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*