[تاريخ:۱ آبان ۱۳۹۱] ~ [Java Script](http://dnpars.ir/blog/?cat=21)،[MVVM](http://dnpars.ir/blog/?cat=20)،[Silverlight](http://dnpars.ir/blog/?cat=22)،[Single Page Application](http://dnpars.ir/blog/?cat=7)،[Win RT](http://dnpars.ir/blog/?cat=9)،[WPF](http://dnpars.ir/blog/?cat=23)،[مطلب](http://dnpars.ir/blog/?cat=6)

همانطور که در انتهای سری پست‌های بررسی [Application Server و Client](http://dnpars.ir/blog/?p=33) قول داده بودم، می‌خواهم یک Application Server را با NET؛ و یک کلاینت را با Java Script ایجاد کنم.

برنامه نمونه ما در اینجا یک برنامه چت با امکانات عالی و ظاهری زیبا خواهد بود، که در یک ارتباط دو طرفه با Application Server امکان چت بین چندین نفر را فراهم می‌کند.

البته در کنار نوشتن این برنامه، اهداف دیگری نیز دنبال می‌شود، که یکی از اصلی‌ترین آن‌ها زیبا کد نوشتن در جاوا اسکریپت است.

استفاده از جاوا اسکریپت روز به روز در حال افزایش است، ولی کمتر کسی دنبال آن است تا از آن به صورت جدی استفاده کند (منظور میزان استفاده نیست، درستی و مدون بودن استفاده است).

جاوا اسکریپت آن نیست که ما فقط بگوییم، jQuery UI، Ext JS، Kendo UI، jQ Grid و …

جاوا اسکریپت یک زبان برنامه نویسی است، و استفاده از آن در ساخت شکیل‌تر ظاهر برنامه فقط یکی از استفاده های آن است.

در طی نوشتن این مثال می‌خواهیم با Design Pattern های مربوط به Java Script آشنا شویم.

و در یکی از مهم‌ترین آنها به Model View – View Model یا به اختصار MVVM بپردازیم.

و همچنین به مباحث بسیار مهم مانند Dependency Injection بپردازیم.

در نهایت کلاینت مورد نظر را با امکانات بیشتر در Silverlight باز نویسی خواهیم کرد.

و اما MVVM چیست؟

در بسیاری از مواقع (چه C# کارها و چه JavaScript کارها)، می‌آییم و یک فرم درست می‌کنیم و سپس می‌آییم و در پشت آن شروع به کد نویسی می‌کنیم.

خب اشکال این کار در چیست؟

در کدنویسی مان می‌گوییم اگر شرط X بر قرار نبود بود، Is Enabled فلان دکمه را غیر فعال کن، یا رکورد جاری را از دیتاگرید بگیر و تغییراتی را روی آن اعمال کن

در تمامی خطوط کدهای ما اسامی مانند Data Grid، Combo Box، Button و … به چشم می‌خورد.

نمونه یک کد کثیف :

[?](http://dnpars.ir/blog/?p=108)

|  |  |
| --- | --- |
| Textbox1.Text = datagrid1[datagrid2.CurrenRow.Index].Value.ToString() | 1 |

مشکل این مدل کدنویسی در چند آیتم است :

۱- منطق شما که یکی از با ارزش‌ترین اجزاء برنامه شما است، هم اکنون به فرم شما وابسته شده است، آیا تا به حال به برنامه‌ها و بازی‌ها و سایت‌هایی که یک کار را در ۳ ظاهر متفاوت انجام می‌دهند دقت کرده‌اید؟ (یکی برای کامپیوتر، یکی برای موبایل‌ها و یکی هم برای پرینت)، منطق عوض نشده، بلکه فقط ظاهر عوض شده است.

در روش اشتباه (ولی رایج)، که شما منطق را به یکی از فرم‌هایتان گره می‌زنید، ارائه چنین امکانی بسیار سخت می‌شود.

دقت کنید، ارائه ظاهر موبایل گونه در یک برنامه تجاری (نه یک سایت خبری! )، فقط کوچک شدن کنترل‌ها و حذف آیتم‌های سنگین نیست.

ممکن است در کامپیوتر، شما دیتاگرید مشتریان را نمایش دهید، که در پایین خریدهایش در یک دیتاگرید مجزا نمایش داده شود، در حالی که برای موبایل ممکن است برای استفاده بیشتر از فضا، دیتاگرید مشتریان را به کمبو باکس تغییر دهید، منطق همان است، فقط ظاهر عوض شده است.

