


پروژه دیگری از آقای [David Ebbo](#) (عضو تیم ASP.NET که پیشتر با پروژه [T4 MVC](#) آن در این سایت آشنا شده‌اید)، جهت کامپایل کامل فایل‌های View و ارائه پروژه نهایی ASP.NET MVC بدون نیاز به ارائه پوشه Views آن به نام Razor Generator وجود دارد که در ادامه خلاصه‌ای از نحوه استفاده از آن را مرور خواهیم کرد.

الف) ابتدا افزونه Razor Generator را [از اینجا](#) دریافت و نصب کنید.

ب) سپس به پروژه MVC خود بسته NuGet زیر را اضافه نمائید:

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

در این حالت باید پروژه پیش فرض، همان وب سایت MVC شما انتخاب گردد:



با اضافه کردن این بسته NuGet تغییرات زیر به پروژه جاری اعمال خواهند شد:

- ارجاعی به اسمبلی RazorGenerator.Mvc.dll به پروژه اضافه خواهد شد.
- در پوشه App_Start، فایل به نام RazorGeneratorMvcStart.cs اضافه گردیده و کار تنظیم موتور View مخصوص کار با Viewهای کامپایل شده را انجام می‌دهد.

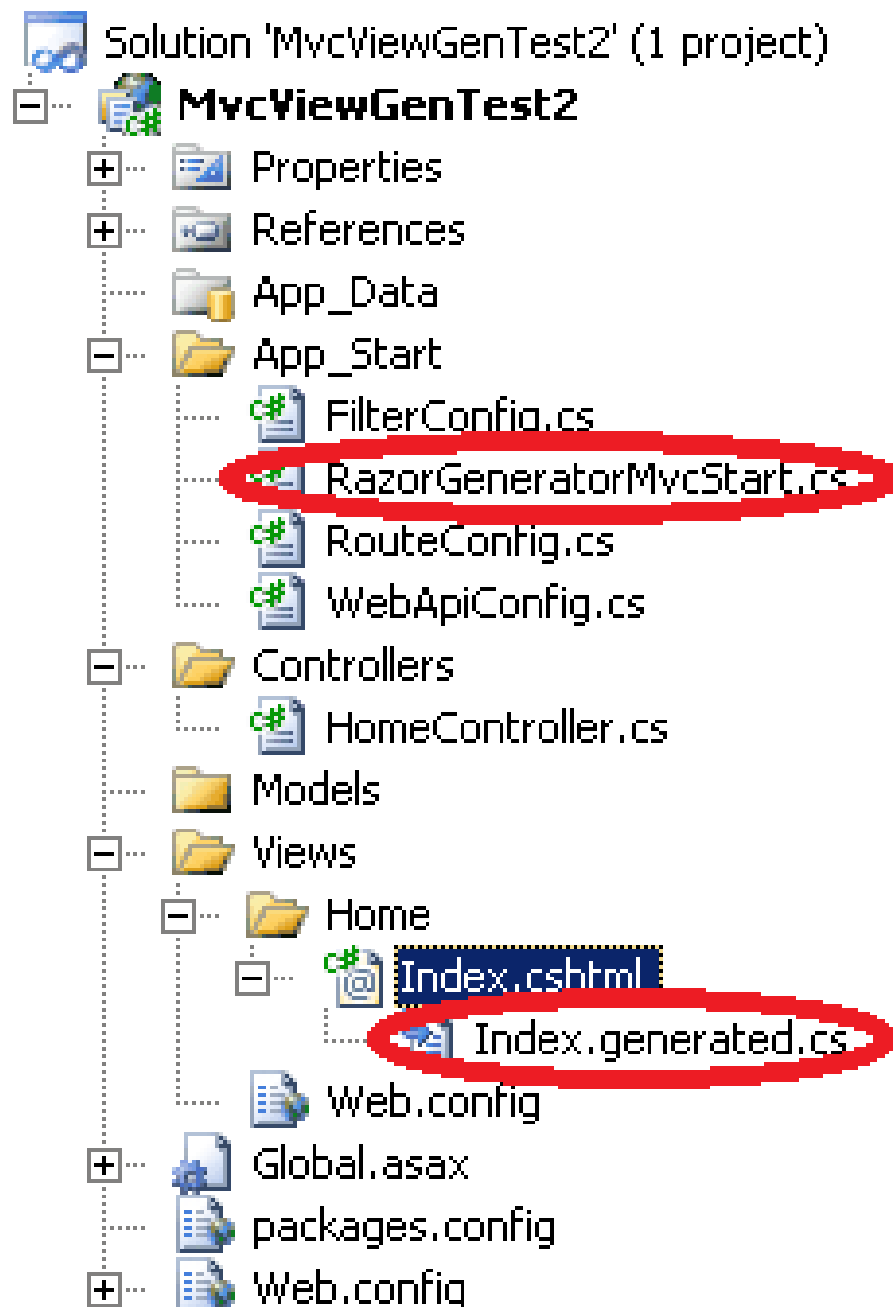
ج) پس از نصب بسته NuGet یاد شده در همان خط فرمان پاورشل نوگت دستور زیر را صادر کنید:

```
PM> Enable-RazorGenerator
```

و ... همین!

پس از انجام اینکار، دو کار صورت خواهد گرفت:

- برای تمام Viewهای برنامه، فایل cs متناظری تولید می‌شود که ذیل فایل‌های View قابل مشاهده است.
- گزینه Custom tool این Viewها نیز به RazorGenerator تنظیم می‌شود.



Properties

Index.cshtml File Properties

Advanced	
Build Action	Content
Copy to Output Directory	Do not copy
Custom Tool	RazorGenerator
Custom Tool Namespace	

بدیهی است اگر از دستور Enable-RazorGenerator استفاده نکنید، نیاز خواهید داشت تا تنظیم گزینه Custom tool به RazorGenerator کلیه Viewها را دستی انجام داده و اگر فایل cs متناظر با View تولید نشد روی فایل view کلیک راست کرده و گزینه run custom tool را انتخاب کنید. اما دستور Enable-RazorGenerator کار را بسیار ساده می‌کند.

مزایا:

- در عمل موتور ASP.NET همین کارها را در زمان اولین بار اجرای Viewها (ی کامپایل نشده) در پشت صحنه انجام می‌دهد. بنابراین با این روش زمان آغاز برنامه سریعتر می‌شود.
- دیگر نیازی به توزیع فایل‌های cshtml نخواهید داشت.
- خطایابی Viewها نیز ساده‌تر می‌شود. از این جهت که کامپایل آن‌ها به زمان اجرا موکول نخواهد شد.

یک سری قابلیت‌های دیگر نیز به همراه این پروژه است مانند انتقال Viewها به یک اسمبلی دیگر و یا استفاده از MSBuild برای انجام عملیات که می‌توانید آن‌ها را در [Wiki پروژه Razor Generator](#) مطالعه کنید. انتقال Viewها به یک اسمبلی دیگر هرچند در این روش کاملاً ممکن شده و کار می‌کند اما صفحه dialog افزودن یک view جدید مهیا در کلیک راست بر روی اکشن متدهای یک کنترلر را غیرفعال می‌کند که در عمل آنچنان جالب نیست.

یک نکته مهم:

اگر در آینده بسته NuGet و افزونه یاد شده را به روز کردید نیاز است دستور زیر را اجرا کنید:

```
PM> Redo-RazorGenerator
```

به این ترتیب بر اساس ساختار و کدهای جدید RazorGenerator، کلیه فایل‌های cs تولید شده مجدداً به روز و تولید خواهند شد.

فایل‌های Helper قرار گرفته در پوشه App_Code

اگر HTML Helperهای خود را توسط فایل‌های Razor قرار گرفته در پوشه App_Code تولید می‌کنید، پس از اجرای دستور Enable-RazorGenerator، برای این موارد نیز فایل‌های cs متناظری تولید می‌شود. با این تفاوت که Build Action آن‌ها بر روی Compile [قرار ندارند](#) که این مورد را باید دستی تنظیم کنید. همچنین حین استفاده از این توابع کمکی نیاز است فضای نام مرتبط را نیز در ابتدای فایل View خود ذکر کنید مثلاً:

```
@using MvcViewGenTest2.app_code
```

البته با استفاده از Razor Generaor دیگر نیازی به استفاده از پوشه App_Code نخواهد بود؛ از این جهت که کار کامپایل خودکار، به زمان اجرا موکول نخواهد شد. بنابراین اینبار در هر جایی که علاقمند بودید می‌توانید این فایل‌های کمکی را تولید و کامپایل کنید. فقط ذکر فضای نام مرتبط را در ابتدای View خود فراموش نکنید.

حذف فایل RazorGeneratorMvcStart.cs

اگر علاقمند به استفاده از فایل پیش فرض RazorGeneratorMvcStart.cs نیستید و می‌خواهید [موتورهای View اضافی](#) را حذف کنید، ابتدا فایل RazorGeneratorMvcStart.cs را حذف کرده و سپس در فایل global.asax.cs تغییر زیر را اعمال نمایید:

```
using System.Web.Http;
using System.Web.Mvc;
using System.Web.Routing;
using System.Web.WebPages;
using RazorGenerator.Mvc;

namespace MvcViewGenTest2
{
    public class MvcApplication : System.Web.HttpApplication
    {
    }
```

```
protected void Application_Start()
{
    AreaRegistration.RegisterAllAreas();

    WebApiConfig.Register(GlobalConfiguration.Configuration);
    FilterConfig.RegisterGlobalFilters(GlobalFilters.Filters);
    RouteConfig.RegisterRoutes(RouteTable.Routes);

    // Adding PrecompiledMvcEngine
    var engine = new PrecompiledMvcEngine(typeof(MvcApplication).Assembly);
    ViewEngines.Engines.Clear();
    ViewEngines.Engines.Add(engine);
    VirtualPathFactoryManager.RegisterVirtualPathFactory(engine);
}
}
```

نظرات خوانندگان

نویسنده:

پندار

تاریخ:

۱۱:۴۴ ۱۳۹۱/۱۲/۱۷

چگونه می‌شود این توزیع را برای فایل‌های JQuery انجام داد؟

نویسنده:

وحید نصیری

تاریخ:

۱۱:۵۲ ۱۳۹۱/۱۲/۱۷

روش قدیمی: [استفاده از Web Resources](#)

روش پیشنهادی: [Bundling and Minification](#)

نویسنده:

وحید نصیری

تاریخ:

۱۹:۳۳ ۱۳۹۱/۱۲/۲۰

یک نکته تکمیلی

این روش چون با MVC3 هم سازگار است مسیرهای Razor2 مانند مسیر زیر را پشتیبانی نمی‌کند:

```

```

مسیر فایل‌ها حتما باید توسط Url.Content مشخص شود:

```

```

نویسنده:

Green

تاریخ:

۱۸:۱۶ ۱۳۹۱/۱۲/۲۳

اتفاقا چند وقت پیش با RazorGenerator کار کردم اونموقع مطلب شما رو در این باره نخونده بودم با کلی مکافات تونستم ارزش خروجی بگیرم که یه سوال برام پیش اومد و ارزش منصرف شدم. با توجه به اینکه Razorgenerator یه برای هر view یه فایل cs میسازه و توی اون با استفاده از writer تمام متن داخل view رو به خروجی میفرسته استفاده از یه همچین روشی درسته و بار اضافه روی ایجاد viewها نمیزاره ؟
از روش‌های دیگه استفاده کردین؟ مثل SparkViewFactory
فکر میکنید کدوم بهتره؟

نویسنده:

وحید نصیری

تاریخ:

۱۹:۵۰ ۱۳۹۱/۱۲/۲۳

اگر یک فایل پروژه MVC رو باز کنید چنین تنظیمی داخل آن هست

```
<MvcBuildViews>false</MvcBuildViews>
```

با true کردن آن (که یک best practice محسوب میشه) هربار «از ابتدا» همین کاری که توسط razor generator انجام می‌شود، برای پیش کامپایل Viewها انجام خواهد شد.

بنابراین با استفاده از Razor generator به یک مزیت مهم دیگر هم خواهیم رسید:

بالا رفتن سرعت بررسی زمان کامپایل Viewها بدون نیاز به تغییر فایل پروژه. اگر تعداد Viewها زیاد باشد، تغییر MvcBuildViews به true خیلی زمانبر می‌شود تا حدی که پس از چندبار کامپایل پروژه، شاید مجبور به false کردن آن شویم. اما در حالت استفاده از Razor Generator واقعا سرعت بررسی بسیار بالاتر است؛ چون فایل‌های cs مورد نیاز سایر Viewها پیشتر تهیه شده و زمان تولید آن‌ها یک مرحله کاهش پیدا می‌کند.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۳ ۱۲:۲۴

نکته تکمیلی دو

در این روش فایل‌های Shared موجود در پوشه views نیاز است توزیع شوند؛ به همراه وب کانفیگ آن و همچنین viewStart. به توزیع سایر فایل‌های view نیازی نیست.

