عنوان: **AngularJS #2**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۰۹ ۱۶:۵۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

بهتر است قبل از این که به ادامه‌ی آموزش بپردازم، دو نکته را متذکر شوم:
1) روند آموزشی این فریمورک از کل به جز است؛ به این معنا که ابتدا تمامی قابلیت‌های اصلی فریمورک را به صورت کلی و بدون وارد شدن به جزئیات بیان می‌کنم و پس از آن، جزئیات را در قالب مثال‌هایی واقعی بیان خواهم کرد.
2) IDE مورد استفاده بنده Visual Studio 2012 است. همچنین از ابتدا پروژه را با ASP.NET MVC شروع می‌کنم. شاید بگویید که می‌شود Angular را بدون درگیر شدن با مباحث ASP.NET MVC بیان کرد؛ اما پاسخ من این است که این مثال‌ها باید قابل پیاده‌سازی در نرم‌افزارهای واقعی باشند و یکی از بسترهای مورد علاقه‌ی من ASP.NET MVC است. اگرچه باز هم تاکید می‌کنم که کلیه‌ی مباحث ذکرشده، برای کلیه‌ی زبان‌های سمت سرور دیگر هم قابل استفاده است و هدف من در اینجا بیان یک سری چالش‌ها در ASP.NET MVC است.

نحوه‌ی دریافت AngularJS

1) NuGet Package Manager
2) دریافت از وب‌سایت angularjs.org

دریافت از طریق Nuget Package Manager

روش ارجح افزودن کتابخانه‌های جانبی در یک پروژه‌ی واقعی، استفاده از NuGet Package Manager است. دلیل آن هم بارها بیان‌شده است از جمله: باخبر شدن از آخرین به‌روزرسانی کتابخانه‌ها، دریافت وابستگی‌های کتاب‌خانه‌ی مورد نظر و نبودن محدودیت تحریم برای دریافت فایل‌ها است.
روش کار هم بسیار ساده است، کافی است که بر روی پروژه کلیک راست کرده و گزینه‌ی Manage NuGet Packages را انتخاب کنید و با جست جو angularjs نسبت به نصب آن اقدام نمایید.
اگر هم با ابزارهای گرافیکی رابطه‌ی خوبی ندارید، می‌توانید از Package Manager Console فراهم‌شده توسط NuGet استفاده کنید. کافی است در کنسول پاورشل آن عبارت زیر را تایپ کنید:

```
Install-Package angularjs
```

پس از نصب angularjs، شاهد تغییراتی در پوشه‌ی Scripts پروژه‌ی خودخواهید بود. تعداد زیادی فایل جاوا اسکریپت که با عبارت angular شروع‌شده‌اند، به این پوشه اضافه شده است. در حال حاضر ما تنها به فایل angular.js نیاز داریم و احتیاجی به فایل‌های دیگر نیست.
همچنین یک پوشه به نام i18n نیز اضافه شده است که برای مباحث Globalization و Internationalization به کار گرفته می‌شود.

دریافت از سایت angularjs.org

برای دریافت Angular از وب سایت رسمی‌اش، به angularjs.org مراجعه کنید؛ اما گویا به دلیل تحریم‌ها این سایت برای IP ایران مسدود شده است (البته افرادی نیز بدون مشکل به آن دسترسی دارند). دکمه‌ی Download را فشار داده و در نهایت کلید دریافت را بزنید. اگر نسخه‌ی کامل آن را دریافت کنید، لیستی از مستندات AngularJS را نیز در فایل دریافتی، خواهید داشت. در هر صورت این روش برای استفاده از angular دریک پروژه‌ی واقعی توصیه نمی‌شود.
پس به عنوان یک best practice، همیشه کتاب‌خانه‌های جانبی را با NuGet دریافت و نصب کنید. رفع موانع تحریم‌ها، یکی از مزایای مهم آن است.
پس از دریافت angular، نوشتن برنامه‌ی معروف Hello, World به وسیله‌ی آن ، می‌تواند بهترین شروع باشد؛ اما اگر اجازه بدهید، نوشتن این برنامه را در قالب توضیح قالب‌های سمت کلاینت انجام دهیم.