۲- امکان نوشتن تست‌های خودکار برای قطعه کد بالا متصور نیست.

۳- مدیریت وضعیت ظاهر برنامه از حالت Declarative خارج خواهد شد، در یک مثال بسیار ابتدایی، شما می‌توانید یک Label داشته باشید برای «نام کامل»، که مقدار دو ویژگی مربوط به نام و نام خانوادگی را با یک حرف فاصله به همدیگر بچسباند، در روش درست فقط در نیم خط آنرا تعریف می‌کنید.

[?](http://dnpars.ir/blog/?p=108)

|  |  |
| --- | --- |
| Data-bind=”value=’FName’ + ‘ ‘ + ‘LName’” | 1 |

این یک تعریف Declarative است، با هرگونه تغییر در مقدار نام و یا نام خانوادگی، نوشته به صورت خودکار به روز می‌شود. (اولین مثال MVVM ای که پیاده سازی خواهیم کرد)

در روش برنامه نویسی اشتباه شما این امکان را ندارید.

۴- محرومیت از امکانات اساسی WPF و Silverlight و Win RT که اساسا مبتنی بر روش خوب هستند.

ممکن است این سوال مطرح شود که MVC و MVVM چه تفاوت‌هایی دارند؟ چه شباهت‌هایی دارند؟ و کجا بهتر است از کدام استفاده کنیم؟ چه ابزارهایی بهتر است استفاده شود؟ Best Practice ها چیست؟ و …

به همه این موارد خواهیم پرداخت.

تکنولوژی‌هایی که برای ساخت برنامه استفاده خواهد شد :

برای سمت سرور، مثال‌ها با ASP.NET Web API خواهد بود، و همچنین مثال برای درک بهتر تفاوتهای [Web API](http://www.asp.net/web-api) و WCF در سری دوم با [WCF Data Services](http://msdn.microsoft.com/en-us/data/odata.aspx) نوشته خواهد شد.

همچنین از Signal R برای مدیریت ارتباطات دو طرفه در برنامه استفاده خواهد شد.

و در سمت کلاینت از HTML و [MVVM Knockout](http://knockoutjs.com/) استفاده خواهد شد.

و در نهایت برای درک بهتر امکانات پیشرفته MVVM، مثال را با Silverlight پیاده سازی می‌کنیم، تا مفاهیمی مانند Data Trigger و … را که در حال حاضر فقط در Silverlight و WPF هستند را نیز به شما آموزش دهیم.

بدیهی است، دیر یا زود این امکانات در پلتفرم جدید مایکروسافت، یعنی Win RT نیز پیاده سازی خواهند شد.

برای تست خودکار از [qunit.js](http://qunitjs.com/) در جاوا اسکریپت، و برای Dependency Injection از [require.js](http://requirejs.org/) استفاده خواهیم کرد.

در نهایت این که، روز اول که یه پروژه بزرگ داره شروع می شه، خیلی راحت فکر می کنیم با $ (jQuery)، می

شه همه کارها رو انجام داد، ولی خیلی زود متوجه می شیم

اگه بخواهیم افسار پروژه از دستمون خارج نشه باید یه مقدار بهتر کدنویسی کنیم.

مطالب این قسمت ها در راستای رسیدن به Single Page Application است.

موفق و پایدار باشید.

From <<http://dnpars.ir/blog/?p=108>>

[تاريخ:۳ آبان ۱۳۹۱] ~ [Java Script](http://dnpars.ir/blog/?cat=21)،[MVVM](http://dnpars.ir/blog/?cat=20)،[N-Tier](http://dnpars.ir/blog/?cat=11)،[Silverlight](http://dnpars.ir/blog/?cat=22)،[Single Page Application](http://dnpars.ir/blog/?cat=7)،[مطلب](http://dnpars.ir/blog/?cat=6)

در کجاها باید از MVVM استفاده کنیم ؟

MVVM یکی از بهترین روش‌های برنامه نویسی Desktop Application ها است.

مسلما با خواندن همین جمله بسیاری از افراد خواهند گفت که ما می‌خواهیم برنامه وبی بنویسیم، پس این آیتم به کار ما نمی‌آید.

برنامه های Desktop ای (فارغ از تکنولوژی)، برنامه‌هایی هستند، که قسمت قابل توجهی از کارها را در سمت سیستم مشتری یا کلاینت انجام می‌دهند، که از بارزترین ویژگی‌های آن‌ها پردازش

ظاهربرنامه در سمت کلاینت است.