نویسنده: مولانا
تاریخ: ۱۳۹۲/۰۲/۲۱ ۱۶:۲۵

با سلام.

کدام بخش‌های یک سایت MVC را برای ارائه نهایی باید توزیع کرد؟
شما کدام ابزار را برای توزیع پیشنهاد میکنید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۲۱ ۱۶:۳۲

در روش متداول: منهای پوشه کنترلر (که کامپایل شده آن در پوشه bin موجود است) و obj، مابقی را باید توزیع کرد.
+ اگر از روش مطرح شده در بحث جاری استفاده می‌کنید، « [نکته تکمیلی دو](#) » را که کمی بالاتر مطرح شده لحاظ کنید.

نویسنده: سمیرا قادری
تاریخ: ۱۳۹۲/۰۴/۰۲ ۱۶:۰۸

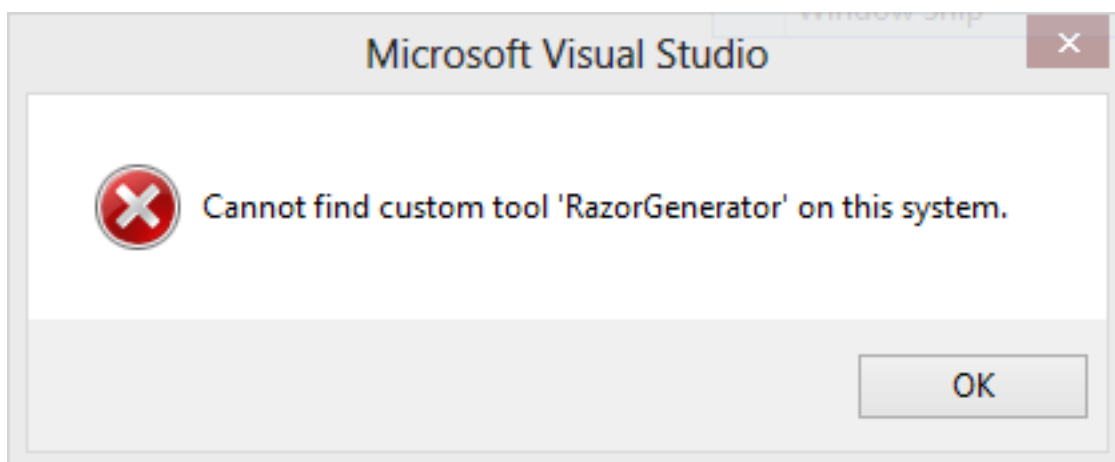
با سلام

چرا بعد از publish دوباره view(ها) ارسال می‌شوند . در صورتی که تمامی کارهای گفته شده در بالا انجام شده ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۴/۰۲ ۱۷:۴۵

publish برای روال معمول و استاندارد کار است. کاری که در اینجا انجام شده غیرمعمول است. مدیریت نهایی انتشار آن دستی خواهد بود (در کل پوشه‌های اسکریپت، bin، content و چند فایل config باید توزیع شوند + [نکته تکمیلی 2](#) که در بالا ذکر شده).

نویسنده: محسن
تاریخ: ۱۳۹۲/۰۷/۱۶ ۱۷:۱۰



سلام آقای نصیری. من نسخه Razor Generator 2.1.2 رو نصب کردم. اما دستور enable-razorgenerator بدون دادن هیچ خطایی انجام می‌شود اما فایل‌های Cs مربوط به View ها تولید نمیشه و وقتی روشن Run Custom Tool رو می‌زنم این پیغام میاد. ممنون اگه راهنماییم کنید.

نویسنده: وحید نصیری
تاریخ: ۱۷:۱۸ ۱۳۹۲/۰۷/۱۶

قسمت (ب) ابتدای بحث کافی نیست. قسمت (الف) توضیح داده شده نیز باید انجام شود .