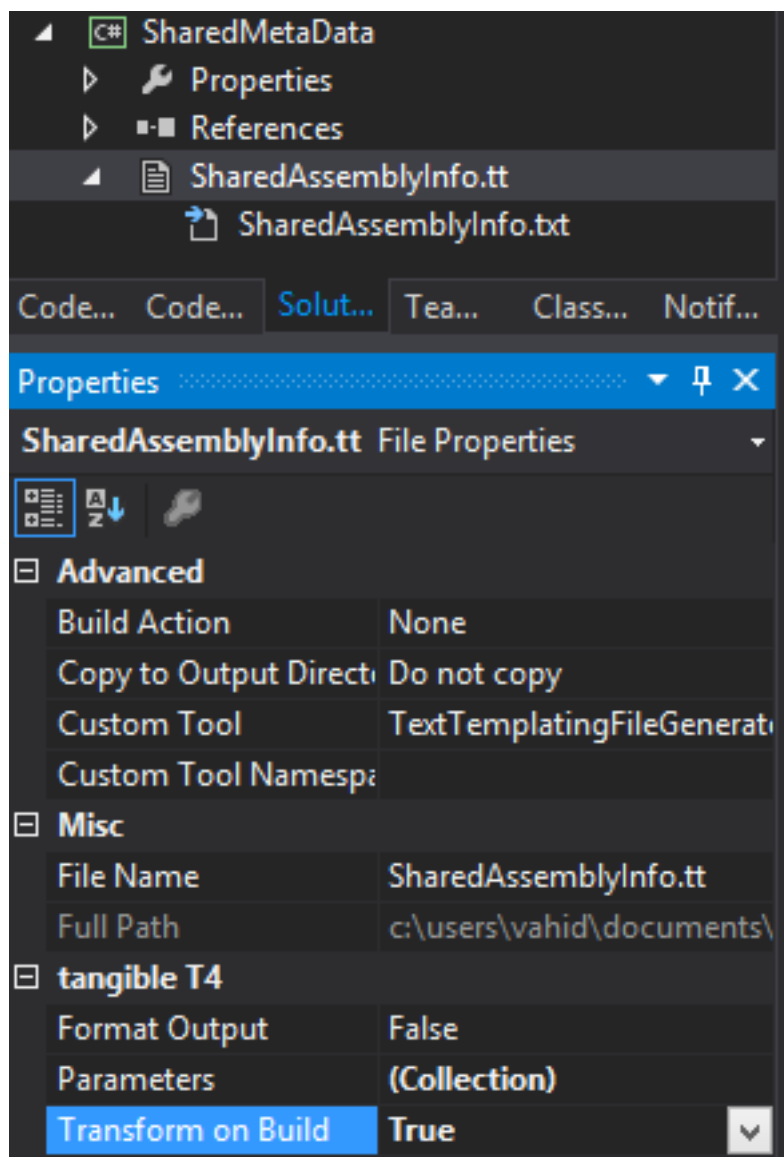
برای نگهداری فایل مشترک SharedAssemblyInfo.cs نهایی و همچنین اطمینان از تولید مجدد آن به ازای هر Build، یک پروژه‌ی class library جدید را به نام SharedMetadata به Solution جاری اضافه کنید.



سپس نیاز است یک فایل text template جدید را به نام SharedAssemblyInfo.tt، به این پروژه اضافه کنید.



به خواص فایل SharedAssemblyInfo.tt مراجعه کرده و [Transform on build](#) آن را [true](#) کنید. به این ترتیب مطمئن خواهیم شد این فایل به ازای هر build جدید، مجدداً تولید می‌گردد.



اکنون محتوای این فایل را به نحو ذیل تغییر دهید:

```
<#@ template debug="false" hostspecific="false" language="C#" #>
//
// This code was generated by a tool. Any changes made manually will be lost
// the next time this code is regenerated.
//
using System.Reflection;
using System.Resources;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;

[assembly: AssemblyCompany("some name")]
[assembly: AssemblyCulture("")]
[assembly: NeutralResourcesLanguageAttribute("en")]

[assembly: AssemblyProduct("product name")]
[assembly: AssemblyCopyright("Copyright VahidN 2014")]
[assembly: AssemblyTrademark("some name")]

#if DEBUG
[assembly: AssemblyConfiguration("Debug")]
#else
[assembly: AssemblyConfiguration("Release")]
#endif

// Assembly Versions are incremented manually when branching the code for a release.
```



```
[assembly: AssemblyVersion("<#> this.MajorVersion #>.<#> this.MinorVersion #>.<#> this.BuildNumber
#>.<#> this.RevisionNumber #>")]
// Assembly File Version should be incremented automatically as part of the build process.
[assembly: AssemblyFileVersion("<#> this.MajorVersion #>.<#> this.MinorVersion #>.<#> this.BuildNumber
#>.<#> this.RevisionNumber #>")]

<#+
// Manually incremented for major releases, such as adding many new features to the solution or
introducing breaking changes.
int MajorVersion = 1;
// Manually incremented for minor releases, such as introducing small changes to existing features or
adding new features.
int MinorVersion = 0;
// Typically incremented automatically as part of every build performed on the Build Server.
int BuildNumber = (int)(DateTime.UtcNow - new DateTime(2013,1,1)).TotalDays;
// Incremented for QFEs (a.k.a. "hotfixes" or patches) to builds released into the Production
environment.
// This is set to zero for the initial release of any major/minor version of the solution.
int RevisionNumber = 0;
#>
```

در این فایل اجزای شماره نگارش برنامه به صورت متغیر تعریف شده‌اند. هر بار که نیاز است یک نگارش جدید ارائه شود، می‌توان این اعداد را تغییر داد.

MajorVersion با افزودن تعداد زیادی قابلیت به برنامه، به صورت دستی تغییر می‌کند. همچنین اگر یک breaking change در برنامه یا کتابخانه وجود داشته باشد نیز این شماره باید تغییر نماید.

MinorVersion با افزودن ویژگی‌های کوچکی به نگارش فعلی برنامه تغییر می‌کند.

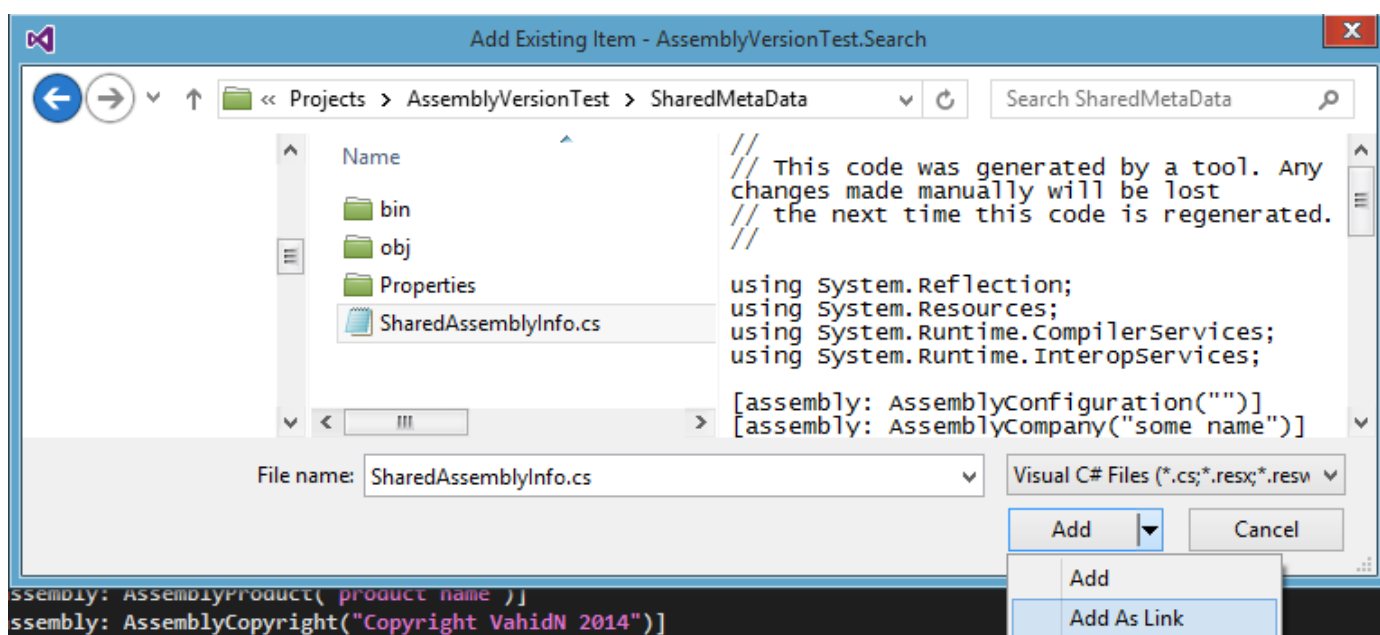
BuildNumber به صورت خودکار بر اساس هر Build انجام شده باید تغییر یابد. در اینجا این عدد به صورت خودکار به ازای هر روز، یک واحد افزایش پیدا می‌کند. ابتدای مبداء آن در این مثال، 2013 قرار گرفته‌است.

