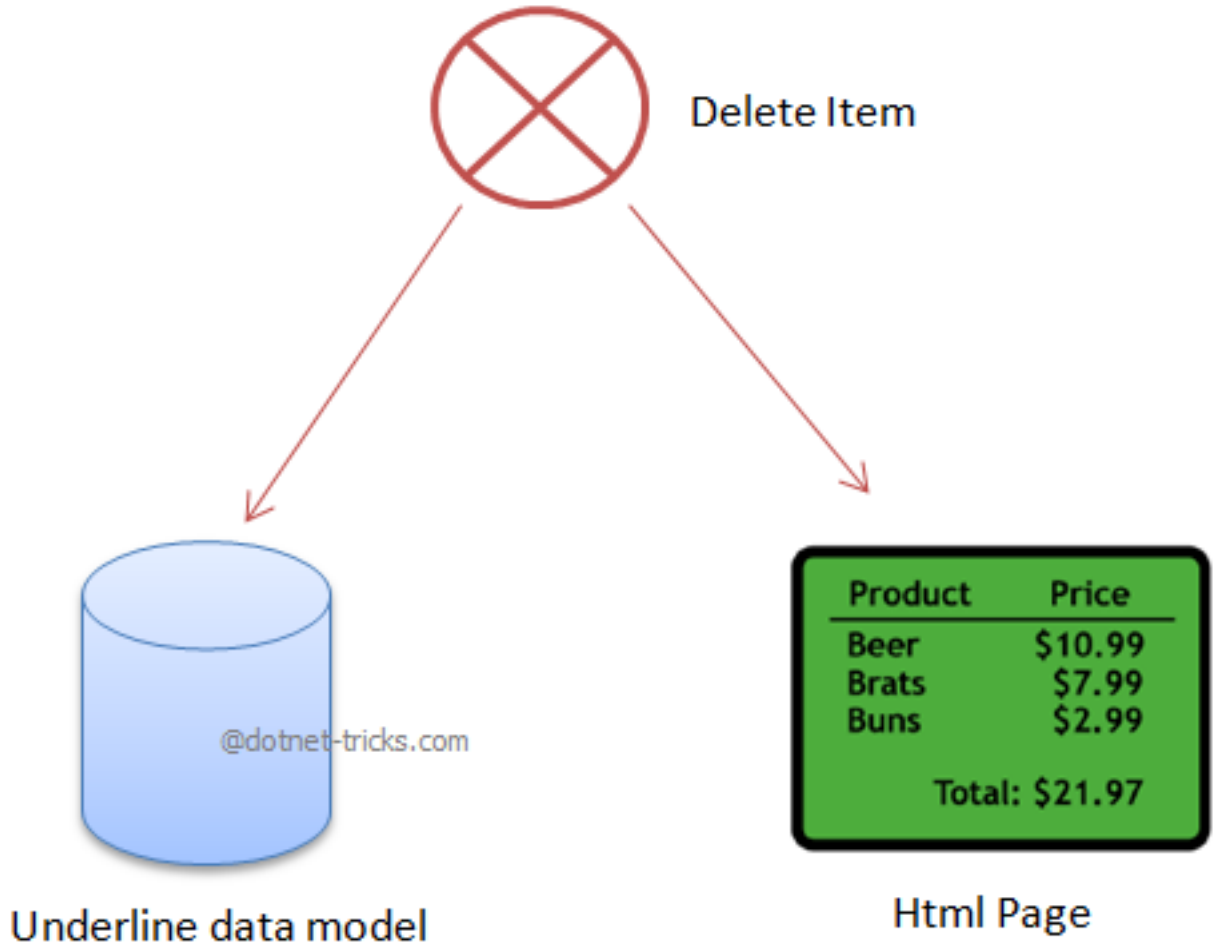
عنوان: **آموزش Knockout.Js #1**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۰۵ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, KnockoutJS, JavaScriptFrameWork

اگر از برنامه نویس‌های پروژه‌های WPF درباره ویژگی‌های مهم الگوی MVVM بپرسید به احتمال زیاد اولین مطلبی که عنوان می‌شود این است که هنگام کار با الگوی MVVM در WPF باید از مباحث data-binding استفاده شود. به صورت خلاصه، data-binding مکانیزمی است که عناصر موجود در Xaml را به آبجکت‌های موجود در ViewModel یا سایر عناصر Xaml مقید می‌کند به طوری که با تغییر مقدار در آبجکت‌های ViewModel، عناصر View نیز خود را به روز می‌کنند یا با تغییر در مقادیر عناصر Xaml، آبجکت‌های متناظر در ViewModel نیز تغییر خواهند کرد(در صورت تنظیم Mode = TwoWay).

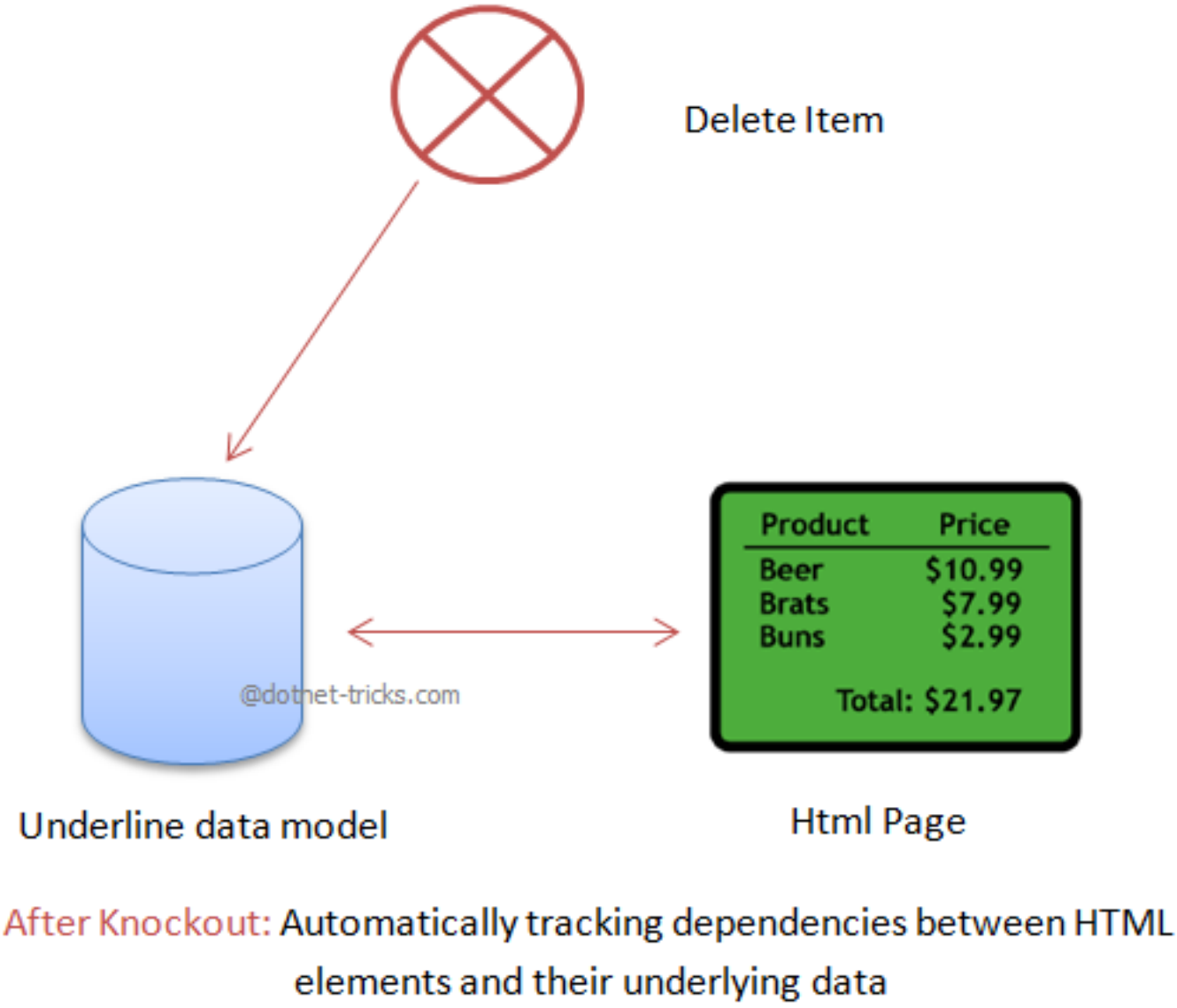
Knockout.Js چیست؟

در یک جمله Knockout.Js یک فریم ورک جاوا اسکریپ است که امکان پیاده سازی الگوی MVVM و مکانیزم data-binding را در پروژه‌های تحت وب به راحتی میسر می‌کند. به عبارت دیگر عناصر DOM را به data-model و آبجکت‌های data-model را به عناصر DOM مقید می‌کند، به طوری که با هر تغییر در مقدار یا وضعیت این عناصر یا آبجکت ها، تغییرات به موارد مقید شده نیز اعمال می‌گردد. به تصاویر زیر دقت کنید!

