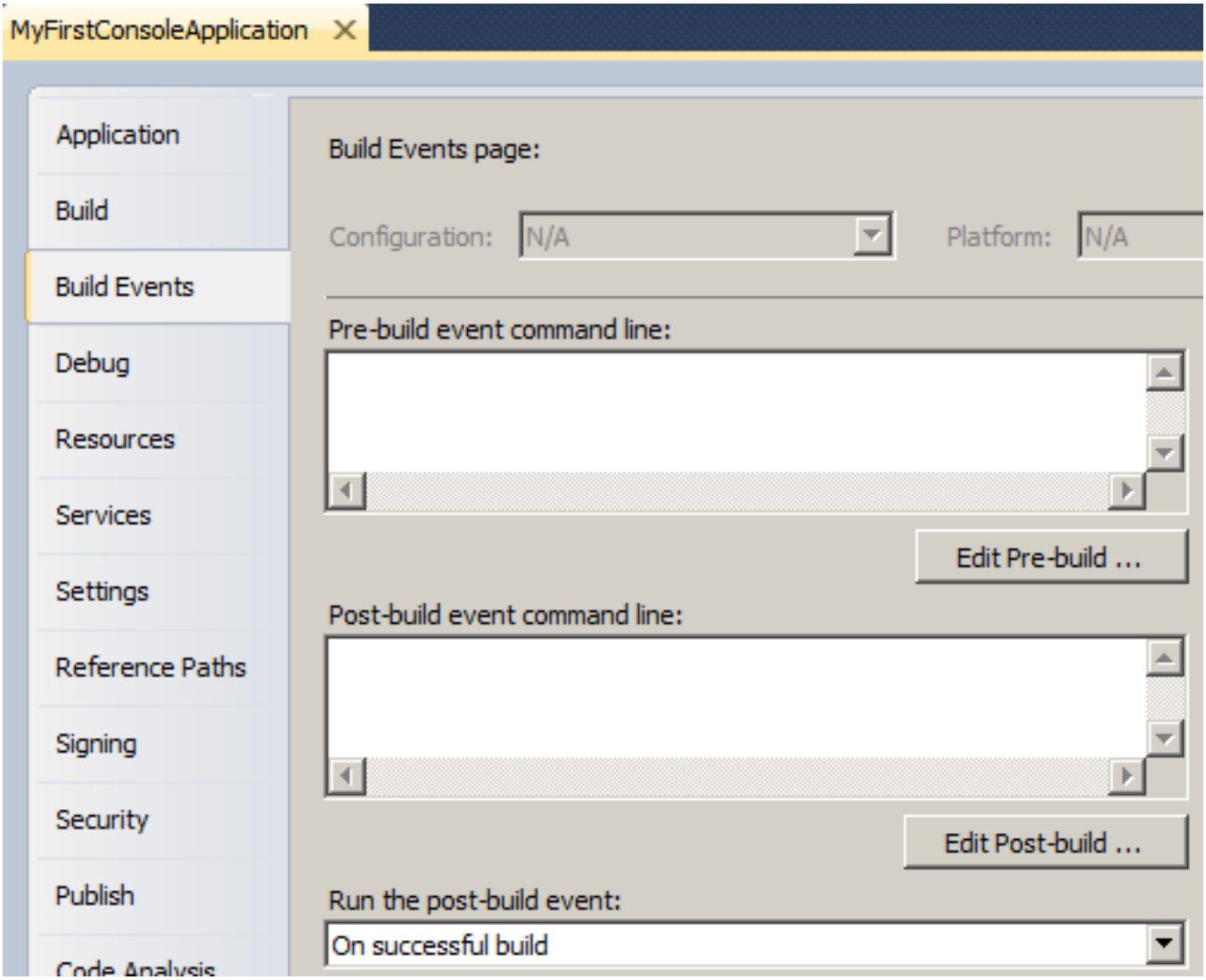
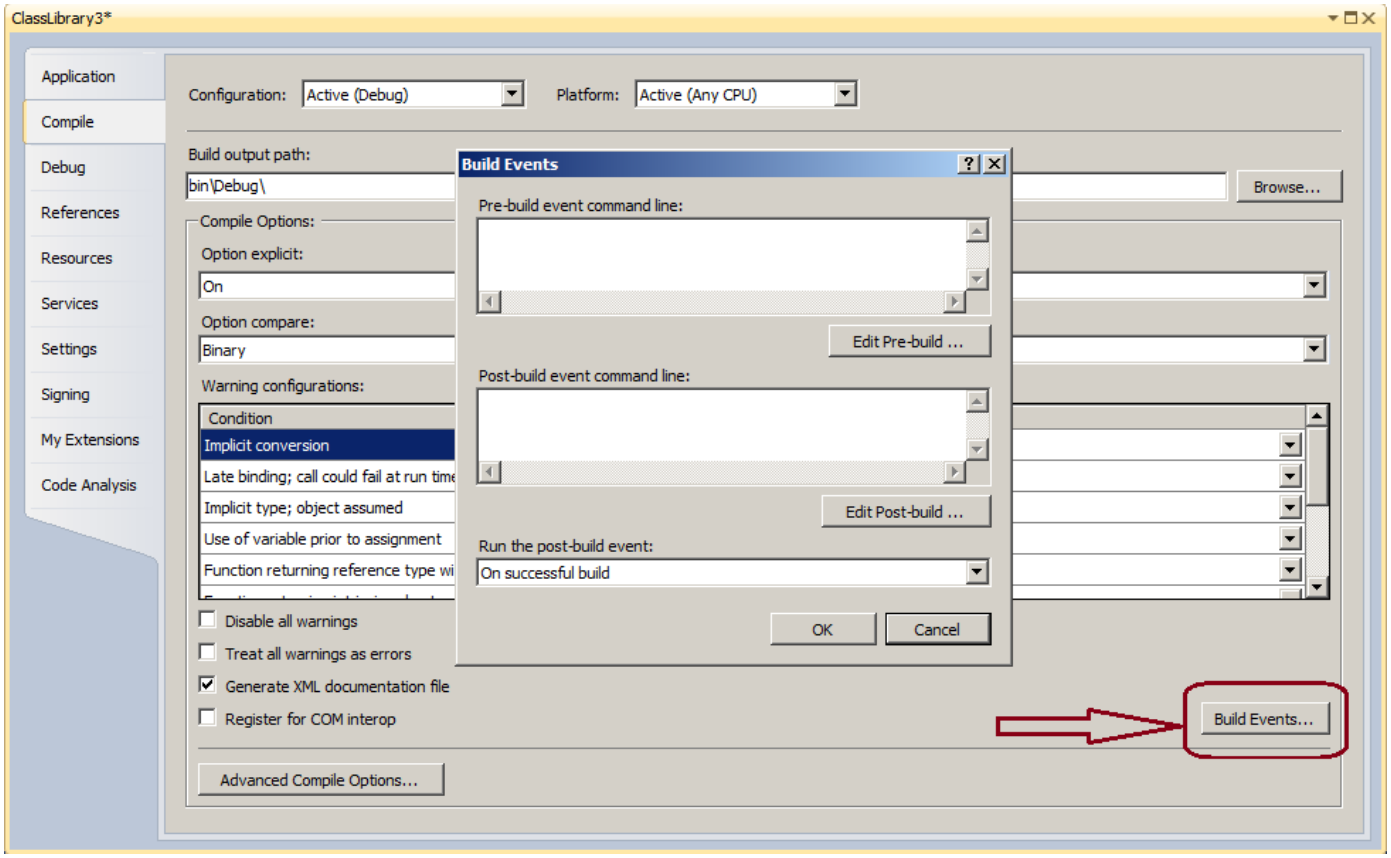
عنوان: **Build Events**
نویسنده: یوسف نژاد
تاریخ: ۱۳۹۱/۱۱/۱۱ ۲۳:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: Visual Studio, MSBuild, Build Events