RevisionNumber با ارائه یک وصله جدید برای نگارش فعلی برنامه، به صورت دستی باید تغییر کند. اگر اعداد شماره نگارش major یا minor تغییر کنند، این عدد باید به صفر تنظیم شود.

اکنون اگر این محتوای جدید را ذخیره کنید، فایل SharedAssemblyInfo.cs به صورت خودکار تولید خواهد شد.

افزودن فایل SharedAssemblyInfo.cs به صورت لینک به تمام پروژه‌ها

نحوه‌ی افزودن فایل جدید SharedAssemblyInfo.cs به پروژه‌های موجود، اندکی متفاوت است با روش معمول افزودن فایل‌های cs هر پروژه. ابتدا از منوی پروژه گزینه‌ی add existing item را انتخاب کنید. سپس فایل SharedAssemblyInfo.cs را یافته و به صورت add as link، به تمام پروژه‌های موجود اضافه کنید.



اینکار باید در مورد تمام پروژه‌ها صورت گیرد. به این ترتیب چون فایل SharedAssemblyInfo.cs به این پروژه‌ها صرفاً لینک شده‌است، اگر محتوای آن در پروژه‌ی metadata تغییر کند، به صورت خودکار و یک دست، در تمام پروژه‌های دیگر نیز منعکس خواهد شد.

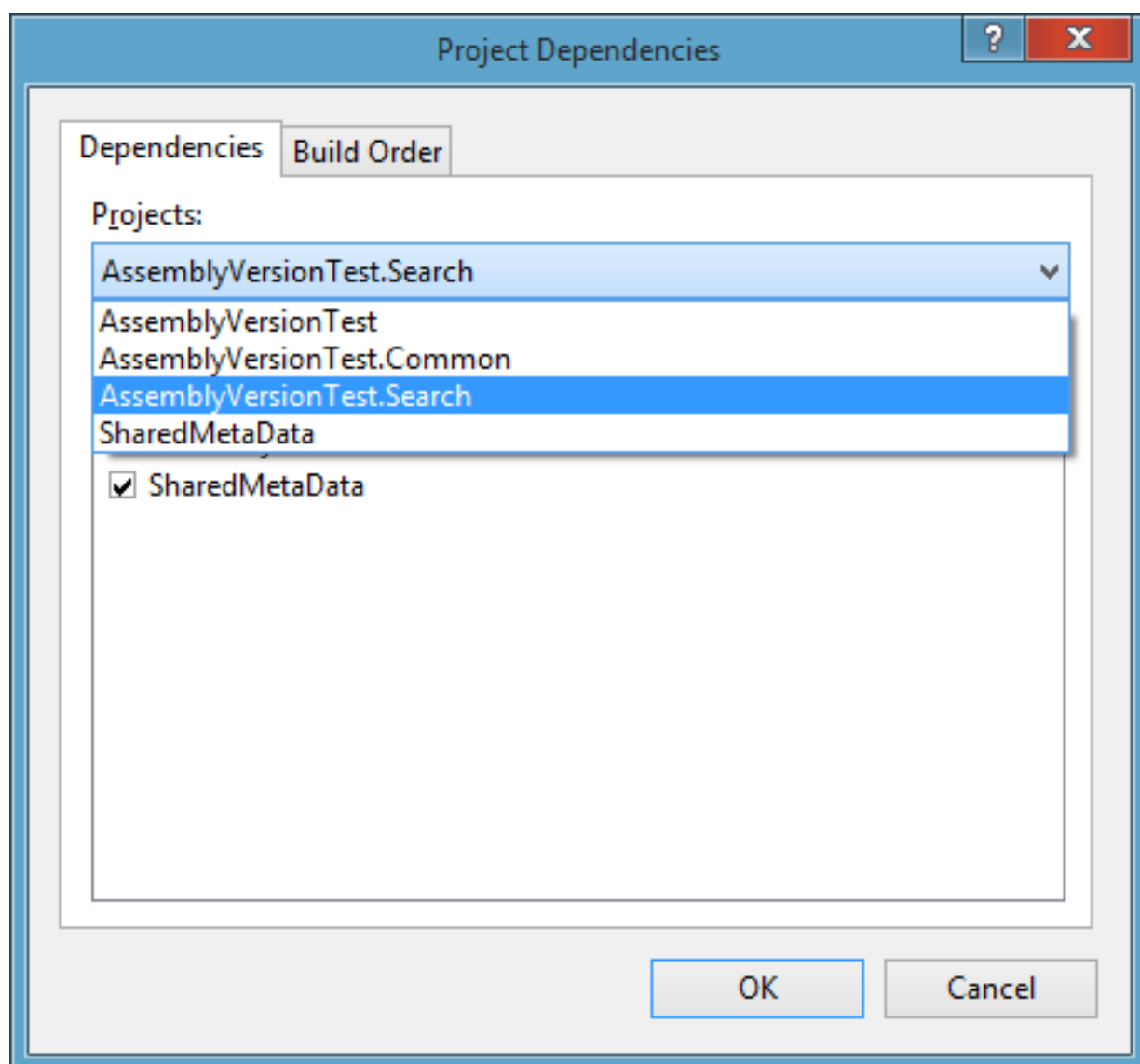
در ادامه اگر بخواهید Solution را Build کنید، پیام تکراری بودن یک سری از ویژگی‌ها را یافت خواهید کرد. این مورد از این جهت رخ می‌دهد که هنوز فایل‌های AssemblyInfo.cs اصلی، در پروژه‌های برنامه موجود هستند. این فایل‌ها را یافته و صرفاً چند سطر همیشه ثابت ذیل را در آن‌ها باقی بگذارید:

```
using System.Reflection;
using System.Runtime.InteropServices;









[assembly: AssemblyTitle("title")]
[assembly: AssemblyDescription("")]
[assembly: ComVisible(false)]
[assembly: Guid("9cde6054-dd73-42d5-a859-7d4b6dc9b596")]
```

اضافه کردن build dependency به پروژه Metadata

در پایان کار نیاز است اطمینان حاصل کنیم، فایل SharedAssemblyInfo.cs به صورت خودکار پیش از Build هر پروژه، تولید می‌شود. برای این منظور، از منوی Project، گزینه‌ی Project dependencies را انتخاب کنید. سپس در برگه‌ی dependencies آن، به ازای تمام پروژه‌های موجود، گزینه‌ی SharedMetadata را انتخاب نمایید.