در واقع با گرفتن اطلاعات ۱۰ عدد از محصولات (در یک سیستم خرید و فروش)، در سمت کلاینت اقدام به ساختن یک دیتاگرید برای نمایش آن می‌کنند.

برنامه های وبی، برنامه‌هایی هستند که عمده پردازش آن‌ها در سمت وب است، در همان مثال ۱۰ محصول، سرور هم ۱۰ محصول را می‌خواند، و هم دیتاگرید لازم برای نمایش آنرا در سمت

سرورمی‌سازد و آنرا آماده برای نمایش به کلاینت تحویل می‌دهد (برای مثال به صورت HTML )

در واقع در مثال ۱۰ محصول، اگر ما بین کلاینت و سرور بشینیم و اطلاعات رد و بدل شده را ببینیم، در یک برنامه دسکتاپی، فقط رد و بدل شدن مشخصات ۱۰ محصول را می بینیم، ولی در یک برنامه

وبی، علاوه بر اطلاعات مربوط به ده محصول، رد و بدل شدن <tr> و <td> و … ( اطلاعات لازم برای ساخته شدن فرم ) را می بینیم که در کلاینت برای نمایش استفاده می شوند.

البته تفاوت‌های دیگری هم وجود دارد، ولی نحوه ساخته شدن ظاهر برنامه و این که قسمت‌های عمده آن در سرور ساخته می‌شوند یا در کلاینت، یکی از بهترین ملاک‌های تشخیص است.

البته همیشه امکان دارد قسمت‌هایی از برنامه وبی در سمت سیستم مشتری ساخته شود، و قسمت‌هایی از سیستم دسکتاپی در سمت سرور ساخته شود، مهم این است که اکثریت کدام است.

با توجه به این تفاسیر، Windows Forms، WPF، Silverlight، Android، iOS، Windows Phone، Windows Mobile، Flash، Shockwave و سایر تکنولوژی‌های مشابه دسکتاپی هستند، فارغ از این که در

یک شبکه محلی کار کنند، یا روی بستر اینترنت به یکدیگر متصل باشند، و همچنین فارغ از این که در پنجره خودشان دیده شوند، یا از طریق مرورگر Browser

هستند افرادی که وقتی یک برنامه Silverlight ای را در مرورگر Google Chrome می‌بینند، می‌گویند چه برنامه وبی عالی‌ای.

از آن طرف، ASP.NET Web Form، ASP.NET MVC، php، و … نیز برنامه های وبی هستند.

برنامه های دسکتاپی عموما تجربه کاری خوش آیندی را در صورت طراحی درست در اختیار مشتری قرار می‌دهند، سرعت پیاده سازی و توسعه بالاتری را با هزینه کمتری فراهم می‌کنند و نیاز به

دانشکمتری برای توسعه نرم افزار دارند.

مزیت برنامه های وبی، اجرا شدن آن‌ها در هر جایی از دنیا است، چرا که خروجی آن‌ها مبتنی بر HTML و Java Script است که تقریبا در هر جایی قابل اجرا است، لکن در مقایسه با برنامه های

دسکتاپی سختی و هزینه پیاده سازی بیشتری را دارند.

از برنامه های دسکتاپی عموما برای نوشتن برنامه های سازمانی استفاده می‌شود، و از برنامه های وبی برای ساخت برنامه‌هایی استفاده می‌شود که باید استفاده عمومی داشته باشند، مانند

سیستم فروش بلیط سینما، استعلام بیمه نامه، خبرگزاری‌ها و …

برای توضیحات بیشتر می‌توانید به [مطلب آقای وحید نصیری](http://dotnettips.info/Post/461/%DA%86%D9%87-%D8%B2%D9%85%D8%A7%D9%86%DB%8C-%D8%A8%D9%87%D8%AA%D8%B1-%D8%A7%D8%B3%D8%AA-%D8%A7%D8%B2-silverlight-%D8%A7%D8%B3%D8%AA%D9%81%D8%A7%D8%AF%D9%87-%D8%B4%D9%88%D8%AF) در رابطه با این که کجاها باید از سیلورلایت استفاده کرد مراجعه کنید (یک نمونه برنامه دسکتاپی)

در طی سال‌های اخیر، تعدادی از گروه‌ها و شرکت‌ها تلاش کرده‌اند، تا به ایجاد سیستم‌های دسکتاپ، مبتنی بر HTML و Java Script روی آورده‌اند، که از بهترین کارهای آنان Single Page Application

شرکت مایکروسافت است.