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 دقیقا 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 متن 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:



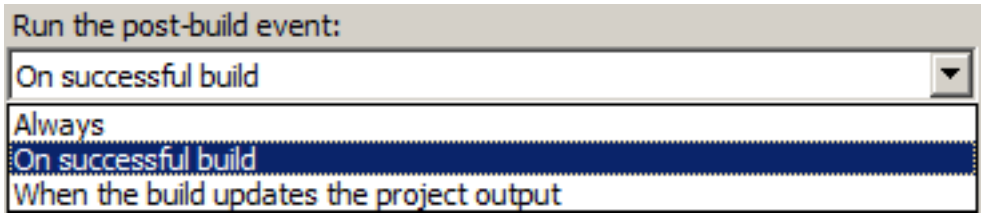
البته در پروژه‌های VB.NET مسیر منتهی به این قسمت کمی فرق میکند که در تصویر زیر نشان داده شده است:



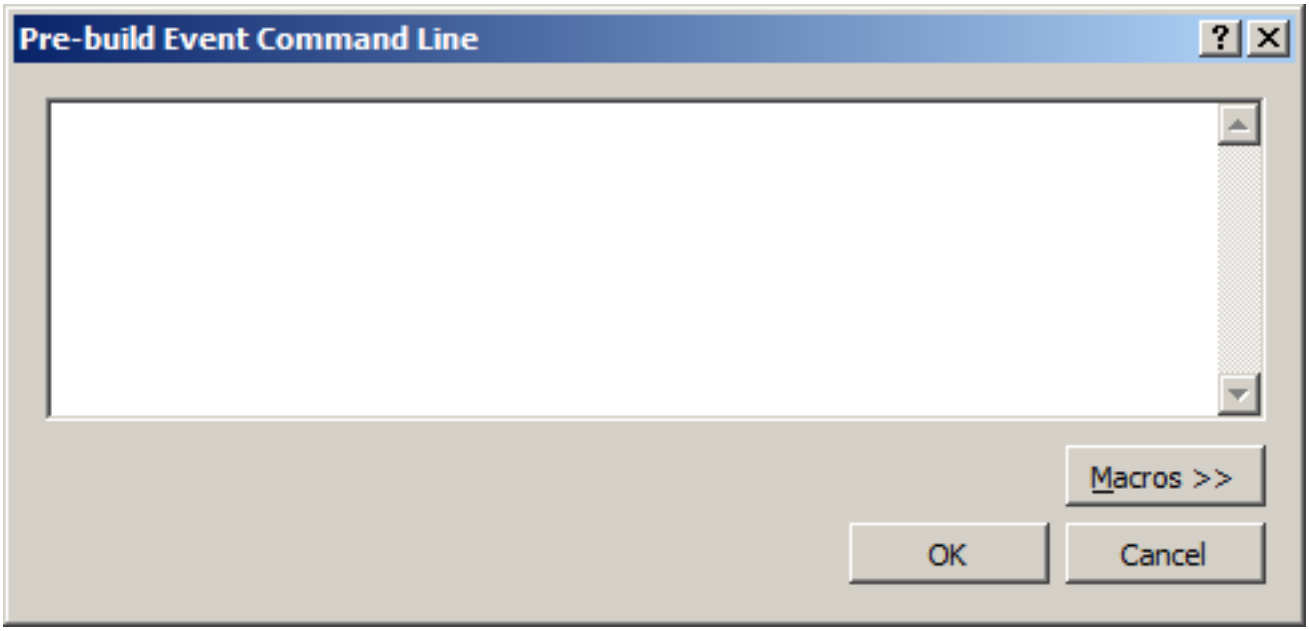
در پروژه‌های مربوط به زبانهای دیگر هم مسیر رسیدن به این رویدادها کمی متفاوت است. برای کسب اطلاعات بیشتر به اینجا مراجعه کنید.

در این قسمت میتوان همانند یک فایل batch دستورات موردنظر را در خطوط مجزا برای اجرا اضافه کرد. از این دستورات معمولا 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: درصورتیکه 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 بید، 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:



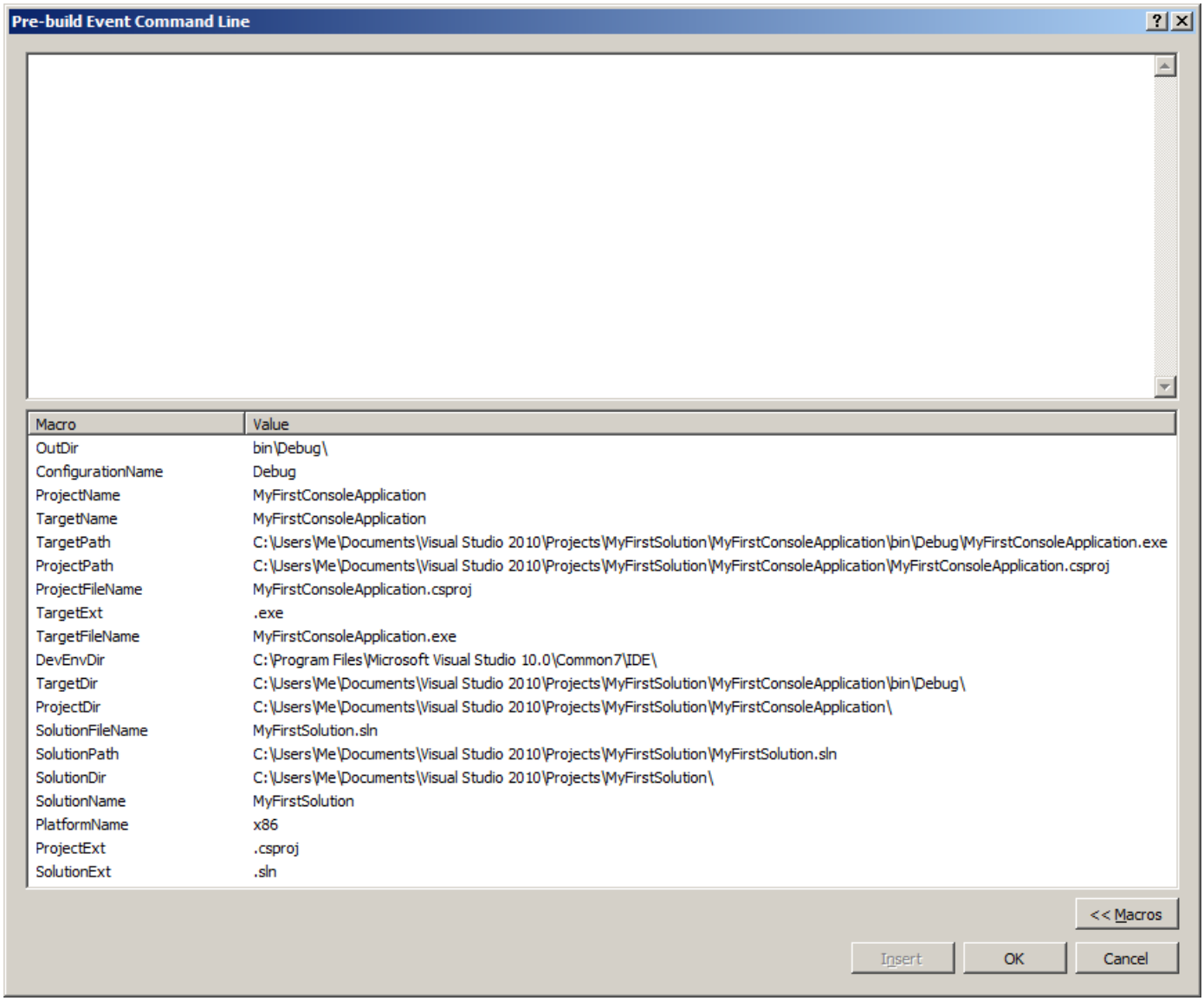
برای استفاده راحتتر از این ویژگی فرمی مخصوص وارد کردن این دستورات در ویژوال استودیو وجود دارد. برای دیدن این فرم بر روی دکمه Edit Pre-build... یا Edit Post-build... کلیک کنید. پنجره زیر نمایش داده میشود:



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. مثلا:

```
$(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. مثلا دستور زیر را درنظر بگیرید:

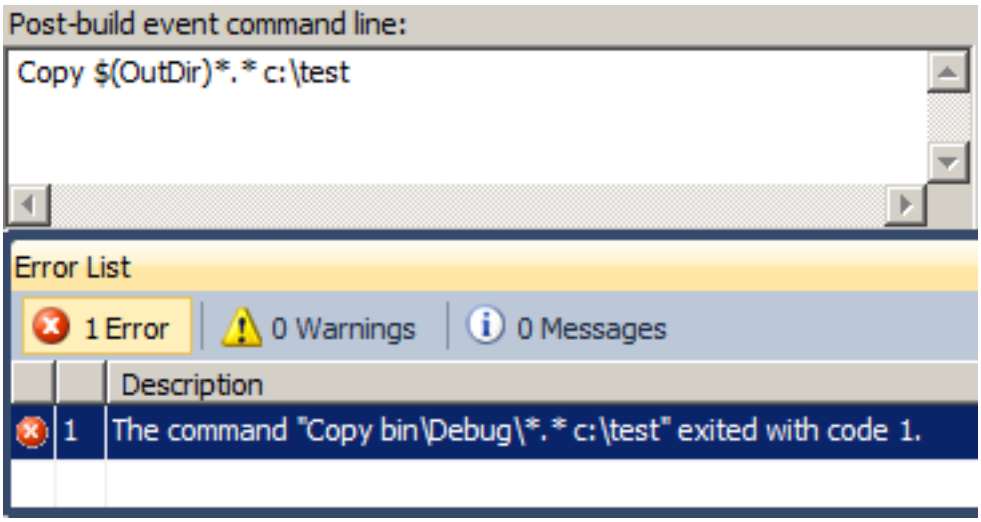
```
Copy $(OutDir)*.* %WinDir%
```

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 رخداده وجود ندارد. یعنی درصورتیکه مثلا 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 ترنزکشنال 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
```

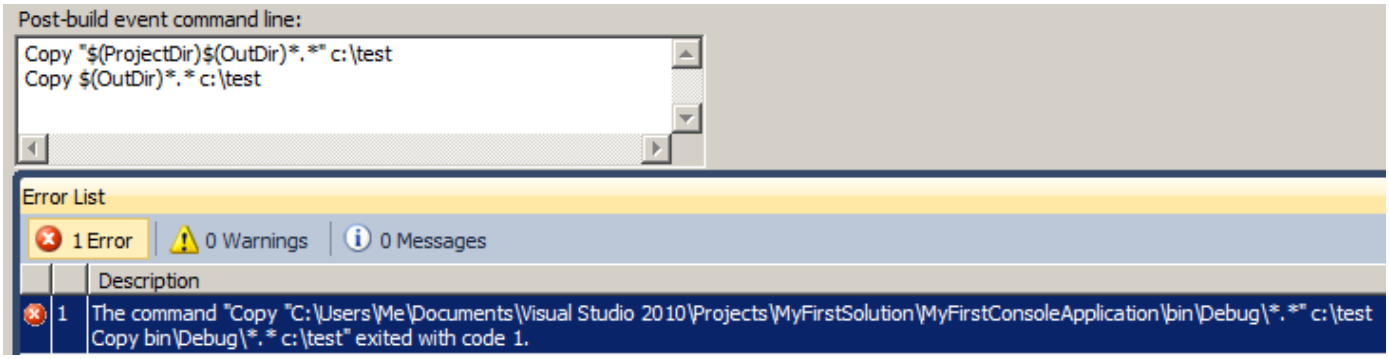
نکته: به دلیل استفاده از کاراکتر فاصله به عنوان جداکننده آرگومانها در دستورات DOS، وجود فاصله در مسیرهای مورد استفاده در این دستورات عملیات را دچار خطا خواهد کرد. راه‌حل استفاده از کاراکتر " در ابتدا و انتهای رشته‌های مربوط به مسیرها همانند دستور بالاست.