این مساله سبب اجرای خودکار فایل SharedAssemblyInfo.tt پیش از هر Build می‌شود و به این ترتیب می‌توان از تازه بودن اطلاعات SharedAssemblyInfo.cs اطمینان حاصل کرد. اکنون اگر پروژه را Build کنید، تمام اجزای آن شماره نگارش یکسانی را خواهند داشت:

Name	File version	Product version
 AssemblyVersionTest.Common.dll	1.0.645.0	1.0.645.0
 AssemblyVersionTest.Common.pdb		
 AssemblyVersionTest.exe	1.0.645.0	1.0.645.0
 AssemblyVersionTest.pdb		
 AssemblyVersionTest.Search.dll	1.0.645.0	1.0.645.0
 AssemblyVersionTest.Search.pdb		
 AssemblyVersionTest.vshost.exe	12.0.30723.0	12.0.30723.0
 AssemblyVersionTest.vshost.exe.manifest		

و در دفعات آتی، تنها نیاز است تک فایل SharedAssemblyInfo.tt را برای تغییر شماره نگارش‌های اصلی، ویرایش کرد.

نظرات خوانندگان

نویسنده: لیبرتاد

تاریخ: ۱۳۹۳/۰۷/۱۹ ۱۱:۴۱

آموزش خوبی بود البته در اکثر اوقات بهتر است که شماره نگارش اسمبلی‌های پروژه‌ها یکی نباشد. ممکن است از چندین پروژه یک یا چندتای آنها در آپدیت‌های مختلف هیچ تغییری نداشته باشند

نویسنده: وحید نصیری

تاریخ: ۱۳۹۳/۰۷/۱۹ ۱۳:۰۰

یک اسمبلی در پروژه، به خودی خود فاقد مفهوم است و در قالب نگارش کلی برنامه مفهوم پیدا می‌کند. فرض کنید برنامه شما از یک فایل exe به همراه دو اسمبلی A و B، تشکیل شده‌است. اسمبلی A، نگارش یک دارد. اسمبلی B نگارش 2 و کل برنامه در نگارش 2.5 است. خطایی به شما گزارش شده‌است که در آن استثنای حاصل، از نگارش یک اسمبلی A صادر شده‌است. این مشکل که در نتیجه‌ی در یافت پردازش اشتباهی از اسمبلی B بوده و در نگارش 2 آن برطرف شده، به صورت خودکار با ارتقاء به آخرین نگارش برنامه، برطرف می‌شود. سؤال: آیا اکنون می‌توانید تشخیص دهید کاربر از آخرین نگارش محصول شما استفاده می‌کند؟ نگارش یک A، آخرین نگارش آن است و اما برنامه در نگارش 2.5 قرار دارد. کاربر هم مدتی است که برنامه را به روز نکرده‌است. یک سیستم از همکاری اجزای مختلف آن مفهوم پیدا می‌کند. برای مطالعه بیشتر: «[Best Practices for .NET Assembly Versioning](#)». عبارت «ensuring all of the various assemblies in the solution share the same version» حداقل دوبار در آن تکرار شده‌است.

نویسنده: مسعود مشهدی

تاریخ: ۱۳۹۳/۰۸/۰۸ ۱۰:۰۳

بعد از گذشت چند وقت پروژه build همیشه و می‌گه باید tangible رو خریداری کنید در صورتیکه لینکی که شما دادید نسخه free بود درسته ؟

```
----- SharedMetadata: tangible T4 Editor transforming text templates marked with TransformOnBuild
WARNING: 'TransformOnBuild' is enabled on file 'SharedAssemblyInfo.tt' but this feature is not
available in the FREE EDITION. Please consider buying PRO EDITION from t4-editor.tangible-
engineering.com
----- SharedMetadata: Text templating transformation complete.
```

شخصی سازی using directives موقعی که یک کلاس جدید را در VS.NET باز میکنید، فضاهای نامی مشخص و تکراری، همیشه به صورت پیش فرض صدا زده شده‌اند و این فضاهای نام را مایکروسافت بر اساس بیشترین کاربرد و استفاده توسط برنامه نویسان قرار داده است؛ ولی در خیلی از اوقات این فضاهای نام پیش فرض، چنان هم برای خیلی از برنامه نویسان کاربردی نداشته یا با توجه به برنامه هایی که می‌نویسند همیشه متفاوت هست و هر بار مجبورند فضاهای نام خاصی را صدا بزنند. برای مثال فضای نام System.ComponentModel.DataAnnotations را در نظر بگیرید که برنامه نویسی می‌خواهد برای مدل‌های نوشته شده خود از تگ‌های متا استفاده کند و باید در هر کلاس ساخته شده، یکبار مورد بالا را صدا بزند که بیشتر باعث کند شدن کار برنامه نویسی می‌شود. پس باید کاری کنیم که پیش فرض‌های فضای نام به آنچه خودمان می‌خواهیم تغییر پیدا کند. برای این منظور، به محل نصب ویژوال استودیو رفته و مسیر زیر را دنبال کنید (به مسیر دقت کنید، در اینجا زبان سی شارپ انتخاب شده است):

X:\...\IDE\ItemTemplates\CSharp\Code\1033

در اینجا تعدادی دایرکتوری با اسامی آشنا می‌بینید که داخل هر کدام از آن‌های یک فایل به اسم class.cs هست و اگر آن را باز کنید یک نمونه یا قالب برای using قابل مشاهده است. برای مثال ما وارد دایرکتوری class می‌شویم و فایل class.cs را باز می‌کنیم:

```
using System;
using System.Collections.Generic;
@if$ ($targetframeworkversion$ >= 3.5)using System.Linq;
$endif$using System.Text;
@if$ ($targetframeworkversion$ >= 4.5)using System.Threading.Tasks;
$endif$
namespace $rootnamespace$
{
    class $safeitemrootname$
    {
    }
}
```

الان باید با یک نگاه به الگو، مشخص باشد که چکار باید بکنید. یک سری از فضاهای نام که در تمامی فریمورک‌ها استفاده میشوند به همان شکل عادی نوشته شده‌اند. ولی آنهایی که از نسخه‌ی خاصی از یک فریم ورک اضافه شده‌اند باید توسط شرط مورد نظر اضافه شده و اعلام شود که این فضای نام از چه نسخه‌ی فریم ورکی به بعد باید اضافه گردد:

```
$if$ ($targetframeworkversion$ >= 3.5)using System.Linq;// مورد نظر
$endif$
```

حالا تغییرات را ذخیره کنید و در VS.NET یک کلاس جدید را ایجاد کنید. همانطور که خواهید دید، تغییرات شما اعمال شده‌است. برای اعمال تغییرات نیازی به بستن و باز کردن مجدد VS.NET نمی‌باشد. در لحظه ایجاد کلاس الگو خوانده می‌شود. حال در همان دایرکتوری سی شارپ دقت کنید، می‌بینید که برای موارد دیگری هم فایل هایی وجود دارند. برای مثال برای اینترفیس‌ها یا silverlight و ... که هر کدام را می‌توانید جداگانه تغییر دهید. نکته: احتمال دارد در نسخه‌های متفاوت به خصوص پایین‌تر مثل نسخه 8 ویژوال استودیو، فایل class.cs به صورت zip باشد که بعد از تغییرات باید دوباره به حالت zip بازگردانده شود.