قالب‌های سمت کلاینت (Client Side Templates)

در برنامه‌های وب چند صفحه‌ای و یا اکثر وب سایت‌های معمول، داده‌ها و کدهای HTML، در سمت سرور اصطلاحا سرهم و مونتاژ

شده و خروجی نهایی که HTML خام است به مرورگر کاربر ارسال می‌شود. با یک مثال بیشتر توضیح می‌دهم: در ASP.NET MVC معمولا از لحظه‌ای که کاربر صفحه‌ای را درخواست می‌کند تا زمانی که پاسخ خود را در قالب HTML می‌بیند، این فرآیند طی می‌شود: ابتدا درخواست به Controller هدایت می‌شود و سپس اطلاعات مورد نیاز از پایگاه داده خوانده‌شده و در قالب یک Model به View که یک فایل HTML ساده است، منتقل می‌شود. سپس به کمک موتور نمایشی Razor، داده‌ها در جای مناسب خود قرار می‌گیرند و در نهایت، خروجی که HTML خام است به مرورگر کلاینت درخواست‌کننده ارسال می‌شود تا در مرورگر خود نتیجه را مشاهده نماید. روال کار نیز در اکثر SPAهای معمول و یا اصطلاحا برنامه‌های AJAX، باکمی تغییر به همین شکل است.

اما در Angular داستان به شکل دیگری اتفاق می‌افتد؛ Angular قالب HTML و داده‌ها را به صورت جداگانه از سرور دریافت می‌کند و در مرورگر کاربر آن‌ها را سرهم و مونتاژ می‌کند. بدیهی است که در اینجا قالب، یک فایل HTML ساده و داده‌ها می‌تواند به فرم JSON باشد. در نتیجه کار سرور دیگر فراهم کردن قالب و داده‌ها برای کلاینت است و بقیه‌ی ماجرا در سمت کلاینت رخ می‌دهد.

خیلی خوب، مزیت این کار نسبت به روش‌های معمول چیست؟ اگر اجازه بدهید این را با یک مثال شرح دهم:

در بسیاری از سایت ها، ویژگی ای به نام اسکرول نامحدود وجود دارد. در همین سایت نیز دکمه ای با عنوان بیشتر در انتهای لیستی از مطالب، برای مشاهده‌ی ادامه‌ی لیست قرار گرفته است. سعی کنید پس از فشردن دکمه‌ی بیشتر، داده‌های دریافتی از سرور را مشاهده کنید. پس از انجام این کار مشاهده خواهید کرد که پاسخ سرور HTML خام است. اگر تعداد 10 پست از سرور درخواست شود، 10 بار محتوای HTML تکراری نیز دریافت خواهد شد؛ در صورتی که ساختار HTML یک پست هم کفایت می‌کرد و تنها داده‌ها در آن 10 پست متفاوتند؛ چرا که قالب کار مشخص است و فقط به ازای هر پست باید آن داده‌ها در جای مناسب خود قرار داد.

دیدگاه‌های یک پست هم به خوبی با Angular قابل پیاده سازی است. قالب HTML یک دیدگاه را برای angular تعریف کرده و داده‌های مناسب که احتمالا JSON خام است از سرور دریافت شود. نتیجه‌ی این کار هم صرفه جوی در پهنای باند مصرفی و افزایش فوق العاده‌ی سرعت است، همچنین در صورت نیاز می‌توان داده‌ها و قالب‌ها راکش کرد تا مراجعه به سرور به حداقل برسد.

چگونگی انجام این کار در AngularJs به صورت خلاصه به این صورت است که در angular یک directive به نام ng-repeat تعریف شده است که مانند یک حلقه‌ی foreach برای HTML عمل می‌کند. شما در داخل حلقه، قالب را مشخص می‌کنید و به ازای تعداد داده‌ها، آن حلقه تکرار می‌شود و بر روی داده‌ها پیمایش صورت می‌گیرد.