نکته: درصورت 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. مثلا:

```
"$(ProjectDir)postBuild.bat"   "$(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 درنظر 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 اسکریپت 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: درصورت 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 غیرصفر 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: درصورتیکه 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 بهر برداری 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: ۱۳۹۱/۱۱/۱۵ ۱۸:۹

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: ۱۳۹۲/۰۳/۰۷ ۱:۵۹

سلام با تشکر از پست شما
من می‌خوام اندازه پشته توی ویزوال 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: ۱۳۹۲/۰۳/۰۷ ۹:۴۹

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

نکته اول: ماکروی TargetFileName فقط اسم فایل خروجی پروژه رو برمی‌گردونه، درصورتیکه 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 کافیه که خط زیر رو در ابتدای مجموعه دستورات build events خودتون قرار بدین:

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

این بچ فایل حاوی دستوراتی نسبتا مفصل برای تنظیم تنظیمات موردنیاز است. درواقع با اجرای این بچ فایل هر خط فرمانی تقریبا 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 کافیه 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 تقریبا 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: ۱۳۹۲/۰۳/۰۷ ۹:۵۳

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.

عنوان: **طراحی افزونه پذیر با ASP.NET MVC 4.x/5.x - قسمت اول**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۷ ۱۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, Build Events, Dependency Injection, Plugin, StructureMap

در طی چند قسمت، نحوه‌ی طراحی یک سیستم افزونه پذیر را با ASP.NET MVC بررسی خواهیم کرد. عناوین مواردی که در این سری پیاده سازی خواهند شد به ترتیب ذیل هستند:
1- چگونه Areaهای استاندارد را تبدیل به یک افزونه‌ی مجزا و منتقل شده‌ی به یک اسمبلی دیگر کنیم.
2- چگونه ساختار پایه‌ای را جهت تامین نیازهای هر افزونه جهت تزریق وابستگی‌ها تا ثبت مسیریابی‌ها و امثال آن تدارک ببینیم.
3- چگونه فایل‌های JS ، CSS و همچنین تصاویر ثابت هر افزونه را داخل اسمبلی آن قرار دهیم تا دیگر نیازی به ارائه‌ی مجزای آن‌ها نباشد.
4- چگونه Entity Framework Code-First را با این طراحی یکپارچه کرده و از آن جهت یافتن خودکار مدل‌ها و موجودیت‌های خاص هر افزونه استفاده کنیم؛ به همراه مباحث Migrations خودکار و همچنین پیاده سازی الگوی واحد کار.

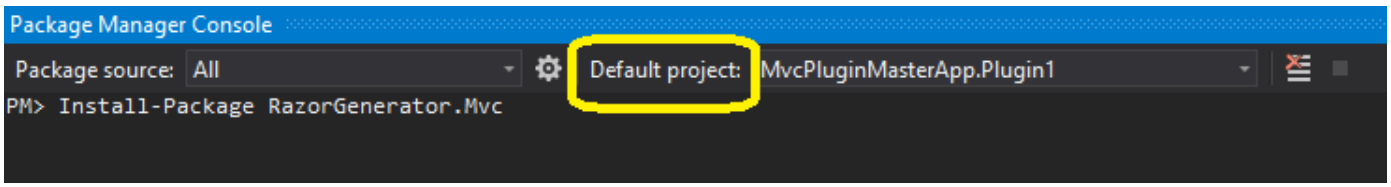
در مطلب جاری، موارد اول و دوم بررسی خواهند شد. پیشنیازهای آن مطالب ذیل هستند:
الف) منظور از یک Area چیست؟
ب) توزیع پروژه‌های ASP.NET MVC بدون ارائه فایل‌های View آن
ج) آشنایی با تزریق وابستگی‌ها در ASP.NET MVC و همچنین اصول طراحی یک سیستم افزونه پذیر به کمک StructureMap
د) آشنایی با رخدادهای Build

تبدیل یک Area به یک افزونه‌ی مستقل

روش‌های زیادی برای خارج کردن Areaهای استاندارد ASP.NET MVC از یک پروژه و قرار دادن آن‌ها در اسمبلی‌های دیگر وجود دارند؛ اما در حال حاضر تنها روشی که نگهداری می‌شود و همچنین اعضای آن همان اعضای تیم نیوگت و ASP.NET MVC هستند، همان روش استفاده از Razor Generator است.
بنابراین ساختار ابتدایی پروژه‌ی افزونه پذیر ما به صورت ذیل خواهد بود:
1) ابتدا افزونه‌ی Razor Generator را نصب کنید.
2) سپس یک پروژه‌ی معمولی ASP.NET MVC را آغاز کنید. در این سری نام MvcPluginMasterApp برای آن در نظر گرفته شده‌است.
3) در ادامه یک پروژه‌ی معمولی دیگر ASP.NET MVC را نیز به پروژه‌ی جاری اضافه کنید. برای مثال نام آن در اینجا MvcPluginMasterApp.Plugin1 تنظیم شده‌است.
4) به پروژه‌ی MvcPluginMasterApp.Plugin1 یک Area جدید و معمولی را به نام NewsArea اضافه کنید.
5) از پروژه‌ی افزونه، تمام پوشه‌های غیر Area را حذف کنید. پوشه‌های Controllers و Models و Views حذف خواهند شد. همچنین فایل global.asax آن‌را نیز حذف کنید. هر افزونه، کنترلرها و Viewهای خود را از طریق Area مرتبط دریافت می‌کند و در این حالت دیگر نیازی به پوشه‌های Controllers و Models و Views واقع شده در ریشه‌ی اصلی پروژه‌ی افزونه نیست.
6) در ادامه کنسول پاور شل نیوگت را باز کرده و دستور ذیل را صادر کنید:

```
PM> Install-Package RazorGenerator.Mvc
```

این دستور را باید یکبار بر روی پروژه‌ی اصلی و یکبار بر روی پروژه‌ی افزونه، اجرا کنید.

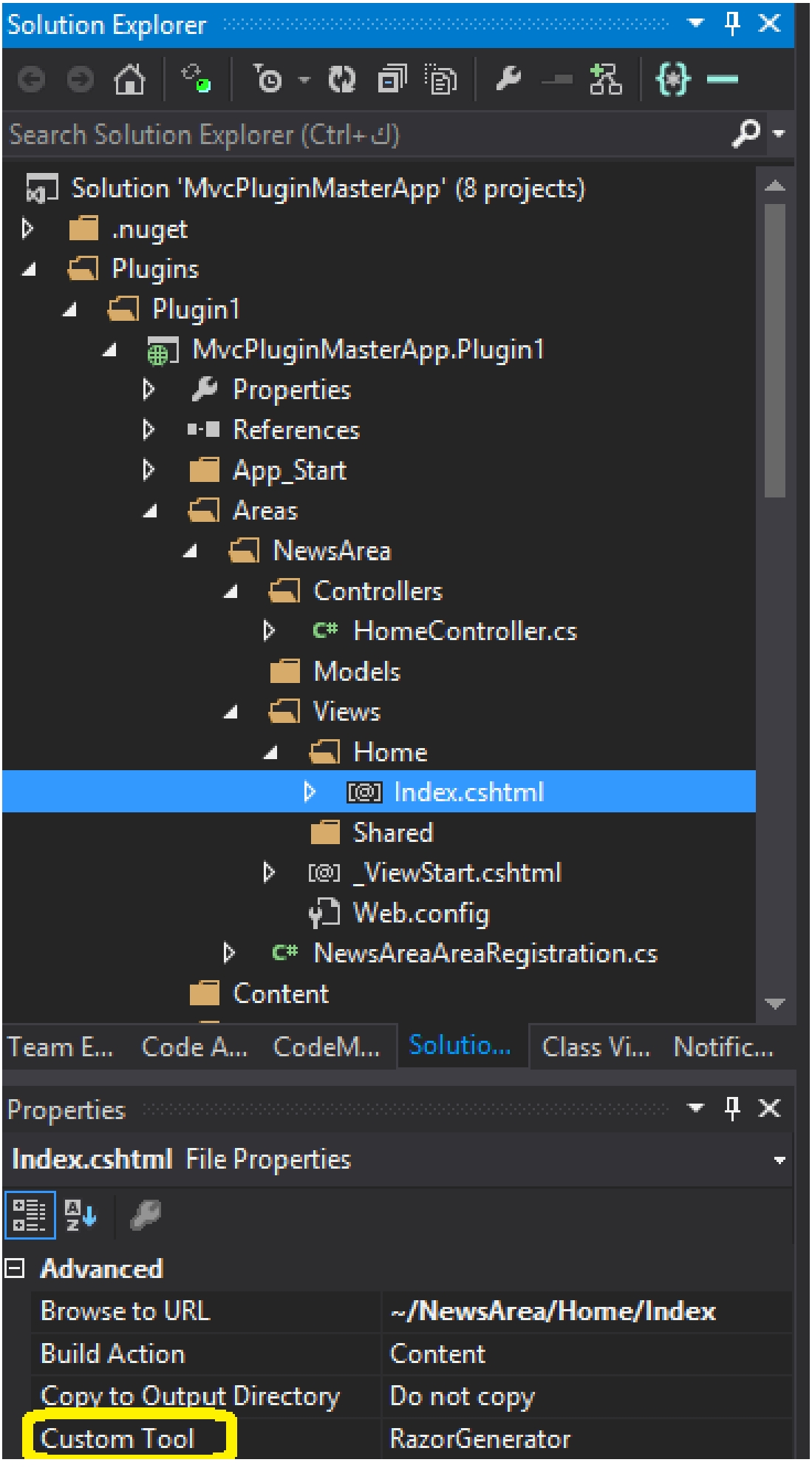


همانطور که در تصویر نیز مشخص شده‌است، برای اجرای دستور نصب RazorGenerator.Mvc نیاز است هربار پروژه‌ی پیش فرض را تغییر دهید.

7) اکنون پس از نصب RazorGenerator.Mvc، نوبت به اجرای آن بر روی هر دو پروژه‌ی اصلی و افزونه است:

```
PM> Enable-RazorGenerator
```

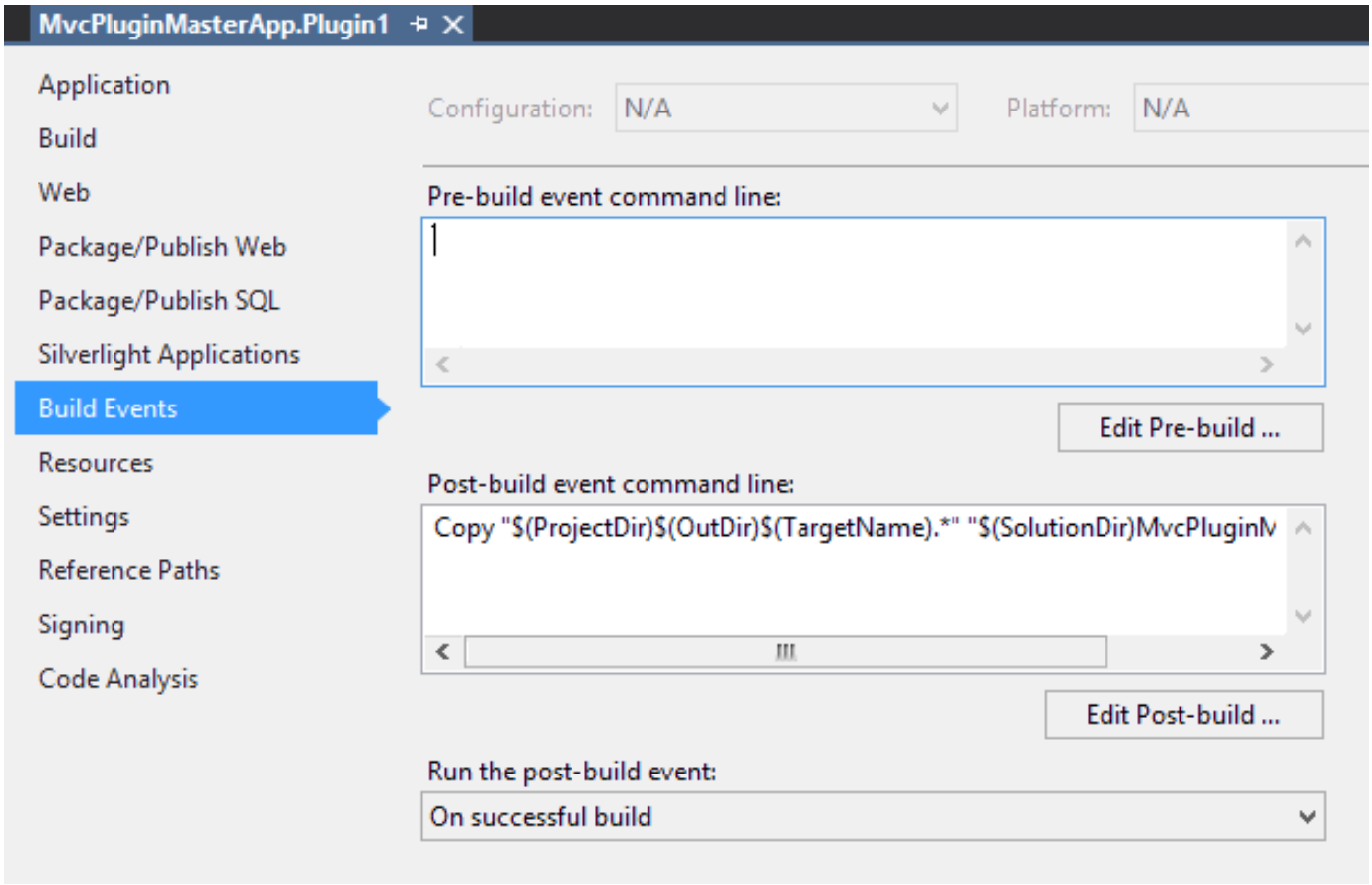
بدیهی است این دستور را نیز باید همانند تصویر فوق، یکبار بر روی پروژه‌ی اصلی و یکبار بر روی پروژه‌ی افزونه اجرا کنید.
همچنین هربار که View جدیدی اضافه می‌شود نیز باید این‌کار را تکرار کنید یا اینکه مطابق شکل زیر، به خواص View جدید مراجعه کرده و Custom tool آن‌را به صورت دستی به RazorGenerator تنظیم نمائید. دستور Enable-RazorGenerator این‌کار را به صورت خودکار انجام می‌دهد.


Solution Explorer
Search Solution Explorer (Ctrl+ک)
Solution 'MvcPluginMasterApp' (8 projects)
.nuget
Plugins
Plugin1
MvcPluginMasterApp.Plugin1
Properties
References
App_Start
Areas
NewsArea
Controllers
HomeController.cs
Models
Views
Home
Index.cshtml
Shared
_ViewStart.cshtml
Web.config
NewsAreaAreaRegistration.cs
Content
Team E...
Code A...
CodeM...
Solutio...
Class Vi...
Notific...
Properties
Index.cshtml File Properties
Advanced
Browse to URL
~/NewsArea/Home/Index
Build Action
Content
Copy to Output Directory
Do not copy
Custom Tool
RazorGenerator

تا اینجا موفق شدیم Viewهای افزونه را داخل فایل dll آن مدفون کنیم. به این ترتیب با کپی کردن افزونه به پوشه‌ی bin پروژه‌ی اصلی، دیگر نیازی به ارائه‌ی فایل‌های View آن نیست و تمام اطلاعات کنترلرها، مدل‌ها و Viewها به صورت یکجا از فایل dll افزونه‌ی ارائه شده خوانده می‌شوند.

کپی کردن خودکار افزونه به پوشه‌ی Bin پروژه‌ی اصلی

پس از اینکه ساختار اصلی کار شکل گرفت، هربار پس از کامپایل افزونه (یا افزونه‌ها)، نیاز است فایل‌های پوشه‌ی bin آن‌را به پوشه‌ی bin پروژه‌ی اصلی کپی کنیم (پروژه‌ی اصلی در این حالت هیچ ارجاع مستقیمی را به افزونه‌ی جدید نخواهد داشت). برای خودکار سازی این کار، به خواص پروژه‌ی افزونه مراجعه کرده و قسمت Build events آن‌را به نحو ذیل تنظیم کنید:



در اینجا دستور ذیل در قسمت Post-build event نوشته شده است:

```
Copy "$(ProjectDir)$(OutDir)$(TargetName).*" "$(SolutionDir)MvcPluginMasterApp\bin\"
```

و سبب خواهد شد تا پس از هر کامپایل موفق، فایل‌های اسمبلی افزونه به پوشه‌ی bin پروژه‌ی MvcPluginMasterApp به صورت خودکار کپی شوند.

تنظیم فضاهای نام کلیه مسیریابی‌های پروژه

در همین حالت اگر پروژه را اجرا کنید، موتور ASP.NET MVC به صورت خودکار اطلاعات افزونه‌ی کپی شده به پوشه‌ی bin را دریافت و به Application domain جاری اعمال می‌کند؛ برای اینکار نیازی به کد نویسی اضافه‌تری نیست و خودکار است. برای آزمایش آن فقط کافی است یک break point را داخل کلاس RazorGeneratorMvcStart افزونه قرار دهید.
اما ... پس از اجرا، بلافاصله پیام تداخل فضاهای نام را دریافت می‌کنید. خطاهای حاصل عنوان می‌کند که در App domain جاری، دو کنترلر Home وجود دارند؛ یکی در پروژه‌ی اصلی و دیگری در پروژه‌ی افزونه و مشخص نیست که مسیریابی‌ها باید به کدامیک ختم شوند.
برای رفع این مشکل، به فایل NewsAreaAreaRegistration.cs پروژه‌ی افزونه مراجعه کرده و مسیریابی آن‌را به نحو ذیل تکمیل کنید تا فضای نام اختصاصی این Area صریحا مشخص گردد.

```
using System.Web.Mvc;

namespace MvcPluginMasterApp.Plugin1.Areas.NewsArea
{
    public class NewsAreaAreaRegistration : AreaRegistration
    {
        public override string AreaName
        {
            get
            {
                return "NewsArea";
            }
        }

        public override void RegisterArea(AreaRegistrationContext context)
        {
            context.MapRoute(
                "NewsArea_default",
                "NewsArea/{controller}/{action}/{id}",
                // تکمیل نام کنترلر پیش فرض
                new { controller = "Home", action = "Index", id = UrlParameter.Optional },
                // مشخص کردن فضای نام مرتبط جهت جلوگیری از تداخل با سایر قسمت‌های برنامه
                namespaces: new[] { string.Format("{0}.Controllers", this.GetType().Namespace) }
            );
        }
    }
}
```

همینکار را باید در پروژه‌ی اصلی و هر پروژه‌ی افزونه‌ی جدیدی نیز تکرار کرد. برای مثال به فایل RouteConfig.cs پروژه‌ی اصلی مراجعه کرده و تنظیم ذیل را اعمال نمائید:

```
using System.Web.Mvc;
using System.Web.Routing;

namespace MvcPluginMasterApp
{
    public class RouteConfig
    {
        public static void RegisterRoutes(RouteCollection routes)
        {
            routes.IgnoreRoute("{resource}.axd/{*pathInfo}");

            routes.MapRoute(
                name: "Default",
                url: "{controller}/{action}/{id}",
                defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional },
                // مشخص کردن فضای نام مرتبط جهت جلوگیری از تداخل با سایر قسمت‌های برنامه
                namespaces: new[] { string.Format("{0}.Controllers", typeof(RouteConfig).Namespace) }
            );
        }
    }
}
```

بدون تنظیم فضاهای نام هر مسیریابی، امکان استفاده‌ی بهینه و بدون خطا از Areaها وجود نخواهد داشت.

طراحی قرارداد پایه افزونه‌ها

تا اینجا با نحوه‌ی تشکیل ساختار هر پروژه‌ی افزونه آشنا شدیم. اما هر افزونه در آینده نیاز به مواردی مانند منوی اختصاصی در منوی اصلی سایت، تنظیمات مسیریابی اختصاصی، تنظیمات EF و امثال آن نیز خواهد داشت. به همین منظور، یک پروژه‌ی class library جدید را به نام MvcPluginMasterApp.PluginsBase آغاز کنید.
سپس قرار داد IPlugin را به نحو ذیل به آن اضافه نمائید:

```
using System;
using System.Reflection;
using System.Web.Optimization;
using System.Web.Routing;
using StructureMap;

namespace MvcPluginMasterApp.PluginsBase
{
    public interface IPlugin
    {
        EfBootstrapper GetEfBootstrapper();
        MenuItem GetMenuItem(RequestContext requestContext);
        void RegisterBundles(BundleCollection bundles);
        void RegisterRoutes(RouteCollection routes);
        void RegisterServices(IContainer container);
    }

    public class EfBootstrapper
    {
        /// <summary>
        /// Assemblies containing EntityTypeConfiguration classes.
        /// </summary>
        public Assembly[] ConfigurationsAssemblies { get; set; }

        /// <summary>
        /// Domain classes.
        /// </summary>
        public Type[] DomainEntities { get; set; }

        /// <summary>
        /// Custom Seed method.
        /// </summary>
        //public Action<IUnitOfWork> DatabaseSeeder { get; set; }
    }

    public class MenuItem
    {
        public string Name { set; get; }
        public string Url { set; get; }
    }
}
```

پروژه‌ی این قرارداد برای کامپایل شدن، نیاز به بسته‌های نیوگت ذیل دارد:

```
PM> install-package EntityFramework
PM> install-package Microsoft.AspNet.Web.Optimization
PM> install-package structuremap.web
```

همچنین باید به صورت دستی، در قسمت ارجاعات پروژه، ارجاعی را به اسمبلی استاندارد System.Web نیز به آن اضافه نمائید.

توضیحات قرار داد IPlugin

از این پس هر افزونه باید دارای کلاسی باشد که از اینترفیس IPlugin مشتق می‌شود. برای مثال فعلا کلاس ذیل را به افزونه‌ی پروژه اضافه نمائید:

```
using System.Web.Mvc;
using System.Web.Optimization;
using System.Web.Routing;
using MvcPluginMasterApp.PluginsBase;
using StructureMap;

namespace MvcPluginMasterApp.Plugin1
{
    public class Plugin1 : IPlugin
```

```
    {
        public EfBootstrapper GetEfBootstrapper()
        {
            return null;
        }

        public MenuItem GetMenuItem(RequestContext requestContext)
        {
            return new MenuItem
            {
                Name = "Plugin 1",
                Url = new UrlHelper(requestContext).Action("Index", "Home", new { area = "NewsArea" })
            };
        }

        public void RegisterBundles(BundleCollection bundles)
        {
            //todo: ...
        }

        public void RegisterRoutes(RouteCollection routes)
        {
            //todo: add custom routes.
        }

        public void RegisterServices(IContainer container)
        {
            // todo: add custom services.