حذف فضای نام‌های اضافی

هر موقع که کلاس جدیدی می‌سازیم، namespace ها به صورت پیش فرض که در بالا اشاره کردیم وجود دارند و شاید اصلا در آن کلاس از آن‌ها استفاده نمی‌کنیم یا حتی خودمان در حین نوشتن کدها چند namespace خاص را اضافه می‌کنیم که شاید در طول برنامه نویسی چندتایی را بلا استفاده بگذاریم. برای همین همیشه فضای نام هایی صدا زده شده‌اند که اصلا در آن کلاس استفاده

نشده‌اند. پس برای همین بهتر هست که این رفرنس‌های بلا استفاده را پیدا کرده و آن‌ها را حذف کنیم. شاید این سوال برای بعضی‌های پدید بیاد که چرا باید این‌ها را حذف کنیم، چون کاری هم با ما ندارند و ما هم کاری با آن‌ها نداریم؟

این کار چند علت میتواند داشته باشد:

تمیزکاری کد و خلوت شدن فضای کدنویسی

ممکن هست بعدها گیج کننده شود که من چرا از این‌ها استفاده کردم؟ در آینده با نگاه به یک کد تمیزتر متوجه میشوید یک کد از چه چیزهایی برای انجام کارش بهره‌مند شده و هم اینکه در کارهای گروهی و تیمی هم این مورد به شدت تاثیرگذار هست.

باعث کند شدن تحلیل‌های ایستا میشه ([اینجا](#) و [اینجا](#))

کمپایل شدن کد کندتر میشه

موقع تست برنامه، اجرای اولیه کندتر خواهد بود چون CLR باید این نوع موارد را شناسایی و حذف کند

همه موارد بالا در مورد رفرنس‌های موجود یا همان dll‌های موجود در شاخه‌ی Bin و References هم صدق می‌کند.

برای حذف فضاهای نام اضافی در یک صفحه می‌توانید از طریق این مسیر انجام بدید:

Edit>

Remove Unused using >Organize Usings>IntelliSense برای مرتب سازی هم گزینه Sort Usings و انجام هر دو کار Remove and Sort موجود هست. البته اگر روی صفحه هم راست کلیک کنید گزینه Organize Usings هم وجود دارد. می‌توانید از ابزارهایی چون [Power tools Extensions](#) هم استفاده کنید (در صورتی که ویژوال استودیوی شما گزینه‌های مورد نظر را ندارد، این ابزار را نصب نمایید)

در صورتی که از ابزارهایی چون [telerik](#) یا [devexpress](#) استفاده می‌کنید یا از هر ابزار اضافی که بر روی IDE نصب می‌شود، عموماً چنین گزینه‌هایی حتی با امکانات وسیع‌تر وجود دارند. مثلاً [whole tomato](#) هم یکی از این ابزارهاست.

این نکته را هم خاطر نشان کنم در صورتیکه فضاهای نامی بین [پیش پردازنده‌ها](#) که در قبل توضیح دادیم محصور شده باشند، حذف نخواهند شد و همانطور باقی خواهند ماند.

در مورد کامنت‌های بین using‌ها به قطعه کد زیر نگاه کنید:

```
using System;
/* Comment before remains */
using /* Comment between removed */ System.Linq;
// Comment after remains
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("My Example");
        }
    }
}
```

و حالا بعد از حذف فضای نام‌های اضافی

```
using System;
/* Comment before remains */
// Comment after remains
namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("My Example");
        }
    }
}
```

برای اینکه این عمل را بتوانید در کل صفحات اعمال کنید می‌توانید از *cleanup selected code* هم استفاده کنید؛ به جز اینکه فضاهای نام اضافی را هم پاک می‌کند، کلیه کدهای شما را در قالبی شکل‌تر و خواناتر قرار خواهد داد.

با کلیدهای `Ctrl+k+d` سند انتخابی و با کلیدهای ترکیبی `Ctrl+k+f` هم محدوده انتخاب شده قالب بندی می‌شود. یکی دیگر از ابزارهایی که می‌توان با آن‌ها به کد سر و سامان بهتری داد، افزونه‌ی [codemaide](https://github.com/mbigler/codemaide) هست.

ویژگی سی شارپ 6 در مورد Using فرض کنید ما یک کلاس ایستا به نام `utilities` ایجاد کردیم که یک متد به اسم `addints` دارد. حالا و این کلاس در `namespace` به نام `SomeNamespace` قرار دارد. مطمئناً در این حالت ما ابتدا فضای نام را `using` میکنیم و سپس در کد کلاس، متد را به شکل زیر صدا می‌زنیم:

```
using System;
using SomeNamespace;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int sum = Utility.AddInts(5, 2);

            Console.ReadLine();
        }
    }
}
```

ولی در سی شارپ 6 می‌توانید بعد از فضای نام، یک `.` گذاشته و سپس اسم کلاس ایستا `static` را بیاورید و در کد مستقیماً متد دلخواه خود را صدا بزنید. به شکل زیر دقت کنید:

```
using System;
using SomeNamespace.Utility;

namespace ConsoleApplication1
{
    class Program
    {
        static void Main(string[] args)
        {
            int sum = AddInts(5, 2);

            Console.ReadLine();
        }
    }
}
```

نکته پایانی: در `visual studio 2014` فضاهای نام اضافی به رنگ خاکستری نمایش داده می‌شوند.

منابع: <http://blogs.msdn.com/b/steve/archive/2007/04/10/changing-the-default-using-directives-in-visual-studio.aspx>

<http://stackoverflow.com/questions/629667/why-remove-unused-using-directives-in-c>

<http://stackoverflow.com/questions/5755942/how-do-you-auto-format-code-in-visual-studio>

[/http://csharp.2000things.com](http://csharp.2000things.com)

نظرات خوانندگان

نویسنده: علی یگانه مقدم
تاریخ: ۲۰:۲۸ ۱۳۹۴/۰۵/۰۲

یک نکته اضافه در مورد فایل class.cs در ابتدای مقاله. مدتی هست که من هر موقع کلاسی به خصوص برای بخش مدل‌ها ایجاد میکنم مرتب هر کلاسی را باید یک public را بنویسم چون به طور پیش فرض کلاس‌هایی که دات نت ایجاد میکند private هستند. برای حل این مشکل در فایل class عبارت public را قبل از کلمه class به آن اضافه کنید:

```
class $safeitemrootname$  
{  
}  
  
===== To  
public class $safeitemrootname$  
{  
}
```

احتمالا این معضل خیلی‌ها هست چون نوشتن تعداد کلاس‌های عمومی بیشتر از خصوصی است

در این نوشتار قصد داریم تا Theme ویژوال استودیو 2013 را برای برنامه‌های ویندوز شبیه سازی کنیم. در مرحله اول یک پروژه از نوع ClassLibrary می‌سازیم و پس از آن یک کلاس که از کلاس [ToolStripProfessionalRenderer](#) ارث بری کند را ایجاد می‌کنیم. در اینجا ما نام کلاس را BlueMenuStrip انتخاب می‌کنیم. از این کلاس برای تغییر رنگ منوها استفاده می‌شود. سپس متد OnRenderMenuItemBackground آن را Override می‌کنیم.

```
using System.Drawing;
using System.Windows.Forms;

namespace Navasser.Theme.VisualStudio
{
    public class BlueMenuStrip:ToolStripProfessionalRenderer
    {
        protected override void OnRenderMenuItemBackground(ToolStripItemRenderEventArgs e)
        {
            var borderColor = ColorTranslator.FromHtml("#E5C365");//Menu Item Border
            var selectedMenuBackColor = ColorTranslator.FromHtml("#FDF4BF");//Menu Item Background
            var menuOpenedBackColor = ColorTranslator.FromHtml("#EAF0FF");
            var borderPen = new Pen(borderColor);

            if (e.Item.Selected)//اگر آیتمی از منوها انتخاب شد
            {
                var selectedMenuBrush = new SolidBrush(selectedMenuBackColor);
                var selectedItemBounds = new Rectangle(Point.Empty, e.Item.Size);//اقدام به پر کردن یک
                //مستطیل به اندازه ابعاد آیتم انتخاب شده می‌کند
                e.Graphics.FillRectangle(selectedMenuBrush,selectedItemBounds);
                e.Graphics.DrawRectangle(borderPen,0,0,selectedItemBounds.Width-
                1,selectedItemBounds.Height-1);//بوردر آیتم را رسم میکند
                e.Item.BackColor = menuOpenedBackColor;
            }
            else
            {
                base.OnRenderMenuItemBackground(e);
                e.ToolStrip.BackColor = ColorTranslator.FromHtml("#EAF0FF");
            }
        }
    }
}
```

تکه کد بالا فقط برای تغییر رنگ زمینه‌ی منوها بکار می‌رود. اما برای تغییر رنگ ToolStrip‌ها یک کلاس جدید ایجاد می‌کنیم که از کلاس ToolStripProfessionalRenderer ارث بری کرده باشد و متدهای OnRenderToolStripBackground و OnRenderButtonBackground آن را Override می‌کنیم.

```
using System.Drawing;
using System.Windows.Forms;

namespace Navasser.Theme.VisualStudio
{
    public class BlueToolStrip : ToolStripProfessionalRenderer
    {
        protected override void OnRenderToolStripBackground(ToolStripRenderEventArgs e)
        {
            var toolStripBackColor = ColorTranslator.FromHtml("#D6DBE9");
            var toolStripBrush = new SolidBrush(toolStripBackColor);
            e.Graphics.FillRectangle(toolStripBrush,0,0,e.AffectedBounds.Width +
            10,e.AffectedBounds.Height);
        }

        protected override void OnRenderButtonBackground(ToolStripItemRenderEventArgs e)
        {
            var borderColor = ColorTranslator.FromHtml("#E5C365");//For border
            var selectedToolItemBackColor = ColorTranslator.FromHtml("#FDF4BF");
            var selectedItemBrush = new SolidBrush(selectedToolItemBackColor);
            var pressedItemBackColor = ColorTranslator.FromHtml("#FFF29D");
            var pressedItemBrush = new SolidBrush(pressedItemBackColor);
        }
    }
}
```

```
var borderPen = new Pen(borderColor);
if (e.Item.Selected)
{
    e.Graphics.FillRectangle(selectedItemBrush,0,0,e.Item.Width,e.Item.Height);
    e.Graphics.DrawRectangle(borderPen,0,0,e.Item.Width-1,e.Item.Height-1);
}
if (e.Item.Pressed)
{
    e.Graphics.FillRectangle(pressedItemBrush, 0, 0, e.Item.Width, e.Item.Height);
    e.Graphics.DrawRectangle(borderPen, 0, 0, e.Item.Width - 1, e.Item.Height - 1);
}
}
}
```

سپس D11 ایجاد شده را در برنامه خود Reference دهید و از Theme های ایجاد شده استفاده نمایید. نتیجه‌ی کدهای بالا به شکل زیر است:



همچنین می‌توانید برای انتخاب رنگ‌های دلخواه خودتان از ابزار [ColorSchemer Studio](#) استفاده کنید.

نظرات خوانندگان

نویسنده: شهریار
تاریخ: ۱۵:۶ ۱۳۹۳/۱۲/۰۸

با سلام و تشکر

من کدهای بالا را استفاده کردم ولی شبیه به رنگهای عکس اول نشد. اگر ممکنه راهنمایی بفرمایین

نویسنده: احمد نواصری
تاریخ: ۱۱:۳۲ ۱۳۹۳/۱۲/۰۹

سلام دوست عزیز. کاشکی یه عکس از فرمتون قرار میدادین تا من بهتر متوجه مشکل میشدم.

نویسنده: وحید نصیری
تاریخ: ۲۳:۴ ۱۳۹۳/۱۲/۱۱

پروژه‌ی مثال بحث جاری برای امتحان: [WinFormsThemes.zip](#)

نویسنده: احمد نواصری
تاریخ: ۲۰:۳۱ ۱۳۹۳/۱۲/۱۲

این هم یک نمونه پروژه دیگه : [VS-2013-Theme.zip](#)

معرفی Roslyn

سکوی کامپایلر دات نت یا **Roslyn** (با تلفظ «رازلین») بازنویسی مجدد کامپایلرهای VB.NET و C# توسط همین زبان‌ها است. این سکوی کامپایلر به همراه یک سری کتابخانه و اسمبلی ارائه می‌شود که امکان آنالیز زبان‌های مدیریت شده را به صورت مستقل و یا یکپارچه‌ی با ویژوال استودیو، فراهم می‌کنند. برای نمونه در VS.NET 2015 تمام سرویس‌های زبان‌های موجود، با Roslyn API جایگزین و بازنویسی شده‌اند. نمونه‌هایی از این سرویس‌های زبان‌ها، شامل Intellisense و مرور کدها مانند go to references and definitions، به همراه امکانات Refactoring می‌شوند. به علاوه به کمک Roslyn می‌توان یک کامپایلر و ابزارهای مرتبط با آن، مانند FxCop را تولید کرد و یا در نهایت یک فایل اسمبلی نهایی را از آن تحویل گرفت.

چرا مایکروسافت Roslyn را تولید کرد؟

پیش از پروژه‌ی Roslyn، کامپایلرهای VB.NET و C# با زبان ++C نوشته شده بودند؛ از این جهت که در اواخر دهه‌ی 90 که کار تولید سکوی دات نت در حال انجام بود، هنوز امکانات کافی برای نوشتن این کامپایلرها با زبان‌های مدیریت شده وجود نداشت و همچنین زبان محبوب کامپایلر نویسی در آن دوران نیز ++C بود. این انتخاب در دراز مدت مشکلاتی مانند کاهش انعطاف پذیری و productivity تیم کامپایلر نویسی را با افزایش تعداد سطرهای کامپایلر نوشته شده به همراه داشت و افزودن ویژگی‌های جدید را به زبان‌های VB.NET و C# سخت‌تر و سخت‌تر کرده بود. همچنین در اینجا برنامه نویسی‌های تیم کامپایلر مدام مجبور بودند که بین زبان‌های مدیریت شده و مدیریت نشده سوئیچ کنند و امکان استفاده‌ی همزمان از زبان‌هایی را که در حال توسعه‌ی آن هستند، نداشتند.

این مسایل سبب شدند تا در طی بیش از یک دهه، چندین نوع کامپایلر از صفر نوشته شوند:

- کامپایلرهای خط فرمانی مانند csc.exe و vbc.exe
- کامپایلر پشت صحنه‌ی ویژوال استودیو (برای مثال کشیدن یک خط قرمز زیر مشکلات دستوری موجود)
- کامپایلر snippetها در immediate window و ویژوال استودیو

هر کدام از این کامپایلرها هم برای حل مسایلی خاص طراحی شده‌اند. کامپایلرهای خط فرمانی، با چندین فایل ورودی، به همراه ارائه‌ی تعدادی زیادی خطا و اخطار کار می‌کنند. کامپایلر پشت صحنه‌ی ویژوال استودیوهای تا پیش از نسخه‌ی 2015، تنها با یک تک فایل در حال استفاده، کار می‌کند و همچنین باید به خطاهای رخ داده نیز مقاوم باشد و بیش از اندازه گزارش خطا ندهد. برای مثال زمانیکه کاربر در حالت تایپ یک سطر است، بدیهی است تا اتمام کار، این سطر فاقد ارزش دستوری صحیحی است و کامپایلر باید به این مساله دقت داشته باشد و یا کامپایلر snippetها تنها جهت ارزیابی یک تک سطر از دستورات وارد شده، طراحی شده‌است.

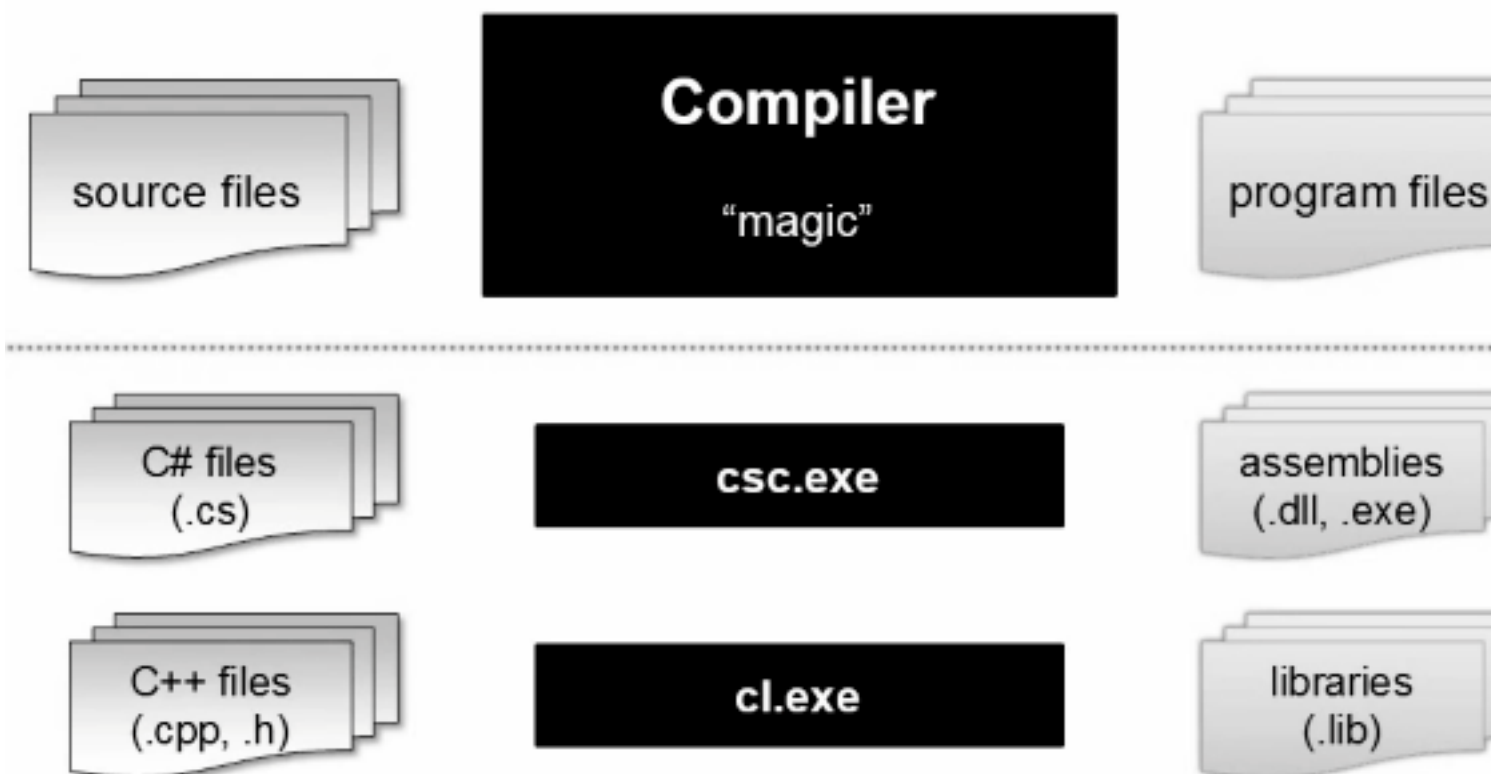
با توجه به این مسایل، مایکروسافت از بازنویسی سکوی کامپایلر دات نت این اهداف را دنبال می‌کند:

- بالا بردن سرعت افزودن قابلیت‌های جدید به زبان‌های موجود
- سبک کردن حجم کاری کامپایلر نویسی و کاهش تعداد آن‌ها به یک مورد
- بالا بردن دسترسی پذیری به API کامپایلرها
- برای مثال اکنون برنامه نویسی‌ها بجای اینکه یک فایل cs را به کامپایلر csc.exe ارائه کنند و یک خروجی باینری دریافت کنند، امکان دسترسی به syntax trees، semantic analysis و تمام مسایل پشت صحنه‌ی یک کامپایلر را دارند.
- ساده سازی تولید افزونه‌های مرتبط با زبان‌های مدیریت شده.
- اکنون برای تولید یک آنالیز کننده‌ی سفارشی، نیازی نیست هر توسعه دهنده‌ای شروع به نوشتن امکانات پایه‌ای یک کامپایلر کند.
- این امکانات به صورت یک API عمومی در دسترس برنامه نویسی‌ها قرار گرفته‌اند.
- آموزش مسایل درونی یک کامپایلر و همچنین ایجاد اکوسیستمی از برنامه نویسی‌های علاقمند در اطراف آن.
- همانطور که اطلاع دارید، Roslyn به صورت سورس باز در [GitHub](https://github.com) در دسترس عموم است.