مزیت این نوع برنامه نویسی، ارائه مزیت برنامه نویسی وبی (اجرا در هر جایی)، و مزیت سیستم‌های دسکتاپی یعنی User Friendly بودن آن است.

برای نمونه به سایت آقای [John Papa](http://johnpapa.aspnet45.cytanium.com/) بروید، و مشاهده کنید چه قدر یک برنامه می‌تواند عالی طراحی شود.

Single Page Application، یک برنامه دسکتاپی است، که به واسطه پیاده سازی آن بر روی HTML و Java Script، تقریبا روی تمامی سیستم‌های موجود در دنیا قابلیت اجرا را دارد، چه یک Tablet

اندرویدی، و چه Mac Book و چه ویندوز ۸ و یا لینوکس و…

حال به ابتدای پست برگردیم، MVVM یکی از بهترین روش‌ها، برای پیاده سازی برنامه های دسکتاپی است.

در پست بعدی به بررسی MVVM می‌پردازیم، و بررسی‌ای را نیز بر روی MVC خواهیم داشت.

From <<http://dnpars.ir/blog/?p=125>>

[تاريخ:۵ آبان ۱۳۹۱] ~ [Java Script](http://dnpars.ir/blog/?cat=21)،[MVVM](http://dnpars.ir/blog/?cat=20)،[Silverlight](http://dnpars.ir/blog/?cat=22)،[مطلب](http://dnpars.ir/blog/?cat=6)

در این قسمت می خواهیم به بررسی دقیق تر MVVM بپردازیم

MVVM مخفف **Model** – **View** – **View Model** است

—

**Model** به مشتری، محصول، سفارش و … که در سیستم شما جزء موجودیت های شما هستند اشاره دارد.

مدل فقط شامل ویژگی های مربوط به موجودیت است، برای مثال نام، نام خانوادگی و …

به این که کجا قرار است نمایش داده شود کاری ندارد.

—

**View Model** که بر پایه دو امکان Binding و Commanding ایجاد شده است.

Binding به اتصال دو آیتم به یکدیگر Binding می گویند، برای مثال اتصال لیستی از مشتریان به دیتاگرید مربوطه، یا به کمبو باکس یا هر چی

شما می توانید هر چیزی را Bind کنید ( چه در Java Script و HTML و چه در C# , XAML )

و Commanding به دستوری گفته می شود که برایش اهمیتی ندارد توسط منو فراخوانی شود، یا توسط دکمه یا هر چیز دیگری

در واقع شما کدنویسی خودتان را فارغ از ظاهر برنامه انجام می دهید، ( یعنی در #C یا JS )

—

**View** نیز به ظاهر برنامه اشاره دارد، و شامل Text Box ای برای نمایش نام است و … ، و در نهایت ممکن است اقلامی مانند Animation و Effect نیز داشته باشد، ولی ابدا درگیر منطق نمی شود.

View وظیفه نمایش دکمه ها و منو ها را برای Command های View Model دارد و همچنین ارائه Check Box ، Data Grid ،List و … را برای Binding های View Model

—

چندین View می توانند از یک View Model استفاده کنند، یا در یک View از چندین View Model استفاده شود، اصلا محدودیتی وجود ندارد.

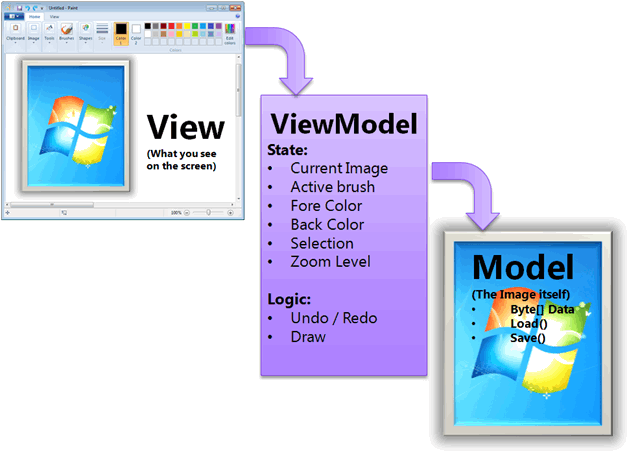
**مزیت ها**

1. جداسازی قسمت های متفاوت پروژه از یکدیگر، امکان کار تیمی، برای مثال یک نفر Java Script یا #C مربوط به منطق را در View Model می نویسد، بدون این که به View کاری داشته باشد، و ممکن است View توسط فