عنوان: AngularJS #1
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۶:۴۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

پیش از اینکه آموزش AngularJs را شروع کنیم بهتر است با مفهوم برنامه‌های تک صفحه ای وب و یا Single Page Web Applications آشنا شویم؛ چرا که AngularJS برای توسعه هر چه ساده‌تر و قوی‌تر این گونه برنامه‌ها متولد شده است.

Single Page Application

برای درک چگونگی کارکرد این برنامه ها، مثالی را میزنیم که هر روزه با آن سرو کار دارید، یکی از نمونه‌های کامل و قدرتمند برنامه‌های Single Page Application و یا به اختصار SPA، سرویس پست الکترونیکی Google و یا همان Gmail است.
اجازه بدهید تا ویژگی‌های SPA را با بررسی Gmail انجام دهم، تا به درک روشنی از آن برسید:

Reload نشدن صفحات

در Gmail هیچ گاه صفحه Reload و یا اصطلاحا بارگیری مجدد نمی‌شود. برای مثال، وقتی شما لیست ایمیل‌های خود را مشاهده میکنید و سپس بر روی یکی از آن‌ها کلیک میکنید، بدون اینکه به صفحه ای دیگر هدایت شوید؛ایمیل مورد نظر را میبینید. در حقیقت تمامی اطلاعات در همان صفحه نمایش داده می‌شوند و بر عکس وب سایت‌های معمول است که از صفحه ای به صفحه‌ی دیگر هدایت میشوید ، در یک صفحه تمام کارهای مورد نیاز خود را انجام می‌دهید و احتیاجی به بارگیری مجدد صفحات نیست. با توجه به این صحبت‌ها برای توسعه دهنده‌های وب آشکار است که تکنیک AJAX، نقشی اساسی در این گونه برنامه‌ها دارد، چون کلیه‌ی عملیات ارتباط با سرور در پشت زمینه انجام می‌شوند.

تغییر URL در نوار آدرس مرورگر

وقتی شما بر روی یک ایمیل کلیک می‌کنید و آن ایمیل را بدون Reload شدن مجدد صفحه مشاهده می‌کنید، آدرس صفحه در مرورگر نیز تغییر می‌کند. خب مزیت این ویژگی چیست؟ مزیت این ویژگی در این است که هر ایمیل شما دارای یک آدرس منحصر به فرد است و به شما امکان Bookmark کردن آن لینک، باز کردن آن در یک Tab جدید و یا حتی ارسال آن به دوستان خود را دارید. حتی اگر این مطلب را جدا از Gmail در نظر بگیریم، به موتورهای جست و جو کمک می‌کند، تا هر صفحه را جداگانه Index کنند؛ جدا از اینکه وبسایت ما SPA است. همچنین این کار یک مزیت مهم دیگر نیز دارد؛ و آن کار کردن کلیدهای back و forward مرورگر، برای بازگشت به صفحات پیمایش شده قبلی است.
شاید قبل از بیان این ویژگی با خود گفته باشید که پیاده سازی Reload نشدن صفحات با AJAX آن چنان کار پیچیده ای نیست. بله درست است، اما آیا شما قبل از این راه حلی برای تغییر URL اندیشیده بودید؟ مطمئنا شما هم صفحات وب زیادی را دیده اید که همه‌ی صفحات آن دارای یک URL در نوار آدرس مرورگر هستند و هیچگاه تغییر نمی‌کنند و با باز کردن یک لینک در یک Tab جدید، باز همان صفحه‌ی تکراری را مشاهده می‌کنند! و یا بدتر از همه که دکمه‌ی back مرورگر غیر عادی عمل می‌کند. بله، اینها تنها تعدادی از صدها مشکلات رایج سیستم‌های نوشته شده ای است که سعی کردند همه‌ی کارها در یک صفحه انجام شود.

Cache شدن اطلاعات دریافتی

شاید خیلی‌ها ویژگی‌های فوق را برای یک SPA کافی بدانند، اما تعدادی هم مانند نگارنده وجود یک کمبود را حس می‌کنند و آن کش شدن اطلاعات دریافتی در مرورگر است. Gmail این امکان را به خوبی پیاده سازی کرده است. لیست ایمیل‌های دریافتی در بار اول از سرور دریافت می‌شود، سپس شما بر روی یک ایمیل کلیک و آن را مشاهده می‌کنید. حال به لیست ایمیل‌های دریافتی بازگردید، آیا رفت و برگشتی به سرور انجام می‌شود؟ مسلما خیر. حتی اگر دوباره بر روی آن ایمیل مشاهده شده ، کلیک کنید، بدون رفت و برگشتی به سرور آن ایمیل را مشاهده می‌کنید.
کش شدن اطلاعات سبب می‌شود که بار سرور خیلی کاهش یابد و رفت و آمدهای بیهوده صورت نگیرد. کش شدن داده‌ها یک مزیت دیگر نیز دارد و آن تبدیل برنامه‌های معمول وب stateless به برنامه‌های شبه دسکتاپ state full است.
تکنیک AJAX در پیاده سازی امکانات فوق نقشی اساسی را بازی می‌کند. کمی به عقب برمیگردیم یعنی زمانی که AJAX برای اولین بار مطرح شد و هدف اصلی به وجود آمدن آن پیاده سازی برنامه‌های وب به شکل دسکتاپ بود و این کار از طریق انجام تمامی ارتباطات سرور با XMLHttpRequest امکان پذیر می‌شد. شاید آن زمان با توجه به محدودیت تکنولوژی‌ها موجود این کار به صورت

تمام و کمال امکان پذیر نبود، اما امروزه به بهترین شکل ممکن قابل پیاده سازی است.

شاید اکنون این سوال پیش بیاید که چرا باید وبسایت خود را به شکل SPA طراحی کنیم؟

برای پاسخ دادن به این سوال باید گفت که سیستم‌های وب امروزی به دو دسته‌ی زیر تقسیم می‌شوند:
- Web Documents و یا همان وب سایت‌های معمول
- Web Applications و یا همان Single Page Web Applications
اگر هدف شما طراحی یک وب سایت معمول است که هدف آن، نمایش یک سری اطلاعات است و به قولی دارای محتواست، مطمئنا پیاده سازی این سیستم به صورت SPA کاری بیهوده به نظر می‌آید؛ ولی اگر هدفتان نوشتن سیستم هایی مثل Gmail، Facebook، Azure، Google Maps و ... است، پیاده سازی آن‌ها به صورت وب سایت‌های معمولی، غیر معقول به نظر می‌آید. حتی بخش‌های مدیریتی یک وبسایت هم می‌تواند به خوبی توسط SPA پیاده سازی شود، چرا که واقعا برای مدیریت اطلاعات یک وب سایت احتیاجی نیست، که از این صفحه به آن صفحه جا به جا شد.

معرفی کتابخانه‌ی AngularJS

AngularJS فریم ورکی متن باز و نوشته شده به زبان جاوا اسکریپت است. هدف از به وجود آمدن این فریم ورک، توسعه هر چه ساده‌تر SPAها با الگوی طراحی MVC و تست پذیری هر چه آسان‌تر آن‌ها است. این فریم ورک توسط یکی از محققان Google در سال 2009 به وجود آمد. بعدها این فریم ورک تحت مجوز MIT به صورت متن باز در آمد و اکنون گوگل آن را حمایت می‌کند و توسط هزاران توسعه دهنده در سرتاسر دنیا، توسعه داده می‌شود.
قبل از اینکه به بررسی ویژگی‌های Angular بپردازم، بهتر است ابتدا مطلبی درباره‌ی به کارگیری Angular از Brad Green که کارمند گوگل است، بیان کنم.
در سال 2009 تیمی در گوگل مشغول انجام پروژه ای به نام Google Feedback بودند. آن‌ها سعی داشتند تا در طی چند ماه، به سرعت کدهای خوب و تست پذیر بنویسند. پس از 6 ماه کدنویسی، نتیجه‌ی کار 17000 خط کد شد. در آن موقع یکی از اعضای تیم به نام Misko Hevery، ادعا کرد که می‌تواند کل این پروژه را در دو هفته به کمک کتابخانه‌ی متن بازی که در اوقات فراغت توسعه داده است، بازنویسی کند. Misko نتوانست در دو هفته این کار را انجام دهد. اما پس از سه هفته همه‌ی اعضای تیم را شگفت زده کرد. نتیجه‌ی کار تنها 1500 خط بود! همین باعث شد که ما بفهمیم که، Misko بر روی چیزی کاری میکند که ارزش دنبال کردن دارد.
پس از آن قضیه Misko و Brad بر روی Angular کار کردند و اکنون هم Angular توسط تیمی در گوگل و هزاران توسعه دهنده‌ی متن باز حرفه ای در سرتاسر جهان، درحال توسعه است.
فکر کنم همین داستان ذکر شده، قدرت فوق العاده زیاد این فریم ورک را برای همگان آشکار سازد.

ویژگی‌های AngularJS:

- **قالب‌های سمت کاربر (Client Side Templates):** انگولار دارای یک template engine قدرتمند برای تعریف قالب است.

- **پیروی از الگوی طراحی MVC:** انگولار، الگوی طراحی MVC را برای توسعه پیشنهاد می‌دهد و امکانات زیادی برای توسعه هر چه راحت‌تر با این الگو فراهم کرده است.

- **Data Binding:** امکان تعریف انقیاد داده دوطرفه (Two-Way Data Binding) در این فریم ورک به راحتی هرچه تمام، امکان پذیر است.

- **Dependency Injection:** این فریم ورک برای دریافت وابستگی‌های تعریف شده، دارای یک سیستم تزریق وابستگی توکار است.

- **تعریف Serviceهای سفارشی:** در این فریم ورک امکان تعریف سرویس‌های دلخواه به صورت ماژول وجود دارد. این ماژول‌های مجزا را به کمک سیستم تزریق وابستگی توکار Angular، به راحتی در هر جای برنامه می‌توان تزریق کرد.

- **تعریف Directiveهای سفارشی:** یکی از جذاب‌ترین و قدرتمندترین امکانات این فریم ورک، تعریف Directiveهای سفارشی

است. Directive ها، امکان توسعه HTML را فراهم کرده اند. توسعه‌ی HTML اکنون در قالب Web Componentsها فراهم شده است، اما هنوز هم خیلی از مرورگرهای جدید نیز از آن پشتیبانی نمی‌کنند.

- **فرمت کردن اطلاعات با استفاده از فیلترهای سفارشی:** با استفاده از فیلترها میتوانید چگونگی الحاق شدن اطلاعات را برای نمایش به کاربر تایین کنید ؛ انگولار همراه با فیلترهای گوناگون مختلفی عرضه میشود که میتوان برایه مثال به فیلتر ، currency uppercase، date کردن رشته‌ها و .... اشاره کرد همچنین شما محدود به فیلترهای تعریف شده در انگولار نیستید و آزادید که فیلترهای سفارشی خودتان را نیز تعریف کنید.

- **سیستم Routing:** دارا بودن سیستم Routing قدرتمند، توسعه SPAها را بسیار ساده کرده است.

- **سیستم اعتبار سنجی:** Angular دارای سیستم اعتبار سنجی توکار قدرتمند برای بررسی داده‌های ورودی است.

- **سرویس تو کار برای ارتباط با سرور:** Angular دارای سرویس پیش فرض ارتباط با سرور به صورت AJAX است.

- **تست پذیری:** Angular دارای بستری آماده برای تست کردن برنامه‌های نوشته شده است و از Unit Tests و Integrated End-to-End Test هم پشتیبانی می‌کند.

- **جامعه‌ی متن باز بسیار قوی**

این‌ها فقط یک مرور کلی بر توانایی‌های این فریم ورک بود و در ادامه هر کدام از این ویژگی را به صورت دقیق بررسی خواهیم کرد.

در مقاله‌ی بعدی، به چگونگی نصب AngularJS خواهیم پرداخت. سپس، اولین کد خود را با استفاده از آن خواهیم نوشت و مطالب Client Side Templates و MVC را دقیق‌تر بررسی خواهیم کرد.

نظرات خوانندگان

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۹:۲۷

سلام
آیا استفاده از این فریمورک باعث این می‌شود که دیگر نیازی به JQuery نباشد؟ و یا باز کمبود هایی دارد که به استفاده از JQuery نیاز باشد؟
تشکر

نویسنده: سالار
تاریخ: ۱۳۹۲/۰۶/۰۶ ۲۳:۱۹

به شدت دنبال آموزش angularjs بودم. سپاس و چند سوال.
- آیا با استفاده از angularjs برای یک SPA دیگر نیازی به asp.net mvc خواهد بود؟
-کامپایلر ویژوال استودیو به خوبی برای razor کار میکند و بسیار intellisence قوی دارد و امکانات Strongly typed نیز دارد. آیا این امکانات برای angular نیز موجود است تا خطایابی راحت‌تر شود؟
- من یک اپلیکیشن دارم که کاربر هربار یکی از آیتم‌های منو را انتخاب میکند و برای آن یک ویو بصورت partialview درون یک تب لود میشود. هر ویو ، فایل جاوااسکریپت مخصوص به خود دارد. کاربر می‌تواند چندین ویوی مجزا را درون تب‌ها باز کند و بین آنها جابه جا شود. اما مدیریت دستی آنها بسیار سخت شده است. آیا انگولار امکان نمایش چند ویو را بطور همزمان و جابه جا شدن بین آنها را میدهد.(بدون بروز تداخل بین فایل‌های جاوا اسکریپت مربوط به هر ویو).
با تشکر.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۰۷ ۸:۰

سلام؛
کتابخانه‌ی jQuery عموما برای دو کار استفاده می‌شود:
1- دستکاری یا پردازش DOM
2- ارتباط با سرور به صورت AJAX
انگولار سرویسی برای ارتباط AJAX دارد که شما را کاملا از jQuery بی نیاز می‌کند.
برای پردازش DOM و الصاق رویداد به عناصر صفحه، انگولار دارای APIهای خیلی خوبی است که می‌تواند تا حدی جایگزین jQuery باشد. اما فراموش نکنیم که jQuery علاوه بر پردازش DOM دارای کلی API متنوع است که انگولار به پای آن نمی‌رسد. دلیل هم مشخص است؛ چرا که هر دو کتابخانه برای کاری متفاوت نوشته شده اند و مقایسه آن‌ها اصلا کار درستی نیست. همچنین به این نکته هم توجه داشته باشید که برای jQuery هزاران افزونه‌ی مختلف نوشته اند که اجرای آن ها، وابسته به jQuery است. پس هیچ کتابخانه ای در این زمینه موازی کاری نمی‌کند، چون توانایی رقابت ندارد.(البته تعدادی را هم با کاملارا انگولار بازنویسی کرده اند.)
انگولار در مورد دستکاری DOM هم یک مورد را شما گوشزد می‌کند: اگر می‌خواهید DOM را دستکاری کنید، این کار را باید از طریق Directiveها انجام دهید. منظور این است کدهای مورد نظرتان را با هر کتابخانه ای که می‌خواهید بنویسید، ولی آن‌ها را درون یک Directive سفارشی بنویسید و روی هوا کد ننویسید تا قابل تست کردن باشد.
در ادامه کاملا متوجه منظورم خواهید شد.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۰۷ ۸:۳۱

- AngularJS یک فریم ورک سمت کلاینتی به زبان جاوا اسکریپت است و ASP.NET MVC یک فریم ورک سمت سرور و این دو نمی‌توانند جای یکدیگر را پر کنند و ربطی به هم ندارند.

- در اینجا Viewهای شما یک HTML ساده است و مفاهیم Strongly Typed View هیچ معنی خاصی ندارد. Model در انگولار یک

کلاس جاوا اسکریپتی ساده (POJO) است. view هم یک فایل HTML که قالب کار را مشخص می‌کند و Controller هم باز یک کلاس ساده‌ی جاوا اسکریپت است.

- اتقاقا یک بار همچنین چیزی را از من خواستند (بدون انگولار) و من انجامش دادم. البته این طراحی اشتباه است. آیا شما ده بخش را در چند TAB مختلف باز می‌کنید، آدرس منحصر به فردی هم در نوار آدرس مرورگر برای آن‌ها در نظر گرفته اید؟
این کار در آن زمان که همه‌ی برنامه با یک URL کار می‌کرد خیلی عالی بود، ولی اکنون که شما می‌توانید برای View یک آدرس منحصر به فرد داشته باشید و آن‌ها را Tab خود مرورگر مدریت کنید، احتیاجی به آن کارها نیست.
به هر حال، در انگولار این جمله بی معنی است که؛ بین فایل‌های جاوا اسکریپت تداخل ایجاد شود!
در انگولار هر View دارای Controller متناظر خود است. اعمال تجاری مشترک هم در سرویس‌ها تعریف می‌شوند و در کنترلرها تزریق می‌شوند. پس کد جاوا اسکریپتی روی هوا نوشته نمی‌شود(حتی یک خط)

صبر کنید تا کمی پیش برویم. این فقط مقدمه بود...

نویسنده: سالار
تاریخ: ۱۳۹۲/۰۶/۰۷ ۹:۵۲

سپاس فراوان.
آیا در یک وب اپلیکیشن و نه وب سایت عمومی ، واقعاً نیازی به داشتن یک url مجزا برای هر ویو داریم. مگر هدف angular ایجاد برنامه هایی شبیه برنامه‌های دسکتاپ نیست. در برنامه‌های دسکتاپ که هر ویو آدرس مجزایی ندارد. با تشکر.

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۰:۲۱

سلام
تشکر از جواب خوبتان
من متوجه شدم که دانسته‌های شما نسبت به فریمورک‌های جاوااسکریپت بسیار خوب است. پس اگر لطف کنید درباره چند تا از فریم ورک‌های دیگر بخصوص Backbone اطلاعاتی بدهید که هر کتابخانه در اصل برای چه کاری ساخته شده است.
منظورم این است که یه مقایسه بین فریم ورک‌های معروف که بیشتر برای چه کاری تهیه شده اند.
اگر چنین مقایسه ای به صورت فارسی هست معرفی کنید.
تشکر

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۰:۲۶

اغلب کارمندان یک سازمان یک عادت عمومی دارند، بر روی لینکشان کلیک راست می‌کنند و گزینه‌ی open in a new tab را می‌زنند و یا بدتر از کلیدهای back و forward مرورگر برای عقب جلو کردن صفحات پیمایش شده استفاده میکنند.
حالا هر چقدر که توی گوششان بخوانید، که فلان لینک را باید روش فقط کلیک کنی، باز هم به خرجشان نمی‌رود.
خیلی از کارمندان برای مثال لینک درج صورت حساب جدید را bookmark کرده اند تا به سرعت به آن دسترسی داشته باشند.
همچنین autocomplete مرورگر هم در یافتن صفحات پیمایش شده به آن‌ها خیلی کمک می‌کند.
یادمه همین مشکل را توی یک برنامه‌ی نوشته شده با سیلورلایت دیدم و مشتری قبول نمی‌کرد که نمی‌تواند در یک tab جدید صفحه باز کند و ...
برنامه‌های شبیه دسکتاپ یعنی اینکه رفرش شدن صفحات را برای تغییر view از بین ببرد. هوز هم ماهیت وب است و باید همیشه این را در نظر داشت، تا بهترین تجربه‌ی رابط کاربری را فراهم کرد. نباید گفت که برنامه‌ی من فلان جور طراحی شده؛ همینی هست که هست، باید با رابط کاربری عموم وب اینترفیس‌ها هماهنگ باشد.

نویسنده: _mehdi
تاریخ: ۱۳۹۲/۰۶/۲۱ ۱۲:۳۸

سلام
چطوری میشه به فایل‌های راهنما و آموزش‌های موجود در فایل دریافتی از angular js دسترسی داشت . (پسوند فایل‌ها ngdoc.
)

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۲۱ ۱۳:۵۶

```
These files are parsed by our docs parser (nodejs script), source can be found here:
https://github.com/angular/angular.js/tree/master/docs/src
It is combined together with docs parsed from the source and result is html, served at
docs.angularjs.org
```

ماخذ

نویسنده: _mehdi
تاریخ: ۱۳۹۲/۰۶/۲۱ ۲۰:۳۵

سلام
ممنون بابت پاسختون ولی من متوجه نشدم که باید چطوری این فایل‌ها را به فایل‌های html تبدیل کرد . من باید nodejs را نصب کنم و با این فایل‌ها که در بالا ذکر کردید (البته کدومشون) ودادن مسیر پوشه راهنما فایل‌ها را به html تبدیل کنم لطفاً دقیق توضیح بدید اگه براتون امکان داره.

نویسنده: آریو
تاریخ: ۱۳۹۲/۰۸/۱۵ ۸:۳۸

سلام. با تشکر از مطالب بسیار خوبتون یه سوال داشتم.
من خیلی علاقه دارم روی الگوی MVVM کار کنم و تا اینجا به دو انتخاب رسیدم یکی KnockoutJS و دیگری AngularJS .
سوالی که دارم اینه که با توجه به اینکه من روی ASP.NET MVC کار میکنم کدوم کتابخانه برای این فریمورک مناسبتره. درسته که این کتابخانه‌ها جاوااسکریپت هستند و تکنلوژی سمت سرور مطرح نیست. اما با توجه به اینکه میخوام روی MVVM وقت بزارم به نظرتون با در نظر گرفتن امکانات ، منابع آموزشی ، میزان استفاده در توسعه دهندگان ایرانی و در نهایت قدرت عملکرد روی کدوم کار کنم؟
آیا اینها دقیقا یک کار اما با امکانات متفاوت انجام میدن یا کاربردهای متفاوتی دارند؟
باتشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۱۵ ۸:۵۲

مقایسه 8 قسمتی AngularJS vs Knockout

نویسنده: گل رز
تاریخ: ۱۳۹۲/۰۸/۱۵ ۱۰:۱۹

منابع آموزشی برای هر کدام وجود دارد . اگه قرار به یادگیری باشه از کجا و از چه منبعی باید شروع کنیم ؟ منبع خاصی مد نظرتون هست ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۱۵ ۱۰:۲۴

- در سایت جاری: AngularJS Knockout
- در سایت‌های دیگر: AngularJS Knockout SPA

نویسنده: مهدی
تاریخ: ۱۳۹۳/۰۸/۰۴ ۱۴:۷

آیا با استفاده از این فریمورک دیگر نیازی به استفاده از knockout نداریم.

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۳/۰۸/۰۴ ۱۶:۴۳

knockoutjs کتاب خانه ای برای انجام DataBinding در وب هست، یعنی برای مقید سازی اشیاء جاوا اسکریپت به کنترل‌های html.

AngularJS یک Toolset برای درست کردن framework و برنامه‌های تک صفحه ای می‌باشد، angular قابلیت مقید سازی داده‌ها را هم فراهم می‌کند.
لطفا برای اطلاعات بیشتر این مطلب را مطالعه کنید.

نویسنده: فخاری
تاریخ: ۱۳۹۳/۱۰/۱۹ ۰:۲۷

با توجه به این مطلب:
" سیستم‌های وب امروزی به دو دسته‌ی زیر تقسیم می‌شوند:
- Web Documents و یا همان وب سایت‌های معمول
- Web Applications و یا همان Single Page Web Applications "
نرم افزارهای تحت وب را با چه قالبی درست می‌کنند؟
ASP.NET MVC
یا
(Single Page application (SPA
یعنی آیا میشه که ما نرم افزار تحت وب را با ASP.NET MVC درست کنیم و از Angularjs استفاده کنیم یا اینکه نه بهتره حتما همون اول کار قالب پروژه را از نوع SPA انتخاب کنیم ؟

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: ۲۳:۵۲ ۱۳۹۲/۰۶/۰۹

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: ۷:۵۷ ۱۳۹۲/۰۶/۱۰

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: ۱۰:۴۹ ۱۳۹۲/۰۶/۱۰

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: ۱۲:۳۱ ۱۳۹۲/۰۶/۱۰

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

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

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

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

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;
                        });
       //...
}
```

نویسنده: فخاری
تاریخ: ۱۳۹۳/۱۰/۲۳ ۱۵:۲۱

سلام
بنده طبق فرمایشات شما به روش زیر عمل کردم ولی بهم کار نمیده و پارشال مورد نظر را به من نشان نمیده. میشه بگید مشکل از کجاست؟

```
public class PostController : Controller
    {
        ApplicationDbContext db;
        // GET: Post
        public ActionResult List()
        {
            using (db=new ApplicationDbContext())
            {
                var query = db.Posts.ToList();
                return PartialView("List",query);
            }
        }
    }
```

و کدهای ویو:

```
<div ng-app="postmodule">

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

```
                <hr />
                {{item.Text}}
            </li>
        </ul>
    </div>

</div>
```

و کدهایی که در فایل _Layout.cshtml نوشته ام:

```
<div class="container body-content" >

        <script src="~/Scripts/angular.js"></script>
        <script src="~/Scripts/angular-route.js"></script>
        <script>
            var PostApp = angular.module('postmodule', []).config(['$routeProvider',
  function ($routeProvider) {
      $routeProvider.
          when('/list', {
              templateUrl: '/Post/List',
              controller: 'PostController'
          });
  }]);
            PostApp.controller('PostController', function ($scope, $http, postServices) {
                //...
                $scope.ListOfItems =
                              postServices.GetPosts()
                                  .success(function (data) {
                                      $scope.ListOfItems = data;
                                  });
                //...
            });
            PostApp.service('postServices', function ($http) {
                this.GetPosts = function () {
                    return $http.get('/Post/List');
                };
            });
        </script>

        <a href="#list">list post</a>
        <div ng-view=""></div>

        @RenderBody()
        <hr />
        <footer>
            <p>&copy; @DateTime.Now.Year - My ASP.NET Application</p>
        </footer>
    </div>
```

نویسنده: عزیزخانی
تاریخ: ۱۳۹۳/۱۰/۲۳ ۱۹:۱۴

باید ngRoute رو اضافه کنید

```
var PostApp = angular.module('postmodule', ['ngRoute']).config( ...
```

view و list هم جداگانه برگشت داده شوند

عنوان: **AngularJS #3**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۶/۱۲ ۲۲:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: AngularJS

در این مقاله مفاهیم انقیاد داده (Data Binding)، تزریق وابستگی (Dependency Injection)،هدایت گرها (Directives) و سرویس‌ها را بررسی خواهیم کرد و از مقاله‌ی آینده، به بررسی ویژگی‌ها و امکانات AngularJS در قالب مثال خواهیم پرداخت.

انقیاد داده (Data Binding)

سناریو هایی وجود دارد که در آن‌ها باید اطلاعات قسمتی از صفحه به صورت نامتقارن (Asynchronous) با داده‌های دریافتی جدید به روز رسانی شود. روش معمول برای انجام چنین کاری؛ دریافت داده‌ها از سرور است که عموما به فرم HTML میباشند و جایگزینی آن با بخشی از صفحه که قرار است به روز رسانی شود، اما حالتی را در نظر بگیرید که با داده هایی از جنس JSON طرف هستید و اطلاعات صفحه را با این داده‌ها باید به روز رسانی کنید. معمولا برای حل چنین مشکلی مجبور به نوشتن مقدار زیادی کد هستید تا بتوانید به خوبی اطلاعات View را به روز رسانی کنید. حتما با خودتان فکر کرده اید که قطعا راهی وجود دارد تا بدون نوشتن کدی، قسمتی از View را به Model متناظر خود نگاشت کرده و این دو به صورت بلادرنگ از تغییرات یکدیگر آگاه شوند. این عمل عموما به مفهوم انقیاد داده شناخته می‌شود و Angular هم به خوبی از انقیاد داده دوطرفه پشتیبانی می‌کند.

برای مشاهده این ویژگی در Angular، مثال مقاله‌ی قبل را به کدهای زیر تغییر دهید تا پیغام به صورت پویا توسط کاربر وارد شود:

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Sample2</title>
</head>
<body>
    <div>
        <input type="text" ng-model="greeting.text" />
        <p>{{greeting.text}}, World!</p>
    </div>
    <script src="../Scripts/angular.js"></script>
</body>
</html>
```

بدون نیاز به حتی یک خط کد نویسی! با مشخص کردن input به عنوان Model از طریق ng-model، خاصیت greeting.text که در داخل {{ }} مشخص شده را به متن داخل textbox مقید (bind) کردیم. نتیجه می‌گیریم که جفت آکلود {{ }} برای اعمال Data Binding استفاده می‌شود.

حال یک دکمه نیز بر روی فرم قرار می‌دهیم که با کلیک کردن بر روی آن، متن داخل textbox را نمایش دهد.

```
<!DOCTYPE html>
<html ng-app>
<head>
    <title>Sample2</title>
</head>
<body>
    <div ng-controller="GreetingController">
        <input type="text" ng-model="greeting.text" />
        <p>{{greeting.text}}, World!</p>
        <button ng-click="showData()">Show</button>
    </div>
    <script src="../Scripts/angular.js"></script>
    <script>
        var GreetingController = function ($scope, $window) {
            $scope.greeting = {
                text: "Hello"
            };

            $scope.showData = function () {
                $window.alert($scope.greeting.text);
            };
        };
    </script>
</body>
</html>
```

به کمک ng-click، تابع showData به هنگام کلیک شدن، فراخوانی می‌شود. $window نیز به عنوان پارامتر کلاس GreetingController مشخص شده است. $window نیز یکی از سرویس‌های پیش فرض تعریف شده توسط Angular است و ما در اینجا در سازنده‌ی کلاس آن را به عنوان وابستگی درخواست کرده ایم تا توسط سیستم تزریق وابستگی توکار، نمونه‌ی مناسب آن در اختیار ما بگذارد. $window نیز تقریبا معادل شی window است و یکی از دلایل استفاده از آن ساده‌تر شدن نوشتن آزمون‌های واحداست.

حال متنی را داخل textbox نوشته و دکمه‌ی show را فشار دهید. متن نوشته شده را به صورت یک popup مشاهده خواهید کرد.

همچنین شی $scope نیز نمونه‌ی مناسب آن توسط سیستم تزریق وابستگی Angular، در اختیار Controller قرار می‌گیرد و نمونه‌ی در اختیار قرارگرفته، برای ارتباط با View Model و سیستم انقیاد داده استفاده می‌شود.

معمولا انقیاد داده در الگوی طراحی ModelView-ViewModel(MVVM) مطرح است و به این دلیل که این الگوی طراحی به خوبی با الگوی طراحی MVC سازگار است، این امکان در Angular گنجانده شده است. **تزریق وابستگی (Dependency Injection)**

تا به این جای کار قطعن بارها و بارها اسم آن را خوانده اید. در مثال فوق، پارامتری با نام $scope را برای سازنده‌ی کنترلر خود در نظر گرفتیم و ما بدون انجام هیچ کاری نمونه‌ی مناسب آن را که برای انجام اعمال انقیاد داده با viewmodel استفاده می‌شود را دریافت کردیم. به عنوان مثال، $window را نیز در سازنده‌ی کلاس کنترلر خود به عنوان یک وابستگی تعریف کردیم و تزریق نمونه‌ی مناسب آن توسط سیستم تزریق وابستگی توکار Angular صورت می‌گرفت.

اگر با IOC Containerها در زبانی مثل #C کار کرده باشید، قطعا با IOC Container فراهم شده توسط Angular هم مشکلی نخواهید داشت.

اما یک مشکل! در زبانی مثل #C که همه‌ی متغیرهای دارای نوع هستند، IOC Container با استفاده از Reflection، نوع پارامترهای درخواستی توسط سازنده‌ی کلاس را بررسی کرده و با توجه به اطلاعاتی که ما از قبل در دسترس آن قرار داده بودیم، نمونه‌ی مناسب آن را در اختیار در خواست کننده می‌گذارد.

اما در زبان جاوا اسکریپت که متغیرها دارای نوع نیستند، این کار به چه شکل انجام می‌گیرد؟

Angular برای این کار از نام پارامترها استفاده می‌کند. برای مثال Angular از نام پارامتر $scope می‌فهمد که باید چه نمونه ای را به کلاس تزریق کند. پس نام پارامترها در سیستم تزریق وابستگی Angular نقش مهمی را ایفا می‌کنند.

اما در زبان جاوا اسکریپت، به طور پیش فرض امکانی برای به دست آوردن نام پارامترهای یک تابع وجود ندارد؛ پس Angular چگونه نام پارامترها را به دست می‌آورد؟ جواب در سورس کد Angular و در تابعی به نام annotate نهفته است که اساس کار این تابع استفاده از چهار عبارت با قاعده (Regular Expression) زیر است.

```
var FN_ARGS = /^function\s*[^\(]*\(\s*([^\)]*)\)/m;
var FN_ARG_SPLIT = /,/;
var FN_ARG = /^\s*(_?)(\S+?)\1\s*$/;
var STRIP_COMMENTS = /((\/\/.*$)|(\/\*[\s\S]*?\*\/))/mg;
```

تابع annotate تابعی را به عنوان پارامتر دریافت می‌کند و سپس با فراخواندن متد toString آن، کدهای آن تابع را به شکل یک رشته در می‌آورد. حال کدهای تابع را که اکنون به شکل یک رشته در دسترس است را با استفاده از عبارات با قاعده‌ی فوق پردازش می‌کند تا نام پارامترها را به دست آورد. در ابتدا کامنت‌های موجود در تابع را حذف می‌کند، سپس نام پارامترها را استخراج می‌کند و با استفاده از "," آن‌ها را جدا می‌کند و در نهایت نام پارامترها را در یک آرایه باز می‌گرداند.

استفاده از تزریق وابستگی، امکان نوشتن کدهایی با قابلیت استفاده مجدد و نوشتن ساده‌تر آزمون‌های واحد را فراهم می‌کند. به خصوص کدهایی که با سرور ارتباط برقرار می‌کنند را می‌توان به یک سرویس انتقال داد و از طریق تزریق وابستگی، از آن در کنترلر استفاده کرد. سپس در آزمون‌های واحد می‌توان قسمت ارتباط با سرور را با یک نمونه فرضی جایگزین کرد تا برای تست، احتیاجی به راه اندازی یک وب سرور واقعی و یا مرورگر نباشد. **Directives**

یکی از مزیت‌های Angular این است که قالب‌ها را می‌توان با HTML نوشت و این را باید مدیون موتور قدرتمند تبدیل گر DOM بدانیم که در آن گنجانده شده است و به شما این امکان را می‌دهد تا گرامر HTML را گسترش دهید.

تا به این جای کار با attributeهای زیادی در قالب HTML روبرو شدید که متعلق به HTML نیست. به طور مثال: جفت آکولادها که برای انقیاد داده به کار برده می‌شود، ng-app که برای مشخص کردن بخشی که باید توسط Angular کامپایل شود، ng-controller که برای مشخص کردن این که کدام بخش از View متعلق به کدام Controller است و ... تمامی Directiveهای پیش فرض Angular هستند.

با استفاده از Directiveها می‌توانید عناصر و خاصیت‌ها و حتی رویدادهای سفارشی برای HTML بنویسید؛ اما واقعا چه احتیاجی به

تعریف عنصر سفارشی و توسعه گرامر HTML وجود دارد؟

HTML یک زبان طراحی است که در ابتدا برای تولید اسناد ایستا به وجود آمد و هیچ وقت هدفش تولید وب سایت‌های امروزی که کاملا پویا هستند نبود. این امر تا جایی پیش رفته است که HTML را از یک زبان طراحی تبدیل به یک زبان برنامه نویسی کرده است و احتیاج به چنین زبانی کاملا مشهود است. به همین دلیل جامعه‌ی وب مفهومی را به نام Web Components مطرح کرده است. Web Components به شما امکان تعریف عناصر HTML سفارشی را می‌دهد. برای مثال شما یک تگ سفارشی به نام datepicker می‌نویسید که دارای رفتار و ویژگی‌های خاص خود است و به راحتی عناصر HTML رابا استفاده از آن توسعه می‌دهید. مطمئنا آینده‌ی وب این گونه است، اما هنوز خیلی از مرورگرها از این ویژگی پشتیبانی نمی‌کنند.

یکی دیگر از معادل‌های Web Componentهای امروز را می‌توان ویجت‌های jQuery UI دانست. اگر بخواهم تعریفی از ویجت ارائه دهم به این گونه است که یک ویجت؛ کدهای HTML، CSS و javascript مرتبط به هم را کپسوله کرده است. مهم‌ترین مزیت ویجت ها، قابلیت استفاده‌ی مجدد آن‌هاست، به این دلیل که تمام منطق مورد نیاز را در خود کپسوله کرده است؛ برای مثال ویجت datepicker که به راحتی در برنامه‌های مختلف بدون احتیاج به نوشتن کدی قابل استفاده است.

خب، متاسفانه Web Componentها هنوز در دنیای وب امروزی رایج نشده اند و ویجت‌ها هم آنچنان قدرت Web Componentها را ندارند. خب Angular با استفاده از امکان تعریف Directiveهای سفارشی به صورت cross-browser امکان تعریف عناصر سفارشیه همانند web Componentها را به شما می‌دهد. حتی به عقیده‌ی عده ای Directiveها بسیار قدرتمندتر از Web Components عمل می‌کنند و راحتی کار با آن‌ها بیشتر است.

با استفاده از Directiveها می‌توانید عنصر HTML سفارشی مثل <datepicker />، خاصیت سفارشی مثل ng-controller، رویداد سفارشی مثل ng-click را تعریف کنید و یا حتی حالت و اتفاقات رخ داده در برنامه را زیر نظر بگیرید.

و این یکی از دلایلی است که می‌گویند Angular دارای ویژگی forward-thinking است.

البته Directiveها یکی از قدرتمندترین امکانات فریم ورک AngularJS است و در آینده به صورت مفصل بر روی آن بحث خواهد شد. **سرویس‌ها در AngularJS**

حتما این جمله را در هنگام نوشتن برنامه‌ها با الگوی طراحی MVC بارها و بارها شنیده اید که در Controllerها نباید منطق تجاری و پیچیده ای را پیاده سازی کرد و باید به قسمت‌های دیگری به نام سرویس‌ها منتقل شوند و سپس در سازنده‌ی کلاس کنترلر به عنوان پارامتر تعریف شوند تا توسط Angular نمونه‌ی مناسب آن به کنترلر تزریق شود. Controllerها نباید پیاده کننده‌ی هیچ منطق تجاری و یا اصطلاحا business برنامه باشد و باید از لایه‌ی سرویس استفاده کنند و تنها وظیفه‌ی کنترلر باید مشخص کردن انقیاد داده و حالت برنامه باشد.

دلیل استفاده از سرویس‌ها در کنترلر ها، نوشتن ساده‌تر آزمون‌های واحد و استفاده‌ی مجدد از سرویس‌ها در قسمت‌های مختلف پروژه و یا حتی پروژه‌های دیگر است.

معمولا اعمال مرتبط در ارتباط با سرور را در سرویس‌ها پیاده سازی می‌کنند تا بتوان در موقع نوشتن آزمون‌های واحد یک نمونه‌ی فرضی را خودمان ساخته و آن را به عنوان وابستگی به کنترلری که در حال تست آن هستیم تزریق کنیم، در غیر این صورت احتیاج به راه اندازی یک وب سرور واقعی برای نوشتن آزمون‌های واحد و در نتیجه کند شدن انجام آزمون را در بر دارد. قابلیت استفاده‌ی مجدد سرویس هم به این معناست که منطق پیاده سازی شده در آن نباید ربطی به رابط کاربری و ... داشته باشد. برای مثال یک سرویس به نام userService باید دارای متد هایی مثل دریافت لیست کاربران، افزودن کاربر و ... باشد و بدیهی است که از این سرویس‌ها می‌شود در قسمت‌های مختلف برنامه استفاده کرد. همچنین سرویس‌ها در Angular به صورت Singleton در اختیار کنترلرها قرار می‌گیرند و این بدین معناست که یک نمونه از هر سرویس ایجاد شده و به بخش‌های مختلف برنامه تزریق می‌شود.

مفاهیم پایه ای AngularJs به پایان رسید. در مقاله بعدی یک مثال تقریبا کامل را نوشته و با اجزای مختلف Angular بیشتر آشنا می‌شویم. *با تشکر از مهدی محزونی برای بازبینی مطلب*

عنوان: **AngularJS #4**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۷:۱۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در این قسمت قصد دارم تا یک سیستم ارسال دیدگاه را به کمک Angular پیاده سازی کنم. هدف از این مثال؛ آشنایی با چند Directive توکار Angular و همچنین آموختن چگونگی کار با سرویس $http برای ارتباط با سرور است.
کدهای HTML زیر را در نظر بگیرید:

```
<div ng-app="myApp">
    <div ng-controller="CommentCtrl">

        <div ng-repeat="comment in comments">
            <div style="float:right;cursor:pointer;" ng-click="remove(comment.Id,$index);">X</div>
            <a href="#">
                <img style="width:32px;" ng-src="/Content/user.gif" alt="{{comment.Name}}">
            </a>
            <div>
                <h4>{{comment.Name}}</h4>
                {{comment.CommentBody}}
            </div>
        </div>

        <div>
            <form action="/Comment/Add" method="post">
                <div>
                    <label for="Name">Name</label>
                    <input id="Name" type="text" name="Name" ng-model="comment.Name" placeholder="Your
Name" />
                </div>
                <div>
                    <label for="Email">Email</label>
                    <input id="Email" type="text" name="Email" ng-model="comment.Email"
placeholder="Your Email" />
                </div>
                <div>
                    <label for="CommentBody">Comment</label>
                    <textarea id="CommentBody" name="CommentBody" ng-model="comment.CommentBody"
placeholder="Your Comment"></textarea>
                </div>
                <button type="button" ng-click="addComment()">Send</button>
            </form>
        </div>

    </div>
</div>
```

خب از ابتدا ساختار را مورد بررسی قرار می‌دهم و موارد ناآشنای آن را توضیح می‌دهم:
ng-app : خاصیت ng-app جز خواص پیش فرض HTML نیست و یک خاصیت سفارشی است که توسط Angular به صورت پیش فرض تعریف شده است. این خاصیت به Angular می‌گوید که کدام بخش از DOM باید توسط Angular مدیریت و پردازش شود. در اینجا div ای که با خاصیت ng-app مزین شده است به همراه تمامی عناصر فرزند آن توسط موتور پردازش گر DOM توکار مورد پردازش قرار گرفته و اصطلاحا کامپایل می‌شود. بله! اینجا از لفظ کامپایل شدن برای بیان این فرآیند استفاده کردم. هیچ کدام از این Directiveهای سفارشی به خودی خود برای مرورگر قابل تفسیر نیست و اینجاست که Angular وارد عمل شده و این Directiveها را به کدهای HTML و جاوا اسکریپت که برای مرورگر قابل فهم است تبدیل می‌کند. به همین جهت با ng-app مشخص می‌کنیم که کدام بخش از DOM باید توسط Angular تفسیر و مدیریت شود.شاید این سوال برای شما مطرح شده باشد که در مثال قبلی ng-app مقداری نداشت و برای تگ html تعریف شده بود. پاسخ این است که در مثال قبلی چون برنامه‌ی ما دارای یک ماژول بیشتر نبود می‌توانستیم از مقدار دهی ng-app صرف نظر کنیم؛ اما در این مثال ما قصد داریم کمی هم مفهوم ماژول را در Angular بررسی کنیم. در نتیجه در این مثال برنامه‌ی ما از ماژولی به نام myApp تشکیل شده است. دلیل اینکه در این مثال ng-app بر روی یک div تعریف شده است این است که همین قسمت از DOM توسط Angular تفسیر شود برای ما کفایت می‌کند. هنگامی ng-app را بر روی html تعریف می‌کنیم که قصد داشته باشیم کل صفحه توسط Angular تفسیر شود. **ng-controller** : در Angular کنترلرها تابع سازنده‌ی کلاس‌های ساده‌ی جاوا اسکریپتی هستند که به کمک آن‌ها بخشی از صفحه را مدیریت می‌کنیم. این که کدام بخش از

صفحه توسط کدام کلاس کنترل و مدیریت شود، توسط ng-controller مشخص می‌شود. در اینجا هم عنصری که با ng-controller مشخص شده به همراه تمامی فرزندانش، توسط کلاس جاوا اسکریپتی به نام CommentCtrl مدیریت می‌شود. در حقیقت ما به کمک ng-controller مشخص می‌کنیم که کدام قسمت از View توسط کدام Controller مدیریت می‌شود. مرسوم است که در Angular نام کنترلرها با Ctrl خاتمه یابد. **ng-repeat** : همه‌ی نظرات دارای یک قالب html یکسان هستند که به ازای داده‌های متفاوت تکرار شده اند. اگر می‌خواستیم نظرات را استفاده از موتور نمایشی Razor نشان دهیم از یک حلقه‌ی foreach استفاده می‌کردیم. خبر خوب این است که ng-repeat هم دقیقا به مانند حلقه‌ی foreach عمل می‌کند.در اینجا عبارت comment in comments دقیقا برابر با آن چیزی است که در یک حلقه‌ی foreach می‌نوشتیم. Comments در اینجا یک لیست به مانند آرایه ای از comment هست که در کنترلر مقدار دهی شده است. پس اگر با حلقه‌ی foreach مشکلی نداشته باشید با مفهوم ng-repeat هم مشکلی نخواهید داشت و دقیقا به همان شکل عمل می‌نماید. **ng-click** : همان طور که گفتیم Directiveهای تعریف شده می‌توانند یک event سفارشی نیز باشند. ng-click هم یک Directive تو کار است که توسط Angular به صورت پیش فرض تعریف شده است. کاملا مشخص است که یک تابع به نام remove تعریف شده است که به هنگام کلیک شدن، فراخوانی می‌شود. دو پارامتر هم به آن ارسال شده است. اولین پارامتر Id دیدگاه مورد نظر است تا به سرور ارسال شود و از پایگاه داده حذف شود. دومین پارامتر $index است که یک متغیر ویژه است که توسط Angular در هر بار اجرای حلقه‌ی ng-repeat مقدارش یک واحد افزایش می‌یابد. $index هم به تابع remove ارسال می‌شود تا بتوان فهمید در سمت کلاینت کدام نظر باید حذف شود. **ng-src** : از این Directive برای مشخص کردن src عکس‌ها استفاده می‌شود. البته در این مثال چندان تفاوتی بین ng-src و src معمولی وجود ندارد. ولی اگر آدرس عکس به صورت Content/{{comment.Name}}.gif می‌بود دیگر وضع فرق می‌کرد. چرا که مرورگر با دیدن آدرس در src سعی به لود کردن آن عکس می‌کند و در این حالت در لود کردن آن عکس با شکست روبرو می‌شود. ng-src سبب می‌شود تا در ابتدا آدرس عکس توسط Angular تفسیر شود و سپس آن عکس توسط مرورگر لود شود. **{{comment.Name}}** : آکلودهای دوتایی برای انقیاد داده (Data Binding) با view-model استفاده می‌شود. این نوع اقیاد داده در مثال‌های قبلی مورد بررسی قرار گرفته است و نکته‌ی بیشتری در اینجا مطرح نیست. **ng-model** : به کمک ng-model می‌توان بین متن داخل textbox و خاصیت شی مورد نظر انقیاد داده بر قرار کرد و هر دو طرف از تغییرات یکدیگر آگاه شوند. به این عمل انقیاد داده دوطرفه (Two-Way Data-Binding) می‌گویند.برای مثال textbox مربوط به نام را به comment.Name و textbox مربوط به email را به comment.Email مقید(bind) شده است. هر تغییری که در محتوای هر کدام از طرفین صورت گیرد دیگری نیز از آن تغییر با خبر شده و آن را نمایش می‌دهد.

تا به اینجای کار قالب مربوط به HTML را بررسی کردیم. حال به سراغ کدهای جاوا اسکریپت می‌رویم:

```
var app = angular.module('myApp', []);
app.controller('CommentCtrl', function ($scope, $http) {
    $scope.comment = {};
    $http.get('/Comment/GetAll').success(function (data) {
        $scope.comments = data;
    })
    $scope.addComment = function () {
        $http.post("/Comment/Add", $scope.comment).success(function () {
            $scope.comments.push({ Name: $scope.comment.Name, CommentBody: $scope.comment.CommentBody
});
            $scope.comment = {};
        });
    };
    $scope.remove = function (id, index) {
        $http.post("/Comment/Remove", { id: id }).success(function () {
            $scope.comments.splice(index, 1);
        });
    };
});
```

در تعریف ng-app اگر به یاد داشته باشید برای آن مقدار myApp در نظر گرفته شده بود. در اینجا هم ما به کمک متغیر سراسری angular که توسط خود کتابخانه تعریف شده است، ماژولی به نام myApp را تعریف کرده ایم. پارامتر دوم را فعلا توضیح نمی‌دهم، ولی در این حد بدانید که برای تعریف وابستگی‌های این ماژول استفاده می‌شود که من آن را برابر یک آرایه خالی قرار داده ام.
در سطر بعد برای ماژول تعریف شده یک controller تعریف کرده ام. شاید دفعه‌ی اول است که تعریف کنترلر به این شکل را مشاهده می‌کنید. اما چرا به این شکل کنترلر تعریف شده و به مانند قبل به شکل تابع سازنده‌ی کلاس تعریف نشده است؟
پاسخ این است که اکثر برنامه نویسان از جمله خودم دل خوشی از متغیر سراسری ندارند. در شکل قبلی تعریف کنترلر، کنترلر به شکل یک متغیر سراسری تعریف می‌شد. اما استفاده از ماژول برای تعریف کنترلر سبب می‌شود تا کنترلرهای ما روی هوا تعریف نشده باشند و هر یک در جای مناسب خود باشند. به این شکل مدیریت کدهای برنامه نیز ساده‌تر بود. مثلا اگر کسی از شما بپرسد که فلان کنترلر کجا تعریف شده است؛ به راحتی می‌گویید که در فلان ماژول برنامه تعریف و مدیریت شده است.
در تابعی که به عنوان کنترلر تعریف شده است، دو پارامتر به عنوان وابستگی درخواست شده است. scope $ که برای ارتباط با view-model و انقیاد داده به کار می‌رود و http $ که برای ارتباط با سرور به کار می‌رود. نمونه‌ی مناسب هر دوی این پارامترها توسط سیستم تزریق وابستگی تو کار angular در اختیار کنترلر قرار می‌گیرد.
قبلا چگونگی استفاده از scope $ برای اعمال انقیاد داده توضیح داده شده است. نکته‌ی جدیدی که مطرح است چگونگی استفاده از سرویس $http برای ارتباط با سرور است. سرویس http $ دارای 4 متد get ، post ، put و delete است.
واقعا استفاده از این سرویس کاملا واضح و روشن است. در متد addComment وقتی که دیدگاه مورد نظر اضافه شد، به آرایه‌ی کامنت‌ها یک کامنت جدید می‌افزاییم و چون انقیاد داده دو طرفه است، بالافاصله دیدگاه جدید نیز در view به نمایش در می‌آید.کار تابع remove هم بسیار ساده است. با استفاده از index ارسالی، دیدگاه مورد نظر را از آرایه‌ی کامنت‌ها حذف می‌کنیم و ادامه‌ی کار توسط انقیاد داده دو طرفه انجام می‌شود. همان طور که مشاهده می‌شود مفاهیم انقیاد داده دو طرفه و تزریق وابستگی خودکار سرویس‌های مورد نیاز، کار با angularjs را بسیار ساده و راحت کرده است. اصولا در بسیاری از موارد احتیاجی به باز اختراع چرخ نیست و کتابخانه‌ی angular آن را برای ما از قبل تدارک دیده است.
کدهای این مثال ضمیمه شده است. این کدها در Visual Studio 2013 و به کمک ASP.NET MVC 5 و Entity Framework 6 نوشته شده است. سعی شده تا مثال نوشته شده به واقعیت نزدیک باشد. اگر دقت کنید مدل کامنت در مثالی که نوشتم به گونه ای است که دیدگاه‌های چند سطحی به همراه پاسخ هایش مد نظر بوده است. به عنوان تمرین نمایش درختی این گونه دیدگاه‌ها را به کمک Angular انجام دهید. کافیست Treeview in Angular را جست و جو کنید؛ مطمئنا به نتایج زیادی می‌رسید. گرچه در مثال ضمیمه شده اگر جست و جو کنید من پیاده سازیش را انجام دادم. هدف از جست و جو در اینترنت مشاهده این است که بیشتر مسائل در Angular از پیش توسط دیگران حل شده است و احتیاجی نیست که شما با چالش‌های جدیدی دست و پنجه نرم کنید.
پس به عنوان تمرین، دیدگاه‌های چند سطحی به همراه پاسخ که نمونه اش را در همین سایتی که درحال مشاهده آن هستید می‌بینید را به کمک AngularJS پیاده سازی کنید.
در مقاله‌ی بعدی چگونگی انتقال منطق تجاری برنامه از کنترلر به لایه سرویس و چگونگی تعریف سرویس جدید را مورد بررسی قرار می‌دهم. AngularSample1.rar

نظرات خوانندگان

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۸:۴۹

ممنون از مطلبتون. یک سوال کوچک :
شما با استفاده از کتابخانه Newtonsoft.Json لیست خودتون رو سریالایز کردید و بعد بازگشت دادید :

```
var comments = _db.Comments.Include(x => x.Children).ToList().Where(x => x.Parent == null).ToList();

var result = JsonConvert.SerializeObject(comments, Formatting.Indented,
        new JsonSerializerSettings
        {
              ReferenceLoopHandling = ReferenceLoopHandling.Ignore,
        });

return Content(result);
```

در حالی که با این روش هم میشه پاسخ داد هر چند در شی‌های تو در تو ابتدا باید فیلدها رو مشخص کنیم :

```
var comments = _db.Comments.Include(x => x.Children).ToList().Where(x => x.Parent == null).ToList();
return Json(comments , JsonRequestBehavior.AllowGet);
```

تفاوت این دو آیا در حجم مقدار دیتای ارسالی تفاوتی ندارند؟
طول محتوا با روش شما برای من 1278 و زمان 584ms و در روش دوم طول محتوا به 760 و زمان 135ms کاهش پیدا کرد. در مثالی دیگر منتها با همین دو روش بالا. البته اینها ربطی به مطلب شما نداشت.فقط میخواستم بدونم تفاوت این دو غیر از طول محتوا و زمان ،با همدیگه چیه؟
بیصبرانه منتظر مطلب بعدی شما هستم.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۹:۲۷

علت استفاده من از Newtonsoft.Json توانایی سریالایز کردن اشیا تودرتو است. فکر کنم با کمک یک خاصیت به نام IgnoreJsonConvert بتوانیم بگیم که چه خواصی را سریالایز نکند. همچنین این کتابخانه خیلی توانایی دیگر هم دارد. یک نمونش که می‌تونه مفید باشه اینه که نام خواص اشیا را به صورت استاندارد camelCase سریالایز کنه. همچنین از نظر سرعت هم نسبت به نمونه‌ی توکار برتری قابل توجهی داره.

نویسنده: زمان
تاریخ: ۱۳۹۲/۰۸/۲۳ ۱۹:۳۶

با سلام. تشکر بابت مقاله و یک سوال.
شما عملیات CRUD رو در کلاس کنترلر تعریف کردید. در برخی مقالات دیدم که اینها توسط کنترلرهای WebApi مدیریت میشوند. آیا تفاوتی از لحاظ کارایی بین این دو روش وجود دارد یا شما ترجیحاً از روش اول استفاده کردید؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۸/۲۳ ۲۰:۱

There is more than one way to skin a cat
On The Coexistence of ASP.NET MVC and WebAPI

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۸/۲۳ ۲۰:۷

شخصا علاقه ای به استفاده از webapi برای دریافت اطلاعات به فرمت json ندارم. webapi محدودیت‌ها و مزیت‌های خاص خودش را دارد که در اینجا کنترلرهای معمولی بدون محدودیتی کار مورد نیاز من را انجام می‌دهند.

نویسنده: آریو
تاریخ: ۱۳۹۲/۱۱/۱۵ ۰:۱۷

با سلام. من طبق آموزش‌های شما که واقعا عالیه پیش رفتم و توی یه کار عملی به یه مشکلی خوردم.

فایل اسکریپت من اینه :

```
var saman = angular.module('SamanApplication', []);

saman.service('loginService', ['$http', function (http) {
    var loginData = [];
    this.login = function () {

        http.get('Saman/LogOn/IsLogedIn').success(function (data, status, headers, config) {
            loginData = data;
        });
        return loginData;
    };
}]);

        saman.controller('loginController', function ($scope, loginService) {
            $scope.response = [];
            $scope.click = function () { $scope.response = loginService.login() };
        });
```

فایل html هم :

```
<body ng-app="SamanApplication">
    <div ng-controller="loginController">
        <button ng-click="click()">Test</button>
        {{data}}
    </div>
</body>
```

اما مشکل اینجاست که بار اول که کلیک میکنم اطلاعات از سرور میاد ولی در {{data}} نمایش داده نمیشه. اما بار دوم که کلیک میکنم نمایش داده میشه !
امکانش هست راهنمایی کنید ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۱۵ ۱:۱۲

http.get به صورت async اجرا میشه و نه synchronous. یعنی زمانیکه فراخوانی شد، سطر return بعدی اجرا میشه و صبر نمی‌کنه تا به اینجا برسه. بهتره از promises استفاده کنید (راه حل استانداردش): ^ و ^

نویسنده: محمد
تاریخ: ۱۳۹۲/۱۱/۲۳ ۱۱:۶

سلام؛ من از ngResource دارم برای گرفتن اطلاعات از سرور استفاده میکنم. آیا استفاده از این ماژول به جای $http مشکلی به وجود نمیاره ؟ کلا میشه در مورد ngResource بیشتر توضیح بدین ؟

نویسنده: سعید رضایی
تاریخ: ۱۳۹۲/۱۲/۲۱ ۱۲:۱۰

با عرض سلام من از آموزشتون استفاده کردم وقتی دیتا رو ذخیره می‌کنم مقادیر به صورت null ذخیره میشه تو دیتابیس اینم کدم:

html

```
<div ng-app="myApp" id="ng-app">



<div ng-controller="MenuCtrl" style="width:300px">

  <div style="height:200px;overflow:auto;">
  <div ng-repeat="menu in menu" >
<div style="float:right;cursor:pointer;" ng-click="remove(menu.ID,$index);">X</div>
<a href="#">
<img style="width:32px;" ng-src="/Content/user.gif" alt="{{menu.Title}}">
</a>
<div>
<h4>{{menu.Title}}</h4>
{{menu.Url}}
</div>
  </div>
</div>

  <form action="/Menu/Add" method="post">
<div>
<label for="Title">عنوان</label>
<input id="Title" type="text" name="Title" ng-model="menu.Title" placeholder="عنوان" />
</div>
<div>
<label for="Url">آدرس</label>
<input id="Url" type="text" name="Url" ng-model="menu.Url" placeholder="آدرس" />
</div>
<div>
<label for="ParentID">والد</label>
<input id="ParentID" type="text" name="ParentID" ng-model="menu.ParentID" placeholder="والد" />
</div>


<button type="button"  ng-click="addmenu()">ذخیره</button>
  </form>
</div>
</div>
```

myapp

```
var app = angular.module('myApp', ['ngAnimate']);

app.controller('MenuCtrl', function ($scope, $http) {

    $scope.menu = {};

    $http.get('/Menu/GetAll').success(function (data) {

        $scope.menu = data;

    })
    $scope.addmenu= function () {

        $http.post("/Menu/Add", $scope.menu).success(function () {

            $scope.menus.push({ Title: $scope.menu.Title, Url: $scope.menu.Url, ParentID:
$scope.menu.ParentID });

            $scope.menu = {};

        });
    };

    $scope.remove = function (ID, index) {

        $http.post("/Menu/Remove", { ID: ID }).success(function () {

            $scope.menu.splice(index, 1);

        });
    };

});
```

controller

```
public class MenuController : Controller
    {
        //
        // GET: /Menu/

        MyContext _db = new MyContext();

        public ActionResult GetAll()
        {
            var menu = _db.Menus.ToList();

            var result = JsonConvert.SerializeObject(menu, Formatting.Indented,
                                new JsonSerializerSettings
                                {
                                    ReferenceLoopHandling = ReferenceLoopHandling.Ignore,
                                });

            return Content(result);
        }

        public ActionResult Add(Menu menu)
        {
            _db.Menus.Add(menu);

            _db.SaveChanges();

            return Json("1");
        }

        public ActionResult Remove(int id)
        {
            var selectedMenu = new Menu { ID = id };

            _db.Menus.Attach(selectedMenu);

            _db.Menus.Remove(selectedMenu);

            _db.SaveChanges();

            return Json("1");
        }

        public ActionResult Index()
        {
            return View();
        }

    }
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۲/۲۱ ۱۳:۹

دیباگ کنید چه اطلاعات JSON ایی به سرور ارسال میشه؟

نویسنده: سعید رضایی
تاریخ: ۱۳۹۲/۱۲/۲۱ ۱۳:۱۳

null ارسال می‌شه یعنی هیچکدوم از فیلدها مقدار ندارن

نویسنده: علی رضایی
تاریخ: ۱۳۹۳/۰۱/۰۳ ۱۲:۴۳

سلام. با تشکر فراوان برای این آموزش.
در زمان اجرای این برنامه اگر پس از ورود اطلاعات جدید بخواهیم رکوردی را حذف کنیم (قبل از ریفرش)، حذف انجام نمیشود؛ دلیل آن عدم وجود Id نظری است که جدیدا ثبت شده است؛ راه حل آن به شرح ذیل است:

ابتدا در سمت سرور اکش Add باید به شکل ذیل تغییر یابد:

```
public ActionResult Add(Comment comment)
        {
            _db.Comments.Add(comment);

            _db.SaveChanges();

            return Json(comment.Id);
        }
```

و سپس در سمت کلاینت متد AddComent به شکل ذیل تغییر یابد:

```
$scope.addComment = function () {

        $http.post("/Comment/Add", $scope.comment).success(function (id) {

            $scope.comments.push({Id:id ,Name: $scope.comment.Name, CommentBody:
$scope.comment.CommentBody });

            $scope.comment = {};

        });
    };
```

نویسنده: مهدی
تاریخ: ۱۳۹۳/۰۳/۰۲ ۱۱:۳۷

آموزشهای خیلی عالی هستن. با اینکه من ASP کار نمیکنم ولی این مقوله برای همه زبانهای تحت وب هست. دستتون درد نکنه.
ممنون

نویسنده: علی اسدی
تاریخ: ۱۳۹۳/۰۳/۱۴ ۱۹:۲۸

```
//define
app.service('objUser', function ($http) {

    this.user = [{
        id: null,
        firstName: null,
        lastName: null,
        email: null
    }];

  this.userList = function () {
        var promise = $http.get('api/user')
            .success(function (res) {
                return res;
            });
        return promise;
    };
});

//call
app.controller('UserListCtrl', function ($scope, objUser) {
    $scope.user = objUser.user;
    objUser.userList().then(function (promise) {
        $scope.user = promise.data;
    });
});
```

نویسنده: حمیدرضا

تاریخ: ۱۳۹۳/۰۷/۲۶ ۱۸:۴۱

سلام ، من مطالب شما در مورد angularJs رو خوندم،اما مزیت این رو نسبت به mvc c# نفهمیدم

چرا باید یه لایه اضافه کنیم در حالی که همین کار هارم با کنترلرها در mvc میشه انجام داد

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۷/۲۶ ۲۲:۵۹

یکی سمت سرور هست. یکی سمت کاربر. AngularJS برای نظم دادن و مدیریت قسمت سمت کاربر که صرفا درون مرورگر اجرا میشه، طراحی شده.

نویسنده: محمدرضا نظری
تاریخ: ۱۳۹۴/۰۱/۳۰ ۱۵:۴۷

سلام ..
پروژه من در هنگام فراخوانی فانکشن addComment ، به آدرس http://localhost:18147/Comment/Add منتقل میشه و مقدار "1" رو نشون میده ، مشکل از کجاست ؟

نویسنده: محمدرضا نظری
تاریخ: ۱۳۹۴/۰۱/۳۰ ۲۳:۳۶

مشکل از این بود که بنده Button را به صورت ذیل تعریف کرده بودم ..

```
<button ng-click="addComment()">Send</button>
```

در صورتی که میبایست به صورت ذیل تعریف میشد :

```
<button type="button" ng-click="addComment()">Send</button>
```

عنوان: **به اشتراک گذاری داده ها بین کنترلرها در AngularJs**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۹/۱۷ ۹:۰
آدرس: www.dotnettips.info
برچسب‌ها: AngularJS

در پست قبلی با مفاهیم کنترلر و مدل در AngularJs آشنا شدید. قصد دارم روشی را بررسی کنم که یک منبع داده را بین کنترل‌های تعریف شده در یک ماژول را به اشتراک بگذاریم.
ابتدا یک فایل جاوااسکریپ به نام module1 ایجاد می‌کنیم . در این فایل ابتدا ماژول خود را به Angular معرفی کرده و سپس با استفاده از دستور factory سرویس مورد نظر برای به اشتراک گذاری داده را می‌سازیم:

```
var app = angular.module('myApp', []);

app.factory('BookData', function () {
    var books = [
        { code: 1, name: 'book1', },
        { code: 2, name: 'book2', },
        { code: 3, name: 'book3', },
        { code: 4, name: 'book4', },
        { code: 5, name: 'book5', }
    ];
    return books;
});
```

همان طور که در پست قبلی شرح داده شده برای تعریف ماژول از دستور angular.module استفاده می‌کنیم. در خط بعدی یک سرویس به نام BookData را با استفاده از دستور factory در ماژول مربوطه ساخته می‌شود. تابع مورد نظر یک آرایه از کتاب‌ها را که هر کدام از آن‌ها شامل کد و نام است برگشت می‌دهد. قصد داریم کنترل‌های تعریف شده در ماژول myApp بتوانند به این لیست این کتاب‌ها دسترسی داشته باشند. در این مرحله ابتدا یک کنترلر به نام به controller1 به صورت زیر می‌سازیم:

```
app.controller('controller1', function ($scope, BookData) {
    $scope.books = BookData;
});
```

تنها نکته قابل ذکر، تزریق مقادیر $scope و BookData به تابع سازنده کنترلر مربوطه است. از $scope برای مقید سازی مقادیر مدل به عناصر dom در view استفاده می‌شود و BookData در این جا دقیقا به مقدار برگشت داده شده از سرویس BookData اشاره می‌کند(نام سرویس مورد نظر دقیقا باید با مقداری که به عنوان آرگومان اول در تابع factory پاس می‌دهید یکی باشد). در نتیجه این مقدار را به متغیر books در $scope نسبت می‌دهیم. برای کنترلر دوم نیز همین مراحل را تکرار می‌کنیم:

```
app.controller('controller2', function ($scope, BookData) {
    $scope.books = BookData;
});
```

در View مورد نظر نیز یک ارجاع به فایل ساخته شده بالا خواهیم داشت و سپس کدهای مربوط به نمایش را به صورت زیر می‌نویسیم(البته ارجاع به فایل اصلی angular.js فراموش نشود):

```
<script type="text/javascript" src="~/scripts/app/controller1.js"></script>


<div ng-app="myApp">
    <div ng-controller="controller1">
        <p>Data from controller1</p>
        <table>
            <tr ng-repeat="book in books">
                <td>
                    {{book.code}}
                </td>
                <td>
                    {{book.name}}
                </td>
```

```
            </tr>
        </table>
    </div>

    <div ng-controller="controller2">
        <p>Data from controller2</p>
        <table>
            <tr ng-repeat="book in books">
                <td>
                    {{book.code}}
                </td>
                <td>
                    {{book.name}}
                </td>
            </tr>
        </table>
    </div>
</div>
```

ابتدا در تگ div اول با استفاده از ng-app محدوده ماژول مورد نظر در صفحه را تعیین کرده سپس با استفاده از تگ‌های div جداگانه هر کدام از نواحی تحت کنترل مربوط به کنترلرهای تعریف شده را مشخص می‌کنیم.

با استفاده از ng-repeat به راحتی در بین آرایه کتاب‌ها پیمایش کرده و لیست مورد نظر در صفحه نمایش داده می‌شود.

(توضیحات مربوط به ng-repeat و {{}} در پست قبلی شرح داده شده است). خروجی به صورت زیر خواهد بود. واضح است که اطلاعات نمایش داده شده توسط هر دو کنترلر به دلیل استفاده از منبع داده ای یکسان، به یک شکل خواهد بود.

Data from controller1

1	book1
2	book2
3	book3
4	book4
5	book5

Data from controller2

1	book1
2	book2
3	book3
4	book4
5	book5

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۱۷ ۹:۵۷

ممنون از شما. با توجه به اینکه در factory ایجاد شده یک نمونه جدید از books ایجاد می‌شود، آیا با تزریق آن در سازنده‌های کنترلرهای مختلف، اگر تغییری نیز در یک کنترلر بر روی این books انجام شود، در بقیه هم منعکس می‌شود؟

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۱۷ ۱۰:۴۳

بله. اگر متدی را به کنترلر 2 اضافه کنید و سپس یک تگ لینک در view برای فراخوانی این متد، به صورت زیر :

```
sampleApp.controller('controller2', function ($scope, BookData) {
    $scope.books = BookData;
    $scope.change = function(){
         BookData[0].name = 'change this book';
    }
});
```

و view :

```
<div ng-controller="controller2">
        <p>Data from controller2</p>
        <table>
            <tr ng-repeat="book in books">
                <td>
                    {{book.code}}
                </td>
                <td>
                    {{book.name}}
                </td>
            </tr>
        </table>
<a href="javascript:void(0)" ng-click="change();" >Click here to change book1</a>
    </div>
```

میبینید که در کنترلر شماره 1 هم تغییر میکند.

عنوان: آشنایی با Directive ها در AngularJs
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۹/۱۹ ۸:۳۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

اگر مطالعه ای اجمالی درباره مزیت‌ها و قدرت‌های فریم ورک Angular داشته باشید یکی از مواردی که بسیار جلب توجه می‌کند مبحث Directiveها است. به کمک Directiveها در Angular می‌توانید کدهای HTML خود را توسعه دهید. این توسعه علاوه بر تعریف تگ‌های جدید، شامل توسعه کلاس‌ها و همچنین ویژگی‌های تگ‌های HTML نیز خواهد بود. کدهای HTML شما بسیار خواناتر و از طرفی با قابلیت استفاده مجدد می‌شود. البته این پست فقط شروع به کار در این مقوله است زیرا مبحث Directiveها بسیار گسترده‌تر از آن است که بتوان مطالب آن را در یک مقاله گنجاند.

برای شروع یک فایل جاوا اسکریپ ایجاد کرده و در ابتدای آن یک ماژول تعریف کنید:

```
var app = angular.module('myApp', []);
```

در این جا نام ماژول را myApp انتخاب کردم. حال یک Directive به نام angry (نام دیرکتیوها را با حروف کوچک آغاز کنید)به صورت زیر ایجاد می‌کنیم:

```
app.directive('angry', function () {
    return {
        restrict: 'E',
        template: '<div style="color:red"> I am angry!</div>'
    }
})
```

تابع سازنده Directive مورد نظر که یک آبجکت را برگشت می‌دهد شامل خواص زیر می‌باشد:

restrict: که چهار مقدار E و A و C و M را می‌پذیرد که به EACM نیز معروف هستند.

E: زمانی که قصد داشته باشیم یک المان جدید بسازیم از E به معنای element در restrict استفاده می‌کنیم(<my-directive></my-directive>):

A: زمانی که قصد داشته باشیم Directive مورد نظر به عنوان Attribute در تگ‌ها استفاده شود از A به معنای Attribute در restrict استفاده می‌شود( ):

C: از C نیز برای تعریف Directive به عنوان مقادیر ویژگی کلاس استفاده می‌کنیم( ):

M: حالت M نیز برای استفاده Directive در کامنت‌ها است(<!-- directive: my-directive exp -->).

در ادامه یک Directive دیگر به نام happy می‌سازیم:

```
app.directive('happy', function () {
    return {
        restrict: 'A',
        template: '<div style="color:blue"> I am happy!</div>'
    };
})
```

تفاوت اصلی بین این دو Directive در نوع restrict آن‌ها می‌باشد. برای استفاده از این Directiveها در View می‌توان به صورت زیر عمل نمود:

```
<script type="text/javascript" src="~/scripts/Modules/module1.js"></script>

<div ng-app="myApp">
    <angry></angry>
    <div happy></div>
</div>
```

همان طور که می‌بینید دستور Directive اول که را با restrict از نوع E است می‌توان به عنوان یک تگ جدید استفاده کرد.

دستور happy به عنوان ویژگی تگ div مورد استفاده قرار می‌گیرد(به دلیل اینکه restrict آن از نوع A است) که در نهایت خروجی ساده مثال بالا به صورت زیر خواهد بود:

ادامه دارد...

عنوان: **ارتباط بین Controller و Directive در AngularJs**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۹/۲۱ ۱۰:۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در پست قبلی با کلیات مفاهیم دیرکتیوها آشنا شدید. در این پست قصد داریم برخی توابع کنترلرهای تعریف شده در Angular را به وسیله دیرکتیوهای تعریف شده در ماژول فراخوانی نماییم. در ادامه این موضوع را طی یک مثال بررسی خواهیم کرد.
ابتدا View مورد نظر را به صور زیر ایجاد می‌کنیم:

```
<script type="text/javascript" src="~/scripts/Modules/module4.js"></script>

<div ng-app="myApp">
    <div ng-controller="myCtrl">
        <span enter>Load More Books</span>
    </div>
</div>
```

برنامه به این صورت است که با ورود نشانگر ماوس بر روی تگ span (فراخوانی رویداد mouseenter برای تگ هایی که دارای دیرکتیو enter باشند) یک تابع به نام loadMoreBook در کنترلر myCtrl فراخوانی می‌شود.
یک فایل جاوااسکریپتی به نام myModule بسازید و ماژول مورد نظر را ایجاد نمایید:

```
var app = angular.module('myApp', []);
```

کنترلر مورد نظر به همراه تابع loadMoreBook را به صورت زیر ایجاد می‌کنیم(البته در اینجا به جای لود واقعی داده از یک alert استفاده کردم):

```
app.controller('myCtrl', function ($scope) {
    $scope.loadMoreBook = function () {
        alert('Loading Books...');
    }
});
```

حال نوبت به دیرکتیو مورد بحث می‌رسد که به صورت زیر ایجاد می‌شود:

```
app.directive('enter', function () {
    return function (scope, element) {
        element.bind('mouseenter', function () {
            scope.loadMoreBook();
        })
    }
});
```

اولین نکته این است که به در تابع سازنده دیرکتیو به جای برگشت آبجک مورد نظر یک تابع برگشت داه می‌شود. برای اینکه بتوان به توابع کنترلر محصور کننده دیرکتیو دسترسی داشت آرگومان اول تابع معادل scope مورد استفاده در کنترلر خواهد بود. آرگومان دوم معادل المانی است که دارای دیرکتیو enter است. در این تابع ابتدا برای رویداد mouseenter رویدادگردان آن پیاده سازی شده است که در آن تابع loadMoreBook کنترلر مورد نظر فراخوانی می‌شود.

خروجی

حال فرض بر این است که در کنترلر بالا تابع دیگری به نام loadMoreAuthor برای فراخوانی نویسندگان نیز وجود دارد. به صورت زیر:

```
app.controller('myCtrl', function ($scope) {
    $scope.loadMoreBook = function () {
        alert('Loading Books...');
    }

    $scope.loadMoreAuthor = function () {
        alert('Loading Authors...');
    }
});
```

اما برای انعطاف پذیری بیشتر برنامه، قصد داریم دیرکتیو بالا را به گونه ای تغییر دهیم که نام تابع مورد نظر در کنترلر را به عنوان مقدار یک ویژگی دریافت کند. به صورت زیر:

```
<script type="text/javascript" src="~/scripts/Modules/module4.js"></script>

<div ng-app="myApp">
    <div ng-controller="myCtrl">
        <span enter="loadMoreBook()">Load More Book</span>
        <hr>
       <span enter="loadMoreAuthor()">Load More Author</span>
    </div>
</div>
```

برای به دست آوردن مقدار دیرکتیوی که به عنوان ویژگی در المان تعیین شده، باید از آرگومان سوم در تابع سازنده دیرکتیو به صورت زیر استفاده کرد.

```
app.directive('enter', function () {
    return function (scope, element , attrs) {
        element.bind('mouseenter', function () {
            scope.$apply(attrs.enter);
        })
    }
});
```

در کدهای بالا، برای اینکه بتوان بر اساس نام یک تابع آن را فراخوانی کرد، از سرویس $apply که به صورت توکار در angular تعبیه شده است استفاده کردم. برای به دست آوردن نام تابع، باید از آرگومان سوم تابع (attrs) به همراه نام دیرکتیو استفاده کرد. به دلیل اینکه نام دیرکتیو enter است باید پارامتر سرویس $apply به صورت attrs.enter باشد. خروجی نیز مانند حالت قبل خواهد بود.

عنوان: $inject در AngularJs
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۹/۲۳ ۸:۴۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

همان طور که در پست‌های قبلی ذکر شده بود در angular تزریق وابستگی به صورت پیش فرض وجود دارد. کافیست نام سرویس مورد نظر با نام‌های پیش فرض تعبیه شده در angular یا با نام سرویس‌های ساخته شده توسط خودتان مطابقت داشته باشد. به عنوان مثال برای تزریق سرویس $scope در توابع سازنده کنترلر کافیست یک پارامتر به همین نام را به عنوان آرگومان در این توابع در نظر بگیرید. همچنین برای استفاده از سرویس $http باید یک پارامتر دیگر به همین نام در این توابع در نظر داشته باشید و همچنین برای $location. در مورد سرویس‌های ساخته شده توسط خودتان نیز باید همین قانون را پیاده کنید. در این پست قصد دارم از injector تعبیه شده در angular برای تغییر رفتار فریم ورک در هنگام شناسایی پارامترهای توابع استفاده کنم.
ابتدا مثال زیر را به روش‌های قبلی پیاده سازی می‌کنیم:

```
var app = angular.module('myApp', []);

app.factory('bookService', function () {
    var books = [
        { name: 'A' },
        { name: 'B' },
        { name: 'C' }
    ];
    return books;
});
app.controller('bookCtrl', function ($scope, bookService) {
    $scope.books = bookService;
})
```

view مورد نظر نیز به صورت زیر خواهد بود:

```
<script type="text/javascript" src="~/scripts/Modules/module5.js"></script>

<div ng-app="myApp">
    <div ng-controller="bookCtrl">
        <table>
            <tr ng-repeat="book in books">
                <td>
                    {{book.name}}
                </td>
            </tr>
        </table>
    </div>
</div>
```

نیاز به توضیح نیست که در هنگام تعریف تابع سازنده کنترلر bookCtrl باید نام پارارمترهای ورودی تابع در هنگام تزریق وابستگی دقیقا مانند مثال بالا باشد. (یعنی $scope برای دسترسی به سرویس scope و bookService برای دسترسی به سرویس ساخته شده توسط factory - *ترتیب پارامترها در اینجا اهمیتی ندارد*). حال مثال بالا را با استفاده از injector موجود در angular برای تزریق وابستگی‌ها پیاده سازی می‌کنم. ابتدا تابع کنترلر bookCtrl را به صورت زیر ایجاد می‌کنیم:

```
var bookCtrl = function (sc,bs) {
    sc.books = bs;
};
```

از پارامتر sc به جای $scope و از bs به عنوان bookService در این تابع استفاده شده است. سپس کنترلر موجود را به ماژول مورد نظر نسبت می‌دهیم:

```
app.controller('bookCtrl',bookCtrl);
```

اگر برنامه را به همین صورت اجرا کنید خروجی مورد نظر حاصل نخواهد شد. زیرا آرگومان‌های sc و bs برای angular تعریف نشده است. کافیست وابستگی‌های تابع کنترلر را به صورت زیر برای angular مشخص نماییم:

```
bookCtrl.$inject = ['$scope','bookService'];
```

در نتیجه تعریف کنترلر بالا به صورت کامل زیر خواهد بود:

```
var app = angular.module('myApp', []);

app.factory('bookService', function () {
    var books = [
        { name: 'A' },
        { name: 'B' },
        { name: 'C' }
    ];
    return books;
});

var bookCtrl = function (sc,bs) {
    sc.books = bs;
};

bookCtrl.$inject = ['$scope','bookService'];

app.controller('bookCtrl',bookCtrl);
```

از این پس در هنگام فراخوانی تابع کنترلر bookCtrl سرویس‌های $scope و bookService به ترتیب به عنوان آرگومان‌های اول و دوم در اختیار کنترلر قرار می‌گیرند. می‌توان به جای فراخوانی مستقیم $inject، تزریق وابستگی‌ها را در هنگام تعریف توابع سازنده به صورت زیر نیز فراهم ساخت:

```
app.controller('bookCtrl', ['$scope', 'bookService', function (sc, bs) {
    sc.books = bs;
}])
```

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۲۴ ۲۳:۰

پس با این حساب AngularJS به minification اسکریپت‌ها حساس است. چون در حین minification تمام نام پارامترها با a و b و c جایگزین می‌شوند. در این مورد چه پیشنهادی وجود دارد؟

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۹/۲۴ ۲۳:۱۰

درسته.چون سیستم تزریق وابستگیش با نام متغیرها کار می‌کند با minification نام متغیرها تغییر می‌کند و در نتیجه برنامه از کار می‌افتد. راه‌های معین کردن صریح وابستگی‌ها در مقاله‌ی فوق ذکر شده.
روش اول:

```
bookCtrl.$inject = ['$scope','bookService'];
```

روش دوم:

```
app.controller('bookCtrl', ['$scope', 'bookService', function (sc, bs) {
    sc.books = bs;
}])
```

در این روش وابستگی‌های کنترلرها صریحا ذکر شده و با تغییر نام متغیرها انگولار می‌داند که چه وابستگی هایی را باید تزریق کند.

نویسنده: وحید
تاریخ: ۱۳۹۲/۱۰/۱۴ ۱۶:۲۳

لطفا توضیحی در مورد $Watch دهید ممنون

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۱:۱

به صورت کلی با استفاده از $watch می‌توان تمامی تغییراتی را که به خواص ViewModel اعمال می‌شوند مشاهده کرد.
تعریف کلی آن به صورت زیر است:

```
$watch(watchExpression, listener, objectEquality)
```

«watchExpression : می‌توان نام خاصیت مورد نظر در ViewModel یا یک تابع را که قصد مشاهده تغییرات آن را داریم تعیین کنیم.
«Listener : با تغییر در مقدار watchExpression اگر مقدار قبلی این عبارت با مقدار فعلی آن برابر نباشد این تابع فراخوانی می‌شود.
« objectEquality : به صورت پیش فرض Angular مقادیر مورد نظر برای تغییرات را فقط از نظر Reference Equal بودن چک می‌کند. اگر بخواهیم که Angular به صورت عمقی و درختی مقادیر ابجکت‌ها را بررسی کند مقدار این پارامتر باید true شود.

در فریم ورک Angular هر زمان که عمل مقید سازی خواص ViewModel به عناصر DOM انجام می‌گیرد در واقع یک نمونه از $watch به لیستی به نام $ watch list اضافه می‌شود. دقت کنید که صرفا تعریف در محدوده کنترلر کافی نیست بلکه باید خاصیت مورد نظر حتما مقید شود. برای مثال

```
app.controller('MainCtrl', function($scope) {
```

```
  $scope.foo = "Foo";
  $scope.world = "World";
});
```

در View نیز

```
Hello, {{ World }}
```

در کنترلر بالا دو خاصیت تعریف شده است، در حالی که در View فقط یک خاصیت مقید شده است. درنتیجه فقط یک $watch به لیست مورد نظر اضافه شده است.

و به عنوان نکته آخر، در Angular نسخه 1.1.4 تابعی به نام watchCollection اضافه شده است که برای ردیابی تغییرات یک مجموعه مورد استفاده قرار می‌گیرد.

یک مثال در این مورد

عنوان: **مسیریابی در AngularJs #بخش اول**
نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۲۳ ۱۸:۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در مطالب قبل کنترلرها و viewها مورد بحث قرار گرفتند. در این پست در نظر داریم یکی از ویژگی‌های دیگر AngularJs به نام مسیر یابی (Routing) را مورد بحث قرار دهیم.
یکی از ویژگی‌های برنامه‌های تک صفحه ای عدم Reload شدن صفحات است ،بر خلاف برنامه‌های وب چند صفحه ای که برای نمایش صفحه ای دیگر ، باید از صفحه ای به صفحه ای دیگر منتقل شد و عمل Reload هم به طبع نیز اتفاق می‌افتد.
در قسمت اول این سری مقالات ، مزایای برنامه‌های وب تک صفحه ای SPA به صورت کاملتری بیان شده است. در ادامه ما قصد داریم برنامه‌ی وب خود را به صفحات مختلف تقسیم کنیم و سپس با استفاده از امکان مسیریابی موجود در AngularJs آن صفحات را که هر کدام به کنترلری مجزا مقید شده اند، در صفحه‌ی اصلی خود بارگزاری کنیم. همچنین استفاده از مسیریابی موجود، میتواند به ما در مدیریت بهتر صفحات کمک فراوانی بکند.
به تصویر زیر دقت کنید :

در تصویر بالا دو مسیر با آدرس‌های : /ShowPage1 و /ShowPage2 تعریف شده است که هر کدام به یک view مشخص و یک Controller برای مدیریت آن اشاره میکند.

زمانی که ما از تزریق وابستگی‌ها در AngularJs استفاده میکنیم و یک شیء را به کنترلر تزریق میکنیم، $Injector توسط Angular سعی در پیدا کردن وابستگی مربوطه و سپس تزریق آن به کنترلر را انجام میدهد. برای استفاده از امکان مسیریابی Route ، ما نیز باید از پروایدر مخصوص آن برای تزریق استفاده کنیم. در Angular مسیرهای برنامه توسط پروایدری به نام $routeProvider شناسایی میشود که خدمات مسیریابی را به ما ارائه میدهد. این سرویس به ما کمک میکند تا بتوانیم اتصال بین کنترلر ها، ویوها و آدرس URL جاری مرورگرها را به آسانی برقرار کنیم.

بهتر است کار را شروع کنیم . یک فایل JS ایجاد و سپس محتویات زیر را در آن قرار دهید :

```
var myFirstRoute = angular.module('myFirstRoute', []);

myFirstRoute.config(['$routeProvider',
  function($routeProvider) {
    $routeProvider.
      when('/pageOne', {
        templateUrl: 'templates/page_one.html',
        controller: 'ShowPage1Controller'
```

```
      }).
      when('/pageTwo', {
        templateUrl: 'templates/page_two.html',
        controller: 'ShowPage2Controller'
      }).
      otherwise({
        redirectTo: '/pageOne'
      });
}]);


myFirstRoute.controller('ShowPage1Controller', function($scope) {
    $scope.message = 'Content of page-one.html';
});


myFirstRoute.controller('ShowPage2Controller', function($scope) {
    $scope.message = 'Content of page-two.html';
});
```

در کدهای بالا ابتدا یک ماژول تعریف کرده ایم و سپس توسط ()config. تنظیمات مربوط به مسیریابی را انجام داده ایم. با استفاده از متدهای when. و otherwise. میتوانیم مسیرها را تعریف کنیم. برای هر مسیر دو پارامتر وجود دارد که اولین پارامتر نام مسیر و دومین پارامتر شامل 2 قسمت میشود که templateUrl آن آدرسی که باز خواهد شد و controller نیز نام کنترلری که ویو را مدیریت میکند.

توسط otherwise میتوانیم مسیر پیشفرض را نیز تعریف کنیم تا درصورتی که مسیری با آدرس‌های بالای آن مطابقت نداشت به این آدرس منتقل شود.

در قطعه کد بالا همچنین دو مسیر با نام‌های pageOne/ و pageTwo/ تعریف کرده ایم که هر کدام به ترتیب به Viewهای : templates/page_one.html و templates/page_two.html مرتبط شده اند. همچنین دو کنترلر برای مدیریت ویوها نیز تعریف شده است.

زمانی که ما آدرس http://appname/#pageOne را در نوار آدرس مرورگر وارد میکنیم، Angular به صورت اتوماتیک آدرس URL را با تنظیماتی که ما در اینجا تعریف کرده ایم مطابقت میدهد و در صورت وجود چنین آدرسی ، view مربوطه را بارگزاری میکند و در این مثال نیز مطابق با تنظیمات بالا، صفحه‌ی templates/page_one.html برای ما بارگزاری و سپس کنترلر ShowPage1Controller را فراخوانی میکند ، جایی که ما منطق کار را در آن قرار میدهیم.

محتویات فایل main.html :

```
<body ng-app="app">

    <div>
        <div>
        <div>
            <ul>
                <li><a href="#pageOne"> Show page one </a></li>
                <li><a href="#pageTwo"> Show page two </a></li>
            </ul>
        </div>
        <div>
            <div ng-view></div>
        </div>
        </div>
    </div>

    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
    <script src="app.js"></script>

  </body>
```

در قطعه کد بالا دو لینک تعریف شده است که ویژگی href از علامت هش و نام صفحه تشکیل شده است. یکی از چیزهایی که

شایان ذکر است ، دایرکتیو ng-view است. مکانی برای بارگزاری صفحات در آن.

ما میتوانیم این تگ را به سه شکل زیر نیز استفاده کنیم :

```
<div ng-view></div>
..
<ng-view></ng-view>
..
<div class="ng-view"></div>
```

محتویات صفحه templates/page_one.html :

```
<h2>Page One</h2>

{{ message }}
```

محتویات صفحه templates/page_two.html :

```
<h2>Page Two</h2>

{{ message }}
```

حال اگر پروژه را اجرا کنید و به کنسول مرور گر خود نگاه کنید متوجه میشوید که مسیریاب از مسیر پیشفرض استفاده کرده است و صفحه‌ی page_one.html را به صورت ایجکسی فراخوانی کرده است :

و اگر روی لینک Show Page two کلیک کنید ، صفحه‌ی page_two.html نیز به صورت ایجکسی فراخوانی میشود.

دوباره بر روی لینک Show page one کلیک کنید. بله. هیچ درخواستی به سمت سرور ارسال نشد و صفحه‌ی page_one.html به خوبی نمایش داده شد. یکی از مزیت‌های سیستم مسیریابی قابلیت کش کردن صفحات است تا در صورت فراخوانی مجدد، درخواستی به سمت سرور ارسال نشود و خیلی سریع به شما نمایش داده شود.

مثال این مطلب : RouteExample.zip

ادامه دارد ...

نظرات خوانندگان

نویسنده: حسین
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۲:۵۹

سلام این نمونه کد در فایرفاکس به درستی کار میکنه ولی در گوگل کروم یا ie با خطاهای زیر مواجه میشم

```
1)OPTIONS file:///E:/Users/admin/Downloads/Compressed/RouteExample/RouteExample/templates/page_two.html
No 'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is
therefore not allowed access. angular.min.js:99


2)XMLHttpRequest cannot load
file:///E:/Users/admin/Downloads/Compressed/RouteExample/RouteExample/templates/page_two.html. No
'Access-Control-Allow-Origin' header is present on the requested resource. Origin 'null' is therefore
not allowed access.
```

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۳:۳۴

برای اجرا نیاز به سرور داره. ولی کروم برای تست با دستور خط فرمان Chrome.exe --allow-file-access-from-files چنین اجازه‌ای رو به شما می‌ده.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۳:۳۴

پروژه را تحت یک سرور مثل Apache یا IIS اجرا کنید. مشکل رفع میشود.

نویسنده: وحید م
تاریخ: ۱۳۹۲/۱۰/۰۵ ۲۳:۵۰

در mvc چطور میشود از حالت روتینگ استفاده کرد بگونه ای که مثلا بجای فایل html بتوان فایل cshtml جهت لود کردن partial view استفاده کرد. و اینکه چگونه مسیر دهی مثل views/home/_partial.cshtml داشته باشیم ممنون از شما

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۰۶ ۰:۶

نمیشه مستقیما. چون مسیر کاری MVC از اکشن متد و کنترلرها شروع میشه و نه از Viewها. Viewهای قرار گرفته در پوشه Views دارای web.config عدم دسترسی از خارج از سایت هستند. کدهای angularjs هم سمت کاربر هست و نه سمت سرور.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۱۰/۰۶ ۰:۷

در این کامنت هم توضیحاتی داده شده.

نویسنده: سعید رضایی
تاریخ: ۱۳۹۳/۰۲/۱۴ ۱۳:۴۶

به جای page_one.html چجوری میشه از cshtml استفاده کرد.
کلا کار درستیه که از cshtml فایل‌ها به templateUrl استفاده کرد یا نه؟
با تشکر

نویسنده: محسن درپرستی
تاریخ: ۱۳۹۳/۰۲/۱۴ ۱۸:۱

اگه بخواید از فایل‌های cshtml استفاده کنید باید از طریق یک اکشن باید اینکار رو بکنید. درست یا غلط بودنش به سناریویی که دارین بستگی داره.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۳/۰۲/۱۴ ۱۸:۵۶

یک مثال هم در این کامنت قرار داده شده است.(+)

نویسنده: رحمان
تاریخ: ۱۳۹۴/۰۲/۲۵ ۱۲:۱۰

با سلام،

آیا مشکل سئو استفاده از angular با سیستم مسیریابی که داره حل شده ؟ یعنی میشه ازشون برای اعمال سئو استفاده کرد؟

عنوان: **مسیریابی در AngularJs #بخش دوم**
نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۰۹/۲۳ ۲۳:۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در قسمت قبل با نحوه پیاده سازی مسیریابی در AngularJs آشنا شدیم و در این پست میخواهیم نحوه تعریف و ارسال پارامترها به سیستم مسیریاب را فرا بگیریم.

فرض کنید که میخواهیم در لیست سفارشات قسمتی داشته باشیم برای مشاهده‌ی جزئیات هر سفارش. پس در صفحه نمایش جزئیات کالا نیاز به کد محصول برای واکشی آن داریم. در Angular زمانی که داریم مسیرها را تعریف میکنیم این امکان را هم داریم که پارامترهایی را هم برای هر مسیر مشخص کنیم. برای این کار فایل app.js مثال قبل را باز کنید و مسیر ذیل را به آن اضافه کنید :

```
when('/showOrderDetails/:orderId', {
     templateUrl: 'templates/show_order.html',
     controller: 'ShowOrderController'
});
```

در بالا ما پارامتری به نام orderId وارد کرده ایم که میتوانیم توسط $routeParams در کنترلر به آن دست پیدا کنیم :

```
myFirstRoute .controller('ShowOrderController', function($scope, $routeParams) {
    $scope.order_id = $routeParams.orderId;
});
```

فراموش نکنید که باید پارامتر $routeParams را به کنترلر خود تزریق کنید.

محتوای فایل index.html را نیز به صورت زیر تغییر دهید :

```
<body ng-app="myFirstRoute" style="

    <div>
<div>
<div>
<table dir="rtl">
<thead>
  <tr>
<th>#</th><th>˜کد</th><th>نام محصول</th><th></th>
  </tr>
</thead>
<tbody>
  <tr>
<td>1</td><td>1234</td><td>15" Samsung Laptop</td>
<td><a href="#showOrderDetails/1234">جزئیات محصول</a></td>
  </tr>
  <tr>
<td>2</td><td>5412</td><td>2TB Seagate Hard drive</td>
<td><a href="#showOrderDetails/5412">جزئیات محصول</a></td>
  </tr>
  <tr>
<td>3</td><td>9874</td><td>D-link router</td>
<td><a href="#showOrderDetails/9874">جزئیات محصول</a></td>
  </tr>
</tbody>
  </table>

<div ng-view></div>
</div>
</div>
    </div>

<script src="js/bootstrap.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
    <script src="app.js"></script>

</body>
```

نکته‌ی مهم در کد بالا قرار دادن کد کالا بعد از مسیر است، مانند : showOrderDetails/5412 # و محتویات فایل templates/show_order.html :

```
<h2>سفارش شماره #{{order_id}}</h2>

محل قرار گیری جزئیات سفارش شماره : <b>#{{order_id}}</b>.
```

برنامه را اجرا کنید تا نتیجه را ببینید.

بارگزاری Viewهای محلی توسط تگ <script> :

در بعضی موارد لزومی ندارد که اطلاعات View را از یک فایل دیگر بخوانید و یا حتی اینقدر View شما کوچک است که تمایل دارید آن را به همراه فایل اصلی index.html حمل کنید به جای اینکه آن را در یک فایل جدا نگهداری کنید.

دایرکتیوی به نام ng-template وجود دارد که این امکان را به ما میدهد تا بتوانیم template Viewهای کوچکمان را در داخل فایل اصلی قرار دهیم. با استفاده از تگ <script> به شکل زیر میشود این کار را انجام داد :

```
<script type="text/ng-template" id="add_order.html">
    <h2> ثبت سفارش </h2>
    {{message}}
</script>
```

برای درک بهتر مثالی را تهیه میکنیم .

فایل app.js مثال قبل را باز کنید و مسیرهای زیر را نیز به آن اضافه کنید :

```
when('/AddNewOrder', {
    templateUrl: 'add_order.html',
    controller: 'AddOrderController'
}).
when('/ShowOrders', {
    templateUrl: 'show_orders.html',
    controller: 'ShowOrdersController'
});
```

سپس دو کنترلر زیر را نیز به آن اضافه کنید :

```
myFirstRoute.controller('AddOrderController', function($scope) {
$scope.message = 'صفحه نمایش ثبت سفارش جدید';
});


myFirstRoute.controller('ShowOrdersController', function($scope) {
$scope.message = 'صفحه نمایش لیست سفارشات';
});
```

فایلی به نام index2.html برای صفحه اصلی برنامه با محتوای زیر تعریف کنید :

```
<body ng-app="myFirstRoute" style="

    <div>
        <div>
        <div>
           <ul>
            <li><a href="#AddNewOrder"> ثبت سفارش جدید </a></li>
            <li><a href="#ShowOrders"> نمایش شفارشات </a></li>
            </ul>
        </div>
        <div>
            <div ng-view></div>
        </div>
        </div>
    </div>

    <script type="text/ng-template" id="add_order.html">

        <h2> ثبت سفارش </h2>
        {{message}}
```

```
    </script>

    <script type="text/ng-template" id="show_orders.html">

        <h2> نمایش سفارشات </h2>
        {{message}}

    </script>

<script src="js/bootstrap.js"></script>
    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
    <script src="app.js"></script>

</body>
```

همانطور که مشاهده میکنید در کد بالا از 2 تگ اسکریپت برای قرار دادن محتوای View استفاده کرده ایم که خاصیت type آن برابر با text/ng-template و خاصیت id آن نام View template است و دیگر فایل مجزایی برای Viewها ایجاد نکردیم. Angular به صورت خودکار محتوای داخل تگ‌های Script را به محض فراخوانی آدرس‌های موجود در ویژگی id هر تگ به وسیله‌ی سیستم مسیر یابی، در داخل دایرکتیو ng-view قرار میدهد.

پروژه را اجرا کنید تا نتیجه را مشاهده کنید.

افزودن داده‌های سفارشی به سیستم مسیریابی :

بیشتر اوقات ممکن است نیاز داشته باشید تا داده‌های خاصی را در مسیرهای معینی ارسال کنید. برای مثال ممکن است شما بخواهید از یک کنترلر در مسیرهای مختلف استفاده کنید و برای هر مسیر یک داده‌ی خاص را نیز ارسال میکنید. به مثال زیر توجه کنید :

```
when('/AddNewOrder', {
    templateUrl: 'templates/add_order.html',
    controller: 'CommonController',
    foodata: 'addorder'
}).
when('/ShowOrders', {
    templateUrl: 'templates/show_orders.html',
    controller: 'CommonController',
    foodata: 'showorders'
});

sampleApp.controller('CommonController', function($scope, $route) {
    //access the foodata property using $route.current
    var foo = $route.current.foodata;

    alert(foo);

});
```

در هر دو مسیر از کنترلر CommonController استفاده کرده ایم با این تفاوت که در مسیر اول یعنی AddNewOrder/ یک خاصیت با نام foodata با مقدار addorder تعریف شده است و در مسیر دوم با مقدار showorder.

ما میتوانیم با تزریق $route به کنترلرمان، توسط دستور :

```
$route.current.foodata
```

مقدار موجود در آن را بخوانیم.

نظرات خوانندگان

نویسنده: علی رضایی
تاریخ: ۱۳۹۳/۰۱/۰۵ ۲۰:۱۲

با سلام و تشکر فراوان جهت اشتراک دانسته هایتان؛
یک سوال:
در صورتی که بخواهیم بخش پایانی مقاله یعنی « **افزودن داده‌های سفارشی به سیستم مسیریابی** » را پیاده کنیم، همه چیز درست کارمیکند، اما اگر از controllerAs استفاده کنیم دیگر route.current$ در دستر نیست و undefiend میشود. شما به این مشکل برنخوردید؟
ممنون

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۳/۰۱/۰۶ ۰:۳

مشکلی رخ نداد.
1 - ماژول مسیریابی (ngRoute) رو باید تزریق کنید به ماژول اصلی :

```
var myFirstRoute = angular.module('myFirstRoute', ['ngRoute']);
```

2- $route رو به کنترلر هم تزریق کنید :

```
myFirstRoute.controller('ShowPage1Controller', function ($route) {
    this.message = 'Content of page-one.html';
    //access the foodata property using $route.current
    var foo = $route.current.foodata;
    alert(foo);
});
```

3- این هم یک نمونه از تنظیمات مسیریابی :

```
myFirstRoute.config(['$routeProvider',
  function($routeProvider) {
    $routeProvider.
      when('/pageOne', {
        templateUrl: 'templates/page_one.html',
        controller: 'ShowPage1Controller',
        controllerAs: 'tCtrlOne',
        foodata: 'valueOne'
      }).
      when('/pageTwo', {
        templateUrl: 'templates/page_two.html',
        controller: 'ShowPage2Controller',
        controllerAs: 'tCtrlTwo',
        foodata: 'valueTwo'
      }).
      otherwise({
        redirectTo: '/pageOne'
      });
}]);
```

البته اینها فقط برای تصحیح اشتباه است. وگر نه به ControllerAs ارتباطی ندارد. به احتمال زیاد مشکل شما عدم تزریق $route به کنترلر بوده.

عنوان: روش Controller as در AngularJs
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۹/۲۸ ۱۲:۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در پست‌های قبلی بیان شد که برای پیاده سازی عملیات مقید سازی عناصر View به مدل در کنترلر باید $scope را به تابع سازنده کنترلر تزریق کرد. برای مثال:

```
var app = angular.module('myApp', []);

app.controller('myController', function ($scope) {
    $scope.name = 'Masoud';
    $scope.family = 'Pakdel';
})
```

View متناظر نیز به صورت می‌باشد:

```
<div ng-app="myApp">
    <div ng-controller="myController">
        <div>
            {{name}} {{family}}
        </div>
    </div>
</div>
```

در Angular 1.2 روشی به نام controller as معرفی شده است که با توجه به نوع پیاده سازی آن نیازی به تزریق $scope در توابع سازنده نیست. فقط در کنترلر به جای وابستگی مستقیم به $scope ا زکلمه کلیدی this و در هنگام عملیات مقید سازی باید از نام مستعار تعیین شده برای کنترلر استفاده نمایید. برای مثال

```
var app = angular.module('myApp', []);

app.controller('myController', function () {
    this.name = 'Masoud';
    this.family = 'Pakdel';
})
```

و استفاده آن در View

```
<div ng-app="myApp">
    <div ng-controller="myController as myCtrl">
        <div>
            {{myCtrl.name}} {{myCtrl.family}}
        </div>
    </div>
</div>
```

در هنگام عملیات routing نیز می‌توان این عناوین مستعار را برای کنترلر با استفاده از controllerAs مشخص نمود. به صورت زیر:

```
app.config(function($routeProvider){
    $routeProvider.when('/first', {
            templateUrl: 'first.html',
            controller: 'FirstCtrl',
            controllerAs:'fc' })
        .when('/second', {
            templateUrl: 'second.html' ,
            controller: 'SecondCtrl',
            controllerAs:'sc' })
        .otherwise({
            redirectTo: '/first'
       });
});
```

نظرات خوانندگان

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۲/۰۹/۲۸ ۱۶:۴۵

ممنون؛ می‌خواستم بپرسم به جز syntax استفاده مزیت دیگری نیز دارد؟ چون به شخصه استفاده از $scope را بیشتر می‌پسندم.

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۹/۲۸ ۲۱:۲۷

خیر. در مجموع نمی‌توان تفاوتی خاص بین این دو روش برشمرد. $scope روشی کلاسیک است در حالی که controller as در نسخه جدید پشتیبانی می‌شود. مهم‌ترین مزیت روش controler as عدم تزریق $scope به تابع سازنده کنترلر است؛ اما علاوه بر این به نظر من روش controller as به دلیل استفاده از this که تقریبا تمام برنامه نویسان جاوااسکریپ با آن آشنایی دارند روشی تمیزتر است. البته بد نیست که نگاهی هم به (^) داشته باشید.

عنوان: ارث بری کنترلرها در AngularJs
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۹/۲۸ ۱۵:۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در Angular مکانیزمی وجود دارد که بر اساس آن می‌توان از توابع و خواص تعریف شده در یک کنترلر در سایر کنترلرها نیز استفاده کرد که در واقع از ان به عنوان ارث بری کنترلرها عنوان می‌شود؛ ولی نکته ای که وجود دارد این است که در جاوااسکریپ OOP پشتیبانی نمی‌شود پس چگونه یک آبجکت کنترلر توابع و خصوصیات کنترلر دیگر را به ارث می‌برد؟ با ذکر یک مثال این مورد را بررسی خواهیم کرد.

ابتدا دو کنترلر به صورت زیر ایجاد می‌کنیم:

```
var app = angular.module('myApp', []);

app.controller('parentController', function () {
    this.title = 'Title from parent controller';
})

app.controller('childController', function () {
    this.title = 'Title from child controller';
})
```

در کدهای بالا دو کنترلر به نام parentController و childController ایجاد کردم. از parentController به عنوان کنترلر والد استفاده خواهد شد و قصد داریم که از title آن در کنترلر child استفاده نماییم. View متناظر را به صورت زیر ایجاد خواهیم کرد.

```
<div ng-app="myApp">
    <div ng-controller="parentController as parentCtrl">
        <div ng-controller="childController as childCtrl">
            <input type="text" ng-model="parentCtrl.title" size="100"/>
            <input type="text" ng-model="childCtrl.title" size="100"/>
        </div>
    </div>
</div>
```

اولین نکته این است که تگ div تعریف شده برای کنترلر child در محدوده تگ div کنترلر parent قرار گرفته است. دومین نکته این است که از روش controller as برای مقید سازی خواص به المان‌های صفحه استفاده شده است. اگر برنامه را اجرا نمایید خروجی زیر ار مشاهده خواهید کرد.

هر دو کنترلر دارای یک title مجزا هستند ولی با استفاده از یک اشاره گر توانستیم این دو title را تفکیک نماییم.
حال برای دسترسی به title کنترلر parent در سایر کنترلرها کافیست از نام مستعار parentCtrl استفاده نماییم. از آن جا که محدوده کنترلر child در داخل محدوده کنترلر parent است دسترسی به کنترلر parent امکان پذیر می‌باشد.

```
<div ng-app="myApp">
    <div ng-controller="parentController as parentCtrl">
        <div ng-controller="childController as childCtrl">
            <input type="text" ng-model="parentCtrl.title" size="100"/>
            <input type="text" ng-model="parentCtrl.title" size="100"/>
        </div>
    </div>
</div>
```

خروجی:

File Edit View Favorites Tools Help

Title from parent controller | Title from parent controller

نکته: دسترسی به کنترلر child در کنترلر parent امکان پذیر نیست.

عنوان: **پیاده سازی کنترلرهای Angular با استفاده از Typescript**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۰۸ ۲۲:۵
آدرس: www.dotnettips.info
گروه‌ها: TypeScript, AngularJS

پیشتر با ویژگی ها و نحوه کد نویسی این زبان آشنا شدید. از طرفی دیگر، نحوه تعریف کنترلرها در Angular نیز آموزش داده شد . در این پست قصد دارم طی یک مثال ساده با استفاده از زبان Typescript یک کنترلر Angular را ایجاد و سپس از آن در یک پروژه Asp.Net MVC استفاده نمایم. از آن جا که به صورت پیش فرض در VS.Net امکانات TypeScript نصب نشده است، برای شروع ابتدا TypeScript را از اینجا دانلود نمایید. بعد از نصب یک پروژه Asp.Net MVC ایجاد نمایید و سپس با استفاده از nuget فایل‌های مربوط به AngularJs را نصب نمایید. در این پست به تفصیل این مورد بررسی شده است (عملیات BundleConfig فایل‌های مورد نیاز به عهده خودتان). در پوشه scripts یک فولدر به نام app ساخته، سپس یک فایل TypeScript به نام ProductController.ts ایجاد کنید. (بعد از نصب TypeScript گزینه TypeScript File مشاهده خواهد شد)

در فایل ProductController.ts کدهای زیر را کپی نمایید:

```
module Product {
    export interface Scope {
        message: string;
    }

    export class Controller {
        constructor($scope: Scope) {
            $scope.message = "Hello from Masoud";
        }
    }
}
```

توضیح کدها بالا :

ابتدا یک ماژول به نام Product ایجاد می‌کنیم. سپس یک اینترفیس برای پیاده سازی آبجکت Scope که جهت مقید سازی عناصر DOM به آبجکت‌های کنترلر مورد استفاده قرار می‌گیرد، ایجاد می‌کنیم. در داخل این اینترفیس متغیری به نام message از نوع string داریم. قصد داریم این متغیر را به یک عنصر مقید کنیم. حال یک کلاس به نام کنترلر ایجاد می‌کنیم که در تابع سازنده آن تزریق

وابستگی برای $scope از نوع اینترفیس Scope تعیین شده است. در نتیجه در بدنه سازنده می‌توانیم به متغیر message مقدار مورد نظر را نسبت دهیم .

کلمه کلیدی export برای تعریف عمومی کلاس استفاده شده است .
یک View ایجاد و کدهای زیر را در آن کپی کنید :

```
<script type="text/javascript" src="~/scripts/app/ProductController.js"></script>
<div ng-app>
    <div ng-controller="Product.Controller">
        <p>{{message}}</p>
    </div>
</div>
```

اولین نکته در تگ script است که فراخوانی فایل TypeScript باید با پسوند js. انجام گیرد. به دلیل اینکه فایل‌های TypeScript بعد از کامپایل تبدیل به فایل‌های JavaScript خواهند شد؛ در نتیجه پسوند آن نیز js. است. دومین نکته در فراخوانی کنترلر مورد نظر است که از ترکیب نام ماژول و نام کلاس است. بعد از اجرای پروژه خروجی به صورت زیر خواهد بود :

نظرات خوانندگان

نویسنده: ایلیا اکبری فرد
تاریخ: ۱۳۹۲/۱۰/۲۵ ۱۵:۲۶

با سلام.
در کنترلر ، چگونه watch$ مربوط به شی scope$ را بوسیله TypeScript میتوان فراخوانی کرد؟

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۲۷ ۱۳:۷

ابتدا کنترلر خود را به صورت زیر تعریف کنید:

```
class MyController {
        thescope: any;
        static $inject = ['$scope'];
```

```
constructor($scope) {
              this.thescope = $scope;
```

```
this.thescope.$watch('watchtext', function(newValue, oldValue) {
              this.thescope.counter = scope.counter + 1;
              this.thescope.lastvalue = oldValue;
              this.thescope.currentvalue = newValue;});
```

```
}
    }
```

حال برای استفاده از کنترلر بالا به صورت زیر عمل نمایید:

```
module myApp.ctrl {
    myApp.controller("MyController", function($scope) {
        return new MyController($scope);
    }
}
```

نویسنده: محمد آزاد
تاریخ: ۱۳۹۴/۰۱/۲۹ ۱:۲۲

من از ورژن angular 1.3.15 استفاده میکنم و در صفحه یک چنین اروری میده Error: [ng:areq]
http://errors.angularjs.org/1.3.15/ng/areq?p0=Product.Controller&p1=not%20a%20function%2C%20got%20undefined
مثل اینکه نوع تعریف controller در این ورژن متفاوت با قبله.چطور میشه کد typescript رو اصلاح کرد تا مشکل حل بشه؟

عنوان: واکشی اطلاعات سرویس Web Api با استفاده از TypeScript و AngularJs
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۰۹ ۸:۴۵
آدرس: www.dotnettips.info
گروه‌ها: TypeScript, AngularJS, Web API

در پست‌های قبلی با AngularJs ، TypeScript و Web Api آشنا شدید. در این پست قصد دارم از ترکیب این موارد برای پیاده سازی عملیات واکشی اطلاعات سرویس Web Api در قالب یک پروژه استفاده نمایم. برای شروع ابتدا یک پروژه Asp.Net MVC ایجاد کنید.

در قسمت مدل ابتدا یک کلاس پایه برای مدل ایجاد خواهیم کرد:

```
public abstract class Entity
    {
        public Guid Id { get; set; }
    }
```

حال کلاسی به نام Book ایجاد می‌کنیم:

```
public class Book : EntityBase
    {
        public string Name { get; set; }
        public decimal Author { get; set; }
    }
```

در پوشه مدل یک کلاسی به نام BookRepository ایجاد کنید و کدهای زیر را در آن کپی نمایید(به جای پیاده سازی بر روی بانک اطلاعاتی، عملیات بر روی لیست درون حافظه انجام می‌گیرد):

```
public class BookRepository
    {
        private readonly ConcurrentDictionary<Guid, Book> result = new ConcurrentDictionary<Guid,
Book>();

        public IQueryable<Book> GetAll()
        {
            return result.Values.AsQueryable();
        }

        public Book Add(Book entity)
        {
            if (entity.Id == Guid.Empty) entity.Id = Guid.NewGuid();

            if (result.ContainsKey(entity.Id)) return null;

            if (!result.TryAdd(entity.Id, entity)) return null;

            return entity;
        }
    }
```

نوبت به کلاس کنترلر می‌رسد. یک کنترلر Api به نام BooksController ایجاد کنید و سپس کدهای زیر را در آن کپی نمایید:

```
public class BooksController : ApiController
    {
        public static BookRepository repository = new BookRepository();

public BooksController()
        {
            repository.Add(new Book
            {
                Id=Guid.NewGuid(),
                Name="C#",
                Author="Masoud Pakdel"
            });

            repository.Add(new Book
```

```
            {
                Id = Guid.NewGuid(),
                Name = "F#",
                Author = "Masoud Pakdel"
            });

            repository.Add(new Book
            {
                Id = Guid.NewGuid(),
                Name = "TypeScript",
                Author = "Masoud Pakdel"
            });
        }

        public IEnumerable<Book> Get()
        {
            return repository.GetAll().ToArray();
        }
    }
```

در این کنترلر، اکشنی به نام Get داریم که در آن اطلاعات کتاب‌ها از Repository مربوطه برگشت داده خواهد شد. در سازنده این کنترلر ابتدا سه کتاب به صورت پیش فرض اضافه می‌شود و انتظار داریم که بعد از اجرای برنامه، لیست مورد نظر را مشاهده نماییم.

حال نوبت به عملیات سمت کلاینت میرسد. برای استفاده از قابلیت‌های TypeScript و AngularJs در Vs.Net از این مقاله کمک بگیرید. بعد از آماده سازی در فولدر script، پوشه ای به نام app می‌سازیم و یک فایل TypeScript به نام BookModel در آن ایجاد می‌کنیم:

```
module Model {
    export class Book{
        Id: string;
        Name: string;
        Author: string;
    }
}
```

واضح است که ماژولی به نام Model داریم که در آن کلاسی به نام Book ایجاد شده است. برای انتقال اطلاعات از طریق سرویس $http در Angular نیاز به سریالایز کردن این کلاس به فرمت Json خواهیم داشت. قصد داریم View مورد نظر را به صورت زیر ایجاد نماییم:

```
<div ng-controller="Books.Controller">
        <table class="table table-striped table-hover" style="width: 500px;">
            <thead>
                <tr>
                    <th>Name</th>
                    <th>Author</th>
                </tr>
            </thead>
            <tbody>
                <tr ng-repeat="book in books">
                    <td>{{book.Name}}</td>
                    <td>{{book.Author}}</td>
                </tr>
            </tbody>
        </table>
    </div>
```

توضیح کدهای بالا:

ابتدا یک کنترلری که به نام Controller که در ماژولی به نام Book تعریف شده است باید ایجاد شود. اطلاعات تمام کتب ثبت شده باید از سرویس مورد نظر دریافت و با یک ng-repeat در جدول نمایش داده خواهند شود.

در پوشه app یک فایل TypeScript دیگر برای تعریف برخی نیازمندی‌ها به نام AngularModule ایجاد می‌کنیم که کد آن به صورت زیر خواهد بود:

```
declare module AngularModule {
    export interface HttpPromise {
        success(callback: Function) : HttpPromise;
    }
    export interface Http {
        get(url: string): HttpPromise;
    }
}
```

در این ماژول دو اینترفیس تعریف شده است. اولی به نام HttpPromise است که تابعی به نام success دارد. این تابع باید بعد از موفقیت آمیز بودن عملیات فراخوانی شود. ورودی آن از نوع Function است. یعنی اجازه تعریف یک تابع را به عنوان ورودی برای این توابع دارید.

در اینترفیس Http نیز تابعی به نام get تعریف شده است که برای دریافت اطلاعات از سرویس api، مورد استفاده قرار خواهد گرفت. از آن جا که تعریف توابع در اینترفیس فاقد بدنه است در نتیجه این جا فقط امضای توابع مشخص خواهد شد. پیاده سازی توابع به عهده کنترلرها خواهد بود:

مرحله بعد مربوط است به تعریف کنترلری به نام BookController تا اینترفیس بالا را پیاده سازی نماید. کدهای آن به صورت زیر خواهد بود:

```
/// <reference path='AngularModule.ts' />
/// <reference path='BookModel.ts' />

module Books {
    export interface Scope {
        books: Model.Book[];
    }

    export class Controller {
        private httpService: any;

        constructor($scope: Scope, $http: any) {
            this.httpService = $http;

            this.getAllBooks(function (data) {
                $scope.books = data;
            });
            var controller = this;
    }

        getAllBooks(successCallback: Function): void {
            this.httpService.get('/api/books').success(function (data, status) {
                successCallback(data);
            });
        }
    }
}
```

توضیح کدهای بالا:

برای دسترسی به تعاریف انجام شده در سایر ماژول‌ها باید ارجاعی به فایل تعاریف ماژول‌های مورد نظر داشته باشیم. در غیر این صورت هنگام استفاده از این ماژول‌ها با خطای کامپایلری روبرو خواهیم شد. عملیات ارجاع به صورت زیر است:

```
/// <reference path='AngularModule.ts' />
/// <reference path='BookModel.ts' />
```

در پست قبلی توضیح داده شد که برای مقید سازی عناصر بهتر است یک اینترفیس به نام Scope تعریف کنیم تا بتوانیم متغیرهای مورد نظر برای مقید سازی را در آن تعریف نماییم در این جا تعریف آن به صورت زیر است:

```
export interface Scope {
        books: Model.Book[];
    }
```

در این جا فقط نیاز به لیستی از کتاب‌ها داریم تا بتوان در جدول مورد نظر در View آنرا پیمایش کرد. تابعی به نام getAllBooks در کنترلر مورد نظر نوشته شده است که ورودی آن یک تابع خواهد بود که باید بعد از واکشی اطلاعات از سرویس، فراخوانی شود.
اگر به کدهای بالا دقت کنید می‌بینید که در ابتدا سازنده کنترلر،سرویس $http موجود در Angular به متغیری به نام httpService نسبت داده می‌شود. با فراخوانی تابع get و ارسال آدرس سرویس که با توجه به مقدار مسیر یابی پیش فرض کلاس WebApiConfig باید با api شروع شود به راحتی اطلاعات مورد نظر به دست خواهد آمد. بعد از واکشی در صورت موفقیت آمیز بودن عملیات تابع success اجرا می‌شود که نتیجه آن انتساب مقدار به دست آمده به متغیر books تعریف شده در $scope می‌باشد.

در نهایت خروجی به صورت زیر خواهد بود:

File Edit View Favorites Tools Help

Name	Author
C#	Masoud Pakdel
TypeScript	Masoud Pakdel
F#	Masoud Pakdel

سورس پیاده سازی مثال بالا در Visual Studio 2013

نظرات خوانندگان

نویسنده: sadegh hp
تاریخ: ۱۳۹۲/۱۲/۲۳ ۱۱:۳۳

چجوری میشه با jasmine یک تست برای متدی که $http.post رو در یک سرویس انگولار پیاده کرده نوشت؟ تست متدهای async در انگولار چجوریه ؟

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۲۳ ۱۳:۱

angularJs کتابخانه ای برای mock آبجکت‌ها خود تهیه کرده است.(angular-mock) . از آن جا که در angular مبحث تزریق وابستگی بسیار زیبا پیاده سازی شده است با استفاده از این کتابخانه می‌توانید آبجکت‌های متناظر را mock کنید. برای مثال:

```
describe('myApp', function() {
var scope;

 beforeEach(angular.mock.module('myApp'));
 beforeEach(angular.mock.inject(function($rootScope) {
    scope = $rootScope.$new();
});
it('...')
});
```

هم چنین برای تست سرویس $http و شبیه سازی عملیات request و response در انگولار سرویس $httpBackend تعبیه شده است که یک پیاده سازی Fake از $http است که در تست‌ها می‌توان از آن استفاده کرد. برای مثال:

```
describe('Remote tests', function() {
    var $httpBackend, $rootScope, myService;
       beforeEach(inject(
function(_$httpBackend_, _$rootScope_, _myService_) {
      $httpBackend = _$httpBackend_;
      $rootScope = _$rootScope_;
      myService = _myService_;
}));
it('should make a request to the backend', function() {
   $httpBackend.expect('GET', '/v1/api/current_user')
     .respond(200, {userId: 123});
     myService.getCurrentUser();

     $httpBackend.flush();
});
});
```

دستور $.expect httpBackend برای ایجاد درخواست مورد نظر استفاده می‌شود که نوع verb را به عنوان آرگومان اول دریافت می‌کند. respond نیز مقدار بازگشتی مورد انتظار از سرویس مورد نظر را بر میگرداند. می‌توانید از دستورات زیر برای سایر حالات استفاده کنید:

«$.expectGet httpBackend
«$.expectPut httpBackend
«$.expectPost httpBackend
«$.expectDelete httpBackend
«$.expectJson httpBackend
«$.expectHead httpBackend
«$.expectPatch httpBackend

Flush کردن سرویس $httpBackend در پایان تست نیز برای همین مبحث async اجرا شدن سرویس‌های $backend http است.

نویسنده: صادق اچ پی
تاریخ: ۱۳۹۲/۱۲/۲۵ ۹:۴۸

ممنون از پاسخ شما.
اما سوال بعد اینکه چرا اصلا باید بیرون از سرویس http رو ساخت؟ فرض کنید که ما دسترسی به محتوی متود درون سرویس نداریم و فقط میخواهیم اون رو صدا کنیم و ببینیم که متود درون سرویس درست کار میکنه یا نه! بدون اینکه بدونیم چجوری داخل متود پیاده سازی شده که در این مورد یک http.post یا get هست.

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۲۵ ۱۰:۴۳

$httpBackend یک پیاده سازی fake از $http است، در نتیجه می‌توانید در هنگام تست، این سرویس را به کنترلرهای خود تزریق کنید. اما قبل از DI باید برای این سرویس مشخص شود که برای مثال در هنگام مواجه شدن با یک درخواست از نوع Get و آدرس X چه خروجی برگشت داده شود. درست شبیه به رفتار mocking framework ها. فرض کنید شما کنترلری به شکل زیر دارید:

```
(function (module) {

    var myController = function ($scope, $http) {

        $http.get("/api/myData")
            .then(function (result) {
                $scope.data= result.data;
            });
    };

    module.controller("MyController",
        ["$scope", "$http", myController]);

}(angular.module("myApp")));
```

همان طور که می‌بینید در این کنترلر از $http استفاده شده است. حال برای تست آن می‌توان نوشت:

```
describe("myApp", function () {

    beforeEach(module('myApp'));

    describe("MyController", function () {

        var scope, httpBackend;
        beforeEach(inject(function ($rootScope, $controller, $httpBackend, $http) {
            scope = $rootScope.$new();
            httpBackend = $httpBackend;
            httpBackend.when("GET", "/api/myData").respond([{}, {}, {}]);
            $controller('MyController', {
                $scope: scope,
                $http: $http
            });
        }));

        it("should have 3 row", function () {
            httpBackend.flush();
            expect(scope.data.length).toBe(3);
        });
    });
});
```

httpBackend ساخته شده با استفاده از سرویس $controller به کنترلر مورد نظر تزریق می‌شود. حال اگر در یک کنترلر 5 بار از سرویس $http برای فراخوانی 5 resource متفاوت استفاده شده باشد باید برای هر حالت $httpBackend را طوری تنظیم کرد که بداند برای هر منبع چه خروجی در اختیار کنترلر قرار دهد.

عنوان: **تفاوت AngularJS با KnockoutJS**
نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۱۰/۱۰ ۱۹:۳۵
آدرس: www.dotnettips.info
گروه‌ها: KnockoutJS, AngularJS, DurandalJS, SPA

با پیشرفت HTML 5 و پدید آمدن چارچوب‌های مختلف JavaScript توسعه‌ی نرم افزارهای تک صفحه ای تحت وب (Single Page Applications) محبوب شده است. اخیرا مطالب خوبی در رابطه با AngularJS در وبسایت جاری منتشر شده است.
KnockoutJS توسط Microsoft معرفی شد و در قالب پیشفرض پروژه‌های SPA قرار گرفت ، بنابراین احتمالا این سوال برای افرادی مطرح شده است که تفاوت بین KnockoutJS و AngularJS چیست ؟
می توان پاسخ داد این مقایسه ممکن نیست.

KnockoutJS : یک پیاده سازی مستقل JavaScript از الگوی MVVM با امکانات Databinding می‌باشد. Knokcout یک کتابخانه‌ی Databinding است نه یک کتاب خانه‌ی SPA
AngularJS : طبق معرفی در این مطلب AngularJS فریم ورکی متن باز و نوشته شده به زبان جاوا اسکریپت است. هدف از به وجود آمدن این فریم ورک، توسعه هر چه ساده‌تر SPAها با الگوی طراحی MVC و تست پذیری هر چه آسان‌تر آن‌ها است. این فریم ورک توسط یکی از محققان Google در سال 2009 به وجود آمد. بعدها این فریم ورک تحت مجوز MIT به صورت متن باز در آمد و اکنون گوگل آن را حمایت می‌کند و توسط هزاران توسعه دهنده در سرتاسر دنیا، توسعه داده می‌شود.

بنابراین شاید بهتر باشد ذکر شود AngularJS یک Presentation Framework مخصوص برنامه‌های وب تک صفحه ای می‌باشد در حالی که KnockoutJS کتاب خانه ای با تمرکز بر Databinding می‌باشد ، بنابراین مقایسه‌ی این‌ها چندان صحیح نیست.

اگر قصد بر بررسی گزینه‌های دیگر در کنار Angular باشد ، می‌توان از Durandal نام برد. Durandal یک چارچوب SPA می‌باشد ، این چارچوب بر فراز RequireJS ، jQuery و Knockout توسعه پیدا کرده است. (سابقا برای routing از SammyJS استفاده می‌کرد که در نسخه‌های اخیر از موتور خودش استفاده می‌کند.)

Durandal از Knockout جهت Databinding و از RequireJS برای مدیریت وابستگی‌ها استفاده می‌کند.
Angular همه‌ی امکانات بالا را مستقل پیاده سازی کرده و حتی نیازی به jQuery ندارد. اگر jQuery وجود داشته باشد Angular از آن استفاده می‌کند در غیر این صورت از jQuery Lite یا jqLite استفاده می‌کند. jqLite پیاده سازی توابع متداول jQuery برای دستکاری DOM می‌باشد. اطلاعات بیشتر در اینجا

بنابراین با استفاده تنها از KnockoutJS نمی‌توان یک برنامه‌ی کامل SPA توسعه داد ، در کنار آن نیاز به کتابخانه‌های دیگری مثل jQuery برای مدیریت درخواست‌های AJAX و استفاده از دیگر APIها ، Sammy برای routing و RequireJS برای مدیریت وابستگی‌ها می‌باشد.

در Knockout و در نتیجه Durandal عمل Databinding به این صورت است :

```
// JavaScript
var vm = {
    firstName = ko.observable('John')
};
ko.applyBindings(vm);
```

```
<!-- HTML -->
<input data-bind="value:firstName"/>
```

در Angular :

```
// JavaScript
// Inside of a personController
this.firstName = 'John';
```

در Angular همچنین از یک روش Controller As استفاده می‌شود :

```
<!-- HTML -->
<div ng-controller="personController as vm">
    <input ng-model="vm.firstName"/>
</div>
```

اگر تنها نیاز به یک کتابخانه‌ی Databinding باشد ، Knockout گزینه‌ی مناسبی است ، به خوبی از عمل مقید سازی داده‌ها پشتیبانی می‌کند و Syntax خوش دستی دارد اما اگر نیاز به چارچوبی برای توسعه‌ی پروژه‌های SPA می‌باشد می‌توان از Angular یا Durandal استفاده کرد.

مقایسه‌ی Knockout با Angular همانند مقایسه‌ی موتور بنز با ماشین پورشه می‌باشد.

مطالعه‌ی بیشتر

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۱۱ ۱:۱۱

برای مطالعه بیشتر: سری 8 قسمتی AngularJS vs Knockout

نویسنده: mohammad sepahvand
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۰:۲۹

به نظر من مقایسه angular و knockout آنقدر هم احمقانه نیست. اگر بخواهیم فقط هم از data binding استفاده کنیم angular خیلی از knockout خوش دست‌تر و ساده‌تر است. تازگی angular بیشتر modular شده و بنابراین مقایسه این دو مانند مقایسه موتور بنز با خود پورشه نیست، چون اگر تنها نیازمان data-binding است لزومی ندارد از moduleهای دیگر angular مانند ng-animate, ng-route..... استفاده کنیم و حتی نیازی نیست آن اسکریپتهارا در پروژه خود include کنیم.

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۰:۴۳

در واقع زمانی که تنها از ماژول Data binding استفاده می‌شود یعنی به عنوان مثال تنها از موتور بنز استفاده شده .

نویسنده: خیام
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۳:۴۳

حالا که زحمت مقایسه AngularJs و knockout رو انجام دادین ، بهتر بود Angular رو با یک فریم ورک قویتری مثل Ember مقایسه کنید و از این دو سخن بگید ؟ نظر شما در مورد این دو چی هست ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۷:۳

فاکتورهایی را که باید حین انتخاب یک فریم‌ورک JavaScript MVC در نظر داشت

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۲/۱۱/۲۱ ۱۸:۲۱

مقایسه از این قبیل زیاد است
اگر نگاهی به جامعه کاربری استفاده کننده کنیم به طور مثال در Stackoverflow با تگ Angular حدود 25 هزار سوال پرسیده شده در حالی که با تگ Backbone حدود 14 هزار سوال پرسیده شده.
Angular امکانات کاملی برای توسعه‌ی SPA در بر دارد.

نویسنده: سعید رضایی
تاریخ: ۱۳۹۲/۱۲/۲۰ ۱۶:۴۳

با عرض سلام.
angularjs با مرورگر ie 11 به پایین مشکل داره..

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۰ ۱۷:۰

خیر؛ با IE 9 به بعد مشکلی ندارد. با IE8 هم کار می‌کند ولی یک سری نکات خاص خودش را دارد. اطلاعات بیشتر را در مستندات رسمی آن در مورد IE مطالعه کنید.

عنوان: آشنایی با Promises در جاوا اسکریپت
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۱۲ ۸:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, jQuery Ajax, AngularJS

در حین انجام اعمال غیرهمزمان جاوا اسکریپتی مانند فراخوانی‌های jQuery AJAX، برای مدیریت دریافت نتایج، عموما از یک سری callback استفاده می‌شود. برای مثال:

```
$.get('http://site-url', function(data) {
//این تابع پس از پایان کار عملیات ای‌جکسی در آینده فراخوانی خواهد شد
});
```

تا اینجا مشکلی به نظر نمی‌رسد. اما مورد ذیل چطور؟

```
$.get('http://site-url/0', function(data0) {
    // callback #1
    $.get('http://site-url/1', function(data1) {
        // callback #2
        $.post('http://site-url/2', function(data2) {
            // callback #3
        });
    });
});
```

در اینجا نیاز است پس از پایان کار عملیات Ajax ایی اول، عملیات دوم و پس از آن عملیات سومی انجام شود. همانطور که مشاهده می‌کنید، این نوع کدها به سرعت از کنترل خارج می‌شوند؛ خوانایی پایینی داشته و مدیریت استثناءهای رخ داده در آن‌ها نیز در این بین مشکل است. از این جهت که خطاهای هر کدام به سطحی بالاتر منتقل نمی‌شود و باید همانجا محلی و داخل هر callback مدیریت گردد.

روش‌های زیادی برای حل این مساله ارائه شده‌است و در حال حاضر کار کردن با promiseها متداول‌ترین روش حل مدیریت فراخوانی کدهای همزمان جاوا اسکریپتی است. برای نمونه اگر از AngularJS استفاده کنید، سرویس‌های آن برای دریافت اطلاعات از سرور، از یک چنین مفهومی استفاده می‌کنند.

Promise در جاوا اسکریپت چیست؟

شیء Promise، نمایانگر قراردادی است که در آینده می‌تواند مورد قبول واقع شود، یا رد گردد. بررسی این قرارداد، تنها یکبار می‌تواند رخ دهد (پذیرش یا رد آن). هنگامیکه این بررسی صورت گرفت (رد یا پذیرش آن و نه هردو)، یک callback برای اطلاع رسانی فراخوانی می‌گردد. سپس این callback می‌تواند یک Promise دیگر را سبب شود. به این ترتیب می‌توان Promiseها را زنجیر وار به یکدیگر متصل کرد. برای نمونه jQuery به صورت توکار از promises پشتیبانی می‌کند:

```
// returns a promise
$.get('http://site-url/0')
.then(function(data) {
    // callback 1
    // returns a promise
    return $.get('http://site-url/1');
})
.then(function(data) {
    // callback 2
    // returns a promise
    return $.post('http://site-url/2');
})
.then(function(data) {
    // callback 3
});
```

متد get در jQuery یک شیء promise را بازگشت می‌دهد. در ادامه می‌توان این نتیجه را توسط متد then، زنجیروار ادامه داد. متدی که به عنوان پارامتر به then ارسال می‌شود، یک callback بوده و پس از پایان کار promise قبلی رخ می‌دهد. آرگومانی که به

این callback ارسال می‌شود، نتیجه‌ی promise قبلی است. در حین اعمال jQuery Ajax، این callback تنها زمانی فراخونی می‌شود که عملیات قبلی موفقیت آمیز بوده باشد و data ارائه شده، اطلاعاتی است که توسط response دریافتی از سرور، دریافت گردیده‌است.

در این حالت، هر callback حداقل سه کار را می‌تواند انجام دهد:

الف) یک promise دیگر را بازگشت دهد. نمونه آن‌را با return $.get در کدهای فوق ملاحظه می‌کنید.

ب) خاتمه عادی. همینجا کار promise با مقدار بازگشت داده شده، پایان می‌یابد.

ج) صدور یک استثناء. سبب برگشت خوردن و عدم پذیرش promise می‌شود.

استفاده از Promises در سایر کتابخانه‌ها

jQuery پیاده سازی توکاری از promises دارد؛ اما سایر کتابخانه‌ها، مانند AngularJS ایی که مثال زده شده چطور عمل می‌کنند؟ استانداردی به نام Promises/A+ جهت یک دست سازی پیاده سازی‌های promise در جاوا اسکریپت پیشنهاد شده‌است. jQuery نیمی از آن‌را پیاده سازی کرده‌است؛ اما کتابخانه‌ی دیگری به نام Q Library ، پیاده سازی نسبتا مفصل‌تری را از این استاندارد ارائه می‌دهد. فریم ورک AngularJS نیز در پشت صحنه از همین کتابخانه برای پیاده سازی promises استفاده می‌کند.

آشنایی با کتابخانه Q

استفاده مقدماتی از Q همانند مثالی است که از jQuery ملاحظه کردید.

```
Q.fcall(callback1)
.then(callback2);
```

اشیاء promise بازگشت داده شده توسط jQuery نیز توسط کتابخانه Q مورد پذیرش واقع می‌شوند:

```
Q.fcall(function() {
    return $.get('http://my-url');
})
.then(callback3);
```

علاوه بر این‌ها مفهومی به نام deferred objects نیز در کتابخانه‌ی Q پیاده سازی شده‌است:

```
function waitForClick() {
    var deferred = Q.defer();

$('#okButton').click(function() {
        deferred.resolve();
    });

$('#cancelButton').click(function() {
        deferred.reject();
    });

return deferred.promise;
}

Q.fcall(waitForClick)
.then(function() {
    //  ok button was clicked
}, function() {
    //  cancel button was clicked
});
```

توسط deferred objects می‌توان بررسی یک promise را به تاخیر انداخت. در مثال فوق، اولین callback فراخوانی شده به نام waitForClick، از اشیاء به تاخیر افتاده استفاده می‌کند. ابتدا توسط فراخوانی متد Q.defer، یک deferred object ایجاد می‌شود. در این بین اگر کاربر بر روی دکمه‌ی OK کلیک کرد، با فراخوانی deferred.resolve، این promise مورد پذیرش واقع خواهد شد و یا اگر کاربر بر روی دکمه‌ی cancel کلیک کند، با فراخوانی متد deferred.reject، این promise رد می‌گردد. نهایتا شیء promise

توسط deferred.promise بازگشت داده خواهد شد.

در ادامه کار، اینبار متد then، دو callback را قبول می‌کند. Callback اول پس از پذیرش قرار داد و Callback دوم پس از رد قرار داد، فراخوانی خواهد گردید.

در رنجیره تعریف شده، اگر معادلی برای reject درنظر گرفته نشده باشد، مانند مثال ذیل:

```
Q.fcall(myFunction1)
.then(success1)
.then(success2, failure1);
```

Q به دنبال نزدیک‌ترین متد callback گزارش خطای کار خواهد گشت. در این حالت متد failure1 در صورت شکست اولین promise فراخوانی خواهد شد.

همچنین اگر نتیجه‌ی success1 با شکست مواجه شود نیز failure1 فراخوانی می‌گردد. اما باید درنظر داشت که شکست success2، توسط failure1 مدیریت نمی‌شود.

Promises در AngularJS

در AngularJS امکانات کتابخانه Q توسط پارامتری به نام $q در اختیار سرویس‌های برنامه قرار می‌گیرد (تزریق می‌شود):

```
var app = angular.module("myApp", []);
app.factory('dataSvc', function($http, $q){
  var basePath="api/books";
  getAllBooks = function(){
   var deferred = $q.defer();
 $http.get(basePath).success(function(data){
    deferred.resolve(data);
   }).error(function(err){
    deferred.reject("service failed!");
   });
   return deferred.promise;
  };

  return{
   getAllBooks:getAllBooks
  };
});

app.controller('HomeController', function($scope, $window, dataSvc){
 function initialize(){
   dataSvc.getAllBooks().then(function(data){
    $scope.books = data;
   }, function(msg){
    $window.alert(msg);
   });
 }

 initialize();
});
```

در اینجا اگر دقت کنید، مباحث و عملکرد آن دقیقا مانند قبل است. ابتدا یک deferred object با فراخوانی متد q.defer ایجاد شده است. سپس با استفاده از امکانات توکار http آن (بجای استفاده از jQuery Ajax)، کار فراخوانی یک restful service صورت گرفته است (مثلا فراخوانی یک ASP.NET Web API). در صورت موفقیت کار، متد deferred.resolve و در صورت عدم موفقیت، متد deferred.reject فراخوانی شده‌است. نهایتا این سرویس، یک deferred.promise را بازگشت می‌دهد.

اکنون در کنترلری که قرار است از این سرویس استفاده کند، متد then کتابخانه Q را ملاحظه می‌کنید که دو Callback متناظر resolve و reject مدیریت promise بازگشت داده شده را به همراه دارد. اگر عملیات Ajaxایی موفقیت آمیز باشد، شیء books را مقدار دهی می‌کند و اگر خیر، پیامی را به کاربر نمایش خواهد داد.

پشتیبانی مرورگرهای جدید از استاندارد Promise

در حال حاضر کروم 32 و نگارش‌های شبانه فایرفاکس، Promise را که جزئی از استاندارد JavaScript شده‌است، به صورت توکار

و بدون نیاز به کتابخانه‌های جانبی، پشتیبانی می‌کنند.

```
if (window.Promise) { // Check if the browser supports Promises
 var promise = new Promise(function(resolve, reject) {
  //asynchronous code goes here
 });
}
```

در اینجا با فراخوانی window.Promise مشخص می‌شود که آیا مرورگر جاری از Promises پشتیبانی می‌کند یا خیر. سپس یک شیء promise ایجاد شده و این شیء توسط پارامترهای resolve و reject که هر دو تابع می‌باشند، کار مدیریت کدهای غیرهمزمان را انجام می‌دهد:

```
if (window.Promise) {
 console.log('Promise found');

 var promise = new Promise(function(resolve, reject) {
      // async
  if (result) {
   resolve(data);
  } else {
   reject('error');
  }
 });

 promise.then(function(data) {
  console.log('Promise fulfilled.');
 }, function(error) {
  console.log('Promise rejected.');
 });
} else {
 console.log('Promise not available');
}
```

در مثال فوق ابتدا یک شیء Promise ایجاد شده است. این شیء استاندارد بوده و با کروم 32 قابل آزمایش است. سپس در callback ابتدایی آن می‌توان یک عملیات AJAX ایی را انجام داد. اگر نتیجه‌ی آن موفقیت آمیز بود، تنها کافی است پارامتر اول این callback را فراخوانی کنیم و اگر خیر، پارامتر دوم آن‌را. برای استفاده از این شیء Promise ایجاد شده، می‌توان از متد then استفاده کرد. این متد نیز در اینجا دو callback پذیرش و رد promise را می‌تواند دریافت کند. برای زنجیر کردن آن کافی است متد then، یک Promise دیگر را بازگشت دهد و از نتیجه‌ی آن در then بعدی استفاده گردد.

عنوان: بررسی angular.bootstrap
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۱۴ ۲۳:۱۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, bootstrap

در پست‌های قبلی با مفهوم ng-app آشنا شدید. دایرکتیو ng-app برای استفاده از راه انداز خودکار فریم ورک Angular (معروف به auto-bootstrap) استفاده می‌شود. در حالت پیش فرض، به ازای هر سند Html فقط می‌توان یک ماژول در Angular تعریف کرد. در سند مربوطه اولین المانی که دارای دایرکتیو ng-app باشد به عنوان عنصر ریشه در نظر گرفته می‌شود و تمام عناصر تعریف شده در محدوده این دایرکتیو قابل استفاده برای ماژول مورد نظر خواهد بود. سایر عناصر حتی اگر ng-app یکسان داشته باشند نادیده گرفته می‌شوند.

ابتدا یک مثال زیر را به روش auto-bootstrap بررسی می‌کنیم:

```
<div ng-app="myApp">
    <div ng-controller="myController as ctrl">
       <span>ng-app #1</span> {{ctrl.firstName}} {{ctrl.lastName}}
    </div>
</div>

<div ng-app="myApp">
    <div ng-controller="myController as ctrl">
        <span>ng-app #2</span> {{ctrl.firstName}} {{ctrl.lastName}}
    </div>
</div>

@section scripts
{
    <script type="text/javascript" src="~/scripts/Modules/module8.js"></script>
}
```

در کنترلر مورد نظر نیز تعاریف به صورت زیر خواهد بود:

```
var app = angular.module('myApp', []);

app.controller('myController', function ()
{
    this.firstName = "Masoud";
    this.lastName = "Pakdel";
});
```

در مثال بالا دو تگ div وجود دارد که به صورت مشترک با استفاده از دایرکتیو ng-app به یک ماژول اشاره می‌کنند. طبق گفته‌ها بالا در روش auto-bootstrap اولین عنصری که دارای دایرکتیو ng-app باشد به عنوان محدوده ماژول مورد استفاده قرار خواهد گرفت در نتیجه سایر المان‌ها (در اینجا منظور تگ div دوم است)نادیده گرفته خواهند شد. پس خروجی به صورت زیر می‌شود:

File Edit View Favorites Tools Help

ng-app #1 Masoud Pakdel
ng-app #2 {{ctrl.firstName}} {{ctrl.lastName}}

اما اگر قصد داشته باشیم که در یک سند html دو نقطه شروع تعریف کنیم در حالی که هر کدام از یک منبع داده استفاده نمایند باید bootstrap برنامه را به صورت دستی تعیین کرد. برای این کار کافیست از دستور angular.bootstrap به صورت زیر استفاده نماییم:

پیاده سازی مثال بالا

```
<div id="myAppContainer1">
    <div ng-controller="myController as ctrl">
        <span>ng-app #1</span> {{ctrl.firstName}} {{ctrl.lastName}}
    </div>
</div>

<div id="myAppContainer2">
    <div ng-controller="myController as ctrl">
        <span>ng-app #2</span> {{ctrl.firstName}} {{ctrl.lastName}}
    </div>
</div>

@section scripts
{
    <script type="text/javascript" src="~/scripts/Modules/module8.js"></script>
}
```

اولین تغییر مورد نظر این است که، دایرکتیو ng-app حذف شد و به جای آن id برای تگ div تعیین کردیم. در فایل کنترلر مورد نظر نیز تغییر زیر را اعمال می‌کنیم:

```
var app = angular.module('myApp', []);

app.controller('myController', function ()
{
    this.firstName = "Masoud";
    this.lastName = "Pakdel";
});

angular.bootstrap(document.getElementById("myAppContainer1"), ["myApp"]);

angular.bootstrap(document.getElementById("myAppContainer2"), ["myApp"]);
```

با استفاده از دستور angular.bootstrap می‌توان بر اساس id تعیین شده تگ مورد نظر در سند را به دست آورد و ماژول مورد نظر را به آن نسبت داد.

خروجی مثال بالا:

برخلاف حالت قبل هر دو نقطه شروع به یک منبع داده اشاره می‌کنند و محدودیت حالت قبل برطرف می‌شود.

نظرات خوانندگان

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۱۰/۱۷ ۱۹:۴

سلام و تشکر
بنظرم جای نمونه برنامه‌های کوچکی که گام به گام توضیح داده بشن خالیه. خیلی خوب و کاربردی مفهوم جا می‌افته. مثلا نمونه‌ی معروف todo

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۰/۱۷ ۲۳:۳

مثال بالا نمونه ای از یک برنامه کوچک است. بهتر است هر نکته و سرنخ در طی یک پست ارائه شود به جای ذکر تاریخچه، هستی و چیستی مسائل در طی یک پست.

عنوان: **مسیریابی در AngularJs #بخش سوم**
نویسنده: حمید صابری
تاریخ: ۱۳۹۲/۱۰/۲۱ ۰:۱۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در بخش‌های پیشین (بخش اول و بخش دوم) به خوبی با اصول و روش مسیریابی (Routing) در AngularJS آشناشدیم. در این بخش می‌خواهم به برخی جزئیات درباره مسیریابی بپردازم. اولین موضوع، تغییراتی است که از نسخه 1.2 به بعد در روش استفاده از سرویس مسیریابی در AngularJS بوجود آمده است. از نسخه 1.2 سرویس مسیریابی از هسته اصلی AngularJS خارج شد و برای استفاده از امکانات این سرویس باید فایل angular-route.js و یا angular-route.min.js را به صفحه خود بیفزاییم:

```
<script src="~/Scripts/angular.min.js"></script>
<script src="~/Scripts/angular-route.min.js"></script>
```

سپس باید هنگام تعریف ماژول، ngRoute را به عنوان وابستگی تزریق کنیم:

```
var app = angular.module("mainApp", ['ngRoute']);
```

روش Controller as در AngularJS که از نسخه 1.2 به بعد امکان استفاده از آن وجود دارد قبلا معرفی شده است. با پاس کردن خصوصیت controllerAs به متد when می‌توان از viewای استفاده کرد که در آن از این روش استفاده شده است.

```
.when('/controllerAS', {
    controller:      'testController',
    controllerAs:    'tCtrl',
    template:        '<div>{{tCtrl.Title}}</div>'
})
```

باقی ماجرا مانند گذشته است.
موضوع دیگری که پرداختن به آن می‌تواند مفید باشد، بررسی بیشتر متد when است. وقتی در متد config ماژول از $routeProvider استفاده می‌کنیم، داریم سرویس $route را تنظیم، مقداردهی اولیه، و نمونه گیری می‌کنیم. درواقع با استفاده از متدهای when و otherwise داریم سرویس $route را مقداردهی اولیه می‌کنیم (برای آشنایی با تفاوت factory، service و provider کلیک کنید). خوب! جریان این مقادیری که به عنوان پارامتر به این متدها پاس می‌کنیم چیست؟
متد when به این صورت تعریف شده است:

```
when(string path, object route)
```

پارامتر path در بخش‌های قبل به اندازه کافی معرفی شده است. پارامتر route یک شی است شامل اطلاعاتی که با تطبیق آدرس صفحه با پارامتر path، به $route.current مقداردهی می‌شود (حالا باید متوجه شده باشید که روال افزودن داده‌های سفارشی به سیستم مسیریابی و دسترسی به آن‌ها که در بخش دوم مطرح شد به چه شکل کار می‌کند). این شی می‌تواند خصوصیات از قبل تعریف شده‌ای داشده باشد که در ادامه آن‌ها را مرور می‌کنیم: **controller:** می‌تواند یک رشته شامل نام کنترلر از قبل تعریف شده، یا یک تابع به عنوان تابع کنترلر باشد. **controllerAs:** رشته‌ای شامل نام مستعار کنترلر. **template:** رشته‌ای شامل قالب html، و یا تابعی که قالب html را بازمی‌گرداند. این خصوصیت بر templateUrl اولویت دارد. اگر مقدار این خصوصیت یک تابع باشد، $routeParams به عنوان پارامتر ورودی به آن پاس می‌شود. **templateUrl:** رشته‌ای شامل مسیر فایل قالب html، و یا تابعی که این رشته را بازمی‌گرداند. اگر مقدار این خصوصیت یک تابع باشد، $routeParams به عنوان پارامتر ورودی به آن پاس می‌شود. **redirectTo:** مقداری برای به روز رسانی $location، و فراخوانی روال مسیر یابی. این مقدار می‌تواند یک رشته، و یا تابعی که یک رشته را بازمی‌گرداند باشد. اگر مقدار این خصوصیت یک تابع باشد، این پارامترها به آن پاس می‌شود:
$routeParams برای دسترسی به پارامترهای آمده در آدرس صفحه جاری.

$location.path() جاری به صورت یک رشته.

$location.search() جاری به صورت یک شی.

caseInsensitiveMatch: یک مقدار منطقی است که مشخص می‌کند بزرگ و کوچک بودن حروف در تطبیق آدرس صفحه با پارامتر route در نظر گرفته بشود یا نه. مقدار پیشفرض این خصوصیت false است. یعنی در همه مثالهایی که تا کنون زده شده، اگر بزرگ و کوچه بودن حروف آدرس صفحه با مقدار مشخص شده برای پارامتر route متفاوت باشد، روال مسیریابی انجام نخواهد شد. برای رفع این مشکل کافی است مقدار این خصوصیت را true قرار دهیم. برای مثال، مسیر '/controllerAS' که بالاتر تعریف کرده‌ایم را درنظر بگیرید. اگر www.mySite.com/#/ControllerAS را وارد کنیم، هیچ اتفاقی نخواهد افتاد و در واقع این آدرس با route مشخص شده تطبیق پیدا نمی‌کند. اگر بخواهیم کوچک و بزرگ بودن حروف در نظر گرفته نشود، کافیست به این ترتیب عمل کنیم:

```
.when('/controllerAS', {
    controller:      'testController',
    controllerAs:    'tCtrl',
    template:        '<div>{{tCtrl.Title}}</div>',
    caseInsensitiveMatch: true
})
```

resolve: نگاشتی از وابستگی‌هایی که می‌خواهیم به کنترلر تزریق شود. قبلا مفهوم promise توضیح داده شده است . اگر هر یک از این وابستگی‌ها یک promise باشد، مسیریاب تا resolve شدن همه آن‌ها یا reject شدن یکی از آن‌ها منتظر می‌ماند. در صورتی که همه promiseها resolve شوند، رخداد $routeChangeSuccess، و در صورتی که یکی از آن‌ها reject شود رخداد $routeChangeError اجرا می‌شود. یکی از کاربردهای resolve زمانیست که بخواهید جلوی تغییر محتویات صفحه، پیش از بارگذاری داده‌ای که از سمت سرور درخواست کرده‌اید را بگیرید.

```
$routeProvider
    .when('/resolveTest',
        {
            resolve: {
                // این وابستگی بلافاصله بازمی‌گردد
                person: function () {
                    return {
                        name: "Hamid Saberi",
                        email: "Hamid.Saberi@Gmail.com"
                    }
                },
                // این وابستگی یک promise بازمی‌گرداند
                // پس تغییر مسیر تا resolve شدن آن به تاخیر می‌افتد
                currentDetails: function ($http) {
                    return $http({
                        method: 'Get',
                        url: '/current_details'
                    });
                },
                //می‌توانیم از یک وابستگی در وابستگی دیگر استفاده کنیم
                facebookId: function ($http, currentDetails) {
                    $http({
                        method: 'GET',
                        url: 'http://facebook.com/api/current_user',
                        params: {
                            email: currentDetails.data.emails[0]
                        }
                    })
                },
                // بارگذاری فایل‌های اسکریپت مورد نیاز
                fileDeps:function($q, $rootScope){
                    var deferred = $q.defer();
                    var dependencies =
                    [
                        'controllers/AboutViewController.js',
                        'directives/some-directive.js'
                    ];

                    //$Script.js بارگذاری وابستگی‌ها با استفاده از
                    $script(dependencies, function(){
                        // همه وابستگی‌ها بارگذاری شده اند
                        $rootScope.$apply(function(){
                            deferred.resolve();
                        });
                    });

                    return deferred.promise;
```

```
                }
            },
            controller: function ($scope, person, currentDetails, facebookId) {
                this.Person = person;
            },
            controllerAs: 'rtCtrl',
            template: '<div>{{rtCtrl.Person.name}}</div>',
            caseInsensitiveMatch: true
        })
```

نظرات خوانندگان

نویسنده: مسعود رمضانی
تاریخ: ۱۳۹۲/۱۲/۲۰ ۱۱:۰

دوست عزیز خیلی مطلب مفیدی بود.

ممنونم :)

عنوان: **زیرنویس فارسی ویدئوهای مقدمات AngularJS - قسمت اول**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۱۰/۲۱ ۲۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

سایت pluralsight یک دوره آموزشی با عنوان AngularJS Fundamentals تهیه کرده است، که به آموزش مقدمات AngularJS و اینکه چگونه می‌توانیم برنامه هایی با قابلیت تست پذیری، SPA و به سبک MVC بنویسیم، می‌پردازد.
فعلاً قسمت اول این مجموعه زیرنویس شده است که از اینجا قابل دریافت می‌باشد، جهت مشاهده ویدئوها نیز پیشنهاد می‌شود از برنامه KMPlayer استفاده کنید.
لیست ویدئوهای قسمت اول این مجموعه به شرح زیر است :

```
Course Introduction
Module Introduction
Introduction to Angular
Angular Architecture
Demo: Hello World in Angular
The Angular Event Reg Application
Angular Seed
Summary
```

6 قسمت دیگر از این مجموعه باقیمانده است، که بعد از آماده شدن به همین ترتیب به صورت یک پست در سایت ارائه خواهد شد. اگر مایل به همکاری بودید در قسمت پروژه‌های سایت می‌توانید اقدام کنید.
برای تهیه زیرنویس‌ها هم از برنامه Subtitle Tools استفاده میکنم، البته ظاهراً خود ویدئوها دارای زیرنویس انگلیسی هستند که رایگان نیستند.

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۰/۲۱ ۲۲:۵

با تشکر از شما. خود ویدیوها رو هم میشه از اینجا دریافت کرد: ajf.7z

نویسنده: مرتضی
تاریخ: ۱۳۹۲/۱۰/۲۳ ۹:۵۲

سلام
ببخشید نمیشه دانلود کرد فکر کنم مسیر اشتباه است
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۰/۲۳ ۱۰:۰

فایل‌های آن، در قسمت فایل‌های پروژه یاد شده، قرار داده شده‌اند. در اینجا

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۱۱/۰۱ ۸:۱۱

سورس پروژه ایی که در طول دوره ساخته می‌شود را نیز اینجا می توانید دانلود کنید.

نویسنده: PersianMan
تاریخ: ۱۳۹۳/۰۵/۰۵ ۷:۵۲

سلام و تشکر از زحمتی که کشیدید.
هیچ کدوم از فایلها رو نشد دانلود کنم و خطای زیر رو توسط دانلود منیجر می‌داد:
An existing connection was forcibly closed by the remote host. This normally results if the peer application on 'the remote host is suddenly stopped, the host is rebooted, or the remote host used a 'hard close

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۵/۰۵ ۹:۲۱

احتمالاً مشکل از IDMتون هست. پلاگین IDM رو توی مرورگر کروم غیرفعال کنید، بعد فایل رو دانلود کنید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۰۵ ۹:۲۱

- خود زیرنویس‌ها از اینجا (بررسی شد، مشکلی با دریافتش نبود)
- اصل ویدیوها از اینجا با لینک مستقیم

نویسنده: PersianMan
تاریخ: ۱۳۹۳/۰۵/۰۵ ۹:۲۴

تشکر، الان تست کردم و دانلود شد، مشکل نمیدونم از چی بود،اما نیازی به غیرفعال کردن IDM هم نشد.

عنوان: **خواندن اطلاعات از سرور و نمایش آن توسط Angular در ASP.NET MVC**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۱۱/۱۸ ۱۶:۴۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, AngularJS

می خواهیم یک مثال ساده از دریافت اطلاعات از سرور و نمایش آن در یک View را توسط AngularJS، با هم بررسی کنیم.
همانطور که می‌دانید برای نمایش تعدادی از اشیاء در انگولار می‌توان به این صورت نیز عمل کرد:

```
<div ng-init="products=[
     {id:1,name:'product1',price:25000,description:'description of product'},
     {id:2,name:'product2',price:5000,description:'description of product'},
     {id:3,name:'product3',price:5000,description:'description of product'},
     {id:4,name:'product4',price:2000,description:'description of product'},
     {id:5,name:'product5',price:255000,description:'description of product'}
     ]">
    <div>
        <div>
            <table>
                <tr>
                    <th>Id</th>
                    <th>Product Name</th>
                    <th>Price</th>
                    <th>Description</th>
                </tr>
                <tr ng-repeat="product in products">
                    <td>{{product.id}}</td>
                    <td>{{product.name}}</td>
                    <td>{{product.price}}</td>
                    <td>{{product.description}}</td>
                </tr>
            </table>
        </div>
    </div>
</div>
```

در کد فوق توسط ویژگی ng-init می‌توانیم داده هایمان را Initialize کنیم و در نهایت توسط ویژگی ng-repeat می‌توانیم داده هایمان را در صفحه نمایش دهیم. این ویژگی دقیقاً مانند یک حلقه foreach عمل میکند؛ مثلا معادل آن در Razor سمت سرور، به این صورت است:

```
@foreach (var product in Model.products)
{
    <td>product.id</td>
    <td>product.name</td>
    <td>product.price</td>
    <td>product.description</td>
}
```

خوب؛ حالا میخواهیم این اطلاعات را از سمت سرور بخوانیم و به صورت فوق نمایش دهیم. ابتدا مدل مان را به این صورت تعریف می‌کنیم :

```
namespace AngularAndMvc.Models
{
    public class Product
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public float Price { get; set; }
        public string Description { get; set; }
    }
}
```

سپس در داخل کنترلر زیر اطلاعات را به صورت in memory data تعریف می‌کنیم (جهت سهولت دموی کار) و به view مورد نظر پاس می‌دهیم. البته شما می‌توانید این اطلاعات را از دیتابیس بخوانید؛ روال کار فرقی نمی‌کند:

```
namespace AngularAndMvc.Controllers
```

```
{
    public class ProductController : Controller
    {
        public ActionResult Index()
        {
            return View("Index", "", GetSerializedProduct());
        }

        public string GetSerializedProduct()
        {
            var products = new[]
            {
                new Product{Id=1,Name="product1",Price=4500,Description="description of this product"},
                new Product{Id=2,Name="product2",Price=500,Description="description of this product"},
                new Product{Id=3,Name="product3",Price=400,Description="description of this product"},
                new Product{Id=4,Name="product4",Price=5500,Description="description of this product"},
                new Product{Id=5,Name="product5",Price=66500,Description="description of this product"}
            };
            var settings = new JsonSerializerSettings { ContractResolver=new
CamelCasePropertyNamesContractResolver()};
            return JsonConvert.SerializeObject(products,Formatting.None,settings);
        }
   }
}
```

همانطور که در کد بالا مشخص است، اطلاعات را به صورت JSON به View مان پاس داده ایم و برای اینکه ابتدای نام مقادیر بازگشتی به صورت حروف بزرگ نباشند (به صورت خودکار تبدیل به camel case شوند) پارامتر settings را برای متد SerializeObject تعیین کرده‌ایم:

```
var settings = new JsonSerializerSettings { ContractResolver=new
CamelCasePropertyNamesContractResolver()};
return JsonConvert.SerializeObject(products,Formatting.None,settings);
```

view را نیز به این صورت تغییر می‌دهیم :

```
@model string
<div ng-init="products = @Model">
    <div>
        <div>
            <table>
                <tr>
                    <th>Id</th>
                    <th>Product Name</th>
                    <th>Price</th>
                    <th>Description</th>
                </tr>
                <tr ng-repeat="product in products">
                    <td>{{product.id}}</td>
                    <td>{{product.name}}</td>
                    <td>{{product.price}}</td>
                    <td>{{product.description}}</td>
                </tr>
            </table>
        </div>
    </div>
</div>
```

تنها تغییری که در کد فوق اعمال شده است، به جای اینکه ویژگی ng-init را به صورت inline مقداردهی کنیم آن را از کنترلر دریافت کرده ایم.

در خروجی هم اطلاعات به این صورت نمایش داده می‌شوند :

سورس مثال فوق را هم از اینجا می توانید دریافت کنید.

نظرات خوانندگان

نویسنده: حمید رضا منصوری
تاریخ: ۱۳۹۳/۰۱/۱۵ ۱۱:۱۵

با تشکر
من از این روش در یک view استفاده کردم (در اینجا هدف نمایش لیست کاربران بود) ولی مشکل من این هست که وقتی این view برای کاربر نمایش داده میشه چون انگولار اون رو کش میکنه با افزودن کاربر جدید این لیست تا درخواست مجدد از سرور بروز نمیشه.
میخواستم بدونم آیا با واکشی اطلاعات توسط $http مشکلم حل میشه؟
یا میشه یک view در انگولار کش نشود و به هر با route به اون view مورد نظر از سرور فراخوانی بشه

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۱/۱۵ ۱۱:۵۲

البته می‌تونید کش رو سمت سرور با اعمال outputcache بر روی اکشن خودتون غیر فعال کنید:

```
[OutputCache(NoStore = true, Duration = 0, VaryByParam = "None")]
public ActionResult Index()
{
      return View("Index", "", GetSerializedProduct());
}
```

نویسنده: محسن درپرستی
تاریخ: ۱۳۹۳/۰۱/۱۶ ۱۰:۵۳

در روش بالا اگر چه اطلاعات از سرور دریافت میشود اما حالتی استاتیک دارد چون لیست محصولات در زمان رندر شدن صفحه تولید و در ng-init قرار می‌گیرد . و در نتیجه برای بروزکردن حتما باید صفحه مجددا درخواست بشود (رفرش). این روش برای لیست هایی که تغییراتی ندارند یا به ندرت تغییر می‌کنند مناسب است.
و بله یک راه برای حل این مشکل استفاده از سرویس http می‌تواند باشد .

عنوان: **ساختار پروژه های Angular**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۱/۲۷ ۹:۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, Project Management

با توجه به پست‌ها منتشر شده قبلی درباره AngularJs به احتمال قوی شما نیز به این نتیجه رسیده اید که این فریم ورک برای انواع پروژه‌ها به ویژه پروژه هایی با مقیاس بزرگ بسیار مناسب است. منظور از ساختار پروژه Angular این است که به چه سبکی فایل‌های پروژه را سازمان دهی کنیم طوری که در هنگام توسعه و تغییرات با مشکل مواجه نشویم. عموما کدهای مربوط به بخش frontend پروژه دارای ساختار قوی نمی‌باشند در نتیجه developerها بیشتر سلیقه ای کدهای مربوطه را می‌نویسند که با گذر زمان این مورد باعث بروز مشکل در امر توسعه نرم افزار می‌شود (نمونه بارز آن کدهای نوشته شده Jquery در صفحات است). AngularJs نیز همانند سایر کتابخانه‌ها و فریم ورک‌های جاوااسکریپتی دیگر از این امر مستثنی نیست و فایل‌های آن باید طبق روشی مناسب پیاده سازی و مدیریت شوند. انتخاب ساختار و روش سازمان دهی فایل‌ها وابستگی مستقیم به مقیاس پروژه دارد. ساختار پروژه‌های کوچک می‌تواند کاملا متفاوت با ساختار پروژه‌های بزرگ باشد. در این پست به بررسی چند روش در این زمینه خواهم پرداخت.
پروژه‌های کوچک عموما دارای ساختاری مشابه تصویر ذیل می‌باشند:

- css/
- img/
- js/
  - app.js
  - controllers.js
  - directives.js
  - filters.js
  - services.js
- lib/
- partials/

این مورد، روش پیشنهادی در Angular Seed است و بدین صورت است که تعاریف ماژول‌ها در فایل app.js انجام می‌گیرد. تعاریف و پیاده سازی تمام کنترلرها در فایل controller.js است. و همچنین دایرکتیوها و فیلترها و سرویس‌ها هر کدام در فایل‌ها جداگانه تعریف و پیاده سازی می‌شوند. این روش راه حلی سریع برای پروژه‌های کوچک با تعداد developerهای کم است. برای مثال زمانی که یک developer در حال ویرایش فایل controller.js است، از آن جا که فایل مورد نظر checkout خواهد شد در نتیجه سایر developerها امکان تغییر در فایل مورد نظر را نخواهند داشت. سورس فایل‌ها به مرور زیاد خواهد شد و در نتیجه debug آن سخت می‌شود.

روش دوم

در این حالت تعاریف کنترلر ها، مدل‌ها و سرویس‌ها هرکدام در یک دایرکتوری مجزا قرار خواهد گرفت. برای هر view یک کنترلر و بنا بر نیاز مدل تعریف می‌کنیم. ساختار آن به صورت زیر می‌شود:

- controllers/
  - LoginController.js
  - RegistrationController.js
  - ProductDetailController.js
  - SearchResultsController.js
- directives.js
- filters.js
- models/
  - CartModel.js
  - ProductModel.js
  - SearchResultsModel.js
  - UserModel.js
- services/
  - CartService.js
  - UserService.js
  - ProductService.js

دایرکتیوها و فیلترها عموما در یک فایل قرار داده خواهند شد تا بنابر نیاز در جای مناسب رفرنس داده شوند. این روش ساختار مناسب‌تری نسبه به روش قبلی دارد اما دارای معایبی هم چون موارد زیر است:
»وابستگی بین فایل‌ها مشخص نیست در نتیجه بدون استفاده از کتابخانه هایی نظیر requireJs با مشکل مواجه خواهید شد.
»refactoring کدها تا حدودی سخت است.

روش سوم
این ساختار مناسب برای پیاده سازی پروژه‌ها به صورت ماژولار است و برای پروژه‌های بزرگ نیز بسیار مناسب است. در این حالت شما فایل‌های مربوط به هر ماژول را در دایرکتوری خاص آن قرار خواهید داد. به صورت زیر:

- cart/
    - CartModel.js
    - CartService.js
- common/
    - directives.js
    - filters.js
- product/
    - search/
        - SearchResultsController.js
        - SearchResultsModel.js
    - ProductDetailController.js
    - ProductModel.js
    - ProductService.js
- user/
    - LoginController.js
    - RegistrationController.js
    - UserModel.js
    - UserService.js

همان طور که ملاحظه می‌کنید سرویس ها، کنترلرها و حتی مدل‌های مربوط به هر بخش در یک مسیر جداگانه قرار می‌گیرند. علاوه بر آن فایل هایی که قابلیت اشتراکی دارند در مسیری به نام common وجود دارند تا بتوان در جای مناسب برای استفاده از آن‌ها رفرنس داده شود. حتی اگر در پروژه خود فقط یک ماژول دارید باز سعی کنید از این روش برای مدیریت فایل‌های خود استفاده نمایید. اگر با ngStart آشنایی داشته باشید به احتمال زیاد با این روش بیگانه نیستید.

بررسی چند نکته درباره کدهای مشترک

در اکثر پروژه‌های بزرگ، فایل‌ها و کد هایی وجود خواهد داشت که حالت اشتراکی بین ماژول‌ها دارند. در این روش این فایل‌ها در مسیری به نام common یا shared ذخیره می‌شوند. علاوه بر آن در Angular تکنیک هایی برای به اشتراک گذاشتن این اطلاعات وجود دارد.

»اگر ماژول‌ها وابستگی شدیدی به فایل‌ها و سورس‌های مشترک دارند باید اطمینان حاصل نمایید که این ماژولها فقط به اطلاعات مورد نیاز دسترسی دارند. این اصل interface segregation principle اصول SOLID است.

»توابعی که کاربرد زیادی دارند و اصطلاحا به عنوان Utility شناخته می‌شوند باید به $rootScope اضافه شوند تا scopeهای وابسته نیز به آن‌ها دسترسی داشته باشند. این مورد به ویژه باعث کاهش تکرار وابستگی‌های مربوط به هر کنترلر می‌شود.

»برای جداسازی وابستگی‌های بین دو component بهتر از eventها استفاده نمایید. AngularJs این امکان را با استفاده از سرویس‌های $on و $emit و $broadcast به راحتی میسر کرده است.

نظرات خوانندگان

نویسنده: ایاک
تاریخ: ۱۳۹۲/۱۱/۲۷ ۹:۵۹

با سلام و تشکر از مقاله خوب شما.
برای بارگذاری اسکریپت‌ها در روش سوم ، از آنجا که ممکن است تعداد دایرکتوری‌ها زیاد باشد ، شما چه روشی را پیشنهاد می‌کنید؟

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۱/۲۷ ۱۰:۵۴

زمانی که تعداد فایل‌ها و دایرکتوری‌ها در پروژه زیاد می‌شود(البته این جزء جدانشدنی پروژه‌های مقیاس بزرگ است) برای جلوگیری از لود یک باره کنترلرها و دایرکتیوها، بهتر از lazy loading برای لود فایل‌های مورد نیاز استفاده شود. متاسفانه Angular به صورت رسمی از lazy loading پشتیبانی نمی‌کند اما با کمی تغییر در ساختار و استفاده از کتابخانه‌های جانبی مثل requireJs یا ScriptJs می‌توان به این مهم دست یافت.
(با عنوان این مطلب که قصد داشتم این مورد را طی یک پست جداگانه بررسی کنم)
برای مثال:
ابتدا ماژول app خود را به این شکل تنظیم کنید:

```
(function()
{
    var app = angular.module('app', []);

    app.config(function($routeProvider, $controllerProvider, $compileProvider, $filterProvider,
$provide)
    {
        app.controllerProvider = $controllerProvider;
        app.compileProvider    = $compileProvider;
        app.routeProvider      = $routeProvider;
        app.filterProvider     = $filterProvider;
        app.provide            = $provide;

    });
})();
```

با استفاده از سرویس $controllerProvider می‌توان چرخه ساخت کنترلر را به دست گرفت. هم چنین سرویس $compileProvider برای نمونه سازی دایرکتیوها و $filterProvider برای فیلترها استفاده می‌شوند.
ساخت کنترلرها و دایرکتیوها نیز به صورت زیر انجام خواهد شد:

```
angular.module('app').controllerProvider.resgister('SomeLazyController', function($scope)
{
    $scope.key = '...';
});
```

و هم چنین یک نمونه از ساخت directive

```
$compileProvider.directive('SomeLazyDirective', function()
{
    return {
        restrict: 'A',
        templateUrl: 'templates/some-lazy-directive.html'
    }
})
```

فقط کافیست در هنگام پیاده سازی routing (که در این مقاله شرح داده شده است) نوع بارگذاری کنترلرها و دایرکتیو و ... را به صورت lazy انجام دهید :

```
$routeProvider.when('/about', {templateUrl:'views/about.html', resolve:{deps:function($q, $rootScope)
{
    var deferred = $q.defer();
    var dependencies =
    [
        'controllers/AboutViewController.js',
        'directives/some-directive.js'
    ];

    //*نکته اول
    $script(dependencies, function()
    {
        // *نکته دوم
        $rootScope.$apply(function()
        {
            deferred.resolve();
        });
    });

    return deferred.promise;
}}})
```

*نکته اول: تمام وابستگی‌ها توسط scriptJs مدیریت می‌شوند.

*نکته دوم: تمام وابستگی‌ها مروبط به این scope بعد از فراخوانی تابع deffered.resolved بارگذاری خواهند شد.

نقطه شروع برنامه نیز به صورت زیر است:

```
$script(['appModule.js'], function()
{
    angular.bootstrap(document, ['app'])
});
```

angular.bootstrap

نویسنده: حمید صابری
تاریخ: ۱۳۹۲/۱۱/۲۸ ۹:۵۳

ضمن تشکر فراوان از جناب آقای پاکدل عزیز، در این مقاله به خوبی درباره lazy loading در angularjs بحث شده. نکته مهم اینکه حتما پروژه‌ی قابل اجرایی که در انتهای مقاله لینک شده را ملاحظه کنید. نکاتی در این پروژه هست از جمله اینکه برای دسترسی به providerها برای lazy loading آنها به این ترتیب به app افزوده شده اند:

```
app.config([
        '$stateProvider',
        '$urlRouterProvider',
        '$locationProvider',
        '$controllerProvider',
        '$compileProvider',
        '$filterProvider',
        '$provide',


        function ($stateProvider, $urlRouterProvider, $locationProvider, $controllerProvider,
$compileProvider, $filterProvider, $provide) {
            //برای رجیستر کردن غیر همروند اجزای انگیولاری در آینده
            app.lazy =
            {
                controller: $controllerProvider.register,
                directive:  $compileProvider.directive,
                filter:     $filterProvider.register,
                factory:    $provide.factory,
                service:    $provide.service
            };
.
.
.
])
```

(البته این کد از پروژه خودمان است و بعضی وابستگی‌های دیگر هم تزریق شده‌اند).

استفاده از app.lazy باعث سهولت بیشتر در استفاده و خواناتر شدن کد می‌شود. در ادامه به این ترتیب می‌توانید از app.lazy استفاده کنید:

```
angular.module('app').lazy.controller('myController',
        ['$scope',  function($scope){
...
}]);
```

به این ترتیب کد نوشته شده به دلیل نام گذاری ارجاع controllerProvider $ با controller به حالت عادی شبیه است، و از طرفی lazy پیش از آن به فهم ماجرا کمک خواهد کرد.

این نقطه شروع یکی از پروژه‌های ماست که به عنوان نمونه بد نیست ملاحظه کنید:

```
<script type="text/javascript">
// --- Scriptjs ---
!function (a, b, c) { function t(a, c) { var e = b.createElement("script"), f = j; e.onload = e.onerror
= e[o] = function () { e[m] && !/^c|loade/.test(e[m]) || f || (e.onload = e[o] = null, f = 1, c()) },
e.async = 1, e.src = a, d.insertBefore(e, d.firstChild) } function q(a, b) { p(a, function (a) { return
!b(a) }) } var d = b.getElementsByTagName("head")[0], e = {}, f = {}, g = {}, h = {}, i = "string", j =
!1, k = "push", l = "DOMContentLoaded", m = "readyState", n = "addEventListener", o =
"onreadystatechange", p = function (a, b) { for (var c = 0, d = a.length; c < d; ++c) if (!b(a[c]))
return j; return 1 }; !b[m] && b[n] && (b[n](l, function r() { b.removeEventListener(l, r, j), b[m] =
"complete" }, j), b[m] = "loading"); var s = function (a, b, d) { function o() { if (!--m) { e[l] = 1,
j && j(); for (var a in g) p(a.split("|"), n) && !q(g[a], n) && (g[a] = []) } } function n(a) { return
a.call ? a() : e[a] } a = a[k] ? a : [a]; var i = b && b.call, j = i ? b : d, l = i ? a.join("") : b, m
= a.length; c(function () { q(a, function (a) { h[a] ? (l && (f[l] = 1), o()) : (h[a] = 1, l && (f[l] =
1), t(s.path ? s.path + a + ".js" : a, o)) }) }, 0); return s }; s.get = t, s.ready = function (a, b,
c) { a = a[k] ? a : [a]; var d = []; !q(a, function (a) { e[a] || d[k](a) }) && p(a, function (a) {
return e[a] }) ? b() : !function (a) { g[a] = g[a] || [], g[a][k](b), c && c(d) }(a.join("|")); return
s }; var u = a.$script; s.noConflict = function () { a.$script = u; return this }, typeof module !=
"undefined" && module.exports ? module.exports = s : a.$script = s }(this, document, setTimeout)

$script(['/Scripts/Lib/jquery/jquery-1.10.2.min.js'], function () {
    $script(['/Scripts/Lib/angular/angular.js'], function () {
        $script(['/Scripts/Lib/angular/angular-ui-router.min.js',
                    '/Scripts/Lib/angular/angular-resource.min.js',
                    '/Scripts/Lib/angular/angular-cache.min.js',
                    '/Scripts/Lib/angular/angular-sanitize.min.js',
                    '/Scripts/Lib/angular/angular-animate.min.js',
                    '/Scripts/Lib/angular/angular-cookie.min.js',
                    '/APP/Common/directives.js'
                ], function () {
                    $script('/app/app.js', function () {
                        angular.bootstrap(document, ['app']);
                    });
                });
            })
        });
</script>
```

این تگ script در صفحه شروع پروژه آمده است.

کد minify شده scriptjs در ابتدا قرار دارد، پس از آن فایل‌های js مورد نیاز با رعایت وابستگی‌های احتمالی به ترتیب بارگذاری شده‌اند.

این قسمت resolve یکی از بخش‌های مسیریابی است:

```
resolve: {
                        fileDeps: ['$q', '$rootScope', function ($q, $rootScope) {
                            var deferred = $q.defer();
                            var deps = ['/app/HotStories/dataContextService.js',
                                        '/app/HotStories/hotStController.js'];
                            $script(deps, function () {
                                $rootScope.$apply(function () {
                                    deferred.resolve();
                                });
                            });

                            return deferred.promise;
                        }]
```

```
                    }
```

این نحوه تعریف سرویسی که فایل آن در وابستگی‌ها آمده و قرار است lazy load شود:

```
angular.module('app').lazy.service('dataContextService',
        ['$rootScope', '$resource', '$angularCacheFactory', '$q', function($rootScope, $resource,
$cacheFactory, $q){
...
}]);
```

و این هم نحوه تعریف کنترلری که فایل آن در وابستگی‌ها آمده و قرار است lazy load شود:

```
angular.module('app').lazy.controller('hotStController',
        ['$scope', 'ipCookie', 'dataContextService', function($scope, ipCookie, dataContextService){
...
}]);
```

نویسنده: علی فخرایی
تاریخ: ۱۳۹۲/۱۲/۲۷ ۱۳:۱۲

ممنون از مطلب مفیدتون. اگر ما یک area مثلا به نام administrator برای مدیریت داشته باشیم، آیا باید فایل‌ها را در مسیر ریشه مثلا در پوشه script قرار دهیم؟ یا باید در همان area؟ چون اگر در ریشه قرار دهیم جالب به نظر نمیرسد. ممکنه راهنمایی کنید؟

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۲/۲۷ ۱۷:۳۱

خیر. می‌توانید فایل‌های مورد نیاز هر ماژول و area را در مسیرهای جداگانه مربوط به area قرار دهید. پوشه Scripts صرفا برای قرار گیری فایل‌های مورد نیاز کتابخانه هاست(نظیر Jqeury و angular و q و ...).

نویسنده: علی فخرایی
تاریخ: ۱۳۹۲/۱۲/۲۷ ۲۰:۱

تشکر.شما در مورد مسیر یابی هم قطعه کدی قرار دادید که میشود وابستگی‌ها و ... را تزریق کرد.منتها اگر ما بیش از 100 مسیر داشته باشیم باید چه کنیم؟ یعنی به ازای هر مسیر باید این قطعه کد تکرار شود :

```
$routeProvider.when('/about', {templateUrl:'views/about.html', resolve:{deps:function($q, $rootScope)
{
    var deferred = $q.defer();
    var dependencies =
    [
        'controllers/AboutViewController.js',
        'directives/some-directive.js'
    ];

    //*نکته اول
    $script(dependencies, function()
    {
        // *نکته دوم
        $rootScope.$apply(function()
        {
            deferred.resolve();
        });
    });

    return deferred.promise;
}}})
```

راه حل پویایی وجود دارد؟
مثلا شما در ساختار سوم بیان کردید که فایل‌های مربوط به هر قسمت در کنار هم باشند. اعم از کنترلر و دایرکتیوها و فیلترها و ...
.
آیا میشود برای هر قسمت مثل product , user ,cart ، یک ماژول app جدا نوشت و در آن طبق مثال شما مسیریابی را تولید کرد؟ یعنی چندین ماژول انگولار app.js برای یک پروژه نوشت؟استاندارد است؟ بدین صورت دیگر نگران تعداد مسیرهای زیاد نیستیم و مشخص میشود که مسیریابی هر قسمت در کنار آن وجود دارد.
امکان پذیر است؟ اگر نیست شما چه راهی برای این کار دارید. ممنون

نویسنده: علی فخرایی
تاریخ: ۱۳۹۳/۰۱/۰۲ ۱۴:۲۳

میشه یک مثال ساده هم در مورد کامنت‌های دوم و سوم قرار بدید؟
من کلیه مراحل رو پیش رفتم و دو روز کامل درگیرش هستم، اما به نتیجه ای نمیرسم.خطاهای زیر رو در کنسول کروم دریافت میکنم.

```
Uncaught Error: [$injector:modulerr] Failed to instantiate module app due to:
Error: [$injector:nomod] Module 'app' is not available! You either misspelled the module name or forgot
to load it. If registering a module ensure that you specify the de...<omitted>...0) angular.js:78
Uncaught ReferenceError: app is not defined selectAllCheckbox.js:3
Uncaught Error: [$injector:modulerr] Failed to instantiate module app due to:
Error: [$injector:unpr] Unknown provider: $routeProvider
http://errors.angularjs.org/1.2.14/$injector/unpr?p0=%24routeProvider
    at http://localhost:8417/Scripts/Angula...<omitted>...0)
```

نویسنده: حمید رضا منصوری
تاریخ: ۱۳۹۳/۰۱/۰۳ ۱۶:۲۷

با تشکر بابت راهنمایتون
لطفا میتونید یه sample ساده از این مطلبتون بزارید. البته اون مثال لینکی که گذاشته بودید رو دیدم ولی نتونستم اجراش کنم و نمونه داخل خودش هم کار نمی‌کرد.

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۱/۰۵ ۱۰:۱۶

بخش اول سوال: بهتر است که کد مربوط به لود وابستگی‌ها در یک تابع مجزا نوشته شود و فقط در زمان نیاز این تابع را با پاس دادن وابستگی فراخوانی نمایید(با فرض اینکه نام این فایل dependencyResolver است):

```
(function()
{
    return function(dependencies)
    {
        var definition =
        {
            resolver: ['$q','$rootScope', function($q, $rootScope)
            {
                var deferred = $q.defer();

                $script(dependencies, function()
                {
                    $rootScope.$apply(function()
                    {
                        deferred.resolve();
                    });
                });

                return deferred.promise;
            }]
        }

        return definition;
    }
```

```
});
```

و برای لود وابستگی نیز تابع dependencyResolver را به این صورت فراخوانی نمایید:

```
angular.forEach(config.routes, function(route, path)
   {
                    $routeProvider.when(path, {templateUrl:route.templateUrl,
resolve:dependencyResolver(route.dependencies)});
    });
```

در مورد سوال دوم نیز باید عنوان کنم که شما می‌توانید مسیریابی هر ماژول را به صورت جداگانه در تعاریف همان ماژول‌ها انجام دهید که البته روشی مرسوم و معمول است. فقط در هنگام عملیات bootstrapping ماژول اصلی برنامه، سایر ماژول‌ها به عنوان وابستگی آن تعیین می‌شوند. به صورت زیر(عنوان ماژول‌ها را یکتا انتخاب نمایید) :

```
var app = angular.module('app', ['anotherModule1' , 'anotherModule2' , 'anotherModule3']);
```

نویسنده: حمید صابری
تاریخ: ۱۳۹۳/۰۱/۰۸ ۱۷:۰

دوست عزیز اینجا میتونید توضیحات بیشتر درباره lazy loading و یک پیاده سازی ساده از اونو مطالعه کنید.

نویسنده: حمید صابری
تاریخ: ۱۳۹۳/۰۱/۰۸ ۱۷:۱

دوست عزیز اینجا میتونید توضیحات بیشتر درباره lazy loading و یک پیاده سازی ساده از اونو مطالعه کنید.

نویسنده: ایاک
تاریخ: ۱۳۹۳/۰۱/۲۷ ۱۲:۴۰

با تشکر.
برای این قسمت در صورت امکان توضیح بیشتری میدهید؟

```
angular.forEach(config.routes, function(route, path)
   {
                    $routeProvider.when(path, {templateUrl:route.templateUrl,
resolve:dependencyResolver(route.dependencies)});
    });
```

این کد باید در کجا نوشته شود و مقدار config.routes از کجا دریافت می‌شود؟

عنوان: **یک نکته درباره Angular routeProvider**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۱/۲۷ ۲۱:۱۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

بعد از مطالعه پست‌های ^ و ^ نکته ای به ذهنم رسید که بیان آن از بنده و مطالعه آن توسط شما خالی از لطف نیست. اگر مثال‌های پیاده سازی شده در پست‌های ^ و ^ را با AngularJs نسخه 1.2 اجرا نمایید به طور حتم با خطا روبرو می‌شوید و نتیجه مورد نظر حاصل نمی‌شود. در این پست نیز توسط یکی از دوستان اشاره ای به این مطلب شد.

دلیل خطا این است که از نسخه 1.2 به بعد در Angular سیستم مسیر یابی به این شکل امکان پذیر نیست و بخش مسیریابی به یک فایل دیگر به نام angular-route.js منتقل شده است. در نتیجه اگر به سبک نسخه‌های قبلی Angular از سیستم مسیریابی استفاده نمایید با خطا مواجه خواهید شد و خطای مورد نظر هم مربوط به عدم توانایی در تزریق وابستگی $routeProvider به ماژول مورد نظر است. حال راه حل چیست؟

کافیست در هنگام تعریف ماژول، ngRoute را به عنوان وابستگی ماژول تعیین نمایید. و از طرفی فایل اسکریپتی angular-route.js را بعد از angular.js فراخوانی کنید.

بررسی مثال:

کدهای زیر مربوط به مثال‌های پست قبلی می‌باشد که شرح کامل آن در این پست است:

```
var myFirstRoute = angular.module('myFirstRoute', []);

myFirstRoute.config(['$routeProvider',
  function($routeProvider) {
    $routeProvider.
      when('/pageOne', {
        templateUrl: 'templates/page_one.html',
        controller: 'ShowPage1Controller'
      }).
      when('/pageTwo', {
        templateUrl: 'templates/page_two.html',
        controller: 'ShowPage2Controller'
      }).
      otherwise({
        redirectTo: '/pageOne'
      });
}]);


myFirstRoute.controller('ShowPage1Controller', function($scope) {
    $scope.message = 'Content of page-one.html';
});


myFirstRoute.controller('ShowPage2Controller', function($scope) {
    $scope.message = 'Content of page-two.html';
});
```

برای هماهنگ کردن مثال بالا با نسخه 1.2 باید به روش زیر عمل نمود:

```
var myFirstRoute = angular.module('myFirstRoute',['ngRoute']);

myFirstRoute.config(['$routeProvider',
  function($routeProvider) {
    $routeProvider.
      when('/pageOne', {
        templateUrl: 'templates/page_one.html',
        controller: 'ShowPage1Controller'
      }).
      when('/pageTwo', {
        templateUrl: 'templates/page_two.html',
        controller: 'ShowPage2Controller'
      }).
      otherwise({
        redirectTo: '/pageOne'
      });
}]);
```

تنها تغییر، مشخص کردن ngRoute به عنوان وابستگی ماژول myFirstRoute است(خط اول). نکته دیگر لود فایل angular-route.js قبل از فراخوانی فایل بالا و بعد از فراخوانی فایل angular.js است:

```
<body ng-app="app">

    <div>
        <div>
        <div>
            <ul>
                <li><a href="#pageOne"> Show page one </a></li>
                <li><a href="#pageTwo"> Show page two </a></li>
            </ul>
        </div>
        <div>
            <div ng-view></div>
        </div>
        </div>
    </div>

    <script src="http://ajax.googleapis.com/ajax/libs/angularjs/1.0.7/angular.min.js"></script>
     <script src="angular-route.js"></script>
    <script src="app.js"></script>

  </body>
```

نظرات خوانندگان

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۲/۱۱/۲۹ ۱۶:۲۶

ممنون از نکته خوبتون.
آیا برای ساماندهی تعداد بالای مسیرها ، راه حلی وجود داره؟

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۱۱/۲۹ ۲۳:۱۷

بله. من از Angular Dynamic Routing استفاده می‌کنم.

نویسنده: ایاک
تاریخ: ۱۳۹۲/۱۲/۰۱ ۲۰:۸

با سلام.
نظرتون راجع به پروژه angular-ui-router برای مسیریابی انگولار که قابلیت بیشتری را نسبت به سیستم مسیریابی پیش فرض آن ارائه می‌دهد چیست؟

نویسنده: محسن کریمی
تاریخ: ۱۳۹۲/۱۲/۰۲ ۱۳:۲۷

تمام مواردی که در routing مربوط به angularjs هستش در ui-router هم وجود دارد و مواردی مثل nested views و multiple named views بهش اضافه شده و عملا در پروژه‌ها کاربردی‌تر خواهد بود.

عنوان: **زیرنویس فارسی ویدئوهای مقدمات AngularJS - قسمت دوم**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۱۲/۰۹ ۱۷:۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, Subtitle

زیرنویس‌های فارسی قسمت دوم را اینجا می توانید دریافت کنید.
لیست سرفصل‌های قسمت دوم به شرح زیر است :

```
01-Introduction
02-Controllers and Scope
03-Demo. Controllers
04-Demo. Displaying Repeating Information
05-Demo Handling Events
06-Built-in Directives
07-Event Directives
08-Other Directives - Part 1
09-Other Directives - Part 2
10-IE Restrictions
11-Expressions
12-Filters
13-Built-in Filters
14-Writing Custom Filters
15-Two Way Binding
16-Demo. Two Way Binding
17-Validation
```

این قسمت به نحوه ایجاد کنترلرها و بررسی شیء scope و چگونگی تعامل کنترلر با شیء scope و همچنین نحوه ارتباط کنترلر با صفحه و اینکه چگونه داده‌ها توسط مکانیزم Binding در خروجی نمایش داده میشوند می‌پردازد، همچنین نحوه مدیریت رخدادها و چگونگی مدیریت این رخدادها در انگولار و کار با عبارات (سینتکس) در انگولار مورد بررسی قرار خواهد گرفت، سپس بعد از بررسی چندین مثال در این رابطه به بررسی فیلترها و چگونگی ساخت فیلترهای سفارشی می‌پردازد، در نهایت در رابطه با سیستم اعتبارسنجی توکار انگولار صحبت خواهد شد.
همچنین مثال‌های جالبی در این بخش مورد بررسی قرار می‌گیرد، به طور مثال در بخش مربوط به مدیریت رخدادها یک سیستم امتیازدهی ساده مورد بررسی قرار میگیرد.

نظرات خوانندگان

نویسنده: آریو
تاریخ: ۱۳۹۲/۱۲/۱۱ ۱۹:۳۱

با سلام و تشکر فراوان. امکانش هست لینک دانلود ویدئوها رو هم بزارید؟ خیلی جستجو کردم تو اینترنت پیدا نکردم. ممنون

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۱۲/۱۱ ۱۹:۴۴

در نظرات قسمت اول ارسال شده.

عنوان: مسیریابی تو در تو در Angularjs با استفاده از UI-Router
نویسنده: حمید صابری
تاریخ: ۱۳۹۲/۱۲/۲۳ ۱۹:۳۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

UI-Router ابزاری برای مسیریابی در AngularJS است که این امکان را برایتان فراهم می‌کند تا بخش‌های برنامه رابط کاربریتان را به شکل یک ماشین حالت ساماندهی کنید. برخلاف سرویس $route که بر اساس مسیریابی URLها ساماندهی شده و کار می‌کند، UI-Router بر اساس حالت‌ها کار می‌کند، که این حالت‌ها می‌توانند در صورت لزوم مسیریابی هم داشته باشند.

UI-Router یکی از افزونه‌های مجموعه Angular-ui ، و پاراگراف بالا معرفی آن در صفحه خانگیش است (تقریبا!). این افزونه جزئیات مفصلی دارد و در این مطلب تنها به معرفی آن خواهم پرداخت (بر اساس مطالب صفحه خانگیش). پیش از ادامه پیشنهاد می‌کنم اگر مطالب زیر را نخوانده‌اید ابتدا آن‌ها را مرور کنید:

مسیریابی در AngularJs #بخش اول
مسیریابی در AngularJs #بخش دوم
مسیریابی در AngularJs #بخش سوم

برای استفاده از UI-Router باید:

فایل جاوا اسکریپت آن را دانلود کنید (released یا minified).
در صفحه اصلی برنامه‌تان پس از include کردن فایل اصلی AngularJS فایل angular-ui-router.js (یا angular-ui-router.min.js) را include کنید.
'ui.router' را به لیست وابستگی‌های ماژول اصلی اضافه کنید.
نتیجه چیزی شبیه این خواهد بود:

```
<!doctype html>
<html ng-app="myApp">
<head>
    <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.1.5/angular.min.js"></script>
    <script src="js/angular-ui-router.min.js"></script>
    <script>
        var myApp = angular.module('myApp', ['ui.router']);
        // For Component users, it should look like this:
        // var myApp = angular.module('myApp', [require('angular-ui-router')]);
    </script>
    ...
</head>
<body>
    ...
</body>
</html>
```

حالت‌ها و viewهای تو در تو قابلیت اصلی UI-Router امکان تعریف حالت‌ها و viewهای تو در تو است. در مطلب مسیریابی در AngularJs #بخش اول دایرکتیو ng-view معرفی شده است. هنگام استفاده از سرویس $route با این دایرکتیو می‌توان محل مورد نظر برای بارگذاری محتویات مربوط به مسیرها را مشخص کرد. دایرکتیو ui-view در UI-Router همین نقش را دارد. فرض کنید این کد فایل index.html باشد:

```
<!-- index.html -->
<body>
    <div ui-view></div>
    <!-- We'll also add some navigation: -->
    <a ui-sref="state1">State 1</a>
    <a ui-sref="state2">State 2</a>
</body>
```

همانطور که ملاحظه می‌کنید در تگ‌های a از دایرکتیو ui-sref استفاده شده است. این دایرکتیو علاوه بر مدیریت تغییر حالت،

خصوصیت href تگ a را در صورتی که حالت مشخص شده URL داشته باشد تولید می‌کند. البته برای استفاده از UI-Router ملزم به استفاده از دایرکتیو ui-sref نیستید و می‌توانید href را مشخص کنید. ولی با استفاده از ui-sref لازم نیست مسیر یک حالت را به یاد داشته باشید، و یا در صورت تغییر آن، همه hrefها را به روز کنید.

در ادامه برای هر کدام از حالت‌ها یک template اضافه می‌کنیم:

فایل state1.html:

```
<!-- partials/state1.html -->
<h1>State 1</h1>
<hr/>
<a ui-sref="state1.list">Show List</a>
<div ui-view></div>
```

فایل state2.html:

```
<!-- partials/state2.html -->
<h1>State 2</h1>
<hr />
<a ui-sref="state2.list">Show List</a>
<div ui-view></div>
```

دو نکته قابل توجه در این templateها وجود دارد. اول اینکه همانطور که می‌بینید templateها خود شامل تگی با دایرکتیو ui-view هستند. و دوم مقدار دایرکتیو ui-sref است که به صورت state1.list و state2.list آمده است. این جدا سازی با نقطه نشان دهنده سلسله مراتب حالت‌هاست. یعنی حالت‌های state1 و state2 هرکدام حالت فرزندی به نام list دارند. در ادامه وقتی حالت‌ها و مسیریابی را در app.config() تعریف کنیم این مسائل از هاله‌ای از ابهام که در آن هستند خارج می‌شوند! فعلا بیاید با راهنمای UI-Router پیش برویم و فایل‌های template حالت‌های فرزند را تعریف کنیم. templateهایی که قرار است در ui-view پدرانشان بارگذاری شوند:

```
<!-- partials/state1.list.html -->
<h3>List of State 1 Items</h3>
<ul>
  <li ng-repeat="item in items">{{ item }}</li>
</ul>
```

```
<!-- partials/state2.list.html -->
<h3>List of State 2 Things</h3>
<ul>
  <li ng-repeat="thing in things">{{ thing }}</li>
</ul>
```

خوب! حالا برویم سراغ شعبده بازی! برای اینکه از UI-Router استفاده کنید لازم است $stateProvider و $urlRouterProvider را به عنوان وابستگی به app.config() تزریق کنید:

```
myApp.config(['$stateProvider', '$urlRouterProvider',
function($stateProvider, $urlRouterProvider) {
  //
  // For any unmatched url, redirect to /state1
  $urlRouterProvider.otherwise("/state1");
  //
  // Now set up the states
  $stateProvider
    .state('state1', {
      url: "/state1",
      templateUrl: "partials/state1.html"
    })
    .state('state1.list', {
      url: "/list",
      templateUrl: "partials/state1.list.html",
      controller: function($scope) {
        $scope.items = ["A", "List", "Of", "Items"];
      }
    })
```

```
    .state('state2', {
      url: "/state2",
      templateUrl: "partials/state2.html"
    })
    .state('state2.list', {
      url: "/list",
        templateUrl: "partials/state2.list.html",
        controller: function($scope) {
          $scope.things = ["A", "Set", "Of", "Things"];
        }
      })
    }]);
```

در ابتدا با متد ()$urlRouterProvider.otherwise مسیر پیشفرض مشخص شده است. متد otherwise را باید از مقالات مسیریابی در AngularJS به یاد داشته باشید. سپس حالت‌های برنامه با استفاده از متد state تعریف شده است. این متد دو پارامتر ورودی دارد؛ اولی نام حالت و دومی یک شی شامل خصوصیات حالت. همانطور که می‌بینید این شی خصوصیاتی شبیه به همان‌ها که در متد ()$routeProvider.when وجود داشت دارد. می‌شود گفت این خصوصیات همان‌ها هستند و همان عملکرد را دارند.

خصوصیت url مشخص کننده مسیر حالت است. این خصوصیت همان مقداریست که به عنوان پارامتر اول به ()$routeProvider.when پاس می‌شد. در این پارامتر می‌شود متغیرهای url را هم به همان ترتیب تعریف کرد. مثلا اگر حالت state1 در آدرسش یک پارامتر id داشته باشد می‌شود آن را به این ترتیب تعریف کرد:

```
.state('state1', {
      url: "/state1/:id",
      templateUrl: "partials/state1.html"
    })
```

برای خواندن مقدار این متغیر باید از $stateParams استفاده کرد:

```
$stateParams.id
```

به خصوصیت url دو حالت state1.list و state2.list دقت کنید. هردو برابر '/list' است. یعنی هردو یک مسیر دارند؟ نه! بلکه مسیر state1.list برابر '/state1/list' و مسیر state2.list برابر '/state2/list' است. در واقع حالت state1.list یعنی list فرزند state1 و به همین ترتیب state2.list یعنی list فرزند state2. و می‌توان گفت UI-Router آدرس url حالت فرزند را، آدرسی نسبی، نسبت به url حالت پدر می‌داند. این رابطه سلسله مراتبی و پدر و فرزندی را می‌توان با استفاده از خصوصیت parent به صورت صریح‌تری مشخص کرد:

```
.state('list', {
    parent: "state1",
    url: "/list",
    templateUrl: "partials/state1.list.html",
    controller: function($scope) {
        $scope.items = ["A", "List", "Of", "Items"];
    }
})
.state('list', {
    parent: "state2",
    url: "/list",
    templateUrl: "partials/state2.list.html",
    controller: function($scope) {
        $scope.items = ["A", "List", "Of", "Items"];
    }
})
```

تا اینجای کار، اگر آدرس "/state1" وارد شود، فایل "partials/state1.html" در "ui-view" فایل "index.html" بارگذاری خواهد شد. اگر آدرس "/state1/list" وارد شود، ابتدا فایل "partials/state1.html" در "ui-view" فایل "index.html" بارگذاری شده، سپس فایل "partials/state1.list.html" در "ui-view" آمده در فایل فایل "partials/state1.html" بارگذاری می‌شود. این همان امکان حالت‌ها و viewهای تو در تو است که UI-Router فراهم می‌کند. اینجا می‌توانید خروجی کدهای بالا را مشاهده کنید.

اگر مستقیما url یک حالت فرزند وارد شود، یا به عبارت دیگر، اگر بخواهیم مستقیما برنامه به حالتی که فرزند حالت دیگر است برود، UI-Router برنامه را ابتدا به حالت پدر، و پس از آن به حالت فرزند خواهد برد. حالت فرزند دو چیز را از حالت پدر به ارث

می‌برد:
وابستگی‌های فراهم شده در حالت پدر به وسیله " resolve "

داده‌های سفارشی مشخص شده در خصوصیت data حالت پدر

استفاده از resolve در UI-Router مشابه استفاده از آن در $route است. ولی افزودن داده‌های سفارشی کمی متفاوت است.
برای افزودن داده‌های سفارشی باید از خصوصیت data یک حالت استفاده کرد:

```
.state('state1', {
    url: "/state1",
    templateUrl: "partials/state1.html",
    data:{
        foodata: 'addorder'
    }
})
```

برای دسترسی به این داده‌ها هم می‌توان از $state.current.data استفاده کرد:

```
$state.current.data.foodata
```

Viewهای نامگذاری شده و چندگانه

یکی دیگر از قابلیت‌های کاربردی UI-Router امکان داشتن چند ui-view در هر template است (استفاده همزمان از این قابلیت و حالت‌های تو در تو، امکان مدیریت واسط کاربری را به خوبی فراهم می‌کند). برای توضیح این قابلیت، با راهنمای UI-Router همراه شویم:

1. دستورالعمل برپایی UI-Router که در بالا آمده را اجرا کنید.
2. یک یا چند ui-view به برنامه‌تان اضافه کنید و آن‌ها را نامگذاری کنید:

```
<!-- index.html -->
<body>
    <div ui-view="viewA"></div>
    <div ui-view="viewB"></div>
    <!-- Also a way to navigate -->
    <a ui-sref="route1">Route 1</a>
    <a ui-sref="route2">Route 2</a>
</body>
```

3. حالت‌های برنامه‌تان را در روال config ماژول تعریف کنید:

```
myApp.config(function ($stateProvider) {
    $stateProvider
      .state('index', {
          url: "",
          views: {
              "viewA": { template: "index.viewA" },
              "viewB": { template: "index.viewB" }
          }
      })
      .state('route1', {
          url: "/route1",
          views: {
              "viewA": { template: "route1.viewA" },
              "viewB": { template: "route1.viewB" }
          }
      })
      .state('route2', {
          url: "/route2",
          views: {
              "viewA": { template: "route2.viewA" },
              "viewB": { template: "route2.viewB" }
          }
```

```
    })
});
```

4. خروجی کدهای بالا را اینجا مشاهده کنید.

چند نکته

UI-Router جزئیات فراوانی دارد و آنچه آمد تنها پرده برداری از آن بود. دلم می‌خواست می‌توانستم بیش از این آن را معرفی کنم، اما متاسفانه این روزها وقت آزاد کافی ندارم. در انتها می‌خواهم به چند نکته اشاره کنم: **روش controller as**

برای استفاده از روش controller as در UI-Router باید به این ترتیب عمل کنید:

```
.state('list', {
    parent: "state1",
    url: "/list",
    templateUrl: "partials/state1.list.html",
    controller: "state1ListController as listCtrl1"
    }
})
.state('list', {
    parent: "state2",
    url: "/list",
    templateUrl: "partials/state2.list.html",
    controller: "state2ListController as listCtrl2"
    }
})
```

حالت‌های انتزاعی

حالت انتزاعی حالتی است که url ندارد و در نتیجه برنامه نمی‌تواند در آن حالت قرار گیرد. حالت‌های انتزاعی بسیار به درد خور هستند! مثلا فرض کنید چند حالت دارید که اشتراکاتی با هم دارند (همه باید در template مشابهی بارگذاری شود، یا وابستگی‌های یکسانی دارند، یا حتی سطح دسترسی یکسان). با تعریف یک حالت انتزاعی و جمع کردن همه وابستگی‌ها در آن، و تعریف حالت‌های مورد نظرتان به عنوان فرزندان حالت انتزاعی، می‌توانید اشتراکات حالت‌های برنامه را ساده‌تر مدیریت کنید.

حساسیت به حروف بزرگ و کوچک

در سرویس $route با مقداردهی خصوصیت caseInsensitiveMatch می‌توانستیم مشخص کنیم که بزرگ و کوچک بودن حروف در تطبیق آدرس صفحه با پارامتر route در نظر گرفته بشود یا نه . خودمانیش اینکه url به حروف بزرگ و کوچک حساس باشد یا نه. متاسفانه در UI-Router از این امکان خبری نیست (البته فعلا) و آدرس‌های تعریف شده به حروف بزگ و کوچک حساس هستند. اینجا روشی برای حل این مشکل پیشنهاد شده، به این ترتیب که همه urlهای وارد شده به حروف کوچک تبدیل شود (راستش من این راه حل را نمی‌پسندم!). چند روز قبل هم تغییراتی در کد UI-Router داده شده که امکان حساس نبودن به حروف کوچک و بزرگ فراهم شود . این تغییر هنوز در نسخه نهایی فایل UI-Router نیامده است. هرچند اگر بیاید هم آنچه تا امروز (23 اسفند 92) انجام شده مشکل را حل نمی‌کند.

اگر شما هم مثل من می‌خواهید کلا آدرس‌ها به حروف بزرگ و کوچک حساس نباشند، و فرصت حل کردن اساسی مشکل را هم ندارید به این ترتیب عمل کنید:

در فایل "angular-ui-router.js" عبارت "(new RegExp(compiled" را پیدا کرده و آن را به "('RegExp(compiled, 'i" تبدیل کنید. و یا در "angular-ui-router.min.js" (هرکدام از فایل‌ها که استفاده می‌کنید) عبارت new RegExp(o) را پیدا کرده و آن را به new RegExp(o, "i") تبدیل کنید. همین؛ صدایش را هم در نیاورید!

وقتی پروژه انگیولاری‌تان کمی گسترش پیدا کند، تعداد زیادی فایل شامل کنترلرها، سرویس‌ها، دایرکتیوها و ... خواهید داشت. واضح است که همه این اجزا همراه با هم مورد نیاز نیستند و برای افزایش سرعت بارگذاری سایت و صرفه جویی در مصرف پهنای باند بهتر است هرکدام از آن‌ها را در هنگام نیاز بارگذاری کنیم. این یعنی همان lazy loading خودمان!

در AngularJS امکانی برای lazy loading فایل‌ها پیشبینی نشده است، پس باید از ابزارهای دیگری که این امکان را فراهم می‌کنند استفاده کرد. من در ادامه از Script.js برای این کار استفاده خواهم کرد، ولی شما می‌توانید از هر کتابخانه دیگری استفاده کنید.

اما مسئله دیگری که پیش از lazy loading فایل‌ها باید تکلیفش را معلوم کنیم، این است که چطور می‌توانیم اجزایی را به ماژولی که قبلا راه‌اندازی (bootstrap) شده اضافه کنیم. اگر بخواهیم برای مثال کنترلری را در یک فایل مجزا تعریف کنیم، باید آن را به شکلی در ماژول برنامه‌مان ثبت کنیم. فرض کنید این کار را به این ترتیب انجام دهیم:

```
angular.module('app').controller('SomeLazyController', function($scope)
{
    $scope.key = '...';
});
```

در این صورت اگر این کنترلر را در قسمتی از برنامه به صورت 'ng-controller='SomeLazyController استفاده کنیم با این خطا مواجه خواهیم شد:

```
Error: Argument ‘SomeLazyController’ is not a function, got undefined
```

برای این کار (افزودن اجزایی به ماژولی که قبلا راه اندازی شده) می‌توانیم بجای استفاده از APIهای ماژول، از provider های AngularJS استفاده کنیم. به این ترتیب برای ثبت یک کنترلر باید از $controllerProvider ، برای ثبت یک directive از $compileProvider ، برای ثبت فیلترها از $filterProvider و برای ثبت سایر اجزا در ماژول از $provide استفاده کنیم:

```
// Registering a controller after app bootstrap
$controllerProvider.register('SomeLazyController', function($scope)
{
   $scope.key = '...';
});

// Registering a directive after app bootstrap
$compileProvider.directive('SomeLazyDirective', function()
{
    return {
        restrict: 'A',
        templateUrl: 'templates/some-lazy-directive.html'
    }
})

// etc
```

اما نکته‌ای که درباره providerها وجود دارد این است که آن‌ها تنها در روال config یک ماژول در دسترس هستند. بنا بر این برای دسترسی به آن‌ها پس از اجرای این روال، ارجاعی به آنها را باید نگهداری کنیم:

```
(function () {
    app = angular.module("app", []);

    app.config([
        '$controllerProvider',
        '$compileProvider',
        '$filterProvider',
        '$provide',
```

```
    function ($controllerProvider, $compileProvider, $filterProvider, $provide) {
        //برای رجیستر کردن غیر همروند اجزای انگیولاری در آینده
        app.lazy =
        {
            controller: $controllerProvider.register,
            directive: $compileProvider.directive,
            filter: $filterProvider.register,
            factory: $provide.factory,
            service: $provide.service
        };
    }]);
})();
```

اکنون SomeLazyController را به این ترتیب می‌توانیم ثبت کنیم:

```
angular.module('app').lazy.controller('SomeLazyController', function($scope)
{
    $scope.key = '...';
});
```

نکته دیگر این است که کجا باید lazy loadign را انجام دهیم. به نظر می‌رسد مناسب‌ترین محل برای انجام این کار خصوصیت resolve مسیریابی است. در این مطلب و این مطلب resolve در $route و UI-Router معرفی شده است:

```
$stateProvider
            .state('state1', {
                url: '/state1',
                template: '<div>{{st1Ctrl.msg}}</div>',
                controller: 'state1Controller as st1Ctrl',
                resolve: {
                    fileDeps: ['$q', '$rootScope', function ($q, $rootScope) {
                        var deferred = $q.defer();
                        var deps = [
                            'app/messageService.js',
                            'app/state1Controller.js'];
                        $script(deps, function () {
                            $rootScope.$apply(function () {
                                deferred.resolve();
                            });
                        });
                        return deferred.promise;
                    }]
                }
            })
            .state('state2', {
                url: '/state2',
                template: '<div>{{st2Ctrl.msg}}</div>',
                controller: 'state2Controller as st2Ctrl',
                resolve: {
                    fileDeps: ['$q', '$rootScope', function ($q, $rootScope) {
                        var deferred = $q.defer();
                        var deps = [
                            'app/messageService.js',
                            'app/state2Controller.js'];
                        $script(deps, function () {
                            $rootScope.$apply(function () {
                                deferred.resolve();
                            });
                        });
                        return deferred.promise;
                    }]
                }
            });
    }]);
```

کنترلر state1Controller که در فایلی با همین نام پیاده‌سازی شده است تنها در مسیر state1/ مورد نیاز است، و state2Controller تنها در مسیر state2/ لازم است بارگذاری شود. هردوی این کنترلرها به messageService وابستگی دارند که در messageService.js پیاده سازی شده است (همانطور که در این مطلب اشاره شده می‌توانیم یک حالت انتزاعی به عنوان پدر دو حالت موجود تعریف کرده و وابستگی مشترک را به آن منتقل کنیم).

برای بارگذاری فایل‌های مورد نیاز در ابتدای کار و راه اندازی اولیه برنامه هم می‌توان به این ترتیب عمل کرد:

```
<script type="text/javascript">
        // ----Script.js----
        !function (a, b, c) { function t(a, c) { var e = b.createElement("script"), f = j; e.onload =
e.onerror = e[o] = function () { e[m] && !/^c|loade/.test(e[m]) || f || (e.onload = e[o] = null, f = 1,
c()) }, e.async = 1, e.src = a, d.insertBefore(e, d.firstChild) } function q(a, b) { p(a, function (a)
{ return !b(a) }) } var d = b.getElementsByTagName("head")[0], e = {}, f = {}, g = {}, h = {}, i =
"string", j = !1, k = "push", l = "DOMContentLoaded", m = "readyState", n = "addEventListener", o =
"onreadystatechange", p = function (a, b) { for (var c = 0, d = a.length; c < d; ++c) if (!b(a[c]))
return j; return 1 }; !b[m] && b[n] && (b[n](l, function r() { b.removeEventListener(l, r, j), b[m] =
"complete" }, j), b[m] = "loading"); var s = function (a, b, d) { function o() { if (!--m) { e[l] = 1,
j && j(); for (var a in g) p(a.split("|"), n) && !q(g[a], n) && (g[a] = []) } } function n(a) { return
a.call ? a() : e[a] } a = a[k] ? a : [a]; var i = b && b.call, j = i ? b : d, l = i ? a.join("") : b, m
= a.length; c(function () { q(a, function (a) { h[a] ? (l && (f[l] = 1), o()) : (h[a] = 1, l && (f[l] =
1), t(s.path ? s.path + a + ".js" : a, o)) }) }, 0); return s }; s.get = t, s.ready = function (a, b,
c) { a = a[k] ? a : [a]; var d = []; !q(a, function (a) { e[a] || d[k](a) }) && p(a, function (a) {
return e[a] }) ? b() : !function (a) { g[a] = g[a] || [], g[a][k](b), c && c(d) }(a.join("|")); return
s }; var u = a.$script; s.noConflict = function () { a.$script = u; return this }, typeof module !=
"undefined" && module.exports ? module.exports = s : a.$script = s }(this, document, setTimeout)

        $script('Scripts/angular.js', function () {
            $script('Scripts/angular-ui-router.js', function () {
                $script('app/app.js', function () {
                    angular.bootstrap(document, ['app']);
                });
            });
        });
    </script>
```

توجه داشته باشید که لازم نیست بارگذاری فایل‌ها حتما یکی پس از دیگری باشد. ترتیب بارگذاری فایل‌ها تنها در آن‌هایی که وابستگی به هم دارند باید رعایت شود. همچنین، می‌توانید همه فایل‌های مورد نیاز در این مرحله را Bundle کنید. از اینجا می‌توانید پروژه بسیار ساده‌ای که در آن lazy loading پیاده شده است را دانلود کرده و مطالب توضیح داده شده را مشاهده کنید.

نظرات خوانندگان

نویسنده: علی فخرائی
تاریخ: ۱۳۹۳/۰۱/۱۰ ۱۵:۵۲

با تشکر.
اگر ممکن است یک نمونه از تعریف دایرکتیو توسط lazy را هم بنویسید.

نویسنده: حمید صابری
تاریخ: ۱۳۹۳/۰۱/۱۰ ۱۸:۲۲

سلام.
یک لینک به index.html اضافه کنید:

```
<div style="direction: rtl">
        <a href="#/state1">1 حالت</a> |
        <a href="#/state2">2 حالت</a> |
        <a href="#/state3">3 حالت</a>
        <div ui-view style="font-weight:bold; text-align:center;"></div>
    </div>
```

فرض کنید محتویات مورد نظر برای این حالت که در فایل app/state3.html قرار دارد، شامل یک دایرکتیو است: state3.html:

```
تگ زیر یک دایرکتیو دارد:
<br/>
<div ng-hello-directive></div>
```

ng-hello-directive در فایل app/helloDirective.js به این صورت تعریف شده است:

```
angular.module('app').lazy.directive('ngHelloDirective', function () {
    return function (scope, elem, attr) {
        elem.html('سلام دایرکتیو تنبل!');
    };
});
```

و در نهایت حالت sate3 را با آدرس state3/ در app.js تعریف کنید:

```
.state('state3', {
            url: '/state3',
            templateUrl: 'app/state3.html',
            resolve: {
                fileDeps: ['$q', '$rootScope', function ($q, $rootScope) {
                    var deferred = $q.defer();
                    var deps = ['app/helloDirective.js'];
                    $script(deps, function () {
                        $rootScope.$apply(function () {
                            deferred.resolve();
                        });
                    });
                    return deferred.promise;
                }]
            }
        });
```

از اینجا می‌توانید پروژه مثال را که این دایرکتیو به آن افزوده شده دانلود کنید.
دقت کنید که در این حالت، این دایرکتیو تنها در ماژولی با نام app که خصوصیتی به نام lazy به صورت توضیح داده شده دارد ثبت می‌شود. اگر تابحال دایرکتیو آماده‌ای را دریافت کرده باشید، دیده‌اید که این دایرکتیوها به این صورت تعریف می‌شوند:

```
angular.module('moduleOfDirective', []).directive('ngDirectiveName', ...
```

همانطور که می‌بینید یک ماژول جدید تعریف شده و دایرکتیو در آن ثبت شده است. برای استفاده از چنین دایرکتیوی باید ماژولِ

دایرکتیو را به وابستگی‌های ماژول خودتان اضافه کنید:

```
app = angular.module("app", ['ui.router', 'moduleOfDirective']);
```

در این حالت حتما باید فایل دایرکتیو را پیش از فایل app خود بارگذاری کرده باشید. یا اینکه تعریف دایرکتیو را تغییر دهید و بجای تعریف ماژول جدید، آن را به همان ماژول خودتان اضافه کنید. یعنی تعریف دایرکتیو را به این شکل تغییر دهید:

```
angular.module('app', []).lazy.directive('ngDirectiveName', ...
```

حالا این دایرکتیو را هم می‌توانید *تنبلانه!* بارگذاری کنید.

نویسنده: علی فخرائی
تاریخ: ۱۳۹۳/۰۱/۱۴ ۱۲:۴۸

بسیار ممنون جناب صابری.
شما در قسمت مسیریابی هم نام کنترلر را وارد کرده اید:

```
.state('state2', {
                url: '/state2',
                template: '<div>{{st2Ctrl.msg}}</div>',
                controller: 'state2Controller as st2Ctrl',
```

و هم فایل مرتبط را به عنوان وابستگی تعریف کرده اید:

```
var deps = ['app/messageService.js',
                 'app/state2Controller.js'];
```

اما چرا محتوای کنترلر state2Controller.js دو بار اجرا میشود؟یعنی با هر بار تغییر مسیر، 2 بار کل محتوای کنترلر اجرا میشود. مثلا اگر 1 تابع را در کنترلر صدا زده باشیم، این تابع 2 بار اجرا میشود.

نویسنده: ناصر طاهری
تاریخ: ۱۳۹۳/۰۱/۱۴ ۱۳:۳۹

چک کنید ببینید در قسمت کدهای HTML ، ویژگی ای به نام ng-controller که به کنترلر شما اشاره کند وجود نداشته باشد

عنوان: **بررسی خروجی IsAjaxRequest در درخواست های $http توسط AngularJS**
نویسنده: ناصر طاهری
تاریخ: ۲۱:۱۰ ۱۳۹۳/۰۱/۱۲
آدرس: www.dotnettips.info
گروه‌ها: MVC, Security, AngularJS

در یکی از پروژه‌هایی که دارم از AngularJS و ASP.NET MVC استفاده میکنم. به هنگام استفاده از درخواست‌های ایجکسی توسط سرویس $http به مشکل عدم تشخیص ایجکسی بودن درخواست برخوردم.
توسط فیلتری که در اینجا توضیح داده شده و قرار دادن آن قبل از اکشن مورد نظر، میتوانیم تشخیص بدهیم که آیا درخواست رسیده از سمت کلاینت، ایجکسی است یا خیر؟ که در صورت ایجکسی نبودن درخواست، با صادر کردن یک استثنا مانع از اجرا شدن اکشن شویم. این فیلتر از اکستنشنی به نام IsAjaxRequest برای این تشخیص استفاده میکند:

```
HttpContext.Request.IsAjaxRequest();
```

اما هنگام استفاده از سرویس $http ، اکستنشن IsAjaxRequest() همیشه مقدار False را برمیگرداند. در حالیکه با متدهای ساده ایجکسی Jquery مثل $.get و... ، مقدار این اکستنشن True میشود و به خوبی هم کار میکند.
درخواست‌های هر دو مورد را که با فایرباگ بررسی کردم به این مقادیر برخوردم.
ویژگی‌های درخواست توسط Jquery - $.get

```
Request Headers
        Accept  application/json, text/plain, */*
Accept-Encoding  gzip, deflate
Accept-Language  fa-ir,en-us;q=0.7,en;q=0.3
        Cookie  MyLanguageCookieName=fa-IR; __RequestVerificationToken=ZuW8imM5USKOyBH2kih4o
                T38mP9854FoY2y817pXhqjOyO9eQ8VUTsln9A_ufopXaC0btSjSKb8A0auWz5hFd5qiaiVO7eZxY
                ; captchastring=55-57-EB-27-6B-06-A7-C4-CC-93-9F-4A-C8-AF-28-63-93-9E-8E-6E-
                3B-7E-5F-1E-32-7D-66-30-3A-1C-FA-A2-DB-B4-43-10-BF-E7-02-56-CF-63-19-56-00-E
                A-2C-E8-45-B8-81-54-3B-60-84-12-9C-AA-19-4F-18-15-D2-82-5E-2D-02-46-C5-5E-D0-
                ; .403MyApp=CE5EF7C684F13C1FAA8DB8F8B43E14E47587D7E9C590EB0D492620F5E01213F7
                0D6E99B1571F1CEC9E3B97BAF77F3FB5C95E92A7462B6490D6B5ECD4FF38F642DE82D8675536
                1EF24201B7C6F3D56B9083A145D8F57B15FA3805E766CBA746A4FC16EA2A6726DC96
           Host  localhost:8417
        Referer  http://localhost:8417/administrator/dashboard
     User-Agent  Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0
X-Requested-With  XMLHttpRequest
```

و ویژگی‌های درخواست توسط سرویس AngularJs - $http :

Request Headers

Accept	application/json, text/plain, */*
Accept-Encoding	gzip, deflate
Accept-Language	fa-ir,en-us;q=0.7,en;q=0.3
Content-Length	112
Content-Type	application/json;charset=utf-8
Cookie	MyLanguageCookieName=fa-IR; __RequestVerificationToken=ZuW8imM5USKOyBH2kih4T38mP9854FoY2y817pXhqjOyO9eQ8VUTsln9A_ufopXaC0btSjSKb8A0auWz5hFd5qiaiVO7eZx; captchastring=55-57-EB-27-6B-06-A7-C4-CC-93-9F-4A-C8-AF-28-63-93-9E-8E-6E3B-7E-5F-1E-32-7D-66-30-3A-1C-FA-A2-DB-B4-43-10-BF-E7-02-56-CF-63-19-56-00-A-2C-E8-45-B8-81-54-3B-60-84-12-9C-AA-19-4F-18-15-D2-82-5E-2D-02-46-C5-5E-D; .403MyApp=CE5EF7C684F13C1FAA8DB8F8B43E14E47587D7E9C590EB0D492620F5E01213F0D6E99B1571F1CEC9E3B97BAF77F3FB5C95E92A7462B6490D6B5ECD4FF38F642DE82D867553; .ASPXROLES=pZCIOmMLs7NqH6S316M05D191zEnr47gvjGCS7ex068PRurznIYsonRQJZ-eAIOyU0RQWwAYiw-fNGa6q6MK0zCso9uQWGiP-O0aezj2yC9wV_xDYY40C0g8kEl5yqMlhmMY4hmkWU6-IlqkRNG9inON8e4ybE5Yc00uZbh-XN0GPgWPM1BvS4i2J3dcMn00af2-f4wPL4qvakegYDJCkMR2NAx8Ku7HzV2JvpKcYpyfQLF-WMo19yK6fUKJwlTJXgWaZVzs4Jkq-GO9RmRuXBAsDXJTYmJMlyWpHICBwkvj_9hQy2Bdf2jggtYTVEE7XLNPw35nuiRUlgbkT3pCVd_Ipoa77aXXz0ClA1j1C-aQWZq__IErhh-4ysOTa1TJgFP-AloFKsD6Tke3a5Q1zqvKQQIOY9SohCEI-NJSNpl61VhpJBtPK
Host	localhost:8417
Referer	http://localhost:8417/administrator/dashboard
User-Agent	Mozilla/5.0 (Windows NT 6.1; WOW64; rv:28.0) Gecko/20100101 Firefox/28.0

همینطور که میبینید، در هدر درخواست $http یک مورد مفقود الاثر شده به نام X-Requested-With داریم و همین مقدار است که مشخص میکند این یک درخواست ایجکسی است یا خیر و اکستنشن IsAjaxRequest() نیر با همین مقدار عمل تشخیص را انجام میدهد. و به همین خاطر بود که این متد مقدار False را برمیگرداند.

بعد از کمی جستجو در این مورد ، به مخزن git انگیولار رسیدم و به صراحت به این موضوع اشاره شده بود که این هدر به صورت پیشفرض از درخواست‌های $http برداشته شده است.

بنابراین تنها راه حل این بود که خودمان به صورت دستی این هدر خاص رو به ماژول برنامه اضافه کنیم. به صورت زیر :

```
myAppModule.config(['$httpProvider', function($httpProvider) {
    $httpProvider.defaults.headers.common["X-Requested-With"] = 'XMLHttpRequest';
}]);
```

با اضافه کردن این هدر به درخواست‌های $http ، اکستنشن IsAjaxRequest() مقدار درست را برمیگرداند.

عنوان: **زیرنویس فارسی ویدئوهای مقدمات AngularJS - قسمت سوم**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۱/۲۵ ۲۳:۵۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, Subtitle

زیرنویس‌های فارسی قسمت سوم را اینجا می‌توانید دریافت کنید.
لیست سرفصل‌های قسمت سوم به شرح زیر است :

```
01. Introduction to Services
02. Demo - Creating Your First Custom Service
03. Demo - Another Custom Service Example
04. Introduction to Built-In AngularJS Services
05. Demo - Using the $http and $q Services Together
06. Demo - Using the $rsource and $q Services
07. Demo - Using the $anchorScroll Service
08. Demo - Using the $cacheFactory Service
09. Demo - Using the $compile Service
10. Demo - Using the $parse Service
11. Demo - Using the $locale Service
12. Demo - Using the $timeout Service
13. Demo - Using the $exceptionHandler Service
14. Demo - Using the $filter Service
15. Demo - Using the $cookieStore Service
16. Overview of Less Common Services
17. Suggested Exercises
```

این قسمت به ساخت سرویس‌های سفارشی و همچنین چگونگی استفاده از سرویس‌های توکار انگولار می‌پردازد؛ در این قسمت تقریباً تمامی سرویس‌های موردنیاز جهت توسعه یک برنامه مبتنی بر انگولار شرح داده می‌شود. همچنین در آن به صورت عملی با سرویس‌ها آشنا می‌شوید و هر سرویسی که معرفی میشود مثال مربوط به آن ارائه شده است. به طور مثال در قسمت پیاده سازی سرویس‌های سفارشی مثال نمایش Gravatar با وارد کردن آدرس ایمیل کاربر مطرح می‌شود که در ادامه کد آن را مشاهده میکنید :

```
'use strict';

eventsApp.factory('gravatarUrlBuilder', function() {
return {
buildGravatarUrl: function(email) {
        // try angularjsdemo@gmail.com as an email
            var defaultGravatarUrl = "http://www.gravatar.com/avatar/000?s=200";

            var regex = /^(([^<>()[\]\\.,;:\s@\"]+(\.[^<>()[\]\\.,;:\s@\"]+)*)|(\".+\"))@((\[[0-
9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
            if (!regex.test(email))
                return defaultGravatarUrl;

            var MD5=function(s){function L(k,d){return(k<<d)|(k>>>(32-d))}function K(G,k){var
I,d,F,H,x;F=(G&2147483648);H=(k&2147483648);I=(G&1073741824);d=(k&1073741824);x=(G&1073741823)+(k&10737
41823);if(I&d){return(x^2147483648^F^H)}if(I|d){if(x&1073741824){return(x^3221225472^F^H)}else{return(x
^1073741824^F^H)}}else{return(x^F^H)}}function r(d,F,k){return(d&F)|((~d)&k)}function
q(d,F,k){return(d&k)|(F&(~k))}function p(d,F,k){return(d^F^k)}function
n(d,F,k){return(F^(d|(~k)))}function u(G,F,aa,Z,k,H,I){G=K(G,K(K(r(F,aa,Z),k),I));return
K(L(G,H),F)}function f(G,F,aa,Z,k,H,I){G=K(G,K(K(q(F,aa,Z),k),I));return K(L(G,H),F)}function
D(G,F,aa,Z,k,H,I){G=K(G,K(K(p(F,aa,Z),k),I));return K(L(G,H),F)}function
t(G,F,aa,Z,k,H,I){G=K(G,K(K(n(F,aa,Z),k),I));return K(L(G,H),F)}function e(G){var Z;var F=G.length;var
x=F+8;var k=(x-(x%64))/64;var I=(k+1)*16;var aa=Array(I-1);var d=0;var H=0;while(H<F){Z=(H-
(H%4))/4;d=(H%4)*8;aa[Z]=(aa[Z]|(G.charCodeAt(H)<<d));H++}Z=(H-
(H%4))/4;d=(H%4)*8;aa[Z]=aa[Z]|(128<<d);aa[I-2]=F<<3;aa[I-1]=F>>>29;return aa}function B(x){var
k="",F="",G,d;for(d=0;d<=3;d++){G=(x>>>(d*8))&255;F="0"+G.toString(16);k=k+F.substr(F.length-
2,2)}return k}function J(k){k=k.replace(/rn/g,"n");var d="";for(var F=0;F<k.length;F++){var
x=k.charCodeAt(F);if(x<128){d+=String.fromCharCode(x)}else{if((x>127)&&(x<2048)){d+=String.fromCharCode
((x>>6)|192);d+=String.fromCharCode((x&63)|128)}else{d+=String.fromCharCode((x>>12)|224);d+=String.from
CharCode(((x>>6)&63)|128);d+=String.fromCharCode((x&63)|128)}}}return d}var C=Array();var
P,h,E,v,g,Y,X,W,V;var S=7,Q=12,N=17,M=22;var A=5,z=9,y=14,w=20;var o=4,m=11,l=16,j=23;var
U=6,T=10,R=15,O=21;s=J(s);C=e(s);Y=1732584193;X=4023233417;W=2562383102;V=271733878;for(P=0;P<C.length;
P+=16){h=Y;E=X;v=W;g=V;Y=u(Y,X,W,V,C[P+0],S,3614090360);V=u(V,Y,X,W,C[P+1],Q,3905402710);W=u(W,V,Y,X,C[
P+2],N,606105819);X=u(X,W,V,Y,C[P+3],M,3250441966);Y=u(Y,X,W,V,C[P+4],S,4118548399);V=u(V,Y,X,W,C[P+5],
Q,1200080426);W=u(W,V,Y,X,C[P+6],N,2821735955);X=u(X,W,V,Y,C[P+7],M,4249261313);Y=u(Y,X,W,V,C[P+8],S,17
70035416);V=u(V,Y,X,W,C[P+9],Q,2336552879);W=u(W,V,Y,X,C[P+10],N,4294925233);X=u(X,W,V,Y,C[P+11],M,2304
563134);Y=u(Y,X,W,V,C[P+12],S,1804603682);V=u(V,Y,X,W,C[P+13],Q,4254626195);W=u(W,V,Y,X,C[P+14],N,27929
65006);X=u(X,W,V,Y,C[P+15],M,1236535329);Y=f(Y,X,W,V,C[P+1],A,4129170786);V=f(V,Y,X,W,C[P+6],z,32254656
64);W=f(W,V,Y,X,C[P+11],y,643717713);X=f(X,W,V,Y,C[P+0],w,3921069994);Y=f(Y,X,W,V,C[P+5],A,3593408605);
```

```
V=f(V,Y,X,W,C[P+10],z,38016083);W=f(W,V,Y,X,C[P+15],y,3634488961);X=f(X,W,V,Y,C[P+4],w,3889429448);Y=f(
Y,X,W,V,C[P+9],A,568446438);V=f(V,Y,X,W,C[P+14],z,3275163606);W=f(W,V,Y,X,C[P+3],y,4107603335);X=f(X,W,
V,Y,C[P+8],w,1163531501);Y=f(Y,X,W,V,C[P+13],A,2850285829);V=f(V,Y,X,W,C[P+2],z,4243563512);W=f(W,V,Y,X
,C[P+7],y,1735328473);X=f(X,W,V,Y,C[P+12],w,2368359562);Y=D(Y,X,W,V,C[P+5],o,4294588738);V=D(V,Y,X,W,C[
P+8],m,2272392833);W=D(W,V,Y,X,C[P+11],l,1839030562);X=D(X,W,V,Y,C[P+14],j,4259657740);Y=D(Y,X,W,V,C[P+
1],o,2763975236);V=D(V,Y,X,W,C[P+4],m,1272893353);W=D(W,V,Y,X,C[P+7],l,4139469664);X=D(X,W,V,Y,C[P+10],
j,3200236656);Y=D(Y,X,W,V,C[P+13],o,681279174);V=D(V,Y,X,W,C[P+0],m,3936430074);W=D(W,V,Y,X,C[P+3],l,35
72445317);X=D(X,W,V,Y,C[P+6],j,76029189);Y=D(Y,X,W,V,C[P+9],o,3654602809);V=D(V,Y,X,W,C[P+12],m,3873151
461);W=D(W,V,Y,X,C[P+15],l,530742520);X=D(X,W,V,Y,C[P+2],j,3299628645);Y=t(Y,X,W,V,C[P+0],U,4096336452)
;V=t(V,Y,X,W,C[P+7],T,1126891415);W=t(W,V,Y,X,C[P+14],R,2878612391);X=t(X,W,V,Y,C[P+5],O,4237533241);Y=
t(Y,X,W,V,C[P+12],U,1700485571);V=t(V,Y,X,W,C[P+3],T,2399980690);W=t(W,V,Y,X,C[P+10],R,4293915773);X=t(
X,W,V,Y,C[P+1],O,2240044497);Y=t(Y,X,W,V,C[P+8],U,1873313359);V=t(V,Y,X,W,C[P+15],T,4264355552);W=t(W,V
,Y,X,C[P+6],R,2734768916);X=t(X,W,V,Y,C[P+13],O,1309151649);Y=t(Y,X,W,V,C[P+4],U,4149444226);V=t(V,Y,X,
W,C[P+11],T,3174756917);W=t(W,V,Y,X,C[P+2],R,718787259);X=t(X,W,V,Y,C[P+9],O,3951481745);Y=K(Y,h);X=K(X
,E);W=K(W,v);V=K(V,g)}var i=B(Y)+B(X)+B(W)+B(V);return i.toLowerCase()};

            var url = 'http://www.gravatar.com/avatar/' + MD5(email) + ".jpg?s=200&r=g";
            console.log(url);
            return url;
        }
};
});
```

اگر زیرنویس‌ها دارای اشکال هستند می‌توانید در این قسمت فایل‌های اصلاح شده را مجدداً آپلود کنید.

نظرات خوانندگان

نویسنده: داود موسی زاده
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۳:۴۲

با سلام و تشکر
خود ویدیوها رو از کجا باید دانلود کنیم ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۸ ۱۳:۴۶

از جستجوی گوگل استفاده کنید (ajf.7z). آقای اکبرزاده هم قبلا ارسالش کرده بودند (^).

عنوان: **بررسی سرویس های $on و $emit و $broadcast در AngularJs**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۲/۱۴ ۲۳:۴۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در این پست درباره به اشتراک گذاری داده‌ها بین کنترلرهای Angular بحث شد. اما استفاده از Factory و Service فقط زمانی کاربرد دارد که بخواهیم یک منبع داده مشخص را در اختیار مصرف کننده قرار دهیم. اگر قصد داشته باشم بر اساس شرایط خاص، داده یا داده‌های مشخصی در سایر کنترلرها تغییر پیدا کنند چه باید کرد؟ به زبان ساده‌تر برای ایجاد ارتباط بین کنترلرها به طوری که از تغییرات یکدیگر باخبر باشند چه راهکارهایی وجود دارد. $on و $emit و $broadcast برای این منظور تعبیه شده اند.
برای شرح موارد بالا بهترین روش بررسی یک مثال است:

$emit:

دو کنترلر به نام‌های FirstCtrl و SecondCtrl داریم. FirstCtrl به عنوان والد کنترلر Second است(در این مورد در این پست توضیح داده شده است).
پس فایل html نیز به صورت زیر خواهد بود:

```
<body ng-app>
  <div ng-controller="FirstCtrl">
    <p>{{title}}</p>
    <div ng-controller="SecondCtrl">
      <button ng-click="onUpdate()">Update First Ctrl Title</button>
    </div>
  </div>
</body>
```

قصد داریم با کلیک بر روی دکمه Update در کنترلر Second، مقدار خاصیت title در FirstCtrl به روز رسانی شود. اگر دقت کرده باشید حرکت رویداد از پایین به بالاست درنتیجه برای این کار باید از سرویس $emit استفاده کنیم. کافیست در کنترلر دوم (Second) کد زیر را وارد نمایید:

```
function ChildCtrl($scope){
  $scope.onUpdate = function(){
    this.$emit("Update_Title", "Good Bye");
  };
}
```

به وسیله سرویس $emit می‌توان از بروز یک رویداد در کنترلر جاری خبر داد. اگر کنترلر والد یک handler برای این رویداد داشته باشد، می‌تواند مقدار جدید را دریافت نماید. برای تعریف handler باید از سرویس $on استفاده کرد. کدهای کنترلر First را به صورت زیر تغییر دهید.

```
function FirstCtrl($scope){
 $scope.title= "Hello";

 $scope.$on("Update_Title", function(event, message){
   $scope.title= message;
 });
}
```

»سرویس $on برای تعریف handler برای رویداد مورد نظر استفاده می‌شود.
»پارامتر دوم سرویس $on برابر با مقدار جدید ارسال شده توسط سرویس $emit است.
»نام رویدادی که به عنوان پارامتر به $on پاس داده می‌شود باید برابر با نام رویداد پاس داده شده به $emit باشد.
»می‌توان چندین پارامتر را با استفاده از $emit ارسال کرد و در سرویس $on با تعریف متغیر به تعداد پارامترها مقادیر آن‌ها را دریافت نمود.

$broadcast

همان طور که مشاهده کردید SecondCtrl در محدوده FirstCtrl تعریف شده است. در نتیجه به راحتی با استفاده از سرویس $emit توانستیم یک رویداد را منتشر نماییم. اما نکته مهم این است که اگر قصد داشته باشیم یک رویداد را از کنترلر والد (در این جا FirstCtrl است) منتشر نماییم به طوری که در کنترلرهای فرزند قابل دریافت باشد(حرکت رویداد بالا به پایین است)، باید از $broadcast استفاده کنیم.

«$broadcast فقط از نظر کاربرد با $emit متفاوت است و در پیاده سازی کاملا مشابه هستند.

یک مثال:

```
function ParentCtrl($scope){
 $scope.foo = "Hello";

 $scope.$on("UPDATE_PARENT", function(event, message){
   $scope.title= message;

   $scope.$broadcast("DO_BIDDING", {
     buttonTitle : "Taken over",
     onButtonClick : function(){
       $scope.title= "HAHA this button no longer works!";
     }
   });
 });
}

function ChildCtrl($scope){
  $scope.buttonTitle = "Update Parent";
  $scope.onButtonClick = function(){
    this.$emit("UPDATE_PARENT", "Updated");
  };

  $scope.$on("DO_BIDDING", function(event, data){
    for(var i in data){
      $scope[i] = data[i];
    }
  });
}
```

مشاهده خروجی مثال

اگر حالت فرزند و والد بین کنترلرها نباشد چه؟

در این حالت باید $rootScope را به کنترلر مورد نظر تزریق نمایید و سپس با استفاده از سرویس $broadcast یا $emit رویدادتان را منتشر کنید. مثال:

```
'use strict';
angular.module('myAppControllers', [])
  .controller('FirstCtrl', function ($rootScope) {

    $rootScope.$broadcast('UPDATE_ALL');

    Or

    $rootScope.$emit('UPDATE_ALL');
});
```

نکته:

از آن جا که حرکت بالا به پایین event bubbling بسیار هزینه برتر است نسبت به حرکت پایین به بالا در نتیجه سعی کنید تا جای ممکن از rootScope$.$broadcast استفاده **نکنید** . در این جا توضیح کاملی درباره دلایل عدم استفاده از rootScope$.$broadcast داده شده است.

هم چنین می‌توانید یک مثال Live را نیز برای مقایسه بین $emit و $broadcast در این جا مشاهده کنید.

Testing in IE 10.0 32-bit on Windows Server 2008 R2 / 7 64-bit		
Test		**Ops/sec**
$broadcast	`window.$rootScope.$broadcast('fooHappened');`	26,406 ±9.58% 90% slower
$emit	`window.$rootScope.$emit('fooHappened');`	250,522 ±7.21% fastest

عنوان: **زیرنویس فارسی ویدئوهای مقدمات AngularJS - قسمت چهارم**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۲/۲۲ ۱۳:۲۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, Subtitle

زیرنویس‌های فارسی قسمت چهارم را از اینجا می‌توانید دانلود کنید.
لیست سرفصل‌های قسمت چهارم به شرح زیر است :

```
01. Introduction to Routing
02. Websites of Yore
03. Single Page Applications
04. Demo - Adding Your First Route
05. Demo - More Routing and Browser History
06. Demo - Creating a Default Route
07. Demo - Accessing Parameters from the Route
08. Demo - Using the $route Service
09. Demo - Enabling HTML5 Routing
10. Demo - Template and Resolve Properties
11. Demo - Using the $location Service
12. Summary
13. Suggested Exercises
```

در این قسمت به مبحث مسیریابی در انگولار و اهمیت آن جهت ساخت برنامه‌های تک صفحه ایی یا به اصطلاح SPA پرداخته می‌شود. همچنین به بررسی اینکه چگونه می‌توان با کمک سیستم مسیریابی، برنامه‌مان را تبدیل به یک برنامه تک صفحه‌ایی کنیم نیز پرداخته می‌شود. در واقع سیستم مسیریابی به ما کمک می‌کند یک برنامه تک صفحه ایی را به viewهای مختلفی تقسیم کنیم؛ هر چقدر برنامه درگیر جزئیات بیشتری شود، مدیریت آن نیز به مراتب سخت‌تر خواهد شد. تقسیم برنامه به viewهای مختلف و بارگذاری قسمت‌های مختلف برنامه با استفاده از Routing، مدیریت برنامه را برای ما ساده‌تر خواهد کرد.
این قسمت ابتدا شما را با ماهیت برنامه‌های تک صفحه ایی وب آشنا می‌کند و بعد از آن به بررسی مثال‌های عملی در این رابطه خواهد پرداخت. همچنین مواردی ازقبیل Browser History، ایجاد routeهای پیش فرض، افزودن پارامتر به route، استفاده از سرویس $route، فعال سازی سیستم مسیریابی مهیا در HTML5، بررسی پراپرتی‌های Template و Resolve، استفاده از سرویس location بخوبی آموزش و به آنها پرداخته می‌شود.

پ.ن. در مورد نحوه‌ی تهیه اصل ویدیوها در نظرات قسمت‌های قبل این سری مطالب، بیشتر بحث شده‌است.

نظرات خوانندگان

نویسنده: آتوسا فتوحی
تاریخ: ۱۳۹۳/۰۵/۱۱ ۲۱:۲۶

با تشکر از زحمت فراوان شما.آیا زیرنویس قسمت پنجم و ششم را هم در دست تهیه دارید ؟

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۵/۱۱ ۲۲:۵۱

بله، قسمت پنجم رو سعی می‌کنم تا یکی دو هفته دیگه منتشر کنم.

عنوان: آشنایی بیشتر با AngularJS Directive
نویسنده: عزیزخانی
تاریخ: ۱۳۹۳/۰۳/۰۵ ۱۲:۵۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

در مطلب آشنایی با Directiveها در AngularJS با نحوه‌ی ایجاد Directive آشنا شدیم. هدف از این مطلب، آشنایی بیشتر با Directive در AngularJS است؛ یکی از بهترین فریم ورک‌های جاوااسکریپتی، با قابلیت ایجاد کتابخانه‌هایی از کامپوننت‌ها که می‌توانند به HTML اضافه شوند .

کتابخانه‌های جاوااسکریپتی زیادی وجود دارند. به عنوان مثال Bootstrap یکی از محبوب‌ترین "front-end framework" ها است که امکان تغییر در ظاهر المنت‌ها را فراهم می‌کند و شامل تعدادی کامپوننت جاوااسکریپتی نیز می‌باشد. مشکل کار، در هنگام استفاده از کامپوننت ها است. شخصی که در حال توسعه‌ی HTML است باید در کد جاوااسکریپتی خود از jQuery استفاده کند و بعنوان مثال یک Popover را فعال یا غیر فعال کند و این، یک فرآیند خسته کننده و مستعد خطا است.

یک مثال ساده از Directives AngularJS و بررسی آن

```
var m = angular.module("myApp");

myApp.directive("myDir", function() {
  return {
  restrict: "E",
  scope: {
   name: "@",
   amount: "=",
   save: "&"
  },
  template:
   "<div>" +
   "  {{name}}: <input ng-model='amount' />" +
   "  <button ng-click='save()'>Save</button>" +
   "</div>",
  replace: true,
  transclude: false,
  controller: [ "$scope", function ($scope) { …  }],
  link: function (scope, element, attrs, controller) {…}
  }
});
```

به الگوی نامگذاری directive دقت کنید. پیشوند my شبیه به یک namespace است. بنابراین اگر یک Application از دایرکتیوهای قرار گرفته در Module های متفاوت استفاده کند، به راحتی می‌توان محل تعریف یک directive را تشخیص داد. این نام می‌تواند نشان دهنده‌ی این باشد که این directive را خودتان توسعه داده‌اید یا از یک directive توسعه داده شده توسط شخص دیگری در حال استفاده هستید. به هر حال این نحوه‌ی نام گذاری یک اجبار نیست و به عنوان یک پیشنهاد است.

سازنده directive یک شیء را با تعدادی خاصیت باز می‌گرداند که تمامی آنها در سایت AngularJS توضیح داده شده‌اند. در اینجا قصد داریم تا توضیحی مختصر در مورد کاری که این خصوصیات انجام می‌دهند داشته باشیم.

- restrict : تشخیص می‌دهد که آیا directive در HTML استفاده خواهد شد. گزینه‌های قابل استفاده 'C' ، 'E' ، 'A' برای class ، element ، attribute و یا comment است . پیش فرض 'A' برای attribute است. اما ما بیشتر علاقه به استفاده از ویژگی element برای ایجاد المنت‌های UI داریم.

- scope : ایجاد یک scope ایزوله که متعلق به directive است و موجب ایزوله شدن آن از scope صدا زننده directive می‌شود. متغیرهای scope پدر از طریق خصوصیات تگ directive ارسال می‌شوند. این ایزوله کردن زمانی کاربردی است که در حال ایجاد کامپوننت هایی با قابلیت استفاده مجدد هستیم، که نباید متکی به scope پدر باشند. شیء scope در directive نام و نوع

متغیرهای scope را تعیین می‌کنند. در مثال بالا سه متغیر برای scope تعریف شده است:

- (name: "@" (by value, one-way : علامت @ مشخص می‌کند که مقدار متغیر ارسال می‌شود. Directive یک رشته را دریافت می‌کند که شامل مقدار ارسال شده از scope پدر می‌باشد. Directive می‌تواند از آن استفاده کند، اما نمی‌تواند مقدار آن را در scope پدر تغییر دهد.

- (amount: "=" (by reference, two-way : علامت = مشخص می‌کند این متغیر با ارجاع ارسال می‌شود. Directive یک ارجاع به مقدار متغیر در scope اصلی دریافت می‌کند. مقدار می‌تواند هر نوع داده ای، شامل یک شیء complex یا یک آرایه باشد. Directive می‌تواند مقدار را در scope پدر تغییر دهد. این نوع متغیر، زمانیکه نیاز باشد directive مقدار را در scope پدر تغییر دهد، استفاده می‌شود.

- (save: "&" (expression : علامت & مشخص می‌کند این متغیر یک expression را که در scope پدر اجرا می‌شود، نگهداری می‌کند. اکنون directive قابلیت انجام کارهایی فراتر از تغییر یک مقدار را دارد. به عنوان مثال می‌توان یک تابع را از scope پدر فراخوانی و نتیجه‌ی اجرا را دریافت کرد.

· template : الگوی رشته ای که جایگزین المنت تعریف شده می‌شود. فرآیند جایگزینی تمامی خصوصیات را از المنت قدیمی به المنت جدید انتقال می‌دهد. به نحوه استفاده از متغیرهای تعریف شده در scope ایزوله دقت کنید. این مورد به شما امکان تعریف directive های macro-style را می‌دهد که نیاز به کد اضافه‌ای، ندارند. اگرچه در بیشتر موارد الگو یک تگ ساده است که از کدهای link که در زیر توضیح داده شده است استفاده می‌کند.

· replace : تعیین می‌کند که آیا الگوی directive باید جایگزین المنت شود. مقدار پیش فرض false است.

· transclude : تعیین کننده این است که محتوای directive باید در المنت کپی شود یا خیر. در مثال زیر المنت tab شامل المنت‌های HTML دیگر است پس transclude برابر true است.

```
<body ng-app="components">
  <h3>BootStrap Tab Component</h3>
    <tabs>
       <pane title="First Tab">
          <div>This is the content of the first tab.</div>
       </pane>
       <pane title="Second Tab">
          <div>This is the content of the second tab.</div>
       </pane>
    </tabs>
</body>
```

· link : این تابع بیشتر منطق directive را شامل می‌شود. Link وظیفه دستکاری DOM ، ایجاد event listener ها و... را دارد. تابع Link پارامترهای زیر را دریافت می‌کند:

- scope : ارجاع به scope ایزوله شده directive دارد.

- element : ارجاع به المنت‌های DOM که directive را تعریف کرده اند. تابع link معمولا برای دستکاری المنت از jQuery استفاده می‌کند. (یا از Angular's **jqLite** در صورتی که jQuery بارگذاری نشده باشد)

- controller : در مواقعی که از دایرکتیوهای تو در تو استفاده می‌شود کاربرد دارد. این پارامتر یک directive فرزند با ارجاعی به پدر را فراهم می‌کند، بنابراین موجب ارتباط directive ها می‌شود.

به عنوان مثال، این directive که پیاده سازی bootstrap tab را انجام داده است، می‌توانید مشاهده نمایید.

موفق باشید

عنوان: **چگونگی استفاده از افزونه Isotope در AngularJS**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۳/۰۴/۰۹ ۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, Isotope, Masonary

حتما تا به حال در وب سایت‌های زیادی قسمت هایی را دیده اید که چیدمان عناصر آن به شکل زیر است:

این گونه چیدمان را حتما در منوی Start ویندوز 8 بارها دیده‌اید! عناصر تشکیل دهنده‌ی این شکل از چیدمان، می‌توانند یک سری عکس باشند که تشکیل یک گالری عکس را داده‌اند و یا یک سری div که محتوای پست‌های یک وبلاگ را در خود جای داده‌اند. چیزی که این شکل از چیدمان عناصر را نسبت به چیدمان‌های معمول متمایز می‌کند این است که طول و عرض هر یک از این عناصر با یکدیگر متفاوت است و هدف از این گونه چیدمان آن است که این عناصر در فضایی که به آن‌ها اختصاص داده شده است، به صورت بهینه قرار گیرند تا کمترین فضا هدر رود.

برای اعمال این شکل از چیدمان در دنیای وب افزونه‌های زیادی بر فراز کتاب خانه‌ی jQuery تدارک دیده شده است که از جمله مطرح‌ترین آن‌ها می‌توان به افزونه های Masonry ، Isotope و Gridster اشاره کرد.

افزونه‌ی Isotope مزایایی را برای من در پی داشت و این افزونه را برای انجام کارهای خود، مناسب دیدم. نکته‌ی مهم اینجا است که هدف من بررسی Isotope نیست، چرا که اگر به وب سایت آن مراجعه کنید، با کوهی از مستندات مواجه می‌شوید که چگونه از آن در وب سایت‌های معمولی استفاده کنید.

در این مقاله قصد من این است که نشان دهم چگونه از افزونه‌ی Isotope در AngularJS استفاده کنیم؛ چگونه چیدمان آن را راست به چپ کنیم و چگونه آن را با محیط‌های واکنش گرا (Responsive) سازگار کنیم.

فرض کنید در یک وب سایت قصد داریم اطلاعات یک سری مطلب خبری را از سرور، به فرمت JSON دریافت کرده و نمایش دهیم. در AngularJS شیوه‌ی کار بدین صورت است که اطلاعاتی که به فرمت JSON هستند را با استفاده از directive ایی به نام

ng-repeat پیمایش کرده و آن‌ها را نمایش دهیم. حال اگر بخواهیم چیدمان مطالب را با استفاده از Isotope تغییر دهیم، می‌بینیم که هیچ چیزی نمایش داده نمی‌شود. دلیل آن بر می‌گردد به مراحل کامپایل کردن AngularJS و نامشخص بودن زمان اعمال چیدمان Isotope به عناصر است.

در AngularJS هنگامیکه با دستکاری DOM سر و کار پیدا می‌کنیم، معمولا باید به سراغ Directiveها رفت و یک Directive سفارشی برای کار با Isotope تعریف کرد تا با مکانیزم‌های Angular سازگار باشد. خوشبختانه Directive Isotope برای Angular موجود می‌باشد. نکته‌ی مهم این است که این Directive برای نگارش 1 افزونه‌ی Isotope نوشته شده است. البته با نگارش 2 هم کار می‌کند که من برای انجام کار خود نسخه‌ی 1 را ترجیح دادم استفاده کنم.

نکته‌ی بعدی که باید رعایت شود این است که چیدمان عناصر باید از راست به چپ شوند. خوشبختانه این کار در نسخه‌ی 1 Isotope با تغییر کوچکی در سورس Isotope و تغییر یک تابع انجام میشود. گویا نسخه‌ی دوم امکان پیش فرضی را برای این کار دارد، اما نتوانستم آن را به خوبی پیاده سازی کنم و به همین دلیل ترجیح دادم از همان نسخه‌ی اول استفاده کنم.

برای اینکه در هنگام جابه جا شدن عناصر، انیمیشن‌ها نیز از راست به چپ انجام شوند، باید cssهای زیر را نیز اعمال نمود:

```
.isotope .isotope-item {
  -webkit-transition-property: right, top, -webkit-transform, opacity;
     -moz-transition-property: right, top, -moz-transform, opacity;
      -ms-transition-property: right, top, -ms-transform, opacity;
       -o-transition-property: right, top, -o-transform, opacity;
          transition-property: right, top, transform, opacity;
}
```

Responsive بودن این عناصر مسئله‌ی دیگری است که باید حل گردد. امروزه اکثر فریم ورک‌های مطرح css، واکنشگرا نیز هستند و برای پشتیبانی از سایزهای متفاوت صفحه نمایش، تدابیری در نظر گرفته‌اند. اساس کار واکنش گرا بودن این فریم ورک‌ها در تعیین ابعاد عناصر، بیان ابعاد به صورت درصدی است. مثلا فلان عرض div برابر 50% باشد بدین معناست که همیشه عرض این div نصف عرض عنصر والد آن باشد.

متاسفانه Isotope میانه‌ی چندانی با این ابعاد درصدی ندارد و باید عرض عناصر به صورت دقیق و بر حسب پیکسل بیان شود. البته نسخه‌ی جدید آن و یا حتی پلاگین هایی برای کار با ابعاد درصدی نیز تدارک دیده شده است که به شخصه به نتیجه‌ی با کیفیتی نرسیدم.

برای حل این مشکل می‌توان از امکانات CSS به مانند دستورات زیر استفاده کرد:

```
@media (min-width: 768px) and (max-width: 980px) {
    .card {
        width: 320px;
    }
}

@media (min-width: 980px) and (max-width: 1200px) {
    .card {
        width: 260px;
    }
}

@media (min-width: 1200px) {
    .card {
        width: 340px;
    }
}
```

بدین صورت می‌توان در ابعاد مختلف نمایشگر تعیین کرد که عرض عناصر ما چقدر باشد.
اکنون یک گالری عکس را در نظر بگیرید که در زیر هر عکس توضیحی نیز نوشته شده است و ساختار HTML آن به این صورت است که داخل هر div عکسی نیز موجود است. اگر به شیوه‌ی ذکر شده عمل کنید با یک اشکال مواجه می‌شوید و عناصر روی هم قرار گرفته و اصطلاحا overlapping اتفاق می‌افتد. دلیل این امر این است که لود شدن عکس‌ها عملی زمان گیر است و

Isotope قبل از این که عکس لود شود، سایز آن عنصر را محاسبه کرده که در حقیقت این سایز بدون احتساب سایز عکس است و ابعاد واقعی عنصر ما نیست؛ در نتیجه وقتی عکس لود می‌شود آن div فضای بیشتری احتیاج دارد و به همین دلیل به زیر divهای دیگر می‌رود.

برای حل این مشکل باید به این صورت عمل کرد که وقتی عکس‌ها کامل لود شدند، Isotope وارد عمل شده و سایز عناصر را به دست آورده و آن‌ها را بچیند. برای این کار معمولا از افزونه‌ی imagesLoaded استفاده می‌کنند که با کمک این افزونه می‌توان مشخص کرد که وقتی تمام عکس‌های موجود در فلان div کامل لود شدند، Isotope وارد عمل شده و عناصر را چیدمان کند.

البته بدون استفاده از افزونه‌ی imagesLoaded و به کمک امکانات AngularJS و تعریف یک Directive سفارشی می‌توان زمان لود شدن عکس‌ها را کنترل کرد.

```
app.directive('imageOnload', function () {
            return {
                restrict: 'A',
                link: function (scope, element, attrs) {
                    element.bind('load', function () {
                        scope.$emit('iso-method', { name: 'reLayout', params: null }); // call reLayout
isotope methode prevent overlaaping the items
                    });
                }
            };
        });
```

کار این directive این است که به ازای بارگذاری هر عکس، متد reLayout را از Isotope، فراخوانی می‌کند. از این جهت فراخوانی reLayout به ازای لود شدن هر عکس بهتر است که لود شدن تمامی عکس‌ها ممکن است مدت زمان زیادی طول بکشد و کاربر برای مدتی با یک ساختار بهم ریخته مواجه شود.

اگر در نمونه کدی که قرار داده‌ام، به انتهای کدهای کنترلر ListController دقت کنید، برای رویداد resize شی window، تابعی تعریف شده است تا به هنگام تغییر سایز صفحه فراخوانی شود. در این رویداد هر بار که سایز پنجره تغییر کرد، پس از یک ثانیه تابع reLayout افزونه‌ی Isotope را فراخوانی می‌کنیم تا مجددا المنت‌های صفحه چیده شوند. البته ضرورتی وجود نداشته ولی در بعضی مواقع عناصر خوب چیده نمی‌شدند که با فراخوانی reLayout از چیدمان صحیح عناصر مطابق با سایز جدید صفحه اطمینان حاصل پیدا می‌کنیم. دلیل یک ثانیه تاخیر این است که اگر به ساز و کار تعاریف متدها در directive Isotope دقت کنید، از سرویس $timeout به وفور استفاده شده است. ظاهرا اگر برای فراخوانی reLayout زودتر عمل کنیم با فراخوانی هایی این متد در ساختار خودش تداخل پیدا می‌کند.

```
$(window).resize(function () {
                $timeout(function myfunction() {
                    $scope.$broadcast('iso-method', { name: 'reLayout', params: null }); // call
reLayout isotope methode prevent overlaaping the items
                },1000);

            });
```

در نهایت تمامی نکات گفته شده را به صورت یک نمونه کد آماده کردم: دانلود نمونه کد

0-گوگل قصد ندارد تولید نکسوس‌ها را متوقف کند
1-چگونه اندروید L را روی نکسوس 5 یا نکسوس 7 نصب کنیم؟
2-مقایسه‌ی تصویری اندروید L با اندروید کیت‌کت
3-شاتر: تصاویری دیدنی از احساسات والدین در قلمرو حیوانات
4-اپل قیمت آیپاد تاچ را ضمن مجهز کردن نسخه‌ی 16 گیگابایتی به دوربین کاهش داد
5-تماشا کنید: بازگرداندن توانایی کنترل ماهیچه‌ها به افراد فلج به کمک فناوری نوروبریج

عنوان: **نمایش تاریخ شمسی توسط JavaScript در AngularJS**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۳/۰۴/۱۱ ۱۷:۵۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Persian, AngularJS, momentjs

در برنامه‌های مبتنی بر وب رایج، معمولا تبدیل تاریخ میلادی به شمسی در سمت سرور انجام می‌گیرد و تاریخ شمسی حاصل از تبدیل، به کاربر نمایش داده می‌شود. اما در برنامه‌های Single Page و یا به اختصار SPAها که کلاینت فقط با یک سری داده به فرمت JSON درگیر است، برای نمایش تاریخ شمسی به چه طریقی باید عمل کرد؟ آیا باید تاریخ را در سمت سرور به فرمت مورد نظر تبدیل کرد و یا در سمت کلاینت؟ همه‌ی این‌ها از جمله سوالاتی هست که به هنگام توسعه‌ی SPAها با آن‌ها حتما درگیر خواهید شد.

شاید بتوان گفت که در SPA ها، هدف این است که از بار سرور تا حد ممکن کم کرد و آن را در بین کلاینت‌ها توزیع کرد. در SPAها نقش اصلی سرور تامین داده هاست و بیشتر پردازش‌ها در صورت امکان در سمت کلاینت انجام می‌شود و می‌بینید که حتی رندر کردن HTML نیز به عهده‌ی قالب‌های سمت کلاینت است. البته هنوز هم می‌توان قبل از اینکه داده را به فرمت JSON سریالایز کرد، سمت سرور بر روی آن‌ها پیمایش انجام داده و تاریخ‌های میلادی را به شمسی تبدیل کرد که هدف ما این نیست و می‌خواهیم این کار را بر عهده‌ی مرورگر کاربر قرار دهیم. **معرفی moment.js**

برای کار با داده‌هایی از جنس تاریخ در سمت کلاینت، کتابخانه‌ی جاوا اسکریپتی قدرتمندی به نام moment.js وجود دارد. این کتابخانه دارای انواع و اقسام API برای نمایش و پردازش تاریخ هست. حتی می‌تواند relative time را نیز نمایش دهد. منظور از relative time این هست که به جای نمایش تاریخ، اختلاف آن را با زمان حال نمایش دهد. برای مثال می‌نویسند فلان پست در دو ساعت پیش ارسال شده و زمان دقیق ارسال پست را نمایش نمی‌دهد.

خوشبختانه برای افزودن تاریخ شمسی به این کتاب خانه، افزونه‌ای به نام moment-jalaali برای آن تدارک دیده شده است. کار با آن نیز بسیار راحت است. کافی است در همان API هایی که برای فرمت کردن تاریخ در moment.js استفاده می‌کردید؛ یک j در ابتدای آن‌ها قرار دهید که مثال‌های کامل استفاده از آن را در مستندات آن می‌توانید مشاهده کنید.

نحوه‌ی استفاده از moment.js در AngularJS و ASP.NET

در ASP.NET فیلد هایی که از جنس DateTime هستند به شکل زیر به فرمت JSON سریالایز می‌شوند:

```
\/Date(1374222094520)\/
```

در moment.js احتیاج به کدنویسی برای parse کردن این نوع فرمت و تبدیل کردن آن به تاریخ وجود ندارد؛ چرا که moment.js به صورت تو کار از این نوع فرمت نیز پشتیبانی می‌کند و احتیاجی به کار اضافه‌تر نیست.

```
moment("/Date(1198908717056-0700)/"); // December 28 2007 10:11 PM
```

در AngularJS هر گاه قصد داشته باشیم که فرمت نمایش داده‌ها را تغییر دهیم از filterها استفاده می‌کنیم. برای مثال فیلتر uppercase داده name را با حروف بزرگ نمایش می‌دهد.

```
{{ name | uppercase }}
```

حال برای تاریخ نیز می‌خواهیم چنین کاری انجام دهیم؛ بدین صورت که یک فیلتر سفارشی به شکل زیر تعریف کرده تا تاریخ میلادی را به صورت شمسی و با فرمت دلخواهی که می‌خواهیم نمایش دهد:

```
{{post.date | jalaliDate:'jYYYY/jMM/jDD hh:mm' }}
```

تعریف فیلتر jalaliDate نیز به شکل زیر است:

```
app.filter('jalaliDate', function () {
            return function (inputDate, format) {
                var date = moment(inputDate);
                return date.fromNow() + " " + date.format(format);
```

```
        }
    });
```

خروجی این فیلتر نیز به شکل "4 ماه پیش 07/12/1392 03:10" است و مشاهده می‌کنید که به کمک filterها در AngularJS انجام این گونه از کارها بسیار ساده و لذت بخش است.

توجه کنید که این فقط یک ایده‌ی ابتدایی و ساده از پیاده سازی فیلتر فوق است. قطعا با کمک APIهای متنوع momentjs و پارامترهای ورودی فیلتر، می‌توان فیلتری بسیار پیشرفته‌تر تعریف کرد. **دریافت کدهای یک مثال پیاده سازی شده با استفاده از کدهای فوق** moment-jalali-AngularJs.rar

نظرات خوانندگان

نویسنده: محمد 92
تاریخ: ۱۳۹۳/۱۰/۱۰ ۷:۵۳

با سلام و ممنون از مقاله خوبتون
سوالی داشتم، moment-jalaali روی nuget نیست یا من نتونستم پیداش کنم

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۰/۱۰ ۱۶:۶

- کتابخانه‌های جاوا اسکریپتی را بهتر است در Bower و امثال آن دنبال کنید.
- برای نمونه آدرس نصب moment-jalaali (^)

```
npm install moment-jalaali
```

- ویژوال استودیو هم افزونه‌ای برای کار با Bower ارائه داده. اطلاعات بیشتر

نویسنده: علی شریعتی
تاریخ: ۱۳۹۴/۰۳/۳۰ ۱۸:۵۲

سلام.
خیلی خوب و سادست
ولی واسه Date Picker باید چی کار کنیم؟
اصلا Date Picker داره؟
اگه نداره بیزحمت یه Date Picker خوب معرفی کنید که با این افزونه سازگار باشه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۴/۰۳/۳۰ ۱۹:۲۲

قسمت اشتراک‌های سایت، یک گروه Persian دارد و همچنین گروه date picker ها. برای نمونه « تقویم شمسی AngularJS »

عنوان: **زیرنویس فارسی ویدئوهای مقدمات AngularJS - قسمت پنجم**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۵/۳۱ ۲۳:۱۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, Subtitle

زیرنویس‌های فارسی قسمت پنجم را از اینجا می‌توانید دانلود کنید.
لیست سرفصل‌های قسمت پنجم به شرح زیر است:

```
01-Introduction to Directives
02-Demo. Creating Your First Directive
03-Demo. Domain Specific Language via Custom Elements
04-Demo. Isolating Directive Scope
05-Demo. Exploring Isolate Scope Bindings
06-Demo. Handling Events with Directives
07-Demo. Observing and Responding to Changes
08-Demo. Using Controllers within Directives
09-Demo. Sharing Directive Controllers via Require
10-Demo. Directive Priority and using Terminal
11-Demo. Using Require with Nested Directives
12-Demo. Understanding Transclusion
13-Demo. Using Compile to Transform the DOM
14-Demo. Making jQuery More Explicit with Directives
15-Summary
```

در این قسمت به مبحث ایجاد دایرکتیوهای سفارشی پرداخته می‌شود. دایرکتیوها در واقع مهمترین قسمت هر برنامه انگولار هستند. انگولار به صورت توکار شامل تعداد زیادی دایرکتیو می‌باشد. در واقع می‌توانیم بگوئیم دایرکتیو می‌تواند یک سینتکس جدید باشد که دارای یک رفتار مخصوص می‌باشد. برای مثال، static HTML هیچ دیدی نسبت به ایجاد و نمایش یک ویجت انتخابگر تاریخ(Date Picker) ندارد. برای اینکار باید به HTML، این سینتکس جدید را توسط دایرکتیوها آموزش دهیم.

نظرات خوانندگان

نویسنده: جواد مسعودیان
تاریخ: ۱۳۹۳/۰۶/۰۲ ۲۳:۳۲

دمت گرم سیروان جان

ویدئو هاشم از torrent دارم دانلود میکنم.

توی سایت مثه اینکه 6 قسمت گذاشته، بی زحمت این زیر نویس 6 اش رو هم به ما برسون، مرسی.

عنوان: **توسعه سرویس‌های Angular به روش OOP**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۶/۱۴ ۲۳:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, OOP, AngularJS

یک نکته‌ای که در توسعه سیستم‌ها و نرم افزارها تاکید فراوانی به آن می‌شود استفاده مجدد از کدهای نوشته شده قبلی است. یعنی تا جای ممکن باید ساختار پروژه به گونه‌ای نوشته شود که از تکرار کدها در جای جای پروژه جلوگیری شود. این مورد به خوبی در زبان‌های شیءگرا نظیر #C رعایت می‌شود اما در پروژه‌هایی که مبتنی بر Javascript هستند نظیر angular، باید با استفاده از خاصیت prototype جاوا اسکریپ این مورد را رعایت نمود. در مقاله Dr. Axel Rauschmayer، قدم به قدم و به خوبی روش‌های وراثت در Javascript توضیح داده شده است.

در این پست با روش‌های وراثت در کنترلرهای انگولاری آشنا شدید. این وراثت محدود به ارث بری scopeها می‌شود. اما یکی از بخش‌های بسیار مهم پروژه‌های انگولار نوشتن سرویس‌هایی با قابلیت توسعه مجدد در سایر بخش‌های پروژه می‌باشد. معادل آن، مفهوم Overriding در OOP است. با ذکر مثالی این مورد را با هم بررسی خواهیم کرد.
ابتدا یک سرویس به نام BaseService ایجاد کنید:

```
angular.module('myApp').service('BaseService', function() {

    var BaseService = function(title) {
        this.title = title;
    };

    BaseService.prototype.getMessage = function() {
        var self = this;
        return 'Hello ' + self.title;
    };

    return BaseService;
});
```

سرویس بالا دارای سازنده‌ای است که مقدار title باید در اختیار آن قرار گیرد. با استفاده از خاصیت prototype تابعی تعریف می‌کنیم که این تابع خروجی مورد نظر را برای ما تامین خواهد نمود.
حال اگر ماژول و کنترلری جهت نمایش خروجی به صورت زیر ایجاد کنیم:

```
var app= angular.module('myApp', []);


app.controller('myCtrl', function ($scope,BaseService) {

    var instance = new BaseService('Masoud');
    $scope.title = instance.getMessage();

});
```

با کدهای Html زیر:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" ng-app="myApp">
<head>
    <title></title>
</head>
    <body ng-controller="myCtrl">

        <div>
            {{title}}
        </div>

    </body>
<script src="Scripts/jquery-2.1.1.min.js"></script>
<script src="Scripts/angular.js"></script>
<script src="App/app.js"></script>
```

```
</html>
```

در نهایت خروجی به صورت زیر قابل مشاهده است:

تا اینجای کار روال معمول تعاریف سرویس در انگولار بوده است. اما قصد داریم سرویس جدیدی را ایجاد نمایم تا خروجی سرویس قبلی را اندکی تغییر دهد. به جای اینکه سرویس قبلی را تغییر دهیم یا بدتر از آن سرویس جدیدی بسازیم و کدهای قبلی را در آن کپی کنیم کافیست به صورت زیر عمل نماییم:

```
app.service('ExtService', function(BaseService) {

    var ExtService = function() {
        BaseService.apply(this, arguments);
    };

    ExtService.prototype = new BaseService();

    ExtService.prototype.getMessage = function() {
        var self = this;
        return BaseService.prototype.getMessage.apply(this, arguments) + ' From Ext Service';

    };

    return ExtService;
});
```

حال می‌توان کنترلر را به صورت زیر بازنویسی کرد.

```
app.controller('myCtrl', function ($scope,BaseService , ExtService) {

    var baseInstance = new BaseService('Masoud');
    var extInstance = new ExtService('Dotnettips');
    $scope.title = baseInstance.getMessage() + ' and ' + extInstance.getMessage();

});
```

در کنترلر بالا هر دو سرویس تزریق شده‌اند. خروجی سرویس دوم متن From Ext Service را نیز به همراه خواهد داشت. پس از اجرای برنامه خروجی زیر قابل مشاهده است:

localhost:2208/index.html
Hello Masoud and Hello Dotnettips From Ext Service

عنوان: Angular Interceptors
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۶/۲۷ ۱۸:۲۰
آدرس: www.dotnettips.info
گروه‌ها: AngularJS

تا پیش از این به احتمال زیاد با Interceptorها در IOC Containerها متفاوت آشنا شدید و برای AOP از آن‌ها استفاده کرده‌اید. در این جا نیز دقیقا همان مفهوم و هدف را دنبال خواهیم کرد؛ اضافه کردن و تزریق کدهای نوشته شده به منطق برنامه. کاربرد Interceptorها در انگولار، زمانی است که قصد داشته باشیم یک سری تنظیمات عمومی را برای درخواست‌های $http انجام دهیم. همچنین می‌توان انجام برخی مراحل مشترک، نظیر اعتبارسنجی یا مدیریت خطاها را نیز توسط Interceptorها انجام دهیم.
سرویس $http در Angular جهت ارتباط و تبادل اطلاعات با دنیای Backend مورد استفاده قرار می‌گیرد. حالت هایی بنابر نیاز به وجود می‌آیند که بخواهیم ارسال اطلاعات به سرور و هم چنین پاسخ دریافتی را capture کنیم و قبل از این که داده‌ها در اختیار App قرار گیرد، آن را مورد بررسی قرار دهیم(برای مثال لاگ اطلاعات) یا حتی نوشتن یک HTTP error handling جهت مدیریت خطاهای به وجود آمده حین ارتباط با سرور (برای مثال خطای 404).
حال با ذکر مثالی این موارد را بررسی می‌کنیم. برای نوشتن یک Interceptor می‌توان با استفاده از سرویس factory این کار را به صورت زیر انجام داد.

```
module.factory('myInterceptor', ['$log', function($log) {
    $log.debug('data');

    var myInterceptor = {
        ....
        ....
        ....
    };

    return myInterceptor;
}]);
```

کد بالا یک Interceptor بسیار ساده است که وظیفه آن لاگ اطلاعات است. در انگولار چهار نوع Interceptor برای سرویس $http داریم:

« **request** : قبل از هر فراخوانی سرویس‌های سمت سرور، ابتدا این Interceptor فراخوانی می‌شود و config سرویس $http در اختیار آن قرار می‌گیرد. می‌توان این تنظیمات را با توجه به نیاز، تغییر داد و نمونه ساخته شده جدید را در اختیار سرویس $http قرار دهیم.

« **response** : هر زمان که عملیات فراخوانی سرویس‌های سمت سرور به درستی انجام شود و همراه با آن پاسخی از سرور دریافت شود، این Interceptor قبل از فراخوانی تابع success سرویس $http، اجرا خواهد شد.

« **requestError** : از آنجا که سرویس $http دارای مجموعه ای از Interceptorها است و آن‌ها نیز یکی پس از دیگری حین انجام عملیات اجرا می‌شوند، اگر در Request Interceptor قبلی خطایی رخ دهد بلافاصله این Interceptor فراخوانی می‌شود.

« **responseError** : درست مانند حالت requestInterceptor است؛ فقط خطای مربوطه باید در تابع response باشد.

با توجه به توضیحات بالا کد قبلی را به صورت زیر تعمیم می‌دهیم.

```
module.factory('myInterceptor',['$q' , '$log', function($q , $log) {
 $log.debug('data');

 return {

request: function(config) {

return config || $q.when(config);
},

requestError: function(rejection) {
```

```
return $q.reject(rejection);
},

response: function(response) {

return response || $q.when(response);
},

responseError: function(rejection) {

return $q.reject(rejection);
}

}

}]);
```

برای رجیستر کردن Interceptor بالا به سرویس‌های $http باید به صورت زیر عمل نمود.

```
angular.module('myApp')
.config(function($httpProvider) {
$httpProvider.interceptors.push('myInterceptor');
});
```

نظرات خوانندگان

نویسنده: محمود راستین
تاریخ: ۱۳۹۴/۰۶/۰۹ ۲۱:۲۳

با سلام.

مطلب خیلی مفیدی بود. مرسی.

من از این استفاده می‌کنیم برای نمایش پیام Loading ، مثلا وقتی دستوری به سرور ارسال میشه پیام Loading نمایش داده میشه :

```
.factory('httpInterceptor', function($q, $rootScope, $log) {

        var numLoadings = 0;

        return {
            request: function(config) {

                numLoadings++;

                // Show loader
                $rootScope.$broadcast("loader_show");
                return config || $q.when(config);

            },
            response: function(response) {

                if ((--numLoadings) === 0) {
                    // Hide loader
                    $rootScope.$broadcast("loader_hide");
                }

                return response || $q.when(response);

            },
            responseError: function(response) {

                if (!(--numLoadings)) {
                    // Hide loader
                    $rootScope.$broadcast("loader_hide");
                }

                return $q.reject(response);
            }
        };
    })
    .config(function($httpProvider) {
        $httpProvider.interceptors.push('httpInterceptor');
    }).directive("loader", function($rootScope) {
            return function($scope, element, attrs) {
                $scope.$on("loader_show", function() {
                    return element.show();
                });
                return $scope.$on("loader_hide", function() {
                    return element.hide();
                });
            };
        }
    );
```

امیدوارم به درد دوستان بخوره.

برای نحوه نمایشش هم باید به این صورت تو Layout برنامه فراخوانی کنید اون رو :

```
<div class="ajax-loader" loader>
    <img src="@Links.Content.Images.loading_gif" />
</div>
```

کلاس ajax-loader هم به این صورت باید باشه :

```
.ajax-loader {
    position: absolute;
    z-index: 100000;
    display: none;
}
```

مکان قرارگیری تصویر loading رو هم میتونید با خاصیت‌های left و top تنظیم کنید.

با این کد ، زمانی که یک درخواست ارسال میشه این تصویر نمایش داده میشه.

عنوان: **پیاده سازی Template تو در تو در AngularJS و ASP.NET MVC**
نویسنده: میثم نوایی
تاریخ: ۱۳۹۳/۰۷/۱۹ ۱۵:۵۵
آدرس: www.dotnettips.info
گروه‌ها: MVC, AngularJS, formbuilder

در Angular می شود یک سری Template و ساختار از پیش تعریف شده داشت و در هر زمان که نیاز بود مدلی را به آنها پاس داد و نمای HTML مورد نظر را تحویل گرفت.
بطور مثال در فرم سازها یا همان فرم‌های داینامیک ما نیاز داریم که مدل یک فرم (مثلا در فرمت JSON) را برای View ارسال کنیم و با استفاده از توانایی‌های Angular بتوانیم فرم مورد نظر را نمایش دهیم و در صورت امکان تغییر دهیم.
ViewModel فرم شما در MVC میتواند چیزی شبیه این باشد

```
public class Form
    {
        public string Name { get; set; }
        public string Title { get; set; }
        public List<BaseElement> Elements { get; set; }
    }

    public abstract class BaseElement
    {
        public string Name { get; set; }
        public string Title { get; set; }
    }
    public class Section : BaseElement
    {
        public List<TextBox> Elements { get; set; }
    }
    public class TextBox : BaseElement
    {
        public string Value { get; set; }
        public string CssClass { get; set; }
    }
```

یک کنترلر هم برای مدیریت فرم ایجاد میکنیم

```
public class FormBuilderController : Controller
    {
        //
        // GET: /FormBuilder/

        public ActionResult Index()
        {
            var form = new Form();
            var section = new Section() { Title = "Basic Info", Name = "section01" };
            section.Elements.Add(new TextBox() { Name = "txt1", Title = "First Text Box" });
            form.Elements.Add(new TextBox() { Name = "txt1", Title = "Second Text Box" });
            var formJson=JsonConvert.SerializeObject(form);
            return View(formJson);
        }
    }
```

در این کنترلر ما تنها یک اکشن داریم که در آن یک فرم خام ساده ایجاد کرده و سپس با استفاده از کتابخانه Json.net آنرا سریال و تبدیل به فرمت Json می‌کنیم و سپس آنرا برای View ایی که از Angular قدرت گرفته است، ارسال می‌نمائیم.
پیاده سازی View با Angular به اشکال گوناگونی قابل پیاده سازی و استفاده است که در اینجا و اینجا می‌توانید ببینید.
اما برای اینکه مشکل کنترلرهای تودرتو(Section) را حل کنید باید بصورت بازگشتی Template را فراخوانی کنید.

```
<script type="text/ng-template" id="ElementTemplate">
    <div ng-if="control.Type == 'JbSection'">
    <h2>{{control.Title}}</h2>
    <ul>
        <li ng-repeat="control in control.Elements" ng-include="'ElementTemplate'"></li>
    </ul>
    </div>
    </script>
```

و یا

```
<script type="text/ng-template" id="element.html">
    {{data.label}}
    <ul>
        <li ng-repeat="element in data.elements" ng-include="'element.html'"></li>
    </ul>
</script>

<ul ng-controller="NestedFormCtrl">
    <li ng-repeat="field in formData" ng-include="'element.html'"></li>
</ul>
```

در اینجا صفحه element.html یک صفحه بیرونی است که Template ما در آن قرار دارد.

عنوان: **ارث بری prototype ای در جاوا اسکریپت به زبان خیلی ساده**
نویسنده: فرید بکران
تاریخ: ۱۳۹۳/۰۸/۲۵ ۲۳:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, AngularJS

انگیزه اصلی این نوشته شروع کار با AngularJs و استفاده از scope در این کتابخانه است. بیشتر دوستانی که کار با این کتابخانه را شروع می‌کنند و تجربه زیادی با جاوا اسکریپت ندارند، با مفهوم ارث بری scope مشکل پیدا می‌کنند.

ارث بری در scope های AngularJs موضوع پیچیده و عجیب و غریبی نیست. در واقع همان ارث بری prototype ای است که جاوا اسکریپت پشتیبانی می‌کند.

این روش توضیح خیلی ساده ای دارد.

در هنگام دسترسی به مقدار یک خصوصیت روی یک شی اگر آن خصوصیت در شی مورد نظر وجود نداشته باشد جاوا اسکریپت یک سطح در زنجیره‌ی prototype ها بالا رفته و به شی پدر دسترسی پیدا کرده و در آن به دنبال مقدار خصوصیت می‌گردد. این کار را آن قدر ادامه می‌دهد تا به بالاترین سطح برسد و دیگر چیزی پیدا نکند.

این بالا رفتن در زنجیره‌ی prototype ها عملا با دسترسی به خصوصیت prototype انجام می‌شود.

فرض کنید دو شی (دقت کنید که می‌گویم شی) به نام‌های employee و person داریم. این دو شی را به صورت زیر تعریف می‌کنیم.

```
var person = { type: '', name:'No Name' };
var employee = {  };
```

شی employee الان هیچ خصوصیت ای ندارد. و دسترسی به هر خصوصیت ای از آن هیچ نتیجه‌ای در بر نخواهد داشت.

```
console.log('Before Inheritance -> employee.name = ' + employee.name);
```

با مقدار دهی کردن خصوصیت prototype مربوط به employee به person این شی را از person ارث بری می‌کنیم.

```
employee.__proto__ = person;
```

بعد از اجرا شدن این خط از برنامه هنگام دسترسی پیدا کردن به مقدار name، مقدار اصلی آن که در شی person وارد شده بود را خواهیم دید.

ملاحظه کردید که وقتی خصوصیت name در شی مورد نظر وجود نداشت به شی پدر رجوع شد و مقدار خصوصیت مربوطه از آن بدست آمد.

الان فرض کنید که در قسمتی از برنامه خواستیم مقدار name در شی employee را به مقدار مشخصی تغییر دهیم. به طور مثال:

```
employee.name = 'farid';
console.log('After Assiginig -> employee.name = ' + employee.name);
console.log('After Assiginig -> person.name = ' + person.name);
```

با چاپ کردن مقادیر person.name و employee.name انتظار دارید چه نتیجه ای ببینید؟

اگر از زبان‌های شی گرایی مانند #C آمده باشید احتمالا خواهید گفت مقادیر یکسان خواهند بود. ولی در واقع این گونه نیست. مقدار person.name همان مقدار اولیه ما خواهد بود و مقدار employee.name نیز ‘farid’.

دلیل این رفتار یک نکته ساده و اساسی است.

جاوا اسکریپت فقط در زمان دسترسی به یک خصوصیت در صورت پیدا نکردن آن در شی مورد نظر ما به سطوح بالاتر prototype ای رفته و دنبال آن خصوصیت می‌گردد.

اگر ما قصد مقدار دهی به یک خصوصیت را داشته باشیم و خصوصیت مورد نظر ما در شی وجود نداشته باشد جاوا اسکریپت یک نسخه محلی از خصوصیت برای آن شی می‌سازد و مقدار ما را به آن می‌دهد.

در واقع در مثال ما هنگام مقدار دهی به employee.name آن خصوصیت در شی موجود نبود و یک نسخه محلی به نام name در شی ایجاد شد و دفعه بعدی که دسترسی به مقدار این خصوصیت اتفاق افتد این خصوصیت به صورت محلی وجود خواهد داشت و جاوا اسکریپت به سطوح بالاتر نخواهد رفت .

تمام کدهای بالا در bin زیر موجود هستند.

Prototypal Inheritance in Javascript

الان فرض کنید شیءهای ما به این صورت هستند:

```
var person = {
  info : { name: 'No Name', type: '' }
};
var employee = {};
```

به همان صورت بالا ارث بری می‌کنیم.

```
employee.__proto__ = person;
```

و سپس name را مقداردهی می‌کنیم.

```
employee.info.name = 'farid';
```

و مقادیر را چاپ می‌کنیم.

```
console.log('After Assiginig -> employee.name = ' + employee.info.name);
console.log('After Assiginig -> person.name = ' + person.info.name);
```

ملاحظه خواهید کرد که مقادیر مساوی هستند.
دلیل این امر به زبان ساده این است که وقتی اقدام به مقدار دهی name در شی employee کردیم در واقع قبل از مقدار دهی اصلی name یک دسترسی به شی info نیاز بود و دسترسی به شیء با استفاده از همان قانونی که مطرح کردیم انجام شده و شیء مربوط به person برگردانده شده است. چون name یک خصوصیت از info است نه employee.
سوالی که می‌توان مطرح کرد این است که در صورت نوشتن این خط کد چه اتفاقی خواهد افتاد؟

```
employee.info = {
  name: 'farhad'
};
```

Prototypal Inheritance with objects

با توجه به مطالب گفته شده باید قادر به حدس زدن نتیجه خواهید بود. **نکته: روش‌های کار با prototype در این نوشته فقط جنبه آموزشی و توضیحی دارد و روش درست استفاده از prototype این نیست.**

عنوان: **زیرنویس فارسی ویدئوهای مقدمات AngularJS - قسمت ششم (قسمت آخر)**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۳/۰۹/۳۰ ۱۲:۱۵
آدرس: www.dotnettips.info
گروه‌ها: AngularJS, Subtitle

زیرنویس‌های فارسی قسمت ششم را می‌توانید از اینجا دانلود کنید.
لیست سرفصل‌های این قسمت به شرح زیر است:

```
01. Introduction
02. Installing Karma
03. Karma with Webstorm
04. Testing Controllers
05. Testing Simple Services
06. Testing Services with Dependencies
07. Testing AJAX Services
08. Testing Filters
09. Testing Directives - Overview
10. Setting up Karma for Testing Directives
11. Testing Directives
12. End to End Testing - Overview
13. Setting up Karma for End to End Testing
14. End to End Testing - Part 1
15. End to End Testing - Part 2
16. Troubleshooting End to End Tests
17. Summary
```

در این قسمت به نحوه نوشتن تست برای کدهای انگولار پرداخته می‌شود. در برنامه‌های انگولار از Karma برای نوشتن تست‌ها استفاده می‌کنیم اگرچه می‌توان با ابزارهای دیگری نیز اینکار را انجام داد، اما برای تست کردن برنامه‌های انگولار Karma بهترین گزینه است. در این بخش همچنین با نحوه‌ی تست کردن کنترلرها، سرویس‌ها، فیلترها و دایرکتیوها آشنا خواهید شد. در نهایت آزمون‌های End-to-End نیز با بررسی مثال‌های متنوع بررسی خواهد شد.

عنوان: **Performance در AngularJS قدم اول**
نویسنده: بهنام محمدی
تاریخ: ۱۳۹۴/۰۳/۲۰ ۹:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Performance, AngularJS

به احتمال زیاد برنامه نویسانی که از AngularJS در پروژه‌های خود استفاده می‌کنند، در برخی موارد کند شدن در Rendering و Binding صفحات را تجربه کرده اند. این مقاله مطالب خیلی ساده و راحتی در خصوص استفاده درست و بهینه از Binding می‌باشد.

قدم اول و مهم بحث on time binding هست:

در برخی موارد ما اطلاعاتی که فقط یکبار Bind می‌شوند و در طول اجرا هیچ تغییری نمی‌کنند را درست Bind نمی‌کنیم. برای مثال فرض کنید می‌خواهیم عنوان صفحه را در یک تگ h1 نمایش دهیم. به صورت معمول همه‌ی ما از روش زیر استفاده می‌کنیم.

```
<h1>{{title}}</h1>
```

اما این روش درست نیست! چرا؟

AngularJS برای Rendering طرف View از Watcherها استفاده می‌کند که در هر لحظه Bindingها را رصد می‌کنند و در صورت تغییر قسمت مورد نظر از View، دوباره Render می‌شود. بحث اصلی خود Watcherها هستند که حتی Bindingهایی که هیچ وقت مقادیر آنها تغییر نمی‌کنند نیز رصد می‌شود و این باعث کندی عمل Watching در AngularJS می‌شود. در AngularJS نسخه‌ی 1.3 به بعد امکانی فراهم شده‌است که شما بتوانید Bindingهایی را که یک بار بیشتر تغییر نمی‌کنند، مشخص کنید. به کد زیر دقت کنید:

```
<h1>{{::title}}</h1>
```

بله دقیقا به همین راحتی! شما با اضافه کردن :: در ابتدای هر Binding مشخص می‌کنید که این قسمت از View فقط یک بار Render شود. این عمل باعث می‌شود Watcher AngularJS کار کمتری انجام دهد.

به زودی مقاله‌های بیشتری در خصوص Performance در AngularJS خواهم نوشت. امیدوارم لذت برده باشید.

نظرات خوانندگان

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۴/۰۳/۲۳ ۱۳:۳۹

در نسخه‌های قدیمی‌تر نیز می‌توانید از Angular bindonce استفاده کنید.

عنوان: Performance در AngularJS قدم دوم
نویسنده: بهنام محمدی
تاریخ: ۱۳۹۴/۰۳/۲۰ ۹:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Performance, AngularJS

در مقاله‌ی قبل روش درست استفاده کردن از Binding را برای بهبود Performance، توضیح دادم. در این مقاله می‌خواهم در مورد ng-if و فرق آن با ng-show صحبت کنم و اینکه کدامیک Performance بهتری را برای AngularJS فراهم می‌کنند.

سول اول، کار ng-show چیست؟

ng-show یکی از پر کاربردترین Directiveهای AngularJS است که وظیفه‌ی Show و Hide قسمتی از Vew را به عهده دارد. به کد زیر توجه کنید:

```
<div ng-show="has">
     <div ng-repeat="item in items">
          <span>{{item.title}}</span>
     </div>
</div>
```

در این کد ما از ng-show استفاده کرده‌ایم. اگر has مقدار true داشته باشد div نمایش داده می‌شود و اگر false باشد، div نمایش داده نمی‌شود. در داخل div، لیستی وجود دارد که توسط ng-repeat تکرار شده و یک لیست ساده را درست می‌کند.

سول دوم، کار ng-if چیست؟

کد بالا را دوباره تکرار می‌کنیم. ولی با این تفاوت که اینبار بجای ng-show از ng-if استفاده خواهیم کرد:

```
<div ng-if="has">
     <div ng-repeat="item in items">
          <span>{{item.title}}</span>
     </div>
</div>
```

خوب؟ آیا عملکرد این دو کد با هم تفاوت دارد؟ جواب سؤال در ظاهر خیر هست. یعنی مانند کد بالایی، اگر has مقدار true داشته باشد، div نمایش داده می‌شود؛ در غیر اینصورت، خیر.

سوال سوم، پس اگر عملکرد یکسانی دارند، تفاوت آن‌ها در چیست؟

تفاوت این دو Directive در Performance هست. اجازه دهید بیشتر توضیح دهم. ng-show اگر مقدار false دریافت کند، tag مورد نظر را نمایش نمی‌دهد؛ ولی تگ‌های داخلی آن توسط AngularJS پردازش می‌شوند. یعنی چه ng-show مقدار true بگیرید و یا false، لیست داخل tag، توسط AngularJS پردازش و Render می‌شود. ولی در ظاهر ما در View چیزی را نمی‌بینیم. ng-if بر حسب مقادیری که دریافت می‌کند، می‌تواند به بالا رفتن Performance AngularJS کمک کند. فرض کنید ng-if مقدار false گرفته است؛ یعنی has مقدار false دارد. علاوه بر اینکه tag div نمایش داده نمی‌شود، بلکه داخل tag نیز پردازش نمی‌شود. یعنی لیستی که ما در کد نوشته‌ایم، به هیچ عنوان توسط AngularJS پردازش نخواهد شد که باعث می‌شود Watcher، کار کمتری انجام دهد. پس در نتیجه بهبودی را در کارآیی Rendering و Binding خواهیم داشت.

عنوان: **Performance در AngularJS قدم سوم**
نویسنده: بهنام محمدی
تاریخ: ۱۳۹۴/۰۳/۲۳ ۱۲:۵۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Performance, AngularJS

خیلی خوشحالم که تا این مرحله، این مقاله‌ها را دنبال می‌کنید. در مقالات قبل مسائل ساده و مهمی در بحث Performance مطرح شد. در این مقاله می‌خواهم قدم سوم در بهبود Performance را توضیح دهم که رعایت کردن این مسائل می‌تواند کمک زیادی در بهبود عملکرد برنامه‌های مبتنی بر AngularJS داشته باشد.

scope؟

همه‌ی برنامه نویسان و توسعه دهندگان، یکی از اولین مفاهیمی را که در AngularJS یاد می‌گیرند، scope هست. اما scope چیست؟ به صورت خیلی ساده می‌توان گفت scope مشخص کننده‌ی حوزه متغیرها و توابعی هست که قرار است در View تاثیر داشته باشند. کد زیر را مشاهده کنید:

```
<div>نام و نام خانوادگی: {{name}}</div>
<div>معدل: {{avg()}}<div>
```

و کد سمت controller

```
scope.nums=[19,20,17,16,15,18,19];
scope.name='بهنام محمدی';
scope.avg= function(){
  return scope.nums.reduce(function(previousValue, currentValue) {
      return previousValue + currentValue;
   })/scope.nums.length;
}
```

و اما خروجی نهایی

```
نام و نام خانوادگی: بهنام محمدی
معدل:17.71
```

خوب چون ما در قسمت controller به صورت scope.name و scope.avg عمل کرده‌ایم، می‌توانیم در View به صورت name و avg از این متغیرها استفاده کنیم. در نتیجه اگر ما در controller، به جای scope.name بنویسیم name و یا به جای scope.avg بنویسیم avg به مشکل بر می‌خوریم؛ چون قسمت View ما متغیرهای داخل scope را در View دخیل می‌کند و متغیرهای داخل scope توسط Watcherها رصد می‌شود.

خوب سؤال، همه چیز که عالی هست پس مشکل در کجاست؟

مشکل در متغیر scope.nums هست! به کد زیر توجه کنید:

```
var nums=[19,20,17,16,15,18,19];
scope.name='بهنام محمدی';
scope.avg= function(){
  return nums.reduce(function(previousValue, currentValue) {
      return previousValue + currentValue;
   })/nums.length;
}
```

فکر کنم متوجه تفاوت این کد با کد بالایی شده‌اید. اما کدام کد درست است؟ یا بهتر بگویم کدام کد بر روی Performance تاثیر مناسبی دارد؟ کد پایینی Performance بالایی دارد. دلیل این موضوع این است وقتی ما از nums در View هیچ استفاده‌ای نمی‌کنیم، بهتر است به صورت var nums تعریف شود. در کد بالایی که این متغیر به صورت scope.nums تعریف شده بود، با اینکه در View استفاده نشده بود، ولی توسط Watcher AngularJS در هر لحظه رصد می‌شود و این کار باعث کندی و کاهش عملکرد AngularJS خواهد شد. بنابراین در کل متغیرهایی را که در View استفاده نمی‌کنید، به صورت var test استفاده نمایید تا Watcher AngularJS این متغیرها را رصد نکند.

امیدوارم از این مقاله لذت برده باشید. منتظر مقاله بعدی من باشید.

عنوان: **Performance در AngularJS قدم چهارم**
نویسنده: بهنام محمدی
تاریخ: ۱۳۹۴/۰۴/۰۶ ۱۴:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Performance, AngularJS

امیدوارم از مقالات قبلی لذت برده باشید. در این مقاله می‌خواهم در مورد watch$ صحبت کنم.

سوال اول: watch$ چیست و چه کاربردی دارد؟

watch$ همان عملکرد Watching در AngularJS را انجام می‌دهد؛ ولی کاربردهای جالبی دارد. به کد زیر دقت کنید.

```
var errorChat=false;
$scope.$watch(function () {
     return errorChat;
}, function (newValue, oldValue) {
     if(newValue ==true){
          alert('قسمت محاوره سامانه با مشکل روبرو شده است لطفا با مدیریت تماس بگیرید')
     }
});
```

فرض کنید سناریوی پروژه به این صورت است که ما قسمت‌های مختلفی را در صفحه داریم و یکی از این قسمت‌ها، چت روم می‌باشد. می‌خواهیم در صورتیکه خطای اتصال و یا هر نوع خطایی در این قسمت بود، با پیغامی به کاربر گزارش داده شود. ما از متغیر errorChat برای این کار استفاده کرده‌ایم. با فرض اینکه در توابع دیگر در صورتیکه خطایی وجود داشته باشد، مقدار این متغیر را true می‌کند و ما بر حسب رصد این متغیر پیغام خطا را نمایش می‌دهیم.

سوال دوم: این کد به نحوه احسن کار می‌کند؛ پس مشکل کجاست؟

مشکل اینجاست بعد از اینکه متغیر errorChat مقدار true گرفت و ما هشدار را نمایش دادیم، باز این متغییر رصد می‌شود که لزومی به این کار نیست (فرض ما بر این است که بعد از بروز خطا دیگر قسمت چت روم کار نخواهد کرد) و متوقف کردن این متغیر باعث می‌شود Performance در نهایت بهتر شود. خوب برای اینکه رصد این متغییر را متوقف کنیم از کد زیر استفاده می‌کنیم. به کد زیر توجه کنید:

```
var errorChat=false;
var stop=$scope.$watch(function () {
     return errorChat;
}, function (newValue, oldValue) {
     if(newValue ==true){
          stop();
          alert('قسمت محاوره سامانه با مشکل روبرو شده است لطفا با مدیریت تماس بگیرید')
     }
});
```

خوب؛ فکر کنم تفاوت این دو کد با هم روشن است. ما یک متغییر stop به کد اضافه کردیم. این متغیر با تابع متوقف کننده watch$ مربوط به errorChat پر می‌شود و در نهایت با اجرای این تابع، عمل watching متغییر errorChat متوقف می‌شود. اگر با setInterval یا setTimeout در Javascript کار کرده باشید، شباهت این موردها را متوجه خواهید شد.

عنوان: **Performance در AngularJS قدم پنجم**
نویسنده: بهنام محمدی
تاریخ: ۱۳۹۴/۰۴/۱۰ ۲۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Performance, AngularJS

در این مقاله موضوعی را مطرح خواهم کرد که شاید برای خیلی‌ها این نوع کد نویسی خوشایند نباشد. حتی برای خود من هم خوشایند نیست؛ ولی نهایتا در بهبود Performance تاثیر خیلی زیادی دارد. به کد زیر دقت کنید.

```
<div ng-repeat="item in items">
     <div ng-if="setting.header">{{item.header}}</div>
     <div>{{item.title}}</div>
     <div ng-if="setting.footer">{{item.footer}}</div>
</div>
```

توضیح کد: فرض کنید سناریوی پروژه ما به این صورت هست که ما یک لیست داریم، شامل 3 فیلد که header، title و footer را در تنظیمات می‌توانیم مشخص کنیم که header و footer در شرایطی نمایش داده شود و در شرایطی نمایش داده نشود و یا حالت‌های دیگر.

خوب مشکل چیست و راهکار چیست؟

فرض کنید لیست ما شامل 100 رکورد هست و در تنظیمات مشخص کرده‌ایم که header نمایش داده شود و footer نمایش داده نشود. اما اتفاقی بدی که میفتد این است که وقتی لیست در View ساخته می‌شود، 100 بار ng-if مربوط به header و footer چک میشود؛ در جمع 200 بار می‌شود. چه این مقادیر true باشند چه false فرقی نمی‌کند و 200 بار بررسی می‌شود.

راهکار این مشکل به این صورت است که ما باید از ng-if داخل ng-repeat استفاده نکنیم. اما برای پیاده سازی تنظیمات باید ng-if‌ها را قبل از ng-repeat بررسی کنیم. پس مسلما ng-repeat‌ها باید قالب پیش بینی کرده ما را نسبت به ng-if‌ها درست کند.

نتیجه‌ی کار به صورت کد زیر است که شاید برای شما هم خوشایند نباشد:

```
<div ng-if="setting.header && setting.footer">
     <div ng-repeat="item in items">
          <div>{{item.header}}</div>
          <div>{{item.title}}</div>
          <div>{{item.footer}}</div>
     </div>
</div>
<div ng-if="setting.header && setting.footer==false">
     <div ng-repeat="item in items">
          <div>{{item.header}}</div>
          <div>{{item.title}}</div>
     </div>
</div>
<div ng-if="setting.header==false && setting.footer">
     <div ng-repeat="item in items">
          <div>{{item.title}}</div>
          <div>{{item.footer}}</div>
     </div>
</div>
<div ng-if="setting.header==false && setting.footer==false">
     <div ng-repeat="item in items">
          <div>{{item.title}}</div>
     </div>
</div>
```

درست است من هم با شما موافق هستم که خوشایند نیست. در این کد ما همه‌ی حالت‌ها را پیش بینی و قالب مناسب هر شرط را درست کرده‌ایم. حجم کد چند برابر شده، ولی از لحاظ Performance در ساخت لیست در View در حد 98% بهبود پیدا کرده‌است. همان مثال قبلی را در نظر بگیرید. ng-if مربوط به header و footer در این کد فقط 4 بار بررسی می‌شود. چه 100 رکورد باشد، چه 1000 تا، چه 10 تا رکورد.

در مورد ng-repeat‌ها هم نگران نباشید فقط یک بار اجرا میشوند. اگر کارکرد ng-if را در مقاله‌ی قبلی من ، خوانده باشید، متوجه‌ی این موضوع می‌شوید که elementهای داخلی و directionهای AngularJS داخلی ng-if زمانی پردازش می‌شوند که شرط true باشد. از این روش زمانی استفاده کنید که تعداد داده‌ها و حالت‌های زیادی دارید و Performance اهمیت بیشتری دارد.

امیدوارم مقاله‌ی مفیدی باشد.

عنوان: **کنترلرهای تو در تو در angularJs و نحوه ارث بری متدها و property ها در آن - بخش اول**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۵/۲۴ ۲۳:۵۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, AngularJS

در این مقاله قصد داریم تا به بررسی و پیاده سازی کنترلرهای تو در تو در AngularJs بپردازیم. به این صورت که میتوانیم در یک صفحه یک کنترلر اصلی را در نظر بگیریم و کنترلرهای دیگر را در این کنترلر قرار دهیم. نحوه‌ی ارث بری متغیرها، انقیاد داده‌ها و مقادیر تعریف شده در $scope، از جمله مواردی است که به آن خواهیم پرداخت. تمامی مواردی که ذکر خواهد شد در قالب یک پروژه قرار داده شده است. AngularJsNestedController.zip

حال به بررسی جزئی کنترلرهای تودرتو یا همان Nested Controllers می‌پردازیم. در این پروژه قصد داریم تا با تعریف سه متغییر firstName و lastName و middleName نحوه ارث بری پارامترها را در این کنترلرهای تو در تو بررسی نماییم. ساختار یک صفحه با کنترلرهای تو در تو به صورت زیر میباشد. همانطور که مشاهده میکنید کنترلر دوم و سوم به صورت nested درون کنترلر اول تعریف شده اند.

```
<body ng-app ="app">
      <div ng-controller="firstControllerScope">
            <div ng-controller="secondControllerScope">
                  <div ng-controller="thirdControllerScope">
                  </div>
             </div>
    </div>
</body>
```

در کدهای پیاده سازی شده این کنترلر به صورت زیر میباشد. همانطور که مشاهده میکنید در کنترلر اول تنها پارامتر firstName و در کنترلر دوم و سوم نیز به ترتیب lastName و middleName اضافه شده اند. همچنین دو تابع utility به نام getFullName در کنترلر دوم و سوم تعریف شده است.

```
var firstControllerScope = function ($scope) {
    // Initialize the model variables
    $scope.firstName = "John";
};

var secondControllerScope = function ($scope) {
    // Initialize the model variables
    $scope.lastName = "Doe";

    // Define utility functions
    $scope.getFullName = function () {
        return $scope.firstName + " " + $scope.lastName;
    };
};

var thirdControllerScope = function ($scope) {
    // Initialize the model variables
    $scope.middleName = "Al";
    $scope.lastName = "Smith";

    // Define utility functions
    $scope.getFullName = function () {
        return $scope.firstName + " " + $scope.middleName + " " + $scope.lastName;
    };
};
```

فایل html حاوی این کنترلرها در زیر نمایش داده شده است. همانطور که مشاهده میکنید تمامی کنترلر دوم یعنی secondControllerScope و کنترلر سوم درون firstControllerScope تعریف شده اند. همچنین دو تابع utility با نام‌های getFullName نیز در زیر مقدار fullName را باز میگردانند.

```
<div ng-controller="firstControllerScope">
        <h3>First controller</h3>
        <strong>First name:</strong> {{firstName}}<br />
        <br />
        <label>first name: <input type="text" ng-model="firstName" /></label><br />
        <br />
```

```
        <div ng-controller="secondControllerScope">
            <h3>Second controller (inside First)</h3>
            <strong>First name (from First):</strong> {{firstName}}<br />
            <strong>Last name (new variable):</strong> {{lastName}}<br />
            <strong>Full name:</strong> {{getFullName()}}<br />
            <br />
            <label>first name: <input type="text" ng-model="firstName" /></label><br />
            <label>last name: <input type="text" ng-model="lastName" /></label><br />
            <br />


            <div ng-controller="thirdControllerScope">
                <h3>Third controller (inside Second and First)</h3>
                <strong>First name (from First):</strong> {{firstName}}<br />
                <strong>Middle name (new variable):</strong> {{middleName}}<br />
                <strong>Last name (from Second):</strong> {{$parent.lastName}}<br />
                <strong>Last name (redefined in Third):</strong> {{lastName}}<br />
                <strong>Full name (redefined in Third):</strong> {{getFullName()}}<br />
                <br />
                <label>first name: <input type="text" ng-model="firstName" /></label><br />
                <label>middle name: <input type="text" ng-model="middleName" /></label><br />
                <label>last name: <input type="text" ng-model="lastName" /></label>
            </div>
        </div>
    </div>
```

با توجه به مقادیر تعریف شده درون هر یک از کنترلرها و با اجرای پروژه نمونه مشاهده میکنیم که اگر یک پارامتر مدل (مانند middleName) را درون کنترلر فرزند به وسیله input fieldها تغییر دهیم، این تغییر به والد منتقل نمیشود. یعنی اگر به عنوان مثال ما lastName را در کنترلر سوم تغییر دهیم دیگر lastName که در کنترلر سوم است تغییر نخواهد کرد. حال اگر همان lastName در کنترلر دوم که در یک فیلد input قرار داده شده تغییر کند، مشاهده میکنیم که در کنترلر سوم نیز مقدار lastName تغییر خواهد کرد.

در بخش بعدی نوع دیگری از ارث بری پارامترها و تابع‌ها در کنترلرهای AngularJs را شرح خواهیم داد.

عنوان: **کنترلرهای تو در تو در angularJs و نحوه ارث بری متدها و property ها در آن - بخش دوم**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۵/۲۵ ۱۷:۳۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, AngularJS

در بخش قبلی به چگونگی ساخت کنترلرهای تو در تو در AngularJs پرداختیم. همچنین بررسی نمودیم که propertyهای تعریف شده در کنترلر ما چگونه قابل استفاده توسط کنترلر فرزند میباشند.
حال روشی دیگر را برای ارث بری تابع‌ها و propertyها، در کنترلرهای تو در تو معرفی می‌نماییم. لازم به ذکر است که سورس پروژه را می‌توانید از لینک زیر دریافت نمایید: AngularJsNestedController.zip

کد جاوااسکریپت زیر سه کنترلر تو در تو را پیاده سازی می‌کند:

```
var firstControllerObj = function ($scope) {
    // Initialize the model object
    $scope.firstModelObj = {
        firstName: "John"
    };
};

var secondControllerObj = function ($scope) {
    // Initialize the model object
    $scope.secondModelObj = {
        lastName: "Doe"
    };

    // Define utility functions
    $scope.getFullName = function () {
        return $scope.firstModelObj.firstName + " " +
          $scope.secondModelObj.lastName;
    };
};

var thirdControllerObj = function ($scope) {
    // Initialize the model object
    $scope.thirdModelObj = {
        middleName: "Al",
        lastName: "Smith"
    };

    // Define utility functions
    $scope.getFullName = function () {
        return $scope.firstModelObj.firstName + " " +
          $scope.thirdModelObj.middleName + " " +
          $scope.thirdModelObj.lastName;
    };
};
```

کد html زیر هم از کنترلرهای تعریف شده‌ی در بالا، به صورت زیر استفاده کرده است. نگاهی کوتاه و مختصر به این فایل و سپس به شرح فرآیند ارث بری propertyها می‌پردازیم:

```
<div ng-controller="firstControllerObj">
        <h3>First controller</h3>
        <strong>First name:</strong> {{firstModelObj.firstName}}<br />
        <br />
        <label>Set the first name: <input type="text" ng-model="firstModelObj.firstName" /></label><br
/>
        <br />


        <div ng-controller="secondControllerObj">
            <h3>Second controller (inside First)</h3>
            <strong>First name (from First):</strong> {{firstModelObj.firstName}}<br />
            <strong>Last name (from Second):</strong> {{secondModelObj.lastName}}<br />
            <strong>Full name:</strong> {{getFullName()}}<br />
            <br />
            <label>Set the first name: <input type="text" ng-model="firstModelObj.firstName"
/></label><br />
            <label>Set the last name: <input type="text" ng-model="secondModelObj.lastName"
/></label><br />
            <br />
```

```
            <div ng-controller="thirdControllerObj">
                <h3>Third controller (inside Second and First)</h3>
                <strong>First name (from First):</strong> {{firstModelObj.firstName}}<br />
                <strong>Middle name (from Third):</strong> {{thirdModelObj.middleName}}<br />
                <strong>Last name (from Second):</strong> {{secondModelObj.lastName}}<br />
                <strong>Last name (from Third):</strong> {{thirdModelObj.lastName}}<br />
                <strong>Full name (redefined in Third):</strong> {{getFullName()}}<br />
                <br />
                <label>Set the first name: <input type="text" ng-model="firstModelObj.firstName"
/></label><br />
                <label>Set the middle name: <input type="text" ng-model="thirdModelObj.middleName"
/></label><br />
                <label>Set the last name: <input type="text" ng-model="thirdModelObj.lastName"
/></label>
            </div>
        </div>
    </div>
```

اگر نگاهی به کنترلرهای فوق بیاندازیم می‌بینیم که در firstControllerObj تنها یک شیء درون scope قرار داده شده که firstModelObj نامیده شده است و این شی تنها دارای یک property با نام firstName است. در کنترلر دوم نیز یک شیء جدید با نام secondModelObj ساخته شده است؛ با این تفاوت که property آن lastName نام دارد. حال به بدنه‌ی فایل html دو کنترلر توجه کنیم. می‌بینیم که در کنترلر دوم مستقیما پراپرتی firstName از شیء درون کلاس اول فراخوانی شده است. با اجرای پروژه‌ی نمونه خواهیم دید که با تغییر input شامل firstName در کنترلر دوم، این مدل در کنترلر اول نیز تغییر خواهد کرد.
بنابراین ما می‌توانیم در کنترلر فرزند به objectهای درون کنترلر والد دسترسی داشته باشیم و مقادیر آنرا تغییر دهیم. باید توجه داشت که این نوع تعریف مدل با آنچه که در بخش اول گفته شد، تفاوتی اساسی دارد. در بخش قبل با تغییر مدل توسط کنترلر فرزند، آن مدل در کنترلر والد تغییری نمی‌کرد؛ اما در روشی که در این مقاله شرح داده شد مشاهده نمودیم که با تغییر مدل در کنترلر فرزند مقدار مدل در کنترلر والد نیز تغییر می‌کند.

عنوان: **اعتبارسنجی درخواست های $http با استفاده از یک Interceptor**
نویسنده: محمود راستین
تاریخ: ۱۳۹۴/۰۶/۱۱ ۱۰:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVC, Security, AngularJS, ASP.NET Identity

پیش نیاز :

هدایت خودکار کاربر به صفحه لاگین در حین اعمال Ajax ایی
Angular Interceptors

ابتدا مشکل و هدف را بیان می‌کنیم:

مشکل : کاربر در صفحه‌ای حضور دارد که نیاز به اعتبارسنجی داشته و مدت اعتبار کاربر نیز تمام شده است، ولی هنوز در صفحه‌ای که نباید حضور داشته باشد، حضور دارد و بدتر از آن این است که می‌تواند درخواست‌های بی نتیجه‌ای را نیز ارسال کند.
هدف : کاربر را سریعا به صفحه‌ای که به آن تعلق دارد هدایت کنیم (یعنی صفحه‌ی ورود به سیستم).

و حالا از ابتدا پروسه را دنبال می‌کنیم. یک Controller سمت سرور داریم به این صورت :

```
[Authorize(Roles = AuthorizeRole.SuperAdministrator)]
    public partial class HomeController : Controller
    {
        [HttpPost]
        [AngularValidateAntiForgeryToken]
        public virtual JsonResult GetUserInfo()
        {
            var userInfoViewModel = _applicationUserManager.GetUserInfoById(User.Identity.GetUserId());
            return Json(userInfoViewModel);
        }
    }
```

همانطور که مشاهده می‌کنید از Authorize پیش فرض استفاده کرده‌ایم. حالا سمت کلاینت با استفاده از HTTP می‌خواهیم درخواستی را ارسال کنیم.

یعنی با کدی همانند کد زیر:

```
$scope.getUserInfo = function() {
            $http({
                    method: 'POST',
                    url: 'Home/GetUserInfo',
                    headers: $scope.getHeaders()
                }).
                success(function(data, status, headers, config) {
                    $scope.userInfo = data;
                }).
                error(function(data, status, headers, config) {

                }).then(function(res) {

                });
        }
```

و نتیجه‌ی کدهای بالا به صورت زیر درخواهد آمد :

```
success(function(data, status, headers, config) {  data = Object {UserName: "dotnettips.info"}, status = 200,
    $scope.userInfo = data;
}).
```

همانطور که می‌بینید داده‌های اولیه کاربر پس از ورود به سیستم، بدون هیچ مشکلی دریافت می‌شوند.

نکته : زمانیکه status برابر با 200 هست، یعنی درخواست OK می‌باشد. (در پیوست ، لیست تمامی کدها قرار داده شده است)

حالا فرض کنید کاربر در صفحه حضور دارد و به هر دلیلی اعتبار حضور کاربر منقضی شده است و حالا پس از مدتی کاربر درخواستی را به سرور ارسال می‌کند و می‌خواهد اطلاعات خودش را مشاهده کند.

درخواست کاربر با همان کدهای اولیه ارسال می‌شود و خروجی اینبار به صورت زیر در خواهد آمد :

```
success(function(data, status, headers, config) {  data = "", status = 200,
    $scope.userInfo = data;
}).
```

همانطور که می‌بینید وضعیت اینبار نیز OK می‌باشد، ولی هیچ داده‌ای از سرور دریافت نشده است. کاربر قطعا در اینجا دچار سردرگمی می‌شود. چون هیچ چیزی را مشاهده نمی‌کند و به هیچ مسیر دیگری نیز هدایت نمی‌شود و هرکار دیگری نیز انجام دهد، پاسخی مشاهده نمی‌کند.

راه حل چیست ؟

ابتدا باید برای درخواست‌های Ajax ایی اعتبارسنجی را اعمال کنیم. برای این کار باید یک Attribute جدید بسازیم. یعنی به این صورت :

```
[AttributeUsage(AttributeTargets.Class | AttributeTargets.Method, Inherited = true, AllowMultiple =
true)]
    public class MyCustomAuthorize : AuthorizeAttribute
    {
        protected override void HandleUnauthorizedRequest(AuthorizationContext filterContext)
        {
            if (filterContext.HttpContext.Request.IsAjaxRequest())
            {
                // یعنی اعتبارسنجی نشده است
                filterContext.HttpContext.Response.StatusCode = (int)HttpStatusCode.Unauthorized;
                filterContext.HttpContext.Response.End();
            }

            base.HandleUnauthorizedRequest(filterContext);
        }
    }
```

با استفاده از Attribute بالا ما درخواست‌های ای‌جکسی را نیز اعتبارسنجی می‌کنیم.

توجه : کد کامل‌تر همراه با توضیحات در بخش پیش نیازها آمده است.

حالا در سمت کلاینت نیز به این صورت عمل می‌کنیم :

```
$http({
                    method: 'POST',
                    url: 'Home/GetUserInfo',
                    headers: $scope.getHeaders()
```

```
                }).
                success(function(data, status, headers, config) {
                    $scope.userInfo = data;
                }).
                error(function(data, status, headers, config) {
                    if (status === 401) {
                        // you are not authorized
                    }
                }).then(function(res) {

                });
```

حالا دیگر متوجه خواهیم شد که کاربر اعتبارسنجی نشده است، یا اعتبار آن منقضی شده است و می‌توانیم کاربر را به مسیر **"ورود به سیستم"** هدایت کنیم.

در console مرورگر نیز خطای زیر رخ می‌دهد :

```
POST http://localhost:000000/Administrator/Home/GetUserInfo 401 (Unauthorized)
```

اکنون به هدف خودمان رسیده‌ایم. اما برای هر درخواست باید این دستوارت را تکرار کنیم ؟! قطعا خیر.

حالا باید از یک Interceptor استفاده و درخواست‌های HTTP خودمان را مدیریت کنیم. برای این منظور ما یک Interceptor جدید را همانند کدهای زیر می‌نویسیم:

```
factory('AuthorizedInterceptor', function ($q) {
        return {
            response: function(response) {
                return response || $q.when(response);
            },
            responseError: function(rejection) {
                if (rejection.status === 401) {
                    // you are not authorized
                }
                return $q.reject(rejection);
            }
        };
    })
```

همانطور که مشاهده می‌کنید کدهای خطای درخواست $http را در Interceptor قرار دادیم. حالا کاربر درخواستی را ارسال می‌کند و ما با توجه به این Interceptor متوجه خواهیم شد که آیا اعتبار آن منقضی شده است یا خیر و در صورت منقضی شدن، کاربر را به صفحه‌ی ورود هدایت خواهیم کرد. یعنی بدین صورت :

```
}).factory('AuthorizedInterceptor', function ($q) {
    return {
        response: function(response) {
            return response || $q.when(response);
        },
        responseError: function(rejection) {  rejection = Object {data: "", status: 401, config: Object, statusText: "Unauthorized"}
            if (rejection.status === 401) {
                // you are not authorized
            }
            return $q.reject(rejection);
        }
    };
});
```

در خروجی بالا می‌بینید که کاربر، اعتبارسنجی نشده است و آن را به مسیر ورود هدایت خواهیم کرد.

با Interceptor بالا دیگر نیازی نیست برای هر درخواستی این وضعیت را چک کنیم و به صورت خودکار این چک کردن رخ خواهد داد.

پیوست : http_status_codes.rar

نظرات خوانندگان

نویسنده: یاسر مرادی
تاریخ: ۱۳۹۴/۰۶/۱۳ ۱۳:۱

با سلام و احترام

ممنون بابت مقاله خوبتون

ولی این شرط

```
filterContext.HttpContext.Request.IsAjaxRequest
```

معنی

یعنی اعتبارسنجی نشده است

را نمی‌دهد.

شما در متد HandleUnauthorizedRequest هستید و بالطبع اعتبار درخواست رد شده است، کاری که می‌خواهید انجام دهید، اعتبارسنجی نیست، بلکه اصلاح Response درخواست‌های Ajax ای است که Status Code آنها ۲۰۰ است که نباید باشد.

این نوع دقت داشتن و شفافیت لازمه ایجاد یک کد صحیح است، چون الآن و در مرحله ای بالاتر، می‌توانیم بگوییم که هر Response ای که وارد متد HandleUnauthorizedRequest شده است، اگر Status Code 401 ندارد، Status Code آن برابر ۴۰۱ قرار داده شود. فارغ از Ajax بودن یا نبودن.

و باز در مرحله ای بالاتر اگر در پروژه از Owin استفاده شده باشد، یک Owin Middleware و اگر نه یک ASP.NET Module جای مناسب‌تری برای نوشتن این چنین کدی است.

از این که IsAjaxRequest همیشه درست کار نمی‌کند و فقط وقتی درخواست Ajax ای بگوید که Ajax ای است، کار خواهد کرد نیز بگذریم.

موفق و پایدار باشید.

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۶/۱۳ ۱۳:۲۷

نویسنده‌ی گرامی، خواهشمند است جهت رفاه حال خوانندگان گرانقدر، جمله‌ی مبهم «یعنی اعتبارسنجی نشده است» را به جمله‌ی بهتر «یعنی درخواست ای‌جکسی در این مرحله از کار اعتبارسنجی‌اش رد شده‌است و اکنون می‌خواهیم وضعیت خروجی آن‌را اصلاح کنیم» تغییر دهید. با تشکر فراوان!

نویسنده: محمود راستین
تاریخ: ۱۳۹۴/۰۶/۱۴ ۲:۵۶

با سلام
عبارت " یعنی اعتبارسنجی نشده است " برای تعریف خط زیرین خودش هستش که داره پاسخ رو اصلاح می‌کنه و برای شرط

استفاده نشده است که این پاسخ ما با توجه به کد وضعیت ،یعنی اعتبار سنجی نشده است . به بیان ساده‌تر معنی تغییر کد پاسخ می‌باشد.
من شخصا از مطلب جناب نصیری در اینجا استفاده می‌کنم برای این اعتبارسنجی. پیشنهاد میکنم این مطلب و نظرات اون رو ببینید. همچنین این مطلب در پیش نیازها درج شد تا در آینده خوانندگان دچار مشکل نشوند و ابتدا این مطلب را بخوانند.
و جناب مرادی در خط آخر مطلبی رو بیان کردید ، خوشحال میشیم جزئیات بیشتری رو بیان کنید. یعنی چی همیشه درست کار نمی‌کند ؟ چه شرایطی ممکن است رخ دهد که اشتباه کار کند ؟

با تشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۶/۱۴ ۱۰:۱۹

چه شرایطی ممکن است رخ دهد که اشتباه کار کند ؟

اگر این درخواست دستی صادر بشه (یک ربات باشه) یا اگر فایروال خاصی در این میان هدرهای Ajax رو حذف کنه . مورد اول مهم نیست، چون نیازی نیست تا ربات را به صفحه‌ی لاگین هدایت کرد (مرورگر باز شده‌ای نداره). مورد دوم هم خیلی خیلی بعید هست. یعنی من ندیدم که در عمل فایروال‌های شرکت‌ها هدرهای HTTP رو تا این حد دستکاری کنند.

عنوان: **ایجاد سرویس Account Manager با تکنولوژی های Identity 2.1 و Web API 2.2**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۶/۲۲ ۱۷:۵۵
آدرس: www.dotnettips.info
گروه‌ها: Entity framework, Security, ASP.NET Web API, AngularJS, ASP.NET Identity, JSON Web Token

ASP.NET Identity 2.1 جدیدترین فریم ورک عضویت و مدیریت کاربر است که چندی پیش توسط شرکت مایکروسافت منتشر شد. این سیستم عضویت می‌تواند به تمامی فریم‌ورک‌های دات نتی مانند Web API، MVC و ... متصل گردد.
در این دوره چند قسمتی به همراه یک پروژه‌ی نمونه، نحوه‌ی ارتباط Identity و Web API را نمایش خواهیم داد. در قسمت front-end این پروژه‌ی SPA، ما از AngularJs استفاده خواهیم نمود. قسمت front-end که توسط AngularJs توسعه داده می‌شود از bearet token based authentication استفاده می‌کند. این متد از JSON Web Token برای فرایند Authorization بهره می‌گیرد که به اختصار آن را JWT نیز می‌نامند. این روش سیستم اعتبار سنجی role based بوده و تمامی آنچه را که در یک سیستم membership داریم، پوشش می‌دهد. توجه داشته باشید که برای فهم بهتر تمامی مراحل، ما پروژه را بدون هیچ قالب از پیش تعریف شده‌ای در VS2013 آغاز می‌کنیم.
به دلیل اینکه پروژه در طی چند قسمت انجام می‌پذیرد آن را به بخش‌های زیر تقسیم می‌کنیم.

تنظیمات اولیه ASP.NET Identity 2.1 با Web API
ایجاد Account Confirmation به وسیله Identity به همراه تنظیمات policy برای user name و password
توسعه OAuth Json Web Token Authentication به کمک Web API و Identity
ایجاد یک سیستم Role Based و تایید صلاحیت‌های مربوط به آن
توسعه Web API Claims Authorization در Identity 2.1
توسعه بخش front-end با AngularJs

تنظیمات اولیه ASP.NET Identity 2.1 با Web API

1. تنظیمات ASP.Net Identity 2.1

1-1. ساخت یک پروژه Web API

در ابتدا ما یک empty solution را با نام "AspNetIdentity" همانند شکل مقابل می‌سازیم.

یک ASP.NET Web Application با نام "AspNetIdentity.WebApi" را به این solution اضافه می‌نماییم. در بخش select template ما empty template را انتخاب می‌کنیم. همچنین در قسمت add folders and core references for: نیز هیچیک از گزینه‌ها را انتخاب نمی‌کنیم.

1-2. نصب packageهای مورد نیاز از Nuget

در زیر packageهای مورد نیاز برای ASP.NET Web API و Owin را مشاهده میکنید. همچنین packageهای مربوط به ASP.Net Identity 2.1 نیز در زیر قرار داده شده‌اند.

```
Install-Package Microsoft.AspNet.Identity.Owin -Version 2.1.0
Install-Package Microsoft.AspNet.Identity.EntityFramework -Version 2.1.0
Install-Package Microsoft.Owin.Host.SystemWeb -Version 3.0.0
Install-Package Microsoft.AspNet.WebApi.Owin -Version 5.2.2
Install-Package Microsoft.Owin.Security.OAuth -Version 3.0.0
Install-Package Microsoft.Owin.Cors -Version 3.0.0
```

1-3. اضافه کردن user class و database context

حال که تمامی پکیج‌های مورد نیاز را به پروژه خود اضافه نمودیم، قصد داریم تا اولین کلاس EF را با نام "ApplicationUser" به پروژه اضافه کنیم. این کلاس، کاربری را که قصد ثبت نام در membership system، دارد را نمایش می‌دهد. برای این کار ما یک کلاس جدید را به نام "ApplicationUser" می‌سازیم و کلاس "Microsoft.AspNet.Identity.EntityFramework.IdentityUser" را در آن به ارث می‌بریم.

برای این کار ما یک پوشه‌ی جدید را در برنامه با نام "Infrastructure" می‌سازیم و درون آن کلاس ذکر شده را اضافه می‌کنیم:

```
public class ApplicationUser : IdentityUser
    {
        [Required]
        [MaxLength(100)]
        public string FirstName { get; set; }
```

```
        [Required]
        [MaxLength(100)]
        public string LastName { get; set; }

        [Required]
        public byte Level { get; set; }

        [Required]
        public DateTime JoinDate { get; set; }

    }
```

پس از افزودن کلاس User، نوبت به اضافه نمودن Db Context است. این کلاس وظیفه‌ی ارتباط با پایگاه داده را بر عهده دارد. ما یک کلاس جدید را با نام ApplicationDbContext، به پوشه‌ی Infrastructure اضافه می‌نماییم. کد مربوط به این کلاس به صورت زیر است:

```
public class ApplicationDbContext : IdentityDbContext<ApplicationUser>
    {
        public ApplicationDbContext()
            : base("DefaultConnection", throwIfV1Schema: false)
        {
            Configuration.ProxyCreationEnabled = false;
            Configuration.LazyLoadingEnabled = false;
        }

        public static ApplicationDbContext Create()
        {
            return new ApplicationDbContext();
        }

    }
```

همانطور که ملاحظه می‌کنید این کلاس از IdentityDbContext ارث بری نموده است. این کلاس یک نسخه‌ی جدیدتر از DbContext است که تمامی نگاشت‌های Entity Framework Code First را انجام می‌دهد. در ادامه ما یک Connection String را با نام DefaultConnection در فایل web.config اضافه می‌نماییم.

همچنین متد static ایی را با نام Create که در تکه کد فوق از سوی Owin Startup class فراخوانی می‌گردد که در ادامه به شرح آن نیز خواهیم پرداخت.

ConnectionString قرار داده شده در فایل web.config در قسمت زیر قرار داده شده است:

```
<connectionStrings>
    <add name="DefaultConnection" connectionString="Data Source=.\sqlexpress;Initial
Catalog=AspNetIdentity;Integrated Security=SSPI;" providerName="System.Data.SqlClient" />
  </connectionStrings>
```

قدم 4: ساخت پایگاه داده و فعال سازی DB Migration

حال ما باید EF CodeFirst Migration را برای آپدیت کردن دیتابیس، بجای دوباره ساختن آن به ازای هر تغییری، فعال نماییم. برای این کار بایستی در قسمت NuGet Package Manager Console عبارت زیر را وارد نماییم:

```
enable-migrations
add-migration InitialCreate
```

اگر تا به اینجای کار تمامی مراحل به درستی صورت گرفته باشند، پوشه‌ی Migration با موفقیت ساخته می‌شود. همچنین پایگاه داده متشکل از جداول مورد نیاز برای سیستم Identity نیز ایجاد می‌شود. برای اطلاعات بیشتر می‌توانید مقالات مرتبط با Code First را مطالعه نمایید. ساختار جداول ما باید به صورت زیر باشد:

AspNetIdentity
Database Diagrams
Tables
System Tables
FileTables
dbo.__MigrationHistory
dbo.AspNetRoles
dbo.AspNetUserClaims
dbo.AspNetUserLogins
dbo.AspNetUserRoles
dbo.AspNetUsers
Columns
Id (PK, nvarchar(128), not null)
FirstName (nvarchar(100), not null)
LastName (nvarchar(100), not null)
Level (tinyint, not null)
JoinDate (datetime, not null)
Email (nvarchar(256), null)
EmailConfirmed (bit, not null)
PasswordHash (nvarchar(max), null)
SecurityStamp (nvarchar(max), null)
PhoneNumber (nvarchar(max), null)
PhoneNumberConfirmed (bit, not null)
TwoFactorEnabled (bit, not null)
LockoutEndDateUtc (datetime, null)
LockoutEnabled (bit, not null)
AccessFailedCount (int, not null)
UserName (nvarchar(256), not null)
Keys

در بخش بعدی، کلاس‌های مربوط به UserManager را ایجاد خواهیم کرد و پس از آن فایل Startup Owin را برای مدیریت کاربران، تشریح می‌کنیم.

عنوان: **قابلیت چند زبانه و Localization در AngularJs بخش اول: معرفی angular-translate**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۶/۲۳ ۱۸:۱۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Localization, AngularJS

در این مقاله قصد داریم با استفاده از ماژول Angular-Translate امکان ایجاد یک سیستم چند زبانه را تشریح کنیم.
angular-translate یک ماژول توسعه داده شده AngularJs میباشد که با استفاده از i18n و l10n، قابلیت چند زبانه را به صورت Lazy Loading برای شما فراهم می‌کند.
شما میتوانید با خط فرمان زیر، در بخش package-manager، کتابخانه‌های مربوط به angular-translate را به نرم افزار خود اضافه نمایید:

```
Install-Package AngularTranslate
```

تکه کد زیر یک مثال ساده از angular translate است که در صفحه‌ی اصلی این ماژول قرار داده شده است.

```
var app = angular.module('at', ['pascalprecht.translate']);

app.config(function ($translateProvider) {
  $translateProvider.translations('en', {
    TITLE: 'Hello',
    FOO: 'This is a paragraph.',
    BUTTON_LANG_EN: 'english',
    BUTTON_LANG_DE: 'german'
  });
  $translateProvider.translations('de', {
    TITLE: 'Hallo',
    FOO: 'Dies ist ein Paragraph.',
    BUTTON_LANG_EN: 'englisch',
    BUTTON_LANG_DE: 'deutsch'
  });
  $translateProvider.preferredLanguage('en');
});

app.controller('Ctrl', function ($scope, $translate) {
  $scope.changeLanguage = function (key) {
    $translate.use(key);
  };
});
```

با نگاهی ساده به تکه کد فوق می‌توان مراحل افزودن این ماژول و استفاده از آن را به صورت زیر ساده کرد:

1. وابستگی pascalprecht.translate را در بخش angular.module قرار می‌دهیم.
2. در بخش app.config وابستگی $translateProvider را تزریق می‌کنیم.
3. با استفاده از رویه‌ی translations زبان‌های مختلف را به همراه لیبل آنها اضافه می‌نماییم. توجه کنید که امکان خواندن این ریسورس‌ها از فایل txt و json نیز وجود دارد.
4. با استفاده از رویه‌ی preferredLanguage زبان پیش فرض سیستم تعیین می‌گردد.
5. در کنترلر فراخواننده‌ی تغییر زبان، باید وابستگی $translate را اضافه نماییم.
6. با استفاده از رویه‌ی use زبان مورد نظر را تغییر می‌دهیم. تمامی مراحل فوق در قالب یک پروژه نمونه در Plunker قرار داده شده است. در بخش بعدی به بررسی اجمالی قابلیت‌های این ماژول خواهیم پرداخت.

عنوان: قابلیت چند زبانه و Localization در AngularJs- بخش دوم: بررسی اجمالی قابلیت های ماژول angular-translate
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۶/۲۴ ۱۵:۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Localization, AngularJS

در بخش قبلی به معرفی ماژول angular-tanslate پرداختیم. در این بخش قصد داریم تا به بررسی مفهومی قابلیت‌های موجود در این ماژول بپردازیم.
شکل زیر یک شمای کلی را از قابلیت‌های angular-translate، نمایش می‌دهد. همانطور که ملاحظه می‌کنید در مواردی نظیر ذخیره سازی زبان‌ها، بارگذاری ریسورس‌های زبان‌ها و گرامرهای استفاده در DOM راهکارهای گوناگونی دارد.

angular-translate دایرکتیو و فیلتر هایی را به صورت کامپوننت عرضه کرده است که شما می‌توانید به وسیله‌ی آنها پروژه خود را به زبان‌های گوناگون localize کنید. آنچه که در تصویر فوق مشاهده می‌کنید در حقیقت ساختار ماژول angular-translate را نمایش می‌دهد. به این صورت که در اولین لایه دایرکتیو translate قرار گرفته است.

سطح بعدی شاید کمی جذاب‌تر به نظر برسد. هر دو بخش دایرکتیو و فیلتر از سرویس تزریق شده $translate استفاده می‌کنند. این بدین معنی است که در هنگام تغییر یک زبان به زبان دیگر به وسیله دایرکتیوهایی که در view نوشته شده است، در حقیقت شما این دایرکتیوها را به سطح فیلتر انتقال داده‌اید و پس از آن، فیلتر آن را به لایه سرویس برده و سرویس کار اصلی را انجام می‌دهد.

در سطح بعدی لایه Interpolator قرار گرفته است که وظیفه تغییر DOMها را بر عهده دارد. این فرآیند متغیرهایی را که در ریسورس قرار داده‌ایم، درایه به درایه در view و در مکان‌های هم نام قرار می‌دهد. این فرآیند به وسیله راهکارهای گوناگونی از قبیل Message Format و یا Angular Core Interpolation Service امکان پذیر است.

جعبه خاکستری رنگ بخش missing translation handler را نشان می‌دهد. این handlerها زمانی فراخوانی می‌گردند که translator به دنبال یک key می‌گردد که تعریف نشده باشد. angular-translate خود شامل یک logging service می‌باشد که به صورت extension باید آن را به پروژه خود اضافه نمایید که در بخش‌های بعدی در مورد آن مفصل‌تر بحث می‌کنیم.

بخش asynchronous loader شما را قادر می‌سازد که داده‌های زبان‌های متفاوت را به صورت غیر هم زمان بارگذاری نمایید. angular-translate از ماژول‌های asynchronous loader پشتیبانی می‌کند. روش معرفی شده در سایت مرجع در مورد بارگذاری غیر همزمان urlLoader و staticFilesLoader هستند که در بخش‌های بعدی به آن خواهیم پرداخت.

angular-translate برای ذخیره سازی داده‌های بازگذاری شده و نگه داشتن زبان سایت در صورت reload شدن صفحه راهکارهای متفاوتی را ارائه نموده است. این قابلیت در ابتدا چک می‌کند که ریسورس‌های زبان درون یک local storage قرار گرفته‌اند یا خیر. این عملیات طی یک فرآیند جستجوی key-value صورت می‌گیرد و در نهایت فایل‌های ریسورس زبان مورد نظر که درون کوکی یا localStorage ذخیره شده‌اند بارگذاری می‌شوند. angular-translate به صورت توکار دارای دو راهکار localStorage و cookieStorage برای ذخیره سازی ریسورس‌ها می‌باشد.

در بخش بعدی در مورد نحوه‌ی ذخیره سازی داده‌ها به دو روش cookieStorage و localStorage بیشتر صحبت خواهیم کرد.

عنوان: **بررسی مشکلات AngularJS 1.x**
نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۶/۲۶ ۱۹:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, MVC, jQuery, AngularJS, Code Review

از اولین مقاله‌ای که در مورد AngularJS در این سایت منتشر کردم، بیش از دو سال می‌گذرد. در آن زمان فقط از این فریمورک تعریف و تمجید کردم؛ اما بد نیست بعد از چند تجربه‌ی کاری دلایل تنفری را که نسبت به آن پیدا کرده ام، نیز بیان کنم.
اگر عبارت why I hate angularjs را در گوگل جستجو کنید، می‌بینید که فقط من این عقیده را پیدا نکرده‌ام و افراد دیگری نیز هستند که مثل من فکر می‌کنند و حتی از لحاظ فنی AngularJS را به چالش کشیده‌اند. برای مثال سایت I hate angular بیشتر مقالاتی را که ضد AngularJS هستند، گردآوری کرده است و برای بررسی مشکلات Angular می‌تواند شروع خوبی باشد.
البته قصد ندارم که از نظر فنی Angular را نقد کنم؛ فقط قصد به اشتراک گذاری یک سری از مشکلات توسعه‌ی Single Page Applicationها را با استفاده از فریمورک Angular، دارم و این را در نظر داشته باشید که بعضی از این مشکلات در هنگام توسعه SPAها با فریمورک‌هایی از این دست، گریبان‌گیر شما می‌شوند و الزاما ربطی به AngularJS ندارند.

سازگار نبودن افزونه‌های jQuery با Angular

برنامه‌های واقعی فقط از تعدادی ng-repeat تشکیل نشده‌اند که ما از دیدن آن‌ها ذوق زده شویم. خواسته یا ناخواسته مجبوریم در برنامه‌های وب خودمان از افزونه‌های محبوب جی‌کوئری نیز استفاده کنیم. خوب، خیلی هم خوب! چندین راه حل پیش روی ماست:

روش اول - نادیده گرفتن angular

انگار نه انگار که از angular استفاده می‌کنیم و افزونه مورد نظر را بدون در نظر گرفتن وجود angular، کاملا عادی فراخوانی کنیم.
نتیجه: ممکن است بعضی وقت‌ها جواب بدهد، ولی اکثر مواقع، نتیجه عجیب غریب است و خطاها قابل فهم نیستند و توانایی اشکال زدایی آن‌ها را نخواهید داشت. دلیلش هم مشخص است؛ چون Angular فازی به نام کامپایل و اصطلاحا context مربوط به خودش را دارد و فراخوانی افزونه مورد نظر، خارج از context انگولار رخ می‌دهد و انگولار از وجود این افزونه بی خبر است. حال ممکن است به طور اتفاقی، فراخوانی افزونه قبل، مابین و یا حتی بعد از فاز کامپایل انگولار رخ دهد. باز هم فرض کنید که بر حسب اتفاق همه چیز خوب پیش رفت، اما اکنون سایر قابلیت‌های خوب انگولار مثل ng- model و model binding آن در دسترس نیستند و در آخر به این نتیجه می‌رسید که پس چرا دارم از انگولار استفاده می‌کنم.

روش دوم - استفاده از directiveهای محصور کننده

راه اصولی برای استفاده از افزونه‌های جی‌کوئری در AngularJS، استفاده از directiveهای تهیه شده برای آن افزونه است. اگر خوش شانس باشید، معمولا برای افرونه‌های معروف، directive انگولاری آن نیز تهیه شده است. اما این همه‌ی داستان نیست؛ فرض کنید که از کتابخانه jQuery file upload ، برای آپلود فایل می‌خواهید استفاده کنید. خوشبختانه directive انگولاری نیز برای آن تهیه شده است و مستندات استفاده از آن هم، تنها مثالی هست که برای آن فراهم شده است. اما فرض کنید که می‌خواهید مانند مثال استفاده از آن در jQuery، یک file input که کاربر تنها بتواند یک فایل را از طریق کشیدن و رها کردن آپلود کند، با استفاده از Directive انگولاری آن پیاده سازی کنید. اما کار با این directive، به آسانی مثال جی‌کوئری آن نیست. یک‌کم که جلوتر بروید می‌بینید که این directive گنگ طراحی شده است. البته بیشتر directive هایی که اصطلاحا wrapper برای افزونه‌های جی‌کوئری هستند این مشکل را دارند و کار با آن‌ها چندان لذت بخش نیست و باید ساعت‌ها با آنها کلنجار رفت تا به نتیجه‌ی دلخواه رسید و همه‌ی این‌ها را در نظر بگیرید که اگر با apiهای jQuery آن کار می‌کردید، دیگر این مشکلات را نداشتید.
قبلا نیز یک نمونه‌ی دیگر از مشکلات استفاده از این گونه directiveهای محصور کننده را تحت مقاله ای با عنوان استفاده از افزونه isotope در انگولار به اشتراک گذاشتم.

روش سوم - استفاده از directive هایی که به صورت native با انگولار نوشته شده‌اند

اما چرا به هنگام استفاده از directiveهای محصور کننده افرونه‌های جی‌کوئری، با مشکلات زیادی روبرو می‌شویم؟ دلیلش این است که انگولار می‌گوید بهتر است این افزونه‌ها با استفاده از خود angular بازنویسی شوند. برای مثال برای آپلود فایل می‌توان از کتابخانه‌ی با کیفیت ng-file-upload که هیچ وابستگی به jQuery ندارد استفاده کرد. اما آیا واقعا برای تمامی افزونه‌های جی‌کوئری معادلی برای AngularJs آن با همان کیفیت تهیه شده است؟ جواب مطمئنا خیر است. برای مثال در حالی که برای datagrid افزونه‌های بی شماری برای جی‌کوئری تهیه شده است، اما برای angular تنها یکی دو تا directive با کیفیت تهیه شده‌است که نه تنها قابلیت رقابت با معادل‌های jQuery شان را ندارند، آنچنان نیز stable نیستند و در مستندات خودشان هشدار

می‌دهند که فلان ویژگی در حال تست هست و هنوز پایدار نیست.

روش چهارم – نوشتن directive توسط خودتان

به عنوان آخرین راه حل باید خودتان دست به کار شده و برای افزونه مورد نظرتان directive بنویسید. اما نوشتن directive برای افزونه‌های پیچیده‌ی جی‌کوئری به سادگی مثال‌های آموزشی AngularJS همانند چگونگی نوشتن directive برای jQueryUI Datepicker نیست. اگر کدهای directiveهای نوشته شده برای افزونه‌های پیچیده را بررسی کنید، کدهایی را می‌بینید که برای شما منطقی نیست. برای مثال ممکن است با تعداد زیادی setTimeOut مواجه شوید که احتمالا با نحوه‌ی کامپایل HTML توسط انگولار مرتبط است. در کل باید بدانید که نوشتن directive برای تعداد زیادی از افزونه‌ها کار راحتی نیست و احتمالش هست که قید این را کار نیز بزنید.

پس اگر قصد توسعه SPA با هر فریمورکی مثل angular را داشته باشید، این را در نظر داشته باشید که دیر یا زود هنگام استفاده از افزونه‌های جی‌کوئری به مشکل برخواهید خورد.

بیشتر امکانات تو کار ASP.NET MVC را از دست خواهید داد

به هنگام توسعه‌ی برنامه با استفاده از فریم ورک‌های SPA، امکانات توکار ASP.NET MVC مثل اعتبارسنجی یکپارچه و strongly typed viewها را از دست خواهید داد. شاید یک سری پروژه در Github پیدا کنید که سعی کرده‌اند این‌ها را با یکدیگر سازگار کنند. اما به محض استفاده متوجه می‌شوید که اگر همه‌ی کارها را خودتان با Angular انجام بدهید راحت‌تر هستید تا استفاده از کتابخانه‌های آزمایشی و ناقص.

البته باز هم نمی‌گویم که این‌ها تقصیر AngularJS است. ذات توسعه‌ی SPAها، این گونه است و در توسعه‌ی SPA با هر فریمورکی به این مشکلات برخواهید خورد.

حال که یکسری مشکلات عمومی را بررسی کردیم، بدنیست نگاهی اختصاصی به خود AngularJS بیندازیم.

ضعف طراحی

اگر به تعدای از لینک‌های سایت ihateangular مراجعه کنید می‌بینید که هر کسی نظری دارد: یکی می‌گوید به هیچ وجه Directive ننویسید، یکی دیگر می‌گوید کنترلر ننویسید و تمامی کارها را در directiveهای سفارشی نوشته شده توسط خودتان انجام بدهید، کلا همه جا علیه performance این فریمورک صحبت می‌کنند و همگی به پیچیده بودن آن اذعان دارند.

اما نمی‌شود با چند مقاله‌ی موجود در اینترنت، یک فریمورک با این محبوبیت را زیر سوال برد. اما واقعا فکر می‌کنید که چرا نسخه‌ی 2 انگولار یک بازنویسی کامل است؟ دلیلش واضح است؛ این فریمورک از پایه اشکال دار بوده است و باید از اساس اصلاح شود. پس می‌توان نتیجه گرفت که اشکالات وارد شده به این فریمورک صحیح هستند.

AngularJs 2 یک بازنویسی کامل است

قبلا این موضوع در این نظرسنجی مطرح شده است. بازنویسی کامل یعنی این که خیلی چیزها به کل تغییر کرده‌اند و کدهای قبلی شما با نسخه‌ی جدید سازگار نیستند. بیشتر مطالبی که فراگرفته بودید دیگر کاربردی ندارد و دوباره مطالب جدیدی را باید یاد بگیرید. این را هم در نظر بگیرید که توسعه دهندگانی که در حال نوشتن directive هستند، احتمالا با آمدن نسخه 2 انگولار، مجبورند directive خود را بازنویسی کنند. آیا خودتان بودید، دیگر دل به کار می‌دادید؟!

نتیجه گیری

AngularJS فریمورک خیلی خوبی برای نوشتن برنامه‌های تست پذیر است و کسی منکر قابلیت‌های آن نیست. ولی این را نیز در نظر بگیرید که برای تست پذیر بودن، خیلی چیزها از جمله سادگی کار را از دست می‌دهید. معمولا می‌گویند که AngularJS کارهای مشکل را ساده می‌کند و کارهای ساده را مشکل.

پیشنهاد من این است که اگر هنوز AngularJS را فرا نگرفته‌اید، حداقل یادگیری آن را تا انتشار نسخه‌ی 2 آن به تعویق بیندازید. اگر AngularJS را بلد هستید، دیگر آن را در پروژه‌ای استفاده نکنید؛ چون دیگر کدهای شما در نسخه‌ی 2 کار نخواهد کرد و احتیاج

به انجام تغییرات گسترده‌ای در کدهای نوشته شده قبلی پیدا می‌کنید.

نظرات خوانندگان

نویسنده: مرتضی حاتمی
تاریخ: ۱۳۹۴/۰۶/۲۶ ۲۳:۲۷

سلام
حالا ما که هنوز سراغش نرفتیم به نظر شما سراغ کدوم فریمورک بریم
ایا ReactJS خوبه؟

نویسنده: محمود راستین
تاریخ: ۱۳۹۴/۰۶/۲۷ ۱۲:۳۵

ممنون بابت این مقاله. یکی از دلایل علاقه من به AngularJs استفاده از Ui Router هستش. میشه بپرسم به نظرتون چه فریمورک دیگه ای کاربردی شبیه به این درش وجود داره که بشه ازش استفاده کرد ؟

نویسنده: شاهین کیاست
تاریخ: ۱۳۹۴/۰۶/۲۷ ۱۳:۱

شاید مطالعه ی سری مقالات EmberJs به شما کمک کنه.

نویسنده: علی پناهی
تاریخ: ۱۳۹۴/۰۶/۲۷ ۲۱:۴۲

نظرتون راجع به آنگولار 2 تا اینجای کار چیه؟
تا اینجای کار توانسته مشکلات را برطرف کند؟
توانسته با کتابخانه‌های دیگه ارتباط برقرار بکند؟

نویسنده: عباسی بهزاد
تاریخ: ۱۳۹۴/۰۶/۲۸ ۱۰:۱۱

سلام مهدی عزیز.
با اومدن Es6 انگولار باید عوض می‌شد.
در انگولار1 می‌توان انگولار 2 نوشت و تبدیلش کرد
توی انگولار 2 فقط استفاده از Decoratorها عوض شده که می‌تونید اونو خودتون باز نویسی کنید و ازش استفاده کنید.
اگه هم انگولار نمیخواید کار کنید اینگونه خرابش نکنید لطفا.
اونایی هم که نمی‌خوان انگولار کار کنن به نظر من Aurelia بهترین گزینه می‌تونه باشه.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۶/۲۸ ۱۱:۵۳

بنده قصد خراب کردن این فریمورک را نداشتم. اشکالات بیان شده همگی مربوط به مقالاتی هستند که توسعه دهندگان به اشتراک گذاشتند. من هم آن‌ها را تایید نکردم ولی وقتی می‌بینید که angular 2 همان اشکالات وارد شده به نسخه 1 را برطرف کرده است می‌توان نتیجه گرفت که اشکالات نسخه 1 صحیح بوده اند و باید اصلاح می‌شدند.
بنده هم نگفتم که انگولار را یاد نگیرید، گفتم "حداقل یادگیری آن را تا انتشار نسخه‌ی 2 آن به تعویق بیندازید."
نکته بحث "الان" هست نه گذشته. الان که قرار هست نسخه‌ی 2 منتشر شه و تغییرات زیادی داره چه دلیلی داره پروژه جدیدی را با آن شروع کنیم و بعد شروع به تغییر کدهایمان کنیم.
بله راهنمایی برای مهاجرت از نسخه‌ی 1 به نسخه 2 وجود دارد: AngularJS 1 to Angular 2 Upgrade Strategy
Angular 1 and Angular 2 integration: the path to seamless upgrade

اگر با این روش با آپگرید پروژتون که ممکن است چند هزار خط داشته باشد مشکلی ندارید، معطل نکنید همین امروز پروژه جدیدتون را با انگولار آغاز کنید.
باز هم نمی‌گویم انگولار بد است، همین الان میشود مقاله ای برای مزیت‌های نوشت. فقط قصدم این بود که به هنگام استفاده از انگولار و توسعه spa این موارد را هم در نظر داشته باشید.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۶/۲۸ ۱۲:۲

سلام؛
تجربه‌ی کاری با فریم ورک‌های دیگر را ندارم. ولی در مورد ReactJ.NET جمله "REACT ♥ C# AND ASP.NET MVC " می‌تواند برای توسعه دهندگان ASP.NET MVC جالب باشد. باید این‌ها را در یک پروژه واقعی استفاده کرد تا متوجه مزایا و معایبشان شد.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۶/۲۸ ۱۲:۱۶

اگر Angular 1 and Angular 2 integration قسمت why upgrade را مطالعه کنید می‌بینید که بیشتر مشکلات برطرف شده اند.
حتی Future work مثل Server-side rendering خیلی جذاب و مفید می‌تونه باشه.

نویسنده: محسن کریمی
تاریخ: ۱۳۹۴/۰۶/۲۸ ۲۱:۴۵

سلام
بنده هم بیش از 2 سال است که با Angular کار می‌کنم مشکلاتی که از نظر من بیشتر مهمه و باید اول پروژه قبل از انتخاب، در نظر گرفته شود شامل موارد زیر هستش
وضعیت Angular (و فکر می‌کنم بقیه SPA ها) در کامپوننت‌ها خیلی بد و شما در خیلی موارد انتخاب ندارید و از کامپوننت‌های pure javascript و جی کوئری هم بیشتر مواقع نمیشه استفاده کرد و تنها یک راه اینه که خودتون کامپوننت‌های مورد نیاز خودتونو بنویسید که اینم با توجه به Resource و زمان پروژه خیلی جاها امکان پذیر نیست. کامپوننت هایی که با Angular پیاده شده اند اکثرا یا از لحاظ Performance واقعا افتضاح هستند مثل Kendo یا خیلی ساده هستند که جوابگو نیستن و بازم مجبورید که به سمت پیاده سازی کامپوننت برید.

بحث Performance هم در صورتی که با یک پروژه سنگین روبرو باشید مطمئنا اذیت خواهد کرد و خیلی جاها مجبورید به سمت Pure javascript حرکت کنید و همیشه باید حواستون به watch ها، Bindها و ... باشه(dirty watch) تو بحث Performance هم اگر بخواهید از Patternهای خاصی استفاده کنید (Flux) تا حدودی نسبت به pure javascript دستتون بسته است و باید خیلی چیزها را خودتون طراحی و پیاده سازی کنید.

مشکل سوم هم لود اولیه صفحتونه که می‌تونه مشکل ساز بشه

البته من فکر می‌کنم این موارد در همه SPAها تقریبا موجود هستش و باید سیر تکاملی خودشونو طی کنن

نویسنده: عباسی بهزاد
تاریخ: ۱۳۹۴/۰۶/۲۹ ۱۶:۴

سلام. به jsblocks.com یک نگاه بندازید تا لااقل خیالتان نسبت به performance آسوده باشه :)

نویسنده: علی پناهی

تاریخ: ۱۳۹۴/۰۷/۱۱ ۱۷:۲۳

مسلما واکشی اطلاعات از صفحات وب انکارناپذیر است ولی وقتی که کل وب سایت با WebApi پیاده سازی می‌شود و از فریم ورک‌های تحت کلاینت استفاده شود، افراد دیگر به اطلاعات با ساختار دست پیدا می‌کنند و خیلی راحت‌تر از وب سایت‌های دیگر امکان واکشی اطلاعات وجود دارد.
راه حلی برای این موضوع وجود دارد؟

نویسنده: محسن خان
تاریخ: ۱۳۹۴/۰۷/۱۱ ۱۹:۴۰

«خیلی راحت‌تر از وب سایت‌های دیگر امکان واکشی اطلاعات وجود دارد» : خیر. مگر آنکه CORS - Cross Origin Resource Sharing را فعال کرده باشید.

نویسنده: مهدی سعیدی فر
تاریخ: ۱۳۹۴/۰۷/۱۱ ۱۹:۵۴

اگر منظورتون نمایش اطلاعاتی که به فرمت JSON دریافت می‌شوند، در قالب‌های HTML هست می‌توانید از کتاب خانه هایی به مانند Handlebars استفاده کنید.

عنوان: **قابلیت چند زبانه و Localization در AngularJs- بخش سوم: Best Practiceهای angular-translate**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۶/۳۱ ۱۶:۲۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Localization, AngularJS

در این بخش قصد دارم تا در قالب یک پروژه، تمامی قابلیت‌هایی را که در angular-translate و ماژول‌های مرتبط با آن وجود دارند، به شما معرفی کنم. پروژه‌ی نمونه را از لینک زیر دریافت نمایید: AngularJs-Translate-BestPractices.zip

این پروژه در 12 بخش گوناگون تقسیم بندی شده‌است که هر کدام در قالب یک فایل HTML می‌باشد و تمامی اسکریپت‌های مورد نیاز به آن افزوده شده‌است. هر بخش به صورت مجزا به شرح یک ویژگی کاربردی در angular-translate می‌پردازد.

ex1_basic_usage

روند کار مثال اول خیلی ساده است. در ابتدا اسکریپت‌های زیر به صفحه اضافه شده‌اند:

```
<script src="Scripts/angular.js"></script>
    <script src="Scripts/angular-translate.js"></script>
```

در ابتدا وابستگی pascalprecht.translate به ماژول اضافه شده‌است و پس از آن زبان‌های مختلف به صورت JSON در بخش اسکریپت وارد شده‌اند.

```
angular.module('app', ['pascalprecht.translate'])
        .config([
        '$translateProvider', function ($translateProvider) {

            // Adding a translation table for the English language
            $translateProvider.translations('en_US', {
                "TITLE": "How to use",
                "HEADER": "You can translate texts by using a filter.",
                "SUBHEADER": "And if you don't like filters, you can use a directive.",
                "HTML_KEYS": "If you don't like an empty elements, you can write a key for the
translation as an inner HTML of the directive.",
                "DATA_TO_FILTER": "Your translations might also contain any static ({{staticValue}}) or
random ({{randomValue}}) values, which are taken directly from the model.",
                "DATA_TO_DIRECTIVE": "And it's no matter if you use filter or directive: static is
still {{staticValue}} and random is still {{randomValue}}.",
                "RAW_TO_FILTER": "In case you want to pass a {{type}} data to the filter, you have only
to pass it as a filter parameter.",
                "RAW_TO_DIRECTIVE": "This trick also works for {{type}} with a small mods.",
                "SERVICE": "Of course, you can translate your strings directly in the js code by using
a $translate service.",
                "SERVICE_PARAMS": "And you are still able to pass params to the texts. Static =
{{staticValue}}, random = {{randomValue}}."
            });

            // Adding a translation table for the Russian language
            $translateProvider.translations('ru_RU', {
                "TITLE": "Как пользоваться",
                "HEADER": "Вы можете переводить тексты при помощи фильтра.",
                "SUBHEADER": "А если Вам не нравятся фильтры, Вы можете воспользоваться директивой.",
                "HTML_KEYS": "Если вам не нравятся пустые элементы, Вы можете записать ключ для
перевода в как внутренний HTML директивы.",
                "DATA_TO_FILTER": "Ваши переводы также могут содержать любые статичные
({{staticValue}}) или случайные ({{randomValue}}) значения, которые берутся прямо из модели.",
                "DATA_TO_DIRECTIVE": "И совершенно не важно используете ли Вы фильтр или директиву:
статическое значение по прежнему {{staticValue}} и случайное - {{randomValue}}.",
                "RAW_TO_FILTER": "Если вы хотите передать \"сырые\" ({{type}}) данные фильтру, Вам
всего лишь нужно передать их фильтру в качестве параметров.",
                "RAW_TO_DIRECTIVE": "Это также работает и для директив ({{type}}) с небольшими
модификациями.",
                "SERVICE": "Конечно, Вы можете переводить ваши строки прямо в js коде при помощи
сервиса $translate.",
                "SERVICE_PARAMS": "И вы все еще можете передавать параметры в тексты. Статическое
значение = {{staticValue}}, случайное = {{randomValue}}."
            });

            // Tell the module what language to use by default
            $translateProvider.preferredLanguage('en_US');
```

```
        }])
```

در تکه کد فوق مشاهده می‌کنید که دو translate table زبان انگلیسی و روسی به صورت JSON وارد شده‌اند. شما قادرید تا چندین زبان را به همین صورت وارد نمایید. در خط آخر نیز زبان پیش فرض سیستم تعریف شده است.
حال به بررسی کدهای درون کنترلر می‌پردازیم:

```
.controller('ctrl', ['$scope', '$translate', function ($scope, $translate) {

        $scope.tlData = {
            staticValue: 42,
            randomValue: Math.floor(Math.random() * 1000)
        };

        $scope.jsTrSimple = $translate.instant('SERVICE');
        $scope.jsTrParams = $translate.instant('SERVICE_PARAMS', $scope.tlData);

        $scope.setLang = function (langKey) {
            // You can change the language during runtime
            $translate.use(langKey);

            // A data generated by the script have to be regenerated
            $scope.jsTrSimple = $translate.instant('SERVICE');
            $scope.jsTrParams = $translate.instant('SERVICE_PARAMS', $scope.tlData);
        };

    }]);
```

می‌بینیم که وابستگی $translate تزریق شده است. پس از آن دو عدد رندم به پارامترهای تعریف شده در translate table ارسال می‌گردد. تغییر زبان نیز توسط متد setLang صورت می‌پذیرد.
در بخش نهایی می‌خواهیم روش‌های گوناگون استفاده از translate tables را درون HTML نمایش دهیم. به بخش HTML همین مثال توجه کنید:

```
<p>
        <a href="#" ng-click="setLang('en_US')">English</a>
        |
        <a href="#" ng-click="setLang('ru_RU')">Русский</a>
    </p>
    <!-- Translation by a filter -->
    <h1>{{'HEADER' | translate}}</h1>
    <!-- Translation by a directive -->
    <h2 translate="SUBHEADER">Subheader</h2>
    <!-- Using inner HTML as a key for translation -->
    <p translate>HTML_KEYS</p>
    <hr>
    <!-- Passing a data object to the translation by the filter -->
    <p>{{'DATA_TO_FILTER' | translate: tlData}}</p>
    <!-- Passing a data object to the translation by the directive -->
    <p translate="DATA_TO_DIRECTIVE" translate-values="{{tlData}}"></p>
    <hr>
    <!-- Passing a raw data to the filter -->
    <p>{{'RAW_TO_FILTER' | translate:'{ type: "raw" }' }}</p>
    <!-- Passing a raw data to the filter -->
    <p translate="RAW_TO_DIRECTIVE" translate-values="{ type: 'directives' }"></p>
    <hr>
    <!-- Using a $translate service -->
    <p>{{jsTrSimple}}</p>
    <!-- Passing a data to the $translate service -->
    <p>{{jsTrParams}}</p>
```

نحوه تعریف هر روش به صورت کامنت پیش از هر تگ نوشته شده است. شما به روش‌های مختلف و بر حسب استانداردهایی که خود از آن پیروی می‌کنید می‌توانید از یکی از روش‌های فوق استفاده نمایید. اما رایج‌ترین روش، دو روش اول یعنی استفاده از دایرکتیو و یا فیلتر است و دلیل آن هم سادگی و خوانا بودن این دو روش می‌باشد.

ex2_remember_language_cookies

در این مثال همانگونه که از اسم آن پیداست قصد داریم تا زبان مورد نظری را که کاربر در سیستم انتخاب نموده است، ذخیره

کنیم تا پس از بستن و بازکردن مجدد وب سایت با همان زبان پیشین به کاربر نمایش داده شود. این کار بسیار ساده است. کافیست که در ابتدا علاوه بر اسکریپت‌های مثال قبل، اسکریپت‌های زیر را نیز به صفحه اضافه کنید:

```
<script src="Scripts/angular-cookies.js"></script>
    <script src="Scripts/angular-translate-storage-cookie.js"></script>
```

با اضافه کردن خط زیر درون بدنه config، یک کوکی جدید برای شما ساخته می‌شود. این کوکی NG_TRANSLATE_LANG_KEY نام دارد که هر بار با id زبان کنونی که در translate table وارد نموده‌اید آپدیت می‌شود.

```
// Tell the module to store the language in the cookie
$translateProvider.useCookieStorage();
```

حال اگر صفحه را refresh کنید می‌بینید که زبان پیشینی که انتخاب نموده‌اید، مجددا بارگذاری می‌گردد.

ex3_remember_language_local_storage

این مثال همانند مثال قبل رفتار می‌کند، با این تفاوت که به جای اینکه کلید زبان کنونی را درون کوکی ذخیره کند، آن را درون Local Storage با نام NG_TRANSLATE_LANG_KEY قرار می‌دهد. برای اجرا کافیست اسکریپت‌ها و تکه کد زیر را با موارد مثال قبل جایگزین کنید.

```
<script src="Scripts/angular-translate-storage-local.js"></script>


// Tell the module to store the language in the local storage
$translateProvider.useLocalStorage();
```

مثال های ex4_set_a_storage_key و ex5_set_a_storage_prefix نام کلیدی که برای ذخیره سازی زبان کنونی در کوکی یا Local Storage قرار می‌گیرد را تغییر می‌دهد که به دلیل سادگی از شرح آن می‌گذریم.

ex6_namespace_support

translate table در angular-translate قابلیت مفید namespacing را نیز داراست. این قابلیت به ما کمک می‌کند که جهت کپسوله کردن بخش‌های مختلف، ترجمه آنها را با namespaceهای خاص خود نمایش دهیم. به مثال زیر توجه کنید:

```
$translateProvider.translations('en_US', {
                "TITLE": "How to use namespaces",
                "ns1": {
                    "HEADER": "A translations table supports namespaces.",
                    "SUBHEADER": "So you can to structurize your translation table well."
                },
                "ns2": {
                    "HEADER": "Do you want to have a structured translations table?",
                    "SUBHEADER": "You can to use namespaces now."
                }
            });
```

همانطور که توجه می‌کنید بخش ns1 خود شامل زیر مجموعه‌هایی است و ns2 نیز به همین صورت. هر کدام دارای کلید HEADER و SUBHEADER می‌باشند. فرض کنید هر کدام از این بخش‌ها می‌خواهند اطلاعات درون یک section را نمایش دهند. حال به نحوه‌ی فراخوانی این translate tableها دقت کنید:

```
<!-- section 1: Translate Table Called by ns1 namespace -->
<h1 translate>ns1.HEADER</h1>
<h2 translate>ns1.SUBHEADER</h2>

<!-- section 2: Translate Table Called by ns2 namespace -->
<h1 translate>ns2.HEADER</h1>
<h2 translate>ns2.SUBHEADER</h2>
```

به همین سادگی می‌توان تمامی بخش‌ها را با namespaceهای مختلف در translate table قرار داد.

در بخش بعدی (پایانی) شش قابلیت دیگر angular translate که شامل فراخوانی translate table از یک فایل JSON، فراخوانی فایل‌های translate table به صورت lazy load و تغییر زبان بخشی از صفحه به صورت پویا هستند، بررسی خواهند شد.

فایل پروژه: AngularJs-Translate-BestPractices.zip

عنوان: قابلیت چند زبانه و Localization در AngularJs- بخش چهارم و نهایی: Best Practiceهای angular-translate
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۷/۱۱ ۱۱:۲۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, Localization, AngularJS

در بخش پیشین چند مورد از قابلیت‌های angular-translate را بررسی نمودیم. در این بخش به بررسی باقی موارد می‌پردازیم.

ex7_load_static_files

در این مثال خواهیم دید که چگونه یک فایل translate table در موقع فراخوانی به صورت On Demand بارگذاری خواهد شد. در قدم اول اسکریپت‌های زیر به صفحه افزوده می‌شوند.

```
<script src="Scripts/angular.js"></script>
    <script src="Scripts/angular-cookies.js"></script>
    <script src="Scripts/angular-translate.js"></script>
    <script src="Scripts/angular-translate-storage-cookie.js"></script>

    <!-- for override loader methods in angular translate -->
    <script src="/src/service/loader-static-files.js"></script>
```

در ادامه درباره‌ی اسکریپت پنجم بیشتر توضیح خواهیم داد. بگذارید از آخر به اول شروع کنیم و ببینیم که نحوه‌ی فراخوانی و استفاده از امکان on demand بارگذاری شدن فایل‌های زبان به چه صورتی می‌باشد. در زیر، تکه کد اضافه شده به ex7 را مشاهده می‌کنید:

```
// Register a loader for the static files
            // So, the module will search missing translation tables under the specified urls.
            // Those urls are [prefix][langKey][suffix].
            $translateProvider.useStaticFilesLoader({
                prefix: '/l10n/',
                suffix: '.json'
            });
```

همانطور که در توضیحات آمده است، ماژول با دریافت prefix و suffix که در حقیقت همان فولدر و پسوند فایل‌های translate table هستند، هر زبانی را که مورد نیاز است و تا کنون بارگذاری نشده، بارگذاری می‌نماید. تصویر زیر محتویات فولدر l10n را نمایش می‌دهد.

حال ببینیم که این فرآیند در loader-static-files چگونه پیاده سازی شده است. در این فایل یک متد load نوشته شده است که فایل‌های static را طبق یک الگوی مشتمل بر prefix و suffix از سرور می‌خواند. لزومی ندارد که شما فایل‌ها را حتما به صورت JSON و با این پسوند ذخیره کنید. اما چیزی که قطعی است این است که فایل‌ها حتما باید به صورت key value ذخیره شده باشند.

تکه کد زیر اطلاعات فایل loader-static-files را نمایش می‌دهد.

```
angular.module('pascalprecht.translate')
.factory('$translateStaticFilesLoader', $translateStaticFilesLoader);
function $translateStaticFilesLoader($q, $http) {

  'use strict';

  return function (options) {

    if (!options || (!angular.isArray(options.files) && (!angular.isString(options.prefix) ||
!angular.isString(options.suffix)))) {
      throw new Error('Couldn\'t load static files, no files and prefix or suffix specified!');
    }

    if (!options.files) {
      options.files = [{
        prefix: options.prefix,
        suffix: options.suffix
      }];
    }

    var load = function (file) {
      if (!file || (!angular.isString(file.prefix) || !angular.isString(file.suffix))) {
        throw new Error('Couldn\'t load static file, no prefix or suffix specified!');
      }

      var deferred = $q.defer();

      $http(angular.extend({
        url: [
          file.prefix,
          options.key,
          file.suffix
        ].join(''),
        method: 'GET',
        params: ''
      }, options.$http)).success(function (data) {
        deferred.resolve(data);
      }).error(function () {
        deferred.reject(options.key);
      });

      return deferred.promise;
    };

    var deferred = $q.defer(),
        promises = [],
        length = options.files.length;

    for (var i = 0; i < length; i++) {
      promises.push(load({
        prefix: options.files[i].prefix,
        key: options.key,
        suffix: options.files[i].suffix
      }));
    }

    $q.all(promises).then(function (data) {
      var length = data.length,
          mergedData = {};

      for (var i = 0; i < length; i++) {
        for (var key in data[i]) {
          mergedData[key] = data[i][key];
        }
      }

      deferred.resolve(mergedData);
    }, function (data) {
      deferred.reject(data);
    });

    return deferred.promise;
  };
}

$translateStaticFilesLoader.displayName = '$translateStaticFilesLoader';
```

همانطور که ملاحظه می‌کنید، کد فوق یک سرویس با نام $translateStaticFilesLoader را تعریف نموده است. در صورتیکه ما

در کنترلر فایل ex7، اصلا نامی از آن نبردیم و تنها از $translateProvider.useStaticFilesLoader استفاده نمودیم! جواب در نحوه‌ی نگارش کد angular-translate نهفته است. در خط 866 فایل angular-translate تکه کد زیر مربوط به تعریف translateStaticFileLoader می‌باشد. همانطور که ملاحظه می‌کنید سرویس translateStaticFilesLoader درون فضای نام سرویس translateTable قرار گرفته است. بنابراین ما تنها با تعریف سرویس translateStaticFilesLoader، در حقیقت آن را override نموده‌ایم. در کد نمونه‌ای که در بخش‌های قبلی قرار داده‌ام یک فایل translate.js نیز قرار دارد که در فولدر src/services قرار گرفته است. این فایل نیز برخی از امکانات و سرویس‌های built-in درون angular-translate را سفارشی نموده است.

```
/**
   * @ngdoc function
   * @name pascalprecht.translate.$translateProvider#useStaticFilesLoader
   * @methodOf pascalprecht.translate.$translateProvider
   *
   * @description
   * Tells angular-translate to use `$translateStaticFilesLoader` extension service as loader.
   *
   * @param {Object=} options Optional configuration object
   */
  this.useStaticFilesLoader = function (options) {
    return this.useLoader('$translateStaticFilesLoader', options);
  };
```

در این 4 مجموعه سعی کردم تمامی آنچه را که برای ایجاد قابلیت چند زبانه و localization نیاز است و حیاتی بود، تشریح کنم. بنابراین تا کنون دانش خوبی درباره‌ی این کتابخانه کسب نموده‌اید. باقی تمرین‌ها را می‌توانید بر حسب نیاز با استفاده از مستندات موجود در angular-translate مطالعه و استفاده نمایید.

عنوان: **توسعه سرویس‌های مبتنی بر REST در AngularJs با استفاده از RestAngular : بخش اول**
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۷/۱۵ ۱۷:۲۵
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, AngularJS

برای مطالعه‌ی این مقاله شما باید به مواردی از قبیل کتابخانه‌ی AngularJs ، تعاملات بین کلاینت و سرور و همچنین معماری RESTful تسلط کافی داشته باشید و ما از توضیح و تفصیلی این سرفصل‌ها اجتناب میکنیم.
خیلی خوب بپردازیم به اصل مطلب:

Restangular چیست؟

کتابخانه RestAngular بنا به گفته ناشر در مستندات Github آن، یک سرویس توسعه داده شده AngularJs می‌باشد که کدهای نوشته شده‌ی برای پیاده سازی فرآیندهای Request/Response کلاینت و سرور را به حداقل رسانده است. این فرآیندها در قالب درخواست‌های GET، POST، DELETE و Update صورت می‌پذیرند. این سرویس برای وب اپلیکیشن‌هایی که که داده‌های خود را از APIهای RESTful همانند Microsoft ASP.NET Web API 2 دریافت می‌کنند نیز بسیار کارآمد است.
کد زیر یک فرآیند درخواست GET را نمایش می‌دهد که در آن از سرویس RestAngular استفاده شده است:

```
// Restangular returns promises
Restangular.all('users').getList()  // GET: /users
.then(function(users) {
  // returns a list of users
  $scope.user = users[0]; // first Restangular obj in list: { id: 123 }
})

// Later in the code...

// Restangular objects are self-aware and know how to make their own RESTful requests
$scope.user.getList('cars');  // GET: /users/123/cars

// You can also use your own custom methods on Restangular objects
$scope.user.sendMessage();  // POST: /users/123/sendMessage

// Chain methods together to easily build complex requests
$scope.user.one('messages', 123).one('from', 123).getList('unread');
// GET: /users/123/messages/123/from/123/unread
```

همانطور که ملاحظه میکنید تمام پروسه‌ی دریافت، در یک خط خلاصه شده است. همچنین می‌بینید که restAngular دارای Promise نیز هست. در تکه کد اول، تمامی userها به صورت Get دریافت می‌شوند. در تکه کد دوم مشاهده می‌کنید که عملیات درخواست لیست ماشین‌های کاربر چگونه در یک خط خلاصه می‌گردد. حال فرض کنید قصد داشتیم تا این عملیات را با وابستگی http پیاده سازی نماییم. برای این کار باید چندین خط کد را درون یک سرویس جا می‌دادیم و آنگاه از متد سرویس در کنترلر استفاده می‌کردیم.
در اینجا همچنین قادرید درخواست‌های خود را به صورت سفارشی نیز اعمال کنید (تکه کد سوم) و در آخر مشاهده می‌کنید که پیچیده‌ترین عملیات‌های درخواست کاربر را می‌توان در یک خط خلاصه نمود. برای نمونه کد آخر یک دستور GET تو در تو را به چه سادگی پیاده سازی کرده است.

وابستگی ها

باید توجه داشته باشید که برای استفاده از RestAngular نیاز به کتابخانه‌ی Lodash نیز می‌باشد.

شروع پروژه

برای شروع پروژه و استفاده از RestAngular، پس از اضافه کردن اسکریپت رفرنس‌ها و وابستگی‌ها (lodash و AngularJs) باید وابستگی restAngular را به صورت زیر به angular.module اضافه نمایید:

```
// Add Restangular as a dependency to your app
angular.module('your-app', ['restangular']);

// Inject Restangular into your controller
angular.module('your-app').controller('MainCtrl', function($scope, Restangular) {
  // ...
```

```
});
```

توجه داشته باشید هنگامیکه یک وابستگی به module اضافه می‌گردد، با حروف کوچک یعنی restangular اضافه می‌گردد؛ اما هنگامیکه به کنترلر اضافه می‌شود، به صورت Restangular و با R بزرگ اضافه می‌گردد.

برخی از متدهای RestAngular

در این بخش قصد داریم تا برخی از پر کاربردترین متدهای RestAngular را تشریح کنیم:

نام متد	پارامترهای ارسالی	توضیحات
one	route, id	این متد یک RestAngular object ایجاد میکند که از آدرسی که در route قرار داده شده با id مشخص دریافت میشود.
all	route	این متد یک RestAngular object که لیستی از المنت هایی را که در آدرس route قرار دارد، دریافت می‌نماید.
oneUrl	route, url	این متد یک RestAngular object ایجاد می‌کند که یک المنت از url خاصی را بازگشت میدهد. مانند: Restangular.oneUrl(' googlers ' , ' http://www.google.com/1 ').get();'
allUrl	route, url	این متد مانند متد قبل است با این تفاوت که یک مجموعه را بازگشت میدهد.
copy	formElement	این متد یک کپی از المنت‌های یک فرم را ایجاد میکند که ما میتوانیم آنها را تغییر دهیم.
restangularizeElement	parent,element, route, queryParams	یک المنت جدید را به صورت Restangularize تغییر میدهد.
restangularizeCollection	parent, element, route, queryParams	یک کالکشن Collection را به صورت Restangularize تغییر میدهد.

در این قسمت قصد داشتیم تا شما را با این کتابخانه کمی آشنا کنیم. در قسمت بعدی سعی می‌کنیم تا با مثال‌هایی کاربردی، شما را بیشتر با این سرویس خارق العاده آشنا کنیم.

عنوان: توسعه سرویس‌های مبتنی بر REST در AngularJs با استفاده از RestAngular : بخش دوم
نویسنده: مهرداد کاهه
تاریخ: ۱۳۹۴/۰۷/۱۸ ۲۱:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScript, ASP.NET Web API, AngularJS, json.net

در بخش پیشین کلیات کتابخانه‌ی Restangular را بررسی کردیم. در این بخش قصد داریم تا در طی یک پروژه، امکانات و قابلیت‌های بی‌نظیر این سرویس را در یک پروژه‌ی واقعی مشاهده کنیم.

کلیات پروژه

در این پروژه قصد داریم تا لیست کتاب‌های یک کتابخانه را نمایش دهیم. این کتابها قابلیت ویرایش نام دارند و همچنین شما می‌توانید کتابهای جدیدی را به لیست کتابها اضافه نمایید. تصویر زیر خروجی این پروژه است:

پایگاه داده‌ی برنامه با نام LibDb درون پوشه‌ی app_data قرار داده شده‌است. این پایگاه داده تنها دارای یک جدول است؛ با نام Books که دیاگرام آن را در شکل زیر مشاهده می‌کنید:

پیاده سازی

در ابتدا یک پروژه‌ی Empty را با رفرنس web API ایجاد می‌کنیم. حال درون فایل WebApiConfig.cs تکه کد زیر را اضافه می‌کنیم. این تکه کد فیلدها و آبجکت‌هایی را که از سمت سرور بازگشت داده می‌شوند، به صورت camelCase تبدیل می‌کند.

```
var jsonFormatter = config.Formatters.OfType<JsonMediaTypeFormatter>().First();
```

```
jsonFormatter.SerializerSettings.ContractResolver = new CamelCasePropertyNamesContractResolver();
```

در ادامه مدل edmx را ساخته و پس از آن Books Web Api را درون پوشه‌ی کنترلر ایجاد می‌کنیم. این کنترلر به صورت Default web Api Scaffold ساخته شده است و به دلیل اینکه به بحث ما مرتبط نیست؛ از توضیح بیشتر آن می‌گذریم. حال پوشه‌ای را با نام app برای نگه داری فایل‌های AngularJs اضافه می‌کنیم و درون آن فایل app.js را ایجاد می‌کنیم:

```
var angularExample = angular.module('angularexample', ["restangular"])
angularExample.config(["RestangularProvider", function (RestangularProvider) {
    //RestangularProvider.setRestangularFields({
    //    id: "id"
    //});
    RestangularProvider.setBaseUrl('/api');
}]);

angularExample.controller("MainCtrl", ["Restangular", "$scope", function (Restangular, $scope) {
    var resource = Restangular.all('books');
    resource.getList().then(function (response) {
        $scope.books = response;
    });
    $scope.add = function () {
        resource.post($scope.newBook).then(function (newResource) {
            $scope.books.push(newResource);
        })
    }
}]);
```

محتویات فایل app.js را مشاهده می‌کنید. در ادامه درباره‌ی این قسمت بیشتر صحبت می‌کنیم.

حال در روت پروژه فایل index.html را ایجاد می‌کنیم:

```
<!DOCTYPE html>
<html ng-app="angularexample" xmlns="http://www.w3.org/1999/xhtml">
<head>
    <title>Rest Angular Sample</title>
    <script src="//ajax.googleapis.com/ajax/libs/angularjs/1.1.5/angular.min.js"></script>
    <script src="http://cdn.jsdelivr.net/underscorejs/1.5.1/underscore-min.js"></script>
    <script src="/Scripts/restangular.min.js"></script>
    <script src="/app/app.js"></script>
</head>
<body>
    <div ng-controller="MainCtrl">
        <div ng-repeat="item in books">
            name is: {{item.name}}<br />
            change name: <input type="text" ng-model="item.name" /><button type="submit" ng-
click="item.put();">update</button>
        </div>
        <div>
            add new: <br />
            <input type="text" ng-model="newBook.name" /><button type="submit" ng-
click="add()">add</button>
        </div>
    </div>
</body>
```

و در نهایت در پوشه‌ی Scripts فایل‌های سرویس Restangular را قرار می‌دهیم.

تا به اینجای کار تمامی کارهای مورد نیاز تشکیل پروژه را انجام داده‌ایم. حال به بررسی بیشتر سرویس‌های این پروژه می‌پردازیم؛ یعنی کدهای درون فایل app.js.

ببینیم که برای دریافت لیست تمامی کتابها، ما چه کارهایی را انجام داده‌ایم! به تکه کد زیر دقت کنید:

```
var resource = Restangular.all('books');
    resource.getList().then(function (response) {
        $scope.books = response;
    });
```

این تمامی آن چیزی است که ما برای دریافت لیست تمامی کتابها انجام داده‌ایم. به همین سادگی! در خط اول از متد all که در

بخش قبل توضیح مختصری راجع به آن داده بودم برای دریافت لیست تمامی کتابها استفاده کردم که پارامتر درون آن آدرس Web Api است. البته همانطور که می‌دانید در ASP.NET Web Api ما همیشه به base address یک api نیز اضافه می‌کنیم. حال اینکه چرا api در اینجا نیامده در ادامه و در بخش تنظیمات کلی Restangular توضیح می‌دهم. در خط دوم نیز از اشاره‌گر resources متد getList را فراخوانی کرده و لیست کتابها را در response دریافت می‌کنیم.

پس از آن می‌خواهیم ببینیم که عملیات ایجاد یک کتاب جدید چگونه انجام می‌گردد. تکه کد زیر این عملیات را انجام می‌دهد:

```
$scope.add = function () {
        resource.post($scope.newBook).then(function (newResource) {
            $scope.books.push(newResource);
        })
    }
```

ما یک متد با نام add ایجاد کرده‌ایم که در سمت View توسط دایرکتیو ng-click آن را فراخوانی می‌کنیم.

همانطور که مشاهده میکنید درون app.js متدی برای update موارد قبلی نیست. بیایید سری به View بزنیم. به المنت div زیر توجه کنید. همانطور که می‌بینید تمامی عملیات update با یک دستور ساده item.put حل و فصل شده.

```
<div ng-repeat="item in books">
            name is: {{item.name}}<br />
            change name: <input type="text" ng-model="item.name" /><button type="submit" ng-
click="item.put();">update</button>
        </div>
```

تمامی آنچه که گفته شد تنها بخشی از قابلیت‌های شگفت انگیز این افزونه بود. امیدوارم که مطلب مفید واقع شده باشد. سورس این پروژه را می‌توانید از لینک زیر دریافت کنید.

سورس پروژه: RestangularSample.rar