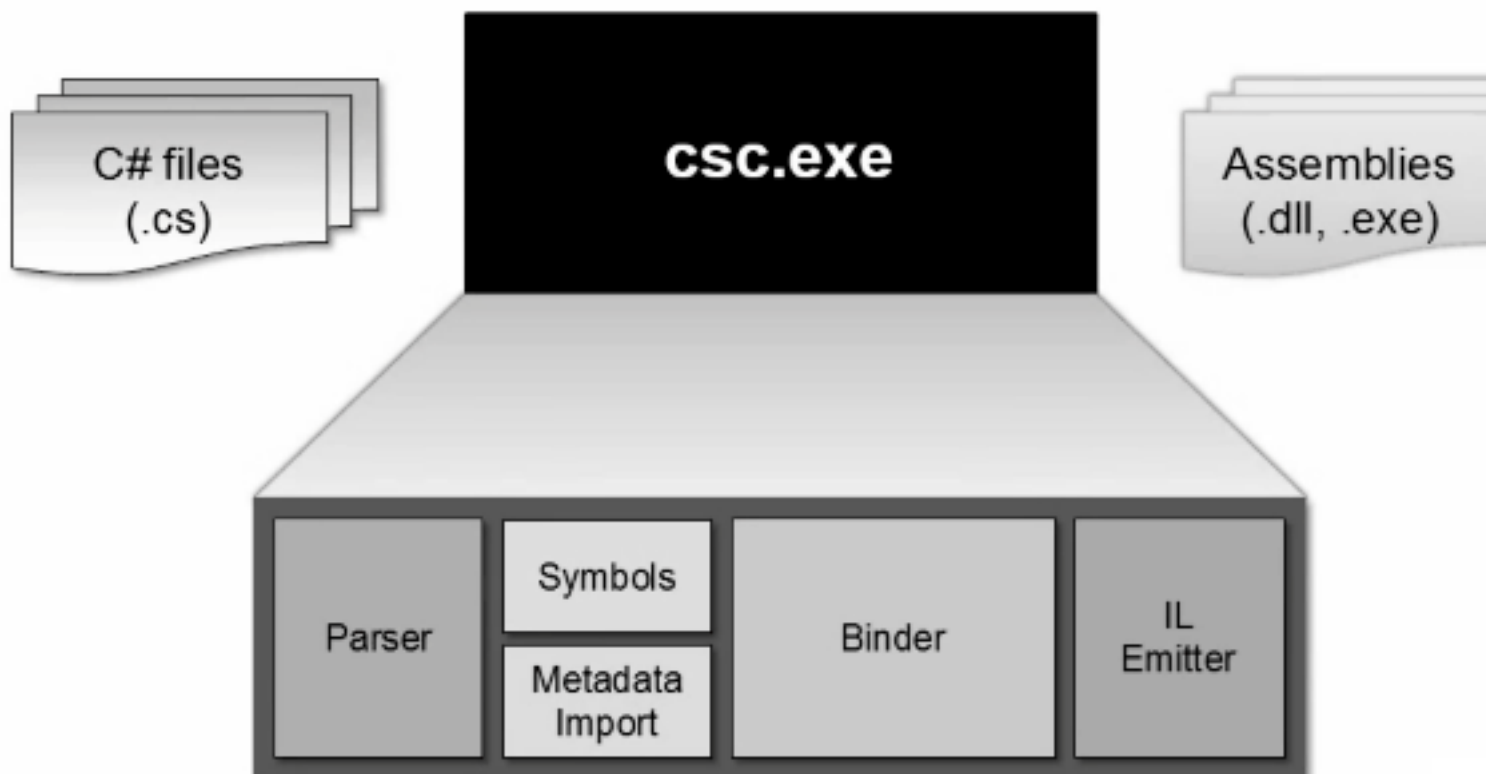
تفاوت Roslyn با کامپایلرهای سنتی

اکثر کامپایلرهای موجود به صورت یک جعبه‌ی سیاه عمل می‌کنند. به این معنا که تعدادی فایل ورودی را دریافت کرده و در نهایت یک خروجی باینری را تولید می‌کنند. اینکه در این میان چه اتفاقاتی رخ می‌دهد، از دید استفاده‌کننده مخفی است.

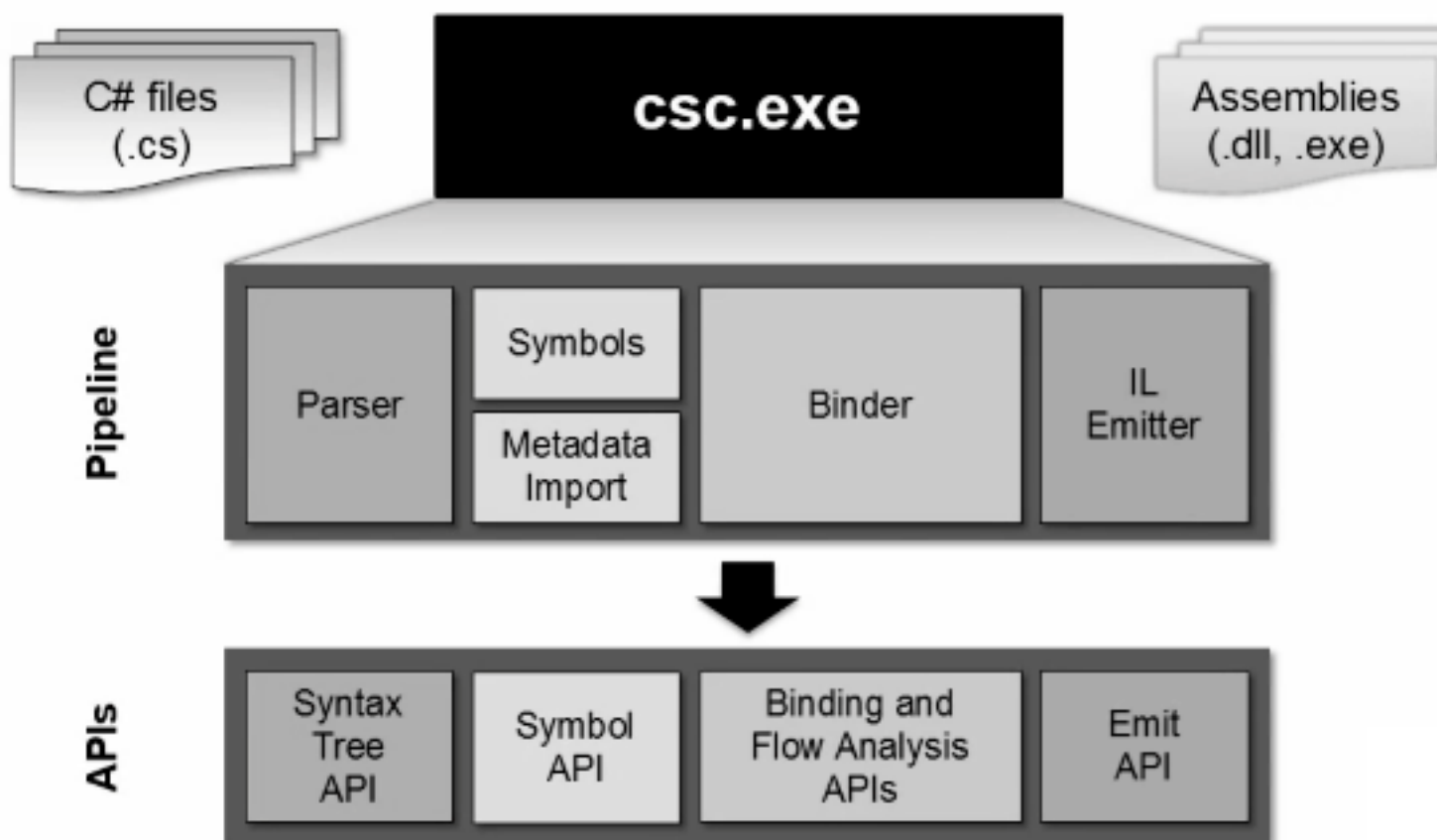
■



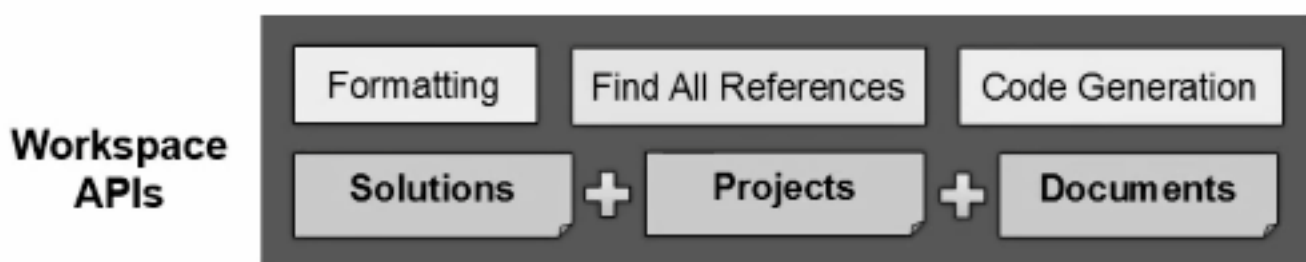
نمونه‌ای از این کامپایلرهای جعبه سیاه را در تصویر فوق مشاهده می‌کنید. در اینجا شاید این سؤال مطرح شود که در داخل جعبه‌ی سیاه کامپایلر سی‌شارپ، چه اتفاقاتی رخ می‌دهد؟



خلاصه‌ی مراحل رخ داده در کامپایلر سی‌شارپ را در تصویر فوق ملاحظه می‌کنید. در اینجا ابتدا کار `parse` اطلاعات متنی دریافتی شروع می‌شود و از روی آن `syntax tree` تولید می‌شود. در مرحله‌ی بعد مواردی مانند ارجاعاتی به `microsoft` و امثال آن پردازش می‌شوند. در مرحله‌ی `binder` کار پردازش حوزه‌ی دید متغیرها، اشیاء و اتصال آن‌ها به هم انجام می‌شود. در مرحله‌ی آخر، کار تولید کدهای `IL` و اسمبلی باینری نهایی صورت می‌گیرد. با معرفی `Roslyn`، این جعبه‌ی سیاه، به صورت یک `API` عمومی در دسترس برنامه نویسی‌ها قرار گرفته‌است:



همانطور که مشاهده می‌کنید، هر مرحله‌ی کامپایلر جعبه‌ی سیاه، به یک API عمومی Roslyn نگاشت شده‌است. برای مثال Parser به Syntax tree API نگاشت شده‌است. به علاوه این API صرفاً به موارد فوق خلاصه نمی‌شود و همانطور که پیشتر نیز ذکر شد، برای اینکه بتواند جایگزین سه نوع کامپایلر موجود شود، به همراه Workspace API نیز می‌باشد:



Roslyn امکان کار با یک Solution و فایل‌های آن را دارد و شامل سرویس‌های زبان‌های مورد نیاز در ویژوال استودیو نیز می‌شود. بر فراز Workspace API، یک مجموعه API دیگر به نام Diagnostics API تدارک دیده شده‌است تا برنامه نویسی‌ها بتوانند امکانات Refactoring جانبی را توسعه داده و یا در جهت بهبود کیفیت کدهای نوشته شده، اختراهایی را به برنامه نویسی‌ها تحت عنوان Code fixes و آنالیز کننده‌ها، ارائه دهند.

Diagnostic APIs

Red
Squiggles

Code
Fixes

Refactorings

عنوان:	استفاده از bower در visual studio
نویسنده:	مهرداد کاهه
تاریخ:	۱۳۹۴/۰۷/۱۷
آدرس:	www.dotnettips.info
گروه‌ها:	JavaScript, Git, Visual Studio, bower

اگر از آن دسته افرادی هستید که با پکیج‌های مختلف و پروژه‌های مختلف تحت کلاینت سر و کار دارید و همچنین اطلاعات چندانی نسبت به NodeJs ندارید (مثل خود من)، حتما به پروژه‌هایی در Github برخوردید که نیازمند نصب وابستگی‌ها از خط فرمان bower و یا npm هستند.

بعد از مطالعه‌ی مطلب [آشنایی با bower](#) این نیاز ایجاد شد تا در پروژه‌هایی که قرار است درون Visual Studio اجرا شوند، وابستگی‌های bower چگونه می‌توانند مدیریت شوند.

خوشبختانه Microsoft این امکان را ایجاد کرده تا شما بتوانید پروژه‌هایی را که وابستگی‌هایشان درون bower تعریف شده را نیز درون Visual Studio حل و فصل کنید. در ادامه تمامی این مراحل، قدم به قدم اضافه تشریح شده است.

قابل ذکر است که هر سه package manager معروف bower، npm و Nuget در ویژوال استدیو 2015 به صورت توکار موجود هستند. [جزئیات بیشتر در مستندات مایکروسافت](#)

معرفی پکیج Bower

همانطور که در مقاله [آشنایی با bower](#) نیز اشاره شد، bower یک package manager برای تکنولوژی‌ها و کتابخانه‌های کلاینت است. این package manager بر روی Git اجرا می‌شود. همانطور که می‌دانید تمامی پکیج‌ها نیز از Git دریافت می‌شود. اما حال اینکه چگونه می‌توان از این package manager در سمت Visual studio بدون نصب NodeJs و Git استفاده کرد، با پکیج توسعه داده شده Bower مایکروسافت رفع شده است. جزئیات این پکیج را می‌توانید در [NuGet](#) مطالعه کنید.

شروع کار با Bower

برای آغاز، یک پروژه‌ی web Application ایجاد می‌کنیم. من Empty را انتخاب و ریسورس‌های MVC را هم اضافه کردم. حال در بخش Package Manager Console دستور زیر را اجرا کنید:

```
Install-Package Bower
```

پس از نصب وابستگی‌ها و خود bower خروجی package manager console به صورت زیر خواهد بود:

```
PM> Install-Package Bower
Attempting to resolve dependency 'Node.js (≥ 0.10.28)'.
Attempting to resolve dependency 'NoGit (≥ 0.0.8)'.
Installing 'Node.js 0.10.36'.
Successfully installed 'Node.js 0.10.36'.
Installing 'NoGit 0.0.9'.
Successfully installed 'NoGit 0.0.9'.
Installing 'Bower 1.3.11'.
Successfully installed 'Bower 1.3.11'.
Adding 'Node.js 0.10.36' to WebApplication5.
Successfully added 'Node.js 0.10.36' to WebApplication5.
Adding 'NoGit 0.0.9' to WebApplication5.
Successfully added 'NoGit 0.0.9' to WebApplication5.
Adding 'Bower 1.3.11' to WebApplication5.
Successfully added 'Bower 1.3.11' to WebApplication5.
```

مشاهده می‌کنید که فولدر bin. به پروژه‌ی شما اضافه شده است.

حال درون صفحه‌ی cmd (توجه کنید cmd، نه package manager console) به آدرس پروژه (نه solution) با دستور زیر منتقل شوید:

```
cd <Project Location>
```

که به جای project location آدرس فایل پروژه را قرار می‌دهیم. شکل زیر نمایانگر این مسیر است:

```
Command Prompt
Microsoft Windows [Version 10.0.10240]
(c) 2015 Microsoft Corporation. All rights reserved.

C:\Users\mehrdad>cd C:\Users\mehrdad\Documents\visual studio 2013\Projects\SimpleBower\SimpleBower
```

با اجرای دستور زیر bower.json را به پروژه اضافه می‌کنیم:

```
bower init
```

```
C:\Users\mehrdad\Documents\Visual Studio 2013\Projects\SimpleBower\SimpleBower>bower init
? name: (SimpleBower)
```

مشاهده می‌کنید که پس از دستور bower init مواردی که قرار است درون bower قرار گیرد، مقدار دهی می‌شوند. من مقادیر را به صورت زیر (حالت‌های پیش فرض) تکمیل کردم:

```
C:\Users\mehrdad\Documents\Visual Studio 2013\Projects\SimpleBower\SimpleBower>bower init
? name: SimpleBower
? version: 0.0.0
? description:
? main file:
? what types of modules does this package expose?
? keywords:
? authors:
? license: MIT
? homepage:
? set currently installed components as dependencies? Yes
? add commonly ignored files to ignore list? No
? would you like to mark this package as private which prevents it from being accidentally published to the registry? (y? would you like to mark this package as private which prevents it from being accidentally published to the registry? No
{
  name: 'SimpleBower',
  version: '0.0.0',
  license: 'MIT'
}
```

حال باید تا اینجای کار یک فایل bower.json برای شما در روت پروژه ساخته شده باشد. حال بیاید اولین اسکریپت رفرنس را به پروژه اضافه نماییم. من قصد دارم تا با دستور زیر JQuery را به پروژه اضافه کنم:

```
bower install jquery
```

پکیج JQuery به صورت زیر دانلود می شود و در پوشه‌ی bower_component در روت پروژه قرار می گیرد.

```
C:\Users\mehrdad\Documents\Visual Studio 2013\Projects\SimpleBower\SimpleBower>bower instal jquery
bower jquery#*      not-cached git://github.com/jquery/jquery.git#*
bower jquery#*      resolve  git://github.com/jquery/jquery.git#*
bower jquery#*      download https://github.com/jquery/jquery/archive/2.1.4.tar.gz
bower jquery#*      extract  archive.tar.gz
bower jquery#*      resolved git://github.com/jquery/jquery.git#2.1.4
bower jquery#~2.1.4  install  jquery#2.1.4

jquery#2.1.4 bower_components\jquery
```

به همین صورت شما می توانید تمامی نیازمندی های پروژه را از Git با استفاده از bower package manager دریافت کنید.