البته این مثال‌ها فقط دو نمونه از کاربرد این ویژگی در دنیای واقعی بود و مطمئن باشید که در مقالات آینده مثال‌های زیادی از این موضوع را پیاده‌سازی خواهیم کرد.

بهتر است که دیگر خیلی وارد جزئیات نشویم و اولین برنامه‌ی خود را به کمک angularjs بنویسیم. این برنامه، همان برنامه‌ی معروف Hello ,World است؛ اما در این برنامه به جای نوشتن یک Hello, World ساده در صفحه، آن را با ساختار angularjs پیاده‌سازی می‌کنیم.

در داخل ویژوال استادیو یک فایل HTML ساده ایجاد کنید و کدهای زیر را داخل آن بنویسید.

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Sample 1</title>
</head>
<body>
    <div ng-controller="GreetingController">
        <p>{{greeting.text}}, World!</p>
    </div>

    <script src="../Scripts/angular.js"></script>
    <script>
        function GreetingController($scope) {
            $scope.greeting = {
                text: "Hello"
            };
        }
    </script>
</body>
</html>
```

سپس فایل فوق را در مرورگر اجرا کنید. بله؛ عبارت Hello, World را مشاهده خواهید کرد. یک بار دیگر خاصیت text را در $scope.greeting به hi تغییر بدهید و باز هم نتیجه را مشاهده کنید.

این مثال در نگاه اول خیلی ساده است، اما دنیایی از مفاهیم angular را در بر دارد. شما خواص جدیدی را برای عناصر HTML

مشاهده می‌کنید: ng-app، ng-controller، آکلودها و عبارت درون آن و متغیر $scope به عنوان پارامتر.
حال بیایید ویژگی‌ها و مفاهیم جالب کدهای نوشته شده را بررسی کنیم؛ چرا که فرصت برای بررسی ng-app و بقیه‌ی موارد نا آشنا زیاد است:

- هیچ id و یا class برای عناصر html در نظر گرفته نشده تا با استفاده از آنها، رویدادی را برای عناصر مورد نظر مشخص کنیم.

- وقتی در GreetingController مقدار greeting.text را مشخص کرده ایم، باز هم هیچ رویدادی را صدا نزده و یا مشخص نکرده ایم.

- GreetingController یک کلاس ساده‌ی جاوا اسکریپت (POJO) است و از هیچ چیزی که توسط angular فراهم شده باشد، ارث بری نکرده است.

- اگر به متد سازنده‌ی کلاس GreetingController دقت کنید، متغیر $scope به عنوان پارامتر تعریف شده است. نکته‌ی جالب این است که ما هیچ گاه به صورت دستی سازنده‌ی کلاس GreetingController را صدا نزده ایم و حتی درون سازنده هم $scope را ایجاد نکرده ایم؛ پس چگونه توانسته ایم خاصیتی را به آن نسبت داده و برنامه به خوبی کار کند. بهتر است برای پاسخ به این سوال خودتان دست به کار شوید؛ ابتدا نام متغیر $scope را به نام دلخواه دیگری تغییر دهید و سپس برنامه را اجرا کنید. بله برنامه دیگر کار نمی‌کند. دلیل آن چیست؟ همان طور که گفتم Angular دارای یک سیستم تزریق وابستگی توکار است و در اینجا نیز $scope به عنوان وابستگی در سازنده‌ی این کلاس مشخص شده است تا نمونه‌ی مناسب آن توسط angular به کلاس GreetingController ما تزریق شود؛ اما چرا به نام آن یعنی $scope حساس است؟ به این دلیل که زبان جاوا اسکریپت یک زبان پویا است و نوع در آن مطرح نیست؛ angular مجبور است که از نام پارامترها برای تزریق وابستگی استفاده می‌کند. در مقالات آینده چگونگی عملکرد سیستم تزریق وابستگی angular را به تشریح بیان می‌کنم.

- همچنین همان طور که در مورد قبلی نیز به آن اشاره کردم، ما هیچ گاه خود دستی سازنده‌ی GreetingController را صدا نزدیم و جایی نیز نحوه‌ی صدا زدن آن را مشخص نکرده ایم.

تا همین جا فکر کنم کاملا برای شما مشخص شده است که ساختار فریمورک Angular با تمامی کتاب خانه‌های مشابه متفاوت است و با ساختاری کاملا اصولی و حساب شده طرف هستیم. همچنین در مقالات آینده توجه شما را به قابلیت‌هایی بسیار قدرتمندتر جلب خواهم کرد.

MVVM ، MVP، MVC و یا MVW

در بخش اول این مقاله، الگوی طراحی پیشنهادی فریمورک Angular را MVC بیان کرده‌ام؛ اما همان طور که گفته بودم AngularJS از انقیاد داده دوطرفه (Two Way Data Binding) نیز به خوبی پشتیبانی می‌کند و به همین دلیل عده ای آن را یک MVVM Framework تلقی می‌کنند. حتی داستان به همین جا ختم نمی‌شود و عده ای آن را به چشم MVP Framework نیز نگاه می‌کنند. در ابتدا سایت رسمی AngularJS الگوی طراحی مورد استفاده را MVC بیان می‌نمود ولی در این چند وقت اخیر عنوانش را به MVW Framework تغییر داده است.
MVW مخفف عبارت Model View Whatever هست و کاملا مفهومش مشخص است. Model و View بخش‌های مشترک تمام الگوها بودند و تنها بخش سوم مورد اختلاف توسعه دهندگان بود؛ در نتیجه انتخاب آن را بر عهده‌ی استفاده کننده قرار داده اند و تمام امکانات لازم برای پیاده‌سازی این الگوهای طراحی را فراهم کرده اند. در طی این مقالات صرف نظر از تمام الگوهای طراحی فوق، من بیشتر بر روی MVC تمرکز خواهم کرد.
الگوی طراحی MVC در سال 1970 به عنوان بخشی از زبان برنامه نویسی Smalltalk معرفی شد و از همان ابتدا به سرعت محبوبیت زیادی در بین محیط‌های توسعه‌ی دسکتاپی از قبیل C++ و Java که رابط کاربری گرافیکی به نوعی در آن‌ها دخیل است، پیدا کرد.
تفکر MVC این را بیان می‌کند که باید جداسازی واضح و روشنی بین مدیریت داده‌ها (Model)، منطق برنامه (Controller) و نمایش داده‌ها به کاربر (View) وجود داشته باشد و در اصل هدفش جداسازی اجزای رابط کاربری به بخش هایی مجزا است.
شاید این سوال برای شما پیش بیاید که چرا باید چنین الگویی را در برنامه‌ها پیاده کرد؟
احتمالا تا کنون از بین برنامه هایی که نوشته اید، رابط کاربری بیشتر از آن‌ها را نیز خودتان مجبور شده اید طراحی کنید؛ به این دلیل که برنامه‌ی شما بدون رابط کاربری قابل اجرا شدن نبوده است. اجرای برنامه‌ی شما منوط به وجود تعدادی دکمه و textbox

و ... بوده است و به قولی منطق برنامه به رابط گرافیکی گره خورده بوده است. پس می‌توان گفت که پیاده‌سازی الگوی طراحی وقتی ضرورت پیدا می‌کند که رابط گرافیکی، قسمتی از برنامه‌ی شما را تشکیل دهد.
آیا با وجود زبان‌های طراحی ساده ای مثل HTML و XAML و ... احتیاجی است که برنامه نویس وقت خود را صرف طراحی رابط کاربری کند؟ مسلما خیر، چون دیگر با این امکانات یک طراح هم از پس این کار به خوبی و یا حتی بهتر بر می‌آید. دیگر وظیفه‌ی برنامه نویس نوشتن کدهای مربوط به منطق برنامه است. کدهایی که بدون UI هم قابل تست شدن باشد و به راحتی بتوان برای آن‌ها آزمون‌های واحد نوشت. برنامه نویس باید این را در نظر بگیرد که UI وجود ندارد و حتی ممکن است هیچ گاه هم ایجاد نشود و این کدها تبدیل به یک کتابخانه شود و مورد استفاده قرار بگیرد تا در یک برنامه با رابط کاربری گرافیکی.