به روز رسانی data-model **بدون** استفاده از KO



به روز رسانی data-model با استفاده از KO



ویژگی‌های مهم KO

»ارائه یک راه حل بسیار ساده و واضح برای اتصال بخش‌های مختلف UI به data-model

»به روز رسانی خودکار عناصر و بخش‌های مختلف UI بر اساس تغییرات صورت گرفته در data-model

»به صورت کامل با کتابخانه و توابع javascript پیاده سازی شده است.

»حجم بسیار کم(سیزده کیلو بایت) بعد از فشرده سازی

»سازگار با تمام مروگرهای جدید(... ,IE 6+, Firefox 2+, Chrome, Safari)

»امکان استفاده راحت بدون اعمال تغییرات اساسی در معماری پروژه هایی که در فاز توسعه هستند و بخشی از مسیر توسعه را طی کرده اند

»و...

آیا KO برای تکمیل JQuery در نظر گرفته شده است یا جایگزین؟

در اینکه JQuery بسیار محبوب است و در اکثر پروژه‌های تحت وب مورد استفاده است شکی وجود ندارد ولی این بدان معنی

نیست که با توجه به وجود JQuery و محبوبیت آن دیگر نیازی به KO احساس نمی‌شود. به عنوان یک مثال ساده : فرض کنید در یک قسمت از پروژه قصد داریم یک لیست از داده‌ها را نمایش دهیم. در پایین لیست تعداد آیتم‌های موجود در لیست مورد نظر نمایش داده میشود. یک دکمه Add داریم که امکان اضافه شدن آیتم جدید را در اختیار ما قرار می‌دهد. بعد از اضافه شدن یک مقدار، باید عددی که تعداد آیتم‌های لیست را نمایش می‌دهد به روز کنیم. خب اگر قصد داشته باشیم این کار را با JQuery انجام دهیم راه حل‌های زیر پیش رو است :

» به دست آوردن تعداد trهای جدول موجود؛

»به دست آوردن تعداد divهای موجود با استفاده از یک کلاس مشخص css؛

» یا حتی به دست آوردن تعداد آیتم‌های نمایشی در span هایی مشخص.

و البته سایر راه حل ها...

حال فرض کنید دکمه‌های دیگر نظیر Delete نیز مد نظر باشد که مراحل بالا تکرار خواهند شد. اما با استفاده از KO به راحتی می‌توانیم تعداد آیتم‌های موجود در یک آرایه را به یک عنصر مشخص bind کنیم به طور با هر تغییر در این مقدار، عنصر مورد نظر نیز به روز می‌شود یا به بیانی دیگر همواره تغییرات observe خواهند شد. برای مثال:

```
Number of items :<span data-bind="text: myList().count"></span>
```

در نتیجه برای کار با KO وابستگی مستقیم به استفاده از JQeury وجود ندارد ولی این امکان هست که بتوانیم هم از JQuery و هم از KO در کنار هم به راحتی استفاده کنیم و از قدرت‌های هر دو فریم ورک بهره ببریم و البته KO جایگزینی برای JQuery نخواهد بود.

در پست بعد، شروع به کار با KO آموزش داده خواهد شد.

ادامه دارد...

نظرات خوانندگان

نویسنده: ابوالفضل رجب پور
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۳:۱۴

سلام و تشکر از آموزش خوبتون
KNOCKOUT در مقایسه با angular ، کدام مناسب‌تر هستند؟
آیا مقایسه این دو درست است؟
شنیدم روی ویژوال استودیو 2013 مایکروسافت پیش فرض آنگولار رو استفاده کرده. این خودش خیلی نقطه قوت هست و حتما روش فکر کرده مایکروسافت!
نظر شما چیه؟
به طور کل برای spa چه مجموعه فریم ورکی رو پیشنهاد میدید؟
مثلا ترکیب jquery + angular +requirejs چطوره؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۰۷ ۱۳:۴۲

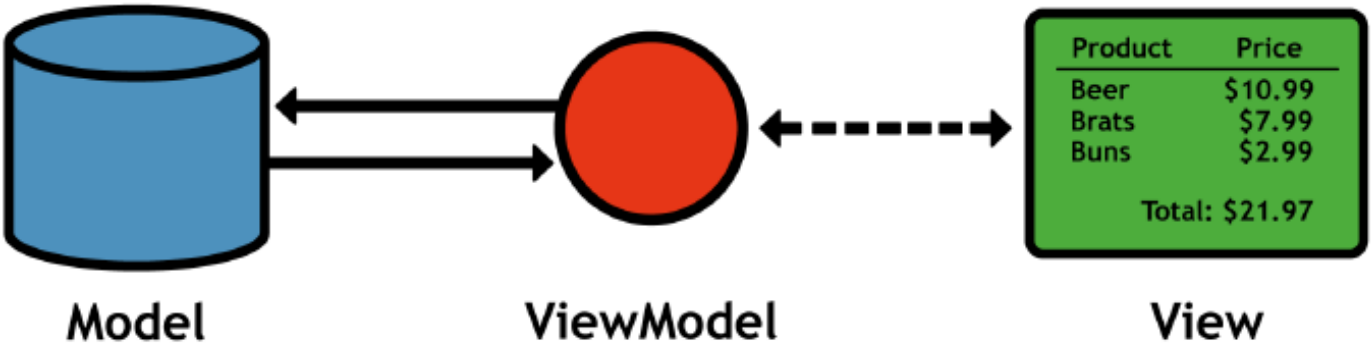
در این مطلب فقط بوت استرپ نگارش 2 در MVC 5 پیش فرض شده. قبلا ناک آوت در MVC4 بود جزو اسکریپت‌های پیش فرض.

نویسنده: آرمان فرقانی
تاریخ: ۱۳۹۲/۰۷/۰۶ ۱۴:۱۷

این سوال خوبی است. اما گمان نمی‌کنم بشود پاسخ دقیقی به بخش کدام مناسب‌تر است به طور کلی داد. شاید بتوانید بر اساس علاقه به MVC یا MVVM یکی را برگزینید. برای کسانی هم که می‌خواهند یکی را شروع کنند شاید Knockout برای شروع با توجه به داکیومنت و بخش آموزش جالب آن بهتر باشد. همچنین مقایسه هایی مانند این یا بحث هایی مانند این کمک کننده است برای انتخاب بین این فریم ورک ها. البته هر دو فریم ورک مدرن و مناسب برای بسیاری موارد هستند. نظر شخصی من این است اگر ASP.NET MVC کار می‌کنید Angular را به صورت راه حل کلی دنبال کنید چون کمی کسب مهارت و آشنایی با تمام مفاهیم آن نسبت به Knockout بیشتر طول می‌کشد. و زمانی که صرف یافتن گزینه بهتر بین این دو می‌کنید را برای مطالعه Knockout با استفاده از مقالات همین سایت یا بخش آموزش سایت رسمی آن اختصاص دهید. گمان نمی‌کنم از صرف وقت برای این دو پشیمان شوید. هر کدام شیرینی خاص خود را دارند.

عنوان: **آموزش Knockout.Js #2**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۰۶ ۸:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: JavaScript, KnockoutJS, JavaScriptFrameWork

در پست قبلی با مفاهیم و ویژگی‌های کلی KO آشنا شدید. KO از الگوی طراحی MVVM استفاده می‌کند. از آن جا که یکی از پیش نیازهای KO آشنایی اولیه با مفاهیم View و Model است نیاز به توضیح در این موارد نیست اما اگر به هر دلیلی با این مفاهیم آشنایی ندارید می‌توانید از اینجا شروع کنید. اما درباره ViewModel که کمی مفهوم متفاوتی دارد، این نکته قابل ذکر است که KO از ViewModel برای ارتباط مستقیم بین View و Model استفاده می‌کند، چیزی شبیه به منطق MVC با این تفاوت که ViewModel به جای Controller قرار خواهد گرفت.



ابتدا باید به شرح برخی مفاهیم در KO بپردازم:

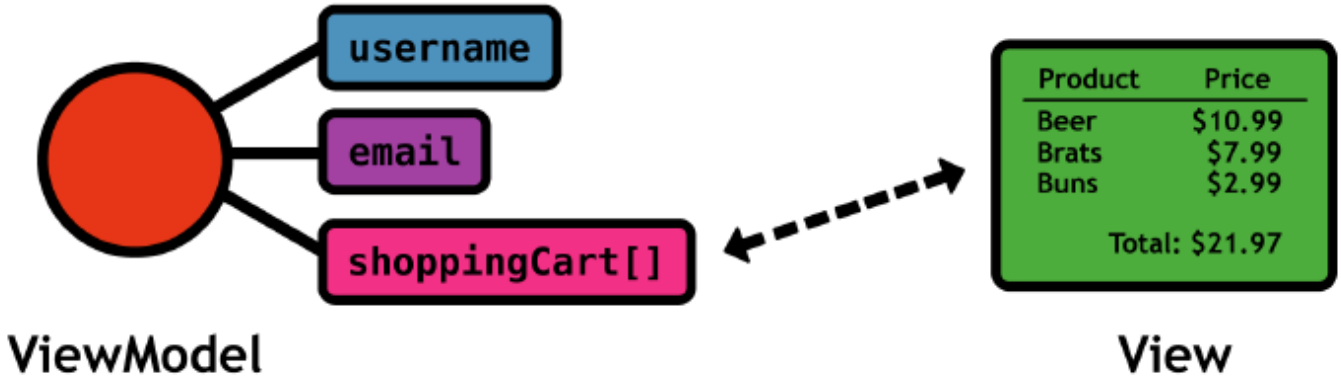
«Observable(قابل مشاهده کردن تغییرات)

KO از Observable برای ردیابی و مشاهده تغییرات خواص ViewModel استفاده می‌کند. در واقع Observable دقیقا شبیه به متغیرها در JavaScript عمل می‌کنند با این تفاوت که به KO اجازه می‌دهند که تغییرات این خواص را پیگیری کند و این تغییرات را به بخش‌های مرتبط View اعمال نماید. اما سوال این است که KO چگونه متوجه می‌شود که این تغییرات بر کدام قسمت در View تاثیر خواهند داشت؟ جواب این سوال در مفهوم Binding است.

«Binding

برای اتصال بخش‌های مختلف View به Observableها باید از binding(مقید سازی) استفاده کنیم. بدون عملیات binding، امکان اعمال تغییرات Observableها بر روی عناصر HTML امکان پذیر نیست.

برای مثال در شکل زیر یکی از خواص ViewModel را به View متناظر مقید شده است.



با کمی دقت در شکل بالا این نکته به دست می‌آید که می‌توان در یک ViewModel، فقط خواص مورد نظر را به عناصر Html مقید کرد .

دانلود فایل‌های مورد نیاز

فایل‌های مورد نیاز برای KO رو می‌توانید از اینجا دانلود نمایید و به پروژه اضافه کنید. به صورت پیش فرض فایل‌های مورد نیاز KO، در پروژه‌های MVC 4 وجود دارد و نیاز به دانلود آن‌ها نیست و شما باید فقط مراحل BundleConfig را انجام دهید.

تعریف ViewModel

برای تعریف ViewModel و پیاده سازی مراحل Observable و binding باید به صورت زیر عمل نمایید:

```
<html lang='en'>
<head>
<title>Hello, Knockout.js</title>
<meta charset='utf-8' />
<link rel='stylesheet' href='style.css' />
</head>
<body>
<h1>Hello, Knockout.js</h1>
<script type='text/javascript' src='knockout-2.1.0.js'>
      <script type='text/javascript'>
             var personViewModel = {
                  firstName: "Masoud",
                  lastName: "Pakdel"
                };
                 ko.applyBindings(personViewModel);
       </script>
</script>
</body>
</html>
```

مشاهده می‌کنید که ابتدا یک ViewModel به نام person ایجاد کردم همراه با دو خاصیت به نام‌های firstName و lastName. تابع applyBinding برای KO بدین معنی است که این آبجکت به عنوان یک ViewModel در این صفحه مورد استفاده قرار خواهد گرفت. اما برای مشاهده تغییرات باید یک عنصر HTML را به این ViewModel مقید(bind) کنیم.

مقید سازی عناصر HTML

برای مقید سازی عناصر HTML به ViewModelها باید از data-bind attribute استفاده نماییم. برای مثال:

```
<p><span data-bind='text: firstName'></span>'s Shopping Cart</p>
```

اگر به data-bind در تگ span بالا توجه کنید خواهید دید که مقدار text در این تگ را به خاصیت firstName در viewModel این صفحه bind شده است. تا اینجا KO می‌داند که چه عنصر از DOM به کدام خاصیت از ViewModel مقید شده است اما هنوز دستور ردیابی تغییرات(Observable) را برای KO تعیین نکردیم.

چگونه خواص را Observable کنیم

در پروژه‌های WPF، فقط در صورتی تغییرات خواص یک کلاس ردیابی می‌شوند که اولا کلاس اینترفیس INotifyPropertyChanged را پیاده سازی کرده باشد ثانیا، در متد set این خواص، متد OnPropertyChanged(البته این متد می‌تواند هر نام دیگری نیز داشته باشد) صدا زده شده باشد. نکته مهم و اساسی در KO نیز همین است که برای اینکه KO بتواند تغییرات هر خاصیت را مشاهده کند حتما خواص مورد نظر باید Observable شوند. برای این کار کافیست به صورت عمل کنید:

```
var personViewModel = {
  firstName: ko.observable("Masoud"),
  lastName: ko.observable("Pakdel")
};
```

مزیت اصلی برای اینکه حتما خواص مورد نظرتان Observable شوند این است که، در صورتی که مایل نباشید تغییرات یک خاصیت بر روی View اعمال شود کافیست از دستور بالا استفاده نکنید. درست مثل اینکه هرگز مقدار آن تغییر نکرده است.

پیاده سازی متدهای get و set

همان طور که متوجه شدید، Observableها متغیر نیستند بلکه تابع هستند در نتیجه برای دستیابی به مقدار یک observable کافیست آن را بدون پارامتر ورودی صدا بزنیم و برای تغییر در مقدار آن باید همان تابع را با مقدار جدید صدا بزنیم. برای مثال:

```
personViewModel.firstName() // Get
personViewModel.firstName("Masoud") // Set
```

البته این نکته را هم متذکر شوم که در ViewModelهای خود می‌توانید توابع سفارشی مورد نیاز را بنویسید و از آن‌ها در جای مناسب استفاده نماید(شبیه به مفاهیم Commandها در WPF)

مقید سازی تعاملی

اگر با WPF آشنایی دارید می‌دانید که در این گونه پروژه‌ها می‌توان رویدادهای مورد نظر را به Commandهای خاص در ViewModel مقید کرد. در KO نیز این امر به آسانی امکان پذیر است که به آن Interactive Bindings می‌گویند. فقط کافیست در data-bind attribute از نام رویداد استفاده نماییم. مثال:
ایتدا یک ViewModel به صورت زیر خواهیم داشت:

```
function PersonViewModel() {
   this.firstName = ko.observable("Masoud");
   this.lastName = ko.observable("Pakdel");
   this.clickMe= function() {
    alert("this is test!");
  };
};
```

تنها نکته قابل ذکر تعریف تابع سفارشی به نام clickMe است که به نوعی معادل Command مورد نظر ما در WPF است. در عنصر HTML مورد نظر که در این جا button است باید data-binding به صورت زیر باشد:

```
<button data-bind='click: clickMe'>Click Me...</button>
```

در نتیجه بعد از کلیک بر روی button بالا تابع مورد نظر در viewModel اجرا خواهد شد.
پس به صورت خلاصه:

ابتدا ViewModel مورد نظر را ایجاد نمایید؛
سپس با استفاده از data-bind عملیات مقید سازی بین View و ViewModel را انجام دهید
در نهایت با استفاده از Obsevable تغییرات خواص مورد نظر را ردیابی نمایید.

ادامه دارد...

نظرات خوانندگان

نویسنده: نوید
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۳:۲۳

اگر امکان داره کدهای مثالهای مربوطه رو هم بذارید . امیدوارم این سری آموزش رو ادامه بدین . با تشکر

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۳:۴۲

در پست‌های بعدی که مفاهیم مهم و اصلی Knockout رو بررسی می‌کنیم حتما مثال‌های مربوطه قرار داده می‌شوند.

نویسنده: دادخواه
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۶:۱۱

سلام
اگر از فریم ورک‌های KnockoutJS و یا AngularJS استفاده کنم. ایا نیاز هست که JQuery را نیز ضمیمه کنم و یا دیگر به JQuery نیازی نیست؟
آیا کارهایی که JQuery انجام میدهد را این دو فریم ورک و یا کلا فریم ورک‌های دیگر می‌توانند انجام دهند؟

تشکر

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۰۶ ۱۶:۴۳

Knockout.js جایگزین jQuery یا MooTools نیست. در این کتابخانه animation یا مدیریت عمومی رخدادها، ساده سازی Ajax و مانند آن پیاده سازی نشده‌اند (هرچند Knockout.js امکان parse اطلاعات Ajax ایی دریافتی را دارد). هدف از Knockout.js ارائه مکملی برای سایر فناوری‌های وب جهت تولید برنامه‌های غنی و *دسکتاپ مانند* وب است. پشتیبانی خوبی از آن توسط مایکروسافت صورت می‌گیره چون نویسنده‌اش عضو تیم ASP.NET MVC است.

نویسنده: سعید یزدانی
تاریخ: ۱۳۹۲/۰۶/۱۸ ۱۷:۵۲

سلام تشکر بابت مطالب ارزشمندی که گذاشتید
ایا میشه در view از چند model view استفاده کرد ؟
اگر میشه چطور باید در هنگام bind کردن به html صفحه از هم تفکیک کرد

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۶/۱۸ ۲۲:۱۵

ممنون دوست عزیز.
بله امکان پذیر است. باید از المان‌های تودرتو استفاده کنیم. به این صورت که المان ریشه با استفاده از with به model مربوطه مقید می‌شود و المان‌های داخلی به خواص مدل bind می‌شوند. برای مثال:

```
<div data-bind="text: teacher"> </div>//مقید سازی به مدل اول
<p data-bind="with: student">//مقید سازی المان ریشه به مدل دوم
    Name: <span data-bind="text: name"> </span>,//مقید سازی خواص به المان‌های داخلی
    Family: <span data-bind="text: family"> </span>
</p>

<script type="text/javascript">
    ko.applyBindings({
        teacher: "myTeacher",
        student: {
```

```
            name:  "Masoud",
            family: "Pakdel"
        }
    });
</script>
```

عنوان: **ExtJs! رویا یا کابوس؟**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۷/۰۱ ۸:۳۰
آدرس: www.dotnettips.info
گروه‌ها: JavaScriptFrameWork, ExtJs

چندی پیش یکی از دوستان درباره فریم ورک ExtJs سوالاتی را پرسیده بود که تصمیم گرفتم جواب‌های مورد نظر را به صورت عمومی در قالب یک پست منتشر کنم.
ExtJs چیست؟
چه زمانی کاربرد دارد؟
تفاوت آن با سایر فریم ورک‌های جاوااسکریپ در چیست؟
شاید خیلی از شما با MODX آشنایی داشته باشید یا حتی با این CMS کار کرده باشید. اگر این طور است پس حتما با پنجره‌های زیبا و کامپوننت‌های قوی و اعتبارسنجی‌های سفارشی و تعاملاتی Ajax ای آن آشنایی دارید و شاید این سوال به ذهنتان خطور کرده باشد که در طراحی این CMS که بر پایه زبان PHP است دقیقا از چه چیز استفاده شده است؟
پاسخ یک کلمه است: ExtJs. بله درست است در طراحی این CMS تنها از یک فریم ورک جاوااسکریپتی به نام ExtJs استفاده شده است. فریم ورکی که به عقیده بعضی‌ها یک رویا برای توسعه دهندگان وب است و به عقیده سایرین شاید یک کابوس. در این پست قصد دارم به عنوان کسی که با این فریم ورک آشنایی دارم این موضوع را بررسی و مزایا و معایب این فریم ورک را عنوان کنم.
ExtJs یک فریم ورک جاوااسکریپ است بر مبنای Sencha و طراحی شده برای توسعه پروژه‌های وب در مقیاس بزرگ و به صورت cross-platform . مجوز استفاده از این فریم ورک به صورت GPLv3 است.(یعنی مجاز به استفاده رایگان از فایل‌های این فریم ورک هستید به شرطی که قصد استفاده تجاری از پروژه تهیه شده را نداشته باشید! در غیر این صورت باید زحمت خرید نسخه تجاری این فریم ورک را متحمل شوید).
نسخه ای که درباره آن بحث می‌کنیم نسخه چهار این فریم ورک (ExtJs 4) که بر مبنای ExtJs 3 تولید شده است. تفاوت عمده آن با نسخه قبلی در تکمیل ابزار و کامپوننت هاست و از طرفی نسخه چهار این فریم ورک بر مبنای مدل MVC توسعه داده شده است. یعنی همانند Angular و BackBoneJs می‌توانید مفاهیم کنترلر و مدل را به راحتی پیاده سازی کنید.

رویایی به نام ExtJs

اگر بخواهیم این فریم ورک را یک رویا برای توسعه دهندگان وب بنامیم می‌توان عناوین زیر را به عنوان مزایا برشمرد:
در درجه اول قابلیتی که این فریم ورک را متفاوت از سایر فریم ورک‌های جاوااسکریپتی می‌کند این است که این فریم ورک انبوهی از کامپوننت‌ها و ویجیت‌های آماده را به همراه خود دارد (با کارایی و انعطاف پذیری قابل قبول) و به نوعی شما را بی نیاز از هرگونه مجموعه کامپوننت‌های دیگر خواهد کرد.

این فریم ورک به خوبی از مباحث OOP پشتیبانی می‌کند و به این صورت است که یک سری مفاهیم و مدل‌های پایه در این فریم ورک تعبیه شده و به راحتی شما می‌توانید مدل‌های مورد نظر خود را بر اساس این مفاهیم و مدل‌های پایه توسعه دهید.

تمام مفاهیم و ابزار لازم جهت درخواست‌های Ajax ای و اعتبار سنجی سفارشی و دستکاری عناصر DOM و... به خوبی در این فریم ورک وجود دارد.

به دلیل وجود کامپوننت‌های یک دست و آماده به راحتی می‌توانید امکان تغییر theme را در پروژه‌های خود بدون کوچکترین زحمت قرار دهید.

کنترل GridPanel،TreeView ، کنترل‌های ورود اطلاعات، کنترل Tab با قابلیت درخواست‌های لود صفحات به صورت Ajax و Async با کمترین زحمت در کد نویسی و هم چنین چارت‌های بسیار گسترده و متنوع از دیگر مزایای این فریم ورک می‌تواند باشد.

ارائه مکانیزمی مناسب برای کار با عملیات داده ای Json. به عنوان نمونه:

```
Ext.data.JsonP.request({
```

```
  url: '@url',
  params: {
  apiKey: '1234'
  },
  callbackKey: 'myCallbackFn',
  success: function(){
 },
  failure: function(){
},
scope: this
});
```

این فریم ورک ابزارهای جالب و کارآمدی برای توسعه به صورت SPA را داراست.
کنترل‌های داده ای این فریم ورک در هنگام کار با حجم داده بسیار زیاد، فراتر از انتظار عمل می‌کنند(برای مثال کنترل GridPanel و DataView)
اگر قصد تولید و توسعه بک پروژه بزرگ درون سازمانی را دارید و سرعت تولید نیز برای شما مهم است ExtJs در این زمینه کمک شایانی به شما خواهد کرد.

و...

حال با همه این تفاسیر آیا این فریم ورک یک رویا برای هر توسعه دهنده وب خواهد بود؟

به طور قطع نه. با توجه به اصل واقع بینی! همواره به خاطر داشته باشید که اگر این فریم ورک یک ابزار بی نقص و همه منظوره بود الآن مطمئنا صدها کتاب و مستندات درباره آن نوشته شده بود و شاید شهرتی بس فراتر از این داشت.

کابوسی به نام ExtJs
اگر قصد ایجاد یک وب سایت کوچک و جمع و جور را دارید به طوری که مباحث مربوط به SEO نیز برای شما اهمیت دارد تجربه نشان داده است که انتخاب ExtJs می‌تواند یکی از بزرگ‌ترین اشتباهات در طول عمر کاری شما شود.
ExtJs هیچ گونه کمکی برای تولید و توسعه اپلیکیشن‌های موبایل یا پروژه‌های وب گرافیکی نمی‌تواند به شما کند.
اگر سرعت یکی از فاکتور خیلی مهم برای شماست بهتر است به این فریم ورک علاقه نشان ندهید.(کتابخانه آن چیزی در حدود 500KB است! البته با فشرده سازی به 150KB خواهد رسید که باز هم قابل قبول نیست)

مجوز استفاده برای پروژه‌های تجاری به صورت رایگان نیست.(^)
به دلیل وجود ابزارهای متنوع و زیاد؛ زمان یادگیری برای آشنایی و کار کردن با ابزارها، نسبتا طولانی خواهد شد.
کد نویسی برای استفاده از ابزارهای آن در مقایسه با Jquery و Angular بیشتر خواهد بود(البته این به نوعی مزیت هم است، به دلیل اینکه خوانایی کدها بسیار بالا می‌رود)

در طراحی کامپونت‌ها آن از تگ div در حد غیر قابل قبول استفاده شده است به طوری که Debug صفحات حتی با Firebug هم در بعضی مواقع سخت می‌شود.
و...

Ext.Net چیست؟

Ext.Net یک پیاده سازی خاص از فریم ورک ExtJs است که برای توسعه پروژه‌های Asp.Net Web Forms و Asp.Net MVC طراحی شده است. تفاوت اصلی بین این دو محصول در نوع کدنویسی برای استفاده در پروژه‌های Asp.Net است. برای مثال در هنگام کار با Ext.Net و پروژه‌های MVC از آنجا که این محصول سازگاری کامل با موتور Razor دارد به راحتی می‌توانید به صورت سینتکس Razor صفحات خود را طراحی کنید.

مثال:

ExtJs

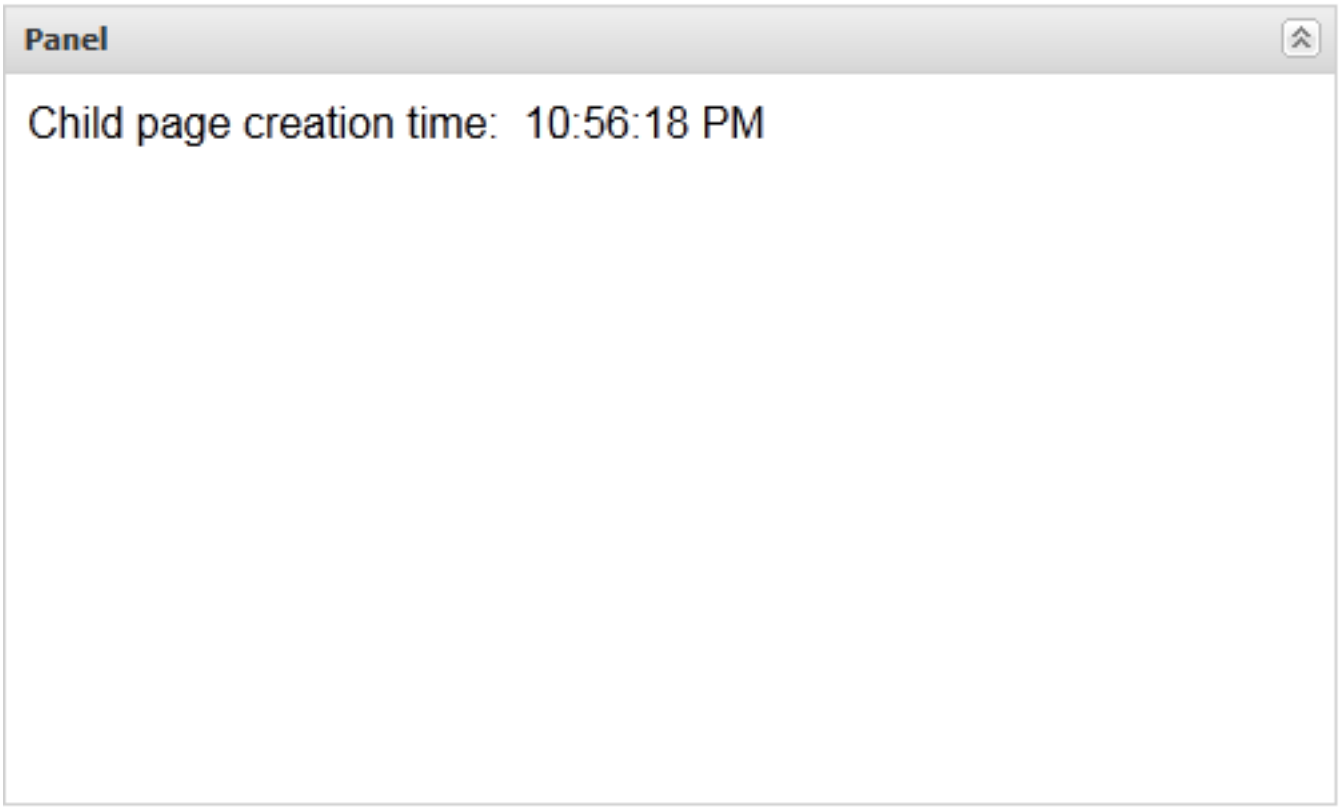
```
Ext.create('Ext.panel.Panel', {
title: 'Fit Layout',
width: 500,
height: 200,
items: {
title: 'Inner Panel',
html: 'Panel content',
bodyPadding: 10,
border: true
},
renderTo: Ext.getBody()
});
```

اجرای کد بالا با استفاده از ExtJs به صورت زیر خواهد بود:

Ext.Net

```
@(X.Panel()
        .ID("ExpandablePanel")
        .Title("Panel")
        .Width(500)
        .Height(300)
        .Collapsible(true)
        .Loader(X.ComponentLoader()
            .Url(Url.Action("RenderChild"))
            .Mode(LoadMode.Frame)
            .DisableCaching(true)
            .Params(new { containerId = "ExpandablePanel" })
            .LoadMask(lm => lm.ShowMask = true)
        )
        .Listeners(l => {
            l.Expand.Handler = "this.reload();";
            l.Collapse.Handler = "this.clearContent();";
        })
    )
```

خروجی مورد نظر برای Ext.Net:



جمع بندی:

با توجه مواردی که ذکر شد می‌توان به یک نکته مهم رسید و آن هم این است که هنگام انتخاب ExtJs یا Ext.Net (البته این شامل اکثر ابزارهای توسعه دیگر نیز خواهد شد) حتما شرایط موجود و حاکم برای توسعه محصول را مد نظر داشته باشید که این شرایط شامل محیط اجرای محصول، مدت زمان لازم برای توسعه، سطح دانش نیروی‌های توسعه دهنده و ... نیز می‌باشد.

نظرات خوانندگان

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۰۱ ۸:۵۷

فارسی سازی و راست به چپ رو هم به لیست اضافه کنید.

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۷/۰۱ ۹:۱۰

درست است!
البته پشتیبانی از زبان‌های RTL از پیش نیازهای فریم ورک‌های Component Oriented است.

نویسنده: سیدعلی
تاریخ: ۱۳۹۲/۰۷/۰۱ ۱۵:۵۲

با تشکر از مطلب منتشر شده.
بنده با جستجو و نظرات گرفته شده از بازخوردهای ExtJs به مطالب مورد اشاره از شما که استفاده از آن می‌تواند مفید با مضر باشد رسیده بودم اما سوالی که داشتم به دلیل عدم استفاده کاربردی از ExtJs این است آیا استفاده آن در یک سیستم ماژولار که دارای زیر سیستمهای مختلف است تداخل در کار سایر قسمت‌ها ایجاد می‌کند؟ در واقع سازگاری آن با دیگر فریم ورک‌ها چگونه است؟(با توجه به اینکه ExtJs بعد از لود اولیه فقط می‌تواند Data بین آن و برنامه مبادله شود و سرعت در واقع افزایش می‌یابد در جایی از برنامه که لود اولیه مهم نباشد مثلا قسمت‌های مدیریتی سیستم می‌تواند مفید باشد.)

نویسنده: Shahrooz
تاریخ: ۱۳۹۲/۰۷/۰۱ ۲۳:۴۴

تو یک پروژه بزرگ ازش استفاده کردیم باگهای بسیار زیادی داشت در حدی که به بعد یکسال داریم می‌بریم رو انگولار

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۷/۰۲ ۹:۲۰

یک مثال از نحوه پیاده سازی پروژه ماژولار با ExtJs (^).

نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۷/۰۲ ۹:۲۲

@دانلود سورس یک مثال از نحوه پیاده سازی پروژه MVC با استفاده از ExtJs(^)

نویسنده: سعید نعمتی
تاریخ: ۱۳۹۲/۰۷/۰۶ ۸:۴۸

به عنوان کسی که توی یه شرکت توی یه پروژه Enterprise تقریباً یک سال با این فریمورک درگیر بودم، فقط یه چیز میتونم بگم:
Get away from Ext JS

نویسنده: امید شریعتی
تاریخ: ۱۳۹۲/۰۷/۰۶ ۱۷:۲

یک مقایسه دیگر: https://www.markhamstra.com/extjs/2013/extjs-dream-nightmare-jquery/

سعید و شهروز، تو رو خدا نون مارو آجر نکنین بابا جان من :D

نویسنده: امید شریعتی
تاریخ: ۱۳۹۲/۰۷/۰۷ ۸:۷

فکر کنم این مطلب هم میتونه واسه اونایی که به هر دلیلی با ExtJs کار میکنن مفید باشه: پانزده اشکال رایج در برنامه نویسی با ExtJs را از اینجا بخوانید. http://omidshariati.com/blog/?p=124

نویسنده: مهدی
تاریخ: ۱۳۹۲/۰۷/۱۹ ۰:۴۹

فارسی سازی و راست چین کردن و حتی تقویم جلالی و فونت فارسی رو به راحتی در عرض 1 هفته به Ext Net اضافه کردیم...
فقط چیزی که در Ext بسیار جذاب هست سرعت بالای Requestهاست که خیلی به درد میخوره! در نسخه جدیدی هم که به زودی منتشر میشه 4000 باگ رفع شده که فکر میکنم خیلی از ایرادات رفع بشه.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۷/۱۹ ۸:۵۹

اگر قرار باشه کسی یک کتابخانه 500 کیلوبایتی یا بیشتر رو مورد استفاده قرار بده، فقط برای اینکه بتونه برنامه‌های دسکتاپ رو در مرورگر شبیه سازی کنه، چرا نره از سیلورلایت استفاده کنه ؟

نویسنده: مهدی
تاریخ: ۱۳۹۲/۰۷/۱۹ ۱۰:۲۶

من 3 سال سیلورلایت کار کردم تنها جایی که ازش تجربه خوبی داشتم نرم افزار GIS آنلاین بود (WebGIS) که به شدت خوب عمل کرد!
اما بزرگترین سوال کاربران در نمایشگاه از من این بود: آیا در موبایل هم اجرا میشه؟ و من پاسخی نداشتم!
سیلورلایت فقط روی موبایل یا تبلت‌های ویندوزی اجرا میشه! دیگه خود دانی!
در ضمن سیلورلایت اگه Crash کنه کل برنامه در جا بسته میشه و دوباره باید لود بشه و این چیزی هست که توی وب معمولی اتفاق نمی‌افته...

نویسنده: سانتا
تاریخ: ۱۳۹۲/۰۹/۲۵ ۱۵:۳۳

دوست عزیز، استفاده از ext.net رو به شما توصیه می‌کنم. تا به حال تو 3 پروژه بزرگ و 3 تا پروژه متوسط استفاده کردم که عالیه. سرعت بالا در حجم اطلاعات زیاد و انعطاف پذیری فوق العاده و امکاناتی مثل راست به چپ، تقویم شمسی ... (حتی میتونی تو یه صفحه همزمان تقویم شمسی و میلادی داشته باشی یا اینکه فقط چند تا کامپونتت راست به چپ باشه و بقیه چپ به راست)
فقط این مخصوص .Net که هم نسخه ASP داره هم MVC

عنوان: آموزش Knockout.Js #4
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۷/۰۲ ۸:۳۰
آدرس: www.dotnettips.info
گروه‌ها: KnockoutJS, JavaScriptFrameWork

مقید سازی رویداد کلیک

Click Binding روشی است برای اضافه کردن یک گرداننده رویداد در زمانی که قصد داریم یک تابع جاوااسکریپتی را در هنگام کلیک بر روی المان مورد نظر فراخوانی کنیم. از این مقید سازی عموما در عناصر button و input و تگ a استفاده می‌شود. اما در حقیقت در تمام عناصر غیر پنهان صفحه مورد استفاده قرار می‌گیرد.

```
<div>
    Number Of Clicks <span data-bind="text: numberOfClicks"></span> times
    <button data-bind="click: clickMe">Click me</button>
</div>

<script type="text/javascript">
    var viewModel = {
        numberOfClicks : ko.observable(0),
        clickMe: function() {
            var previousCount = this.numberOfClicks();
            this.numberOfClicks(previousCount + 1);
        }
    };
</script>
```

رویداد کلیک button در کد بالا به تابعی با نام clickMe مقید شده است. این تابع در viewModel جاری صفحه تعریف شده است و در بدنه آن تعداد کلیک‌های قبلی را به علاوه یک خواهد کرد. از آنجا که تگ span در بالای صفحه به تعداد کلیک‌ها مقید شده است در نتیجه همواره مقدار این تگ به روز خواهد بود.

***نکته اول:** اگر قصد داشته باشیم که عنصر جاری در viewModel را به گرداننده رویداد پاس دهیم چه باید کرد؟
هنگام فراخوانی رویدادها، KO به صورت پیش فرض مقدار جاری مدل را به عنوان اولین پارامتر به این گرداننده پاس می‌دهد. این روش مخصوصا در هنگامی که قصد اجرای عملیاتی خاص بر روی تک تک عناصر یک مجموعه را داشته باشید(مثل حلقه foreach) بسیار مفید خواهد بود.

```
<ul data-bind="foreach: places">
    <li>
        <span data-bind="text: $data"></span>
        <button data-bind="click: $parent.removePlace">Remove</button>
    </li>
</ul>

 <script type="text/javascript">
     function MyViewModel() {
         var self = this;
         self.places = ko.observableArray(['Tehran', 'Esfahan', 'Shiraz']);

         self.removePlace = function(place) {
             self.places.remove(place)
         }
     }
     ko.applyBindings(new MyViewModel());
</script>
```

در تابع removePlace می‌بینید که مقدار آیتم جاری در لیست به عنوان اولین آرگومان به این تابع پاس داده می‌شود، در نتیجه می‌دانیم که کدام عنصر را باید از لیست مورد نظر حذف کنیم. برای به دست آوردن آیتم جاری در لیست از $parent یا $root می‌توان استفاده کرد.
همان طور که پست قبل توضیح داده شد؛ برای اینکه بتوانیم از یک viewModel به مجموعه از عناصر در یک حلقه foreach مقید کنیم امکان استفاده از اشاره گر this میسر نیست. در نتیجه بهتر است در ابتدای viewModel مقدار این اشاره گر را در یک متغیر معمولی (در اینجا به نام self است) ذخیره کنیم و از این پس این متغیر را برای اشاره به عناصر viewModel به کار بریم. در اینجا self به عنواتن یک alias برای this خواهد بود.

***نکته دوم:** دسترسی به عنصر رویداد

در بعضی مواقع نیاز است در حین فراخوانی رویداد ،عنصر رویداد DOM به عنوان فرستنده در اختیار تابع گرداننده قرار گیرد. خبر خوش این است که KO به صورت پیش فرض این عنصر را نیز به عنوان پارامتر دوم به توابع گرداننده رویداد پاس می‌دهد. برای مثال:

```
<button data-bind="click: myFunction">
    Click me
</button>

 <script type="text/javascript">
    var viewModel = {
        myFunction: function(data, event) {
            if (event.shiftKey) {

            } else {
            }
        }
    };
    ko.applyBindings(viewModel);
</script>
```

تابع myFunction در مثال بالا دارای دو پارامتر است. پارامتر دوم در این تابع به عنوان عنصر فرستنده رویداد مورد استفاده قرار خواهد گرفت. بدین ترتیب در توابع event Handlerها می‌توان به راحتی اطلاعات مورد نیاز درباره آبجکت رویداد را به دست آورد.

***نکته سوم:** به صورت پیش فرض KO از اجرای عملیات پیش فرض رویدادها جلوگیری به عمل می‌آورد. این به این معنی است که اگر برای رویداد کلیک تگ a بک تابع گرداننده تعریف کرده باشید، بعد از کلیک بر روی این المان؛ مرورگر فقط این تابع تعریف شده توسط شما را فراخوانی خواهد کرد و دیگر عملیات راهبری به صفحه مورد نظر در خاصیت href صورت نخواهد گرفت. اگر به هر دلیلی قصد داشته باشیم که این رفتار صورت نگیرد کافیست در انتهای تابع گرداننده رویداد مقدار true برگشت داده شود.

***نکته چهارم:** مفهوم clickBubble

ابتدا به کد زیر دقت کنید:

```
<div data-bind="click: myDivHandler">
    <button data-bind="click: myButtonHandler">
        Click me
    </button>
</div>
```

همان طور که مشاهده می‌کنید در کد بالا برای عنصر button یک رویداد کلید تعریف شده است. از طرف دیگر این button درون تگ div قرار دارد که برای این تگ نیز این رویداد کلیک با تابع گرداننده متفاوتی تعریف شده است. نکته این جاست که به صورت پیش فرض بعد از فراخوانی رویداد کلیک عنصر داخلی، رویداد کلیک عنصر خارجی نیز فراخوانی خواهد شد. به این رفتار event bubbling می‌گویند. اگر قصد داشته باشیم که این رفتار را غیر فعال کنیم(یعنی با کلیک بر روی button، رویداد کلیک تگ div اجرا نشود باید مقدار خاصیت clickBubble رویداد عنصر داخلی را برابر false قرار دهیم) به صورت زیر:

```
<div data-bind="click: myDivHandler">
    <button data-bind="click: myButtonHandler, clickBubble: false">
        Click me
    </button>
</div>
```

نظرات خوانندگان

نویسنده: مهرداد اشکانی
تاریخ: ۱۳۹۲/۰۷/۰۳ ۱۵:۵۷

عالی بود دوست عزیز خیلی استفاده کردیم

نویسنده: سعید
تاریخ: ۱۳۹۲/۱۰/۱۶ ۲۳:۵۸

سلام ضمن تشکر از ارایه آموزش فارسی مفیدتون که باعث میشه زمان کمتری برای درک مفهوم صرف بشه اما لطفا درصورت امکان روند را بایک فرآیند عملی قابل درک توضیح دهید چون بعضی اصطلاحات فارسی نمیتونه گویا باشد مثلا در نکته دوم جملات برای من واضح نبود و چون مثال عملی نیست مجبورم که به مطالب زبان اصلی مراجعه کنم تا مفهوم را بهتر درک کنم. من آموزش قبلیتون که با مثال بود را بخوبی درک کردم و از کاراتون تشکر میکنم

عنوان: آماده سازی Jasmine برای پروژه های Asp.Net MVC
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۷/۰۸ ۹:۴۵
آدرس: www.dotnettips.info
برچسب‌ها: Unit testing, test, TDD, JavaScriptFrameWork, Jasmine

با گسترش روز افزون برنامه‌های تحت وب، نیاز به یک سری ابزار برای تست و اطمینان از نحوه عملکرد صحیح کدهای نوشته شده احساس می‌شود. Jasmine یکی از این ابزارهای قدرتمند برای تست کدهای JavaScript است.
چندی پیش در سایت جاری چند مقاله خوب توسط یکی از دوستان درباره Qunit منتشر شد. Qunit یک ابزار قدرتمند و مناسب برای تست کدهای جاوااسکریپت است و در اثبات صحت این گفته همین کافیست که بدانیم برای تست کدهای نوشته شده در پروژه‌های متن بازی هم چون Backbone.Js و JQuery از این فریم ورک استفاده شده است. اما به احتمال قوی در ذهن شما این سوال مطرح شده است که خب! در صورت آشنایی با Qunit چه نیاز به یادگیری Jasmine یا خدای نکرده Mocha و FuncUnit است؟ هدف صرفا معرفی یک ابزار غیر برای تست کد است نه مقایسه و نتیجه گیری برای تعیین میزان برتری این ابزارها. اصولا مهم‌ترین دلیل برای انتخاب، علاوه بر امکانات و انعطاف پذیری، فاکتور راحتی و آسان بودن در هنگام استفاده است که به صورت مستقیم به شما و تیم توسعه نرم افزار بستگی دارد.

اما به عنوان توسعه دهنده نرم افزار که قرار است از این ابزار استفاده کنیم بهتر است با تفاوت‌ها و شباهت‌های مهم این دو فریم ورک آشنا باشیم:

»Jasmine یک فریم ورک تست کدهای جاوا اسکریپ بر مبنای Behavior-Driven Development است در حالی که Qunit بر مبنای Test-Driven Development است و همین مسئله مهم‌ترین تفاوت بین این دو فریم ورک می‌باشد.
»اگر قصد دارید که از Qunit نیز به روش BDD استفاده نمایید باید از ترکیب Pavlov به همراه Qunit استفاده کنید.
»Jasmine از مباحث مربوط به Spies و Mocking به خوبی پشتیبانی می‌کند ولی این امکان به صورت توکار در Qunit فراهم نیست. برای اینکه بتوانیم این مفاهیم را در Qunit پیاده سازی کنیم باید از فریم ورک‌های دیگر نظیر SinonJS به همراه Qunit استفاده کنیم.
»هر دو فریم ورک بالا به سادگی و راحتی کار معروف هستند
»تمام موارد مربوط به الگوهای Matching در هر دو فریم ورک به خوبی تعبیه شده است
» هر دو فریم ورک بالا از مباحث مربوط به Asynchronous Testing برای تست کدهای Ajax ای به خوبی پشتیبانی می‌کنند.

بررسی چند مفهوم
قبل از شروع، بهتر است که با چند مفهوم کلی و در عین حال مهم این فریم ورک آشنا شویم

```
describe('JavaScript addition operator', function () {
 it('adds two numbers together', function () {
  expect(1 + 2).toEqual(3);
 });
});
```

در کد بالا یک نمونه از تست نوشته شده با استفاده از Jasmine را مشاهده می‌کنید. دستور describe برای تعریف یک تابع تست مورد استفاده قرار می‌گیرد که دارای دو پارامتر ورودی است. ابتدا یک نام را به این تست اختصاص دهید(بهتر است که این عنوان به صورت یک جمله قابل فهم باشد). سپس یک تابع به عنوان بدنه تست نوشته می‌شود. به این تابع Spec گفته می‌شود.
در تابع it کد بالا شما می‌توانید کدهای مربوط بدنه توابع تست خود را بنویسید. برای پیاده سازی Assert در توابع تست مفهوم expectationها وجود دارد. در واقع expect برای بررسی مقادیر حقیقی با مقادیر مورد انتظار مورد استفاده قرار می‌گیرد و شامل مقادیر true یا false خواهد بود.
برای Setup و Teardown توابع تست خود باید از توابع beforeEach و afterEach که بدین منظور تعبیه شده اند استفاده کنید.

```
describe("A spec (with setup and tear-down)", function() {
```

```
  var foo;

  beforeEach(function() {
    foo = 0;
    foo += 1;
  });

  afterEach(function() {
    foo = 0;
  });

  it("is just a function, so it can contain any code", function() {
    expect(foo).toEqual(1);
  });

  it("can have more than one expectation", function() {
    expect(foo).toEqual(1);
    expect(true).toEqual(true);
  });
});
```

کاملا واضح است که در تابع beforeEach مجموعه دستورالعمل‌های مربوط به setup تست وجود دارد. سپس دو تابع it برای پیاده سازی عملیات Assertion نوشته شده است. در پایان هم دستورات تابع afterEach ایجاد می‌شوند.

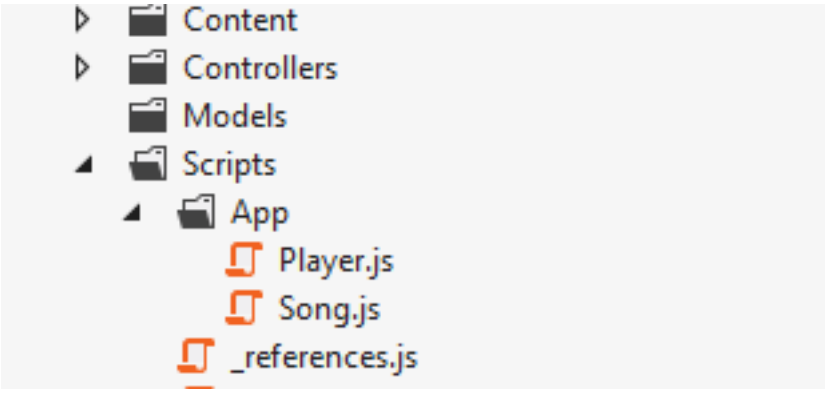
اگر در کد تست خود قصد دارید که یک تابع describe یا it را غیر فعال کنید کافیست یک x به ابتدای آن‌ها اضافه کنید و دیگر نیاز به هیچ کار اضافه دیگری برای comment کردن کد نیست.

```
xdescribe("A spec", function() {
  var foo;

  beforeEach(function() {
    foo = 0;
    foo += 1;
  });

  xit("is just a function, so it can contain any code", function() {
    expect(foo).toEqual(1);
  });
});
```

توابع describe و it بالا در هنگام تست نادیده گرفته می‌شوند و خروجی آن‌ها مشاهده نخواهد شد.

درادامه قصد پیاده سازی یک مثال را با استفاده از Jasmine و RequireJs در پروژه Asp.Net MVC دارم.
برای شروع آخرین نسخه Jasmine را از اینجا دریافت نمایید. یک پروژه Asp.Net MVC به همراه پروژه تست به صورت Empty ایجاد کنید(در هنگام ایجاد پروژه، گزینه create unit test را انتخاب نمایید). فایل دانلود شده را unzip نمایید و دو پوشه lib و spec ،به همراه فایل specRunner.html را در پروژه تست خود کپی نمایید.
فولدر lib شامل فایل‌ها کدهای Jasmine برای setup و tear down و spice و تست کدهای شما می‌باشد.
فایل specRunner.html به واقع یک فایل برای نمایش فایل‌های تست و همچنین نمایش نتیجه تست است.
فولدر spec نیز شامل کدهای Jasmine برای کمک به نوشتن تست می‌باشد.

در این مثال قصد داریم فایل‌های player.js و song.js که به عنوان نمونه به همراه این فریم ورک قرار دارد را در قالب یک پروژه MVC به همراه RequireJs، تست نماییم. در نتیجه این فایل‌ها را از فولدر src انتخاب نمایید و آن‌ها را در قسمت Scripts پروژه اصلی خود کپی کنید(ابتدا یک پوشه به نام App بسازید و فایل‌ها را در آن قرار دهید)



برای استفاده از requireJs باید دستور define را در ابتدا این فایل‌ها اضافه نماییم. در نتیجه فایل‌های Player.js و Song.js را باز کنید و تغییرات زیر را در ابتدای این فایل‌ها اعمال نمایید.

Song.js

```
define(function () {
    function Song() {
    }

    Song.prototype.persistFavoriteStatus = function (value) {
        // something complicated
        throw new Error("not yet implemented");
    };
});
```

Player.js

```
define(function () {
    function Player() {
    }
    Player.prototype.play = function (song) {
        this.currentlyPlayingSong = song;
        this.isPlaying = true;
    };

    Player.prototype.pause = function () {
        this.isPlaying = false;
    };

    Player.prototype.resume = function () {
        if (this.isPlaying) {
            throw new Error("song is already playing");
        }

        this.isPlaying = true;
    };

    Player.prototype.makeFavorite = function () {
        this.currentlyPlayingSong.persistFavoriteStatus(true);
    };
});
```

حال فایل SpecRunner.html را بازکنید و کدهای مربوط به تگ script که به مسیر اصلی فایل‌های تست اشاره می‌کند را Comment نمایید و به جای آن تگ Script مربوط به RequireJs را اضافه نمایید. برای پیکر بندی RequireJs باید از baseUrl و paths استفاده کرد.

```
<link rel="shortcut icon" type="image/png" href="lib/jasmine-1.2.0/jasmine_favicon.png">
<link rel="stylesheet" type="text/css" href="lib/jasmine-1.2.0/jasmine.css">
<script type="text/javascript" src="lib/jasmine-1.2.0/jasmine.js"></script>
<script type="text/javascript" src="lib/jasmine-1.2.0/jasmine-html.js"></script>

<script type="text/javascript" src="../RequireJsMvcStarter/Scripts/require.js"></script>

<!-- include source files here... -->
<!--<script type="text/javascript" src="spec/SpecHelper.js"></script>-->
<!--<script type="text/javascript" src="spec/PlayerSpec.js"></script>-->

<!-- include spec files here... -->
<!--<script type="text/javascript" src="src/Player.js"></script>
<script type="text/javascript" src="src/Song.js"></script>-->

<script type="text/javascript">

    require.config({
        baseUrl: '../RequireJsMvcStarter/Scripts/App',
        paths: {
            spec: '../../../RequireJsMvcStarter.Scripts.Test/spec'
        }
    });
</script>
```

baseUrl در پیکر بندی requireJs به مسیر فایل‌های پروژه که در پروژه اصلی MVC قرار دارد اشاره می‌کند. paths برای تعیین مسیر فایل‌های تست که در پوشه spec در پروژه تست قرار دارد اشاره می‌کند. اگر دقت کرده باشید به دلیل اینگه تگ‌های script مربوط به لود فایل‌های SpecHelper.js و PlayerSpec.js به صورت comment در آمده اند در نتیجه این فایل‌ها لود نخواهند شد و خروجی مورد نظر مشاهده نمی‌شود. در این جا باید از مکانیزم AMD موجود در RequireJs استفاده نماییم و فایل‌های مربوطه را لود کنیم. برای این کار نیاز به اضافه کردن دستور require در ابتدای تگ script به صورت زیر در این فایل است. در نتیجه فایل‌های PlayerSpec و SpecHelper نیز توسط RequireJs لود خواهند شد.

```
<script type="text/javascript">

    require(['spec/PlayerSpec', 'spec/SpecHelper'], function() {
        var jasmineEnv = jasmine.getEnv();
        jasmineEnv.updateInterval = 1000;

        var htmlReporter = new jasmine.HtmlReporter();

        jasmineEnv.addReporter(htmlReporter);

        jasmineEnv.specFilter = function(spec) {
            return htmlReporter.specFilter(spec);
        };

        var currentWindowOnload = window.onload;

        window.onload = function() {
            if (currentWindowOnload) {
                currentWindowOnload();
            }
            execJasmine();
        };

        function execJasmine() {
            jasmineEnv.execute();
        }
    });
</script>
```

نیاز به یک تغییر کوچک دیگر نیز وجود دارد. فایل PlayerSpec را باز نمایید و وابستگی فایل‌های آن را تعیین نمایید. از آن جا که این فایل برای تست فایل‌های Player , Song ایجاد شده است در نتیجه باید از define برای تعیین این وابستگی‌ها استفاده نماییم.

PlayerSpec.js

```
1  define(['Player', 'Song'], function(Player, Song) {
2      describe("Player", function() {
3          var player;
4          var song;
5
6          beforeEach(function() {
7              player = new Player();
8              song = new Song();
9          });
10
```

یادآوری :

»دستور describe در فایل بالا برای تعریف تابع تست است. همان طور که می‌بینید بک نام به آن داده می‌شود به همراه بدنه تابع تست.

»دستور beforeEach برای آماده سازی مواردی است که قصد داریم در تست مورد استفاده قرار گیرند. همانند متدهای Setup در UnitTest.

» دستور expect نیز معادل Assert در UnitTest است و برای بررسی صحت عملکرد تست نوشته می‌شود.

اگر فایل SpecRunner.html را دوباره در مرورگر خود باز نمایید تصویر زیر را مشاهده خواهید کرد که به عنوان موفقیت آمیز بودن پیکر بندی پروژه و تست‌های آن می‌باشد.