            container.Configure(cfg =>
            {
                //cfg.For<INewsService>().Use<EfNewsService>();
            });
        }
    }
}
```

در قسمت جاری فقط از متد GetMenuItem آن استفاده خواهیم کرد. در قسمت‌های بعد، تنظیمات EF، تنظیمات مسیریابی‌ها و Bundling و همچنین ثبت سرویس‌های افزونه را نیز بررسی خواهیم کرد.
برای اینکه هر افزونه در منوی اصلی ظاهر شود، نیاز به یک نام، به همراه آدرسی به صفحه‌ی اصلی آن خواهد داشت. به همین جهت در متد GetMenuItem نحوه‌ی ساخت آدرسی را به اکشن متد Index کنترلر Home واقع در Areaایی به نام NewsArea، مشاهده می‌کنید.

بارگذاری و تشخیص خودکار افزونه‌ها

پس از اینکه هر افزونه دارای کلاسی مشتق شده از قرارداد IPlugin شد، نیاز است آن‌ها را به صورت خودکار یافته و سپس پردازش کنیم. این‌کار را به کتابخانه‌ی StructureMap واگذار خواهیم کرد. برای این منظور پروژه‌ی جدیدی را به نام MvcPluginMasterApp.IoCConfig آغاز کرده و سپس تنظیمات آن‌را به نحو ذیل تغییر دهید:

```
using System;
using System.IO;
using System.Threading;
using System.Web;
using MvcPluginMasterApp.PluginsBase;
using StructureMap;
using StructureMap.Graph;

namespace MvcPluginMasterApp.IoCConfig
{
    public static class SmObjectFactory
    {
        private static readonly Lazy<Container> _containerBuilder =
            new Lazy<Container>(defaultContainer, LazyThreadSafetyMode.ExecutionAndPublication);

        public static IContainer Container
        {
            get { return _containerBuilder.Value; }
        }

        private static Container defaultContainer()
```

```
        {
            return new Container(cfg =>
            {
                cfg.Scan(scanner =>
                {
                    scanner.AssembliesFromPath(
                        path: Path.Combine(HttpRuntime.AppDomainAppPath, "bin"),
                            // یک اسمبلی نباید دوبار بارگذاری شود
                        assemblyFilter: assembly =>
                        {
                            return !assembly.FullName.Equals(typeof(IPlugin).Assembly.FullName);
                        });

                    scanner.WithDefaultConventions(); //Connects 'IName' interface to 'Name' class
automatically.
                    scanner.AddAllTypesOf<IPlugin>().NameBy(item => item.FullName);
                });
            });
        }
    }
}
```

این پروژه‌ی class library جدید برای کامپایل شدن نیاز به بسته‌های نیوگت ذیل دارد:

```
PM> install-package EntityFramework
PM> install-package structuremap.web
```

همچنین باید به صورت دستی، در قسمت ارجاعات پروژه، ارجاعی را به اسمبلی استاندارد System.Web نیز به آن اضافه نمائید.

کاری که در کلاس SmObjectFactory انجام شده، بسیار ساده است. مسیر پوشه‌ی Bin پروژه‌ی اصلی به structuremap معرفی شده‌است. سپس به آن گفته‌ایم که تنها اسمبلی‌هایی را که دارای اینترفیس IPlugin هستند، به صورت خودکار بارگذاری کن. در ادامه تمام نوع‌های IPlugin را نیز به صورت خودکار یافته و در مخزن تنظیمات خود، اضافه کن.

تامین نیازهای مسیریابی و Bundling هر افزونه به صورت خودکار

در ادامه به پروژه‌ی اصلی مراجعه کرده و در پوشه‌ی App_Start آن کلاس ذیل را اضافه کنید:

```
using System.Linq;
using System.Web.Optimization;
using System.Web.Routing;
using MvcPluginMasterApp;
using MvcPluginMasterApp.IoCConfig;
using MvcPluginMasterApp.PluginsBase;

[assembly: WebActivatorEx.PostApplicationStartMethod(typeof(PluginsStart), "Start")]

namespace MvcPluginMasterApp
{
    public static class PluginsStart
    {
        public static void Start()
        {
            var plugins = SmObjectFactory.Container.GetAllInstances<IPlugin>().ToList();
            foreach (var plugin in plugins)
            {
                plugin.RegisterServices(SmObjectFactory.Container);
                plugin.RegisterRoutes(RouteTable.Routes);
                plugin.RegisterBundles(BundleTable.Bundles);
            }
        }
    }
}
```

بدیهی است در این حالت نیاز است ارجاعی را به پروژه‌ی MvcPluginMasterApp.PluginsBase به پروژه‌ی اصلی اضافه کنیم.
دراینجا با استفاده از کتابخانه‌ای به نام WebActivatorEx (که باز هم توسط نویسندگان اصلی Razor Generator تهیه شده‌است)، یک متد PostApplicationStartMethod سفارشی را تعریف کرده‌ایم.

مزیت استفاده از اینکار این است که فایل Global.asax.cs برنامه شلوغ نخواهد شد. در غیر اینصورت باید تمام این کدها را در انتهای متد Application_Start قرار می‌دادیم.
در اینجا با استفاده از structuremap، تمام افزونه‌های موجود به صورت خودکار بررسی شده و سپس پیشنیازهای مسیریابی و Bundling و همچنین تنظیمات IoC Container مورد نیاز آن‌ها به هر افزونه به صورت مستقل، تزریق خواهد شد.

اضافه کردن منوهای خودکار افزونه‌ها به پروژه‌ی اصلی

پس از اینکه کار پردازش اولیه‌ی IPluginها به پایان رسید، اکنون نوبت به نمایش آدرس اختصاصی هر افزونه در منوی اصلی سایت است. برای این منظور فایل جدیدی را به نام _PluginsMenu.cshtml، در پوشه‌ی shared پروژه‌ی اصلی اضافه کنید؛ با این محتوا:

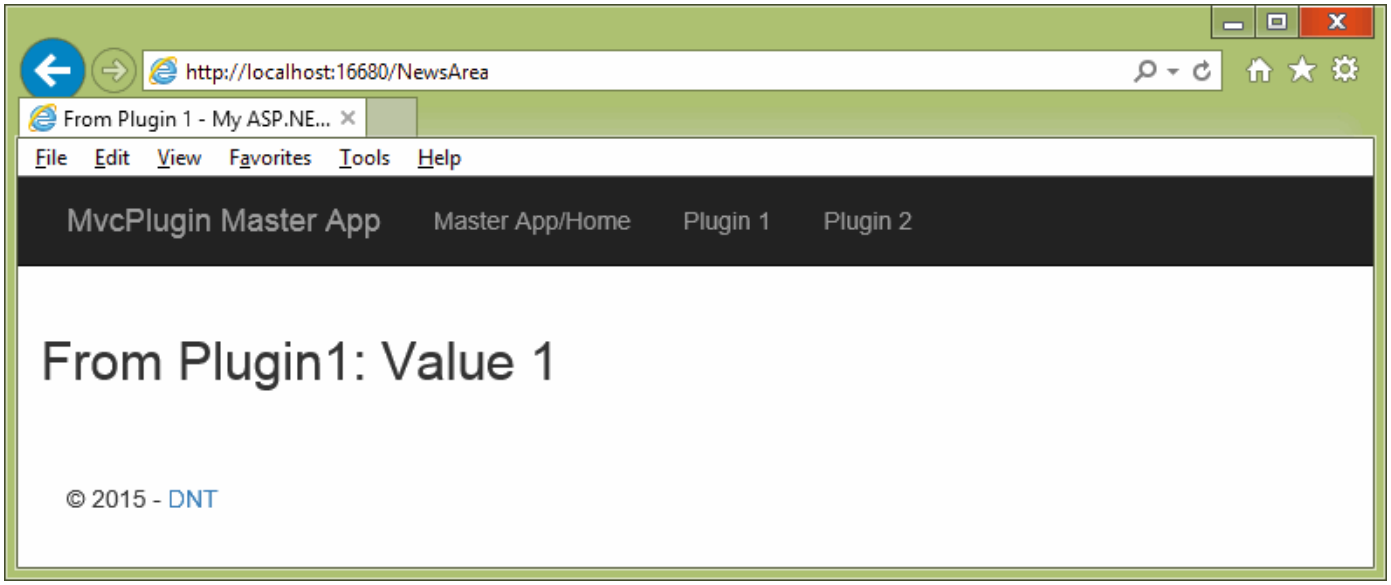
```
@using MvcPluginMasterApp.IoCConfig
@using MvcPluginMasterApp.PluginsBase
@{
    var plugins = SmObjectFactory.Container.GetAllInstances<IPlugin>().ToList();
}