در MVC، روال عمومی کار به این شکل است که View داده‌ها را از Model دریافت می‌کند و به کاربر نمایش می‌دهد. وقتی که کاربر با کلیک کردن و تایپ کردن با برنامه ارتباط برقرار می‌نماید، Controller به این درخواست‌ها پاسخ می‌دهد و داده‌های موجود در Model را به روز رسانی می‌کند. در نهایت هم Model تغییرات خود را به View منعکس می‌کند تا View آن چه را که پیش از آن نمایش می‌داده است، تغییر دهد و View را از تغییرات رخ داده آگاه نماید.
اما در برنامه‌های Angular قضیه از چه قرار است؟ در Angular، قالب HTML یا اگر بخواهم دقیق‌تر بگویم (Document Object Model(DOM معادل View است؛ کلاس‌های جاوا اسکریپتی نقش Controller را دارند؛ و خواص اشیای جاوا اسکریپتی و یا حتی خود اشیا نقش Model را بر عهده دارند.

ساختار بخشیدن به برنامه با استفاده MVC یک مزیت مهم دیگر نیز دارد: ساختار کار کاملا مشخص است و هر کسی نمی‌تواند به صورت سلیقه ای آن را پیاده سازی کند. با یک مثال این موضوع را تشریح می‌کنم: اگر کسی پروژه‌ی بنده را که با ASP.NET MVC نوشتم، بررسی کند، اصلا احساس غریبی نمی‌کند و به راحتی می‌تواند آن را توسعه دهد. دلیل این موضوع این است که ASP.NET MVC یک ساختار مشخص را به توسعه دهندگان اجبار کرده است و هر کسی این ساختار را رعایت کند و با آن آشنا باشد، به راحتی می‌تواند با آن کار کند. توسعه دهنده می‌داندکه من Model را کجا تعریف کرده ام، Controller مربوط به هر View کجاست و در کدام قسمت با پایگاه داده ارتباط برقرار کرده‌ام؛ اما در مورد کدهای JavaScript و سمت کلاینت چه طور؟ توسعه دهنده ای که می‌خواهد کار من را ادامه بدهد دچار وحشت می‌شود! الگوی مشخصی وجود ندارد؛ معلوم نیست که کجا DOM را دستکاری کرده‌ام، در کدام قسمت با سرور ارتباط برقرار شده و... به قول معروف با یک اسپاگتی کد تمام عیار طرف می‌شود. AngularJS این مشکل را حل نموده و ساختار خاصی را سعی کرده به شما دیکته کند و تا حد ممکن دست شما را نیز باز گذاشته است. جدا از همه‌ی اینها، برنامه‌های مبتنی بر Angular به راحتی نگه داری و تست می‌شوند و بدون هیچ دغدغه ای آن‌ها را می‌توان توسعه داد.

در حاشیه

شاید در هنگام دریافت فایل angularjs و افزودن آن به پروژه‌ی خود شروع به اعتراض کرده اید که نسخه‌ی فشرده شده‌ی آن 87 کیلو بایت حجم دارد در صورتی که این حجم در کتابخانه‌های مشابه ممکن است حتی به 10 کیلوبایت هم نرسد. اگر دقت کرده باشید من در بیان AngularJS از واژه‌ی کتاب خانه استفاده نکردم و فقط از واژه‌ی فریمورک استفاده کردم. بله نمی‌شود angular را با کتاب خانه هایی مقایسه کرد که مهمترین ویژگی خود را Data Binding می‌دانند. AngularJS یک بستر کاری قدرتمند است که تمام راه حل‌های موجود را در خود جمع کرده است. تیم توسعه دهنده‌ی آن هم هیچ ادعایی ندارد و می‌گویند که ما هیچ چیزی را خودمان اختراع نکرده ایم، بلکه راه حل‌های عالی را برگزیدیم، تفکرهای خوب را ارتقا بخشیده و در فریمورک خود استفاده کردیم و حتی از ایده‌های خوب دیگر کتاب خانه‌ها هم استفاده کرده ایم. بنابر این نباید به حجم آن در مقابل توانایی هایی که دارد اعتراض کرد.
همچنین به نظر می‌آید که AngularJS یک فریمورک پیچیده است. ولی من همیشه بین پیچیده و پیچیده شده تفاوت قائل می‌شوم. به نظر شخصی خودم Angular به دلیل مشکلات خاص و پیچیده ای که حل می‌کند پیچیده است و پیچیده شده نیست. اگر آن را پیچیده شده حس می‌کنید، تنها دلیلش، نحوه‌ی آموزش دادن بنده است، تمام سعی خود را می‌کنم که مفاهیم را تا حد ممکن ساده بیان کنم و امیدوارم در آینده که با مثال‌های بیشتری روبرو می‌شوید، این مفاهیم به کارتان بیاید.
در مقاله‌ی بعدی به مفاهیم انقیاد داده، تزریق وابستگی، هدایت گرها (Directives) و سرویس‌ها در AngularJS می‌پردازم.
دریافت مثال این قسمت

نظرات خوانندگان

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۰۶/۰۹ ۲۳:۵۲

سلام؛ بحث امنیت مخصوصا در طراحی چی میشه؟ در این روش تمام تمپلیت‌ها و طراحی‌ها در سمت کاربر میشه. ولی در روش ASP.net مثلا کدهای Razor برای کاربر فرستاده نمی‌شه. فقط کد HTML ساخته شده فرستاده میشه و سخت‌تر میشه فهمید طراح دقیقا چه کرده.
تشکر

نویسنده: چارلی
تاریخ: ۱۳۹۲/۰۶/۱۰ ۷:۵۷

مشکل امنیت برای چی؟ شما از سمت سرور اطلاعات مورد نیاز دریافت میکنید بدون نیاز به اینکه چجوری تولید شدند مثل Razor و با این فریمورک باهاش کار میکنید قرار نیست منطق برنامه بیارید سمت کلاینت که اگه این روش دیدید مطئمن باشید یه جای کار مشکل داره!

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۰۶/۱۰ ۱۰:۴۹

مگر در این روش در سمت کلاینت تمپلیت‌ها را تعریف نمی‌کنید؟
تمپلیت‌ها و کنترلرها و توابع نیز در فایل JS هست که ان هم به سمت کاربر ارسال میشه.خب حالا کاربر صفحه را ذخیره می‌کنه فایل‌های JS را هم داره. فقط تنها چیزی را که نداره اطلاعات بانک هست که انرا هم از طریق Ajax ی که صادر میشه حدودا میشه فهمید قضیه چیه. فکر کنم به راحتی میشه یه کپی از روی سایت تهیه کرد.

نویسنده: احمد
تاریخ: ۱۳۹۲/۰۶/۱۰ ۱۲:۳۱

-1 منطق برنامه در Razor نوشته نمیشود ، بلکه در کلاسها و توابع پیاده سازی شده است.
-2سایت بدون data به درد کسی نمیخورد.(مثلا جی میل که به کدهای سمت کلاینت(html,js,...) دسترسی وجود دارد)
-3بهتر است بار تولید UI سمت کلاینت باشد تا سرور.در سمت سرور باید فقط data رد و بدل شود.

نویسنده: سعید پیروز
تاریخ: ۱۳۹۲/۰۶/۱۰ ۱۲:۵۴

سلام؛ اگه می‌شه در مورد فعال کردن intellisense برای angular در vs2012 هم یک توضیحی بدید.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۱۰ ۱۳:۴

اینجا

پلاگینی برای resharper

visual studio 2013 به صورت پیش فرض از angular پشتیبانی می‌کند. در ضمن به آن صورت هم فکر نکنم احتیاجی به intellisense باشد. من به شخصه بدون intellisense به راحتی ازش استفاده می‌کنم.

نویسنده: محسن خان

تاریخ: ۱۳۹۲/۰۶/۱۰ ۲۲:۱۶

در مورد ترکیب Client Side Templates با MVC: یکی از خوبی‌های بازگشت دادن یک partial view کامل در MVC (که بله، یک HTML کامل رو بر می‌گردونه در حالت Ajax ایی مثلا) نسبت به این روش، امکان استفاده از متدهای کمکی سمت سرور برای رندر کردن View هست. مثلا فرض کنید یک لیست فایل‌ها قراره نمایش داده بشه. در View یا Partial View میشه بدون تعریف یک کلاس اضافه‌تر برای بازگشت دادن اطلاعات به صورت JSON که بخواد در AngularJS سمت کلاینت استفاده بشه، اطلاعات رو خیلی ساده برای نمایش، با razor و سی‌شارپ فرمت کرد. مثلا تاریخ رو شمسی کرد. اندازه رو به کیلوبایت یا مگابایت نمایش داد (در حد فراخوانی یک متد الحاقی). یک if و else گذاشت که اگر کاربر لاگین بود این قسمت از partial view رو که درون حلقه داره تولید میشه، مشاهده نکنه یا برعکس. یک قسمت از حلقه هم یک فرم کوچک درست کرد برای ارسال