@foreach (var plugin in plugins)
{
    var menuItem = plugin.GetMenuItem(this.Request.RequestContext);
    <li>
        <a href="@menuItem.Url">@menuItem.Name</a>
    </li>
}
```

در اینجا تمام افزونه‌ها به کمک structuremap یافت شده و سپس آیتم‌های منوی آن‌ها به صورت خودکار دریافت و اضافه می‌شوند.
سپس به فایل _Layout.cshtml پروژه‌ی اصلی مراجعه و توسط فراخوانی Html.RenderPartial، آن‌را در بین سایر آیتم‌های منوی اصلی اضافه می‌کنیم:

```
<div class="navbar navbar-inverse navbar-fixed-top">
    <div class="container">
        <div class="navbar-header">
            <button type="button" class="navbar-toggle" data-toggle="collapse" data-target=".navbar-
collapse">
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
                <span class="icon-bar"></span>
            </button>
            @Html.ActionLink("MvcPlugin Master App", "Index", "Home", new { area = "" }, new { @class =
"navbar-brand" })
        </div>
        <div class="navbar-collapse collapse">
            <ul class="nav navbar-nav">
                <li>@Html.ActionLink("Master App/Home", "Index", "Home", new {area = ""}, null)</li>
                @{ Html.RenderPartial("_PluginsMenu"); }
            </ul>
        </div>
    </div>
</div>
```

اکنون اگر پروژه را اجرا کنیم، یک چنین شکلی را خواهد داشت:



بنابراین به صورت خلاصه

1) هر افزونه، یک پروژه‌ی کامل ASP.NET MVC است که پوشه‌های ریشه‌ی اصلی آن حذف شده‌اند و اطلاعات آن توسط یک Area جدید تامین می‌شوند.
2) تنظیم فضای نام مسیریابی‌های تمام پروژه‌ها را فراموش نکنید. در غیر اینصورت شاهد تداخل پردازش کنترلرهای هم نام خواهید بود.
3) جهت سهولت کار، می‌توان فایل‌های bin هر افزونه را توسط رخداد post-build، به پوشه‌ی bin پروژه‌ی اصلی کپی کرد.
4) Viewهای هر افزونه توسط Razor Generator در فایل dll آن مدفون خواهند شد.
5) هر افزونه باید دارای کلاسی باشد که اینترفیس IPlugin را پیاده سازی می‌کند. از این اینترفیس برای ثبت اطلاعات هر افزونه یا دریافت اطلاعات سفارشی از آن کمک می‌گیریم.
6) با استفاده از استراکچرمپ و قرارداد IPlugin، منوهای هر افزونه را به صورت خودکار یافته و سپس به فایل layout اصلی اضافه می‌کنیم.

کدهای کامل این قسمت را از اینجا می‌توانید دریافت کنید:

MvcPluginMasterApp-Part1.zip

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۴/۰۱/۲۷ ۱۶:۲۱

یک سوال، هنگام حذف افزونه با توجه به اینکه ممکنه کاربری در حال کار با بخش‌های مختلف اون باشه چه اتفاقی برای حذف ارجاع‌های اون به برنامه می‌افتد؟ آیا اجازه حذف لازم است؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۷ ۱۶:۲۵

- برنامه‌ی اصلی ارجاع مستقیمی را به هیچ افزونه‌ای ندارد.
+ هر نوع تغییری در پوشه‌ی bin برنامه سبب ری استارت آن خواهد شد. بنابراین اگر افزونه‌ای اضافه شود، برنامه به صورت خودکار ری استارت شده و بلافاصله افزونه‌ی جدید، قابل استفاده خواهد بود. اگر فایل افزونه‌ای از پوشه‌ی bin حذف شود، باز هم سبب ری استارت برنامه و بارگذاری خودکار منوها و محاسبه‌ی مجدد آن‌ها می‌گردد که اینبار دیگر شامل اطلاعات افزونه‌ی حذف شده نیست.

نویسنده: حامد67
تاریخ: ۱۳۹۴/۰۱/۲۷ ۲۱:۴۱

سلام؛ یه سوال امنیتی، آیا راهکاری دارید که کسی به طور غیر مجاز برای برنامه پلاگین ننویسه منظور این هستش که فردی که پلاگین رو نوشته فقط با تایید بتونه فعالش کنه و از لحاظ امنیتی قابل چک باشه و بدون تایید اجرایی نشه چون من نگران هستم فردی پلاگین بنویسد و عمدا یا غیر عمد پلاگینی توسعه دهد که اطلاعات و روند فعالیت برنامه را جاسوسی کند
خودم این ذهنیت رو دارم که هش کد هر پلاگین باید توسط مدیر تایید بشه و سپس قابل اجرا باشه تا کسی نتونه بعدا پلاگین را تغییر بده و امنیت سیستم را به خطر بنداره
در کل ملاحظات امنیتی پاگین‌ها را چگونه در نظر بگیریم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۲۷ ۲۲:۵

از مطلب « تهیه XML امضاء شده جهت تولید مجوز استفاده از برنامه » ایده بگیرید. یک متد GetLicense به اینترفیس IPlugin اضافه کنید و در آن مجوز ارائه شده توسط افزونه را در برنامه‌ی اصلی بررسی کنید (در کلاس PluginsStart و همچنین فایل _PluginsMenu.cshtml). فقط کسانی می‌توانند «XML امضاء شده» تولید کنند که دسترسی به کلیدهای خصوصی و امن شما را داشته باشند.

نویسنده: میثم99
تاریخ: ۱۳۹۴/۰۱/۳۰ ۱۵:۵۳

سلام؛ اگر بخواهیم مسیر یابی پروژه را به attribute routing تغییر بدهیم چه کارهایی باید انجام دهیم.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۱/۳۰ ۱۸:۲۶

از این مطالب تکمیلی استفاده کنید:
- « قابلیت Attribute Routing در ASP.NET MVC 5 »
- « Attribute Routing در ASP.NET MVC 5 »