دیتا به سرور اون هم فرمی که آدرسش رو از T4MVC به صورت strongly typed می‌گیره و یا فیلدهاش از Html Helperهای MVC استفاده می‌کنند که این‌ها هم سمت سرور رندر می‌شن. الان چون تمام کار با جاوا اسکریپت باید انجام بشه، یعنی تمام این مراحل رو باید به صورت JSON بازگشت داد که AngularJS بخواد اون‌ها رو سمت کلاینت، سر هم کنه. به علاوه امکان کامپایل کردن Viewهای razor و یافتن خطاهای احتمالی رو هم از دست می‌دیم چون همه چیز قراره سمت کلاینت رندر بشه.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۱۱ ۸:۵۱

احساس می‌کنم، کمی از صحبت‌های من اشتباه برداشت شده است. قالب باید HTML باشد، اما مهم نیست که این قالب توسط چه کسی تولید شده است. برای مثال من در پروژه‌ی خودم یک کنترلر تعریف کرده ام که درآن همه‌ی actionها فقط partialview بر می‌گردانند. حال قبل از اینکه این فایل‌های cshtml تبدیل به html شوند و به کلاینت برگردانده شوند، من با razor عملیات دلخواه خود را انجام می‌دهم.
برای اینکه تاریخ‌ها را شمسی کرده و از این قبیل چیدمان داده ها، قبل از اینکه تبدیل به json شده و به کلاینت پاس داده شوند، باید در یک حلقه داده‌های مورد نظر را به فرمت مورد نظر درآورده و در نهایت تبدیل به json کرده و به کلاینت بگرداند.

برای حل مشکل T4MVC می توان در همان ابتدای کار تمام آدرس‌های مورد نظر را در یک شی جاوا اسکریپت global تعریف کرد و در سراسر برنامه از آن استفاده کرد.

شاید دلایل شما برای این مثال کوچک منطقی به نظر آید، اما هدف angular حل مشکلات برنامه‌های بزرگ و حرفه ای است.
برای مثال نوشتن یک filemanager با استفاده از angular فوق العاده لذت بخش است و به راحتی می‌توان یک فایل منیجر حرفه ای را با آن نوشت. برای آنهم برنامه دارم ، اما اگر وقت شود...

نویسنده: سالار
تاریخ: ۱۳۹۲/۰۶/۱۱ ۹:۵۹

با سلام. اگر partialview ما نیاز به اعتبارسنجی داشته باشد، ما در سمت سرور می‌توانستیم یک سری attribute برای اعتبارسنجی هر خاصیت درون ویومدل‌های خود ایجاد کنیم، و خود razor این attributeها را درون html نهایی رندر می‌کرد. آیا این امکان در سمت کلاینت برای انگولار نیز وجود دارد؟

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۱۱ ۱۰:۵۰

خود angularjs دارای یک سری api قدرتمند برای اعتبار سنجی هستند. البته انتظار helperهای شسته رفته asp.net mvc را نداشته باشید که بر اساس model بتواند اعتبار سنجی سمت کلاینت کند.
باید برایش خودتان دستی کد بنویسید. البته فکر نکنم تهیه‌ی helper برای angular که بر اساس model، کدهای متناظر اعتبار سنجی را تولید کند کار مشکلی باشد.

نویسنده: وحید م
تاریخ: ۱۳۹۲/۰۹/۲۹ ۲۳:۶

با سلام سوالی داشتم در بالا فرمودید: "من در پروژه خودم یک کنترلر تعریف کردم که در آن همه‌ی actionها فقط partial view برمیگرداند" یعنی چی؟
"حال قبل از اینکه این فایل‌های cshtml تبدیل به html شوند وبه کلاینت برگردانده شوند" این جمله هم نامفهوم بود. ممنون میشم اگر توضیح بیشتری بدید.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۳۰ ۰:۱۷

مطلب بارگذاری PartialView با استفاده از jQuery در زمان اجرا را مطالعه کنید.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۳۰ ۱۲:۷

راه دیگه برای بارگزاری صفحات تعریف مسیر یاب است فرض کنید مسیر زیر را تعریف کرده ایم :

```
var PostApp = angular.module('PostApp', []).config(['$routeProvider',
  function ($routeProvider) {
      $routeProvider.
          when('/list', {
              templateUrl: '/Administrator/Post/Index',
              controller: 'PostController'
          });
  }]);
```

در قسمت templateUrl مسیر یک اکشن Index در کنترلری به نام Post است که یک partial view بر میگرداند به شکل زیر :

```
public virtual ActionResult Index(int? id)
{
      var model = new PostViewModels
      {
          //......
      };
      return PartialView(viewName: "_Index", model: model);
}
```

در این اکشن ما مدل را به partial view ارسال میکنیم و ویو توسط razor رندر میشود و نتیجه که یک فایل html است بازگشت داده میشود و ما میتوانیم داخل این html از امکانات Angular استفاده کنیم یعنی:
" قبل از اینکه این فایل‌های cshtml تبدیل به html شوند و به کلاینت برگردانده شوند، من با razor عملیات دلخواه خود را انجام می‌دهم. "

```
@using ViewModels.Administrator.Post

//استفاده از امکانات Razor
@(Html.EnumDropDownListMenu<PostPermition, AppViewPostResource>("permition-", "{{item.id}}"))

//استفاده از امکانات Angular
<div  ng-controller="PostController">
     <ul>
          <li ng-repeat="item in ListOfItems">
              {{item.Title}}
           </li>
    </ul>
</div>
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۳۰ ۱۳:۸

ListOfItems در مدل MVC قابل استفاده است در AngularJS؟

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۳۰ ۱۳:۲۸

ما داخل صفحه‌ی Partial View میتونیم از امکانات Angular برای زمان بازگشت به سمت کلاینت استفاده کنیم. ListOfItem مربوط به زمانی میشود که صفحه‌ی رندر شده و در اختیار کلاینت قرار گرفته است. و آماده استفاده از داده‌های در اختیار قرار داده شده توسط متغیری آرایه ای به نام ListOfItem درکنترلر موجود در Angular است .یعنی صفحه رندر شده میشود یه چیزی شبیه به این :

```
//استفاده از امکانات Razor
لیست مطالب

//استفاده از امکانات Angular
<div  ng-controller="PostController">
     <ul>
          <li ng-repeat="item in ListOfItems">
                {{item.Title}}
           </li>
    </ul>
</div>
```

وحالا که یک صفحه‌ی HTML خام شده است میتوانید از آن استفاده کنید. و این هم کنترلری که این صفحه را مدیریت میکندبرای مثال :

```
PostApp.controller('PostController', function ($scope, $http, postServices) {
       //...
      $scope.ListOfItems =
                    postServices.GetPosts(post)
                        .success(function (data) {
                            $scope.ListOfItems = data;
                        });
       //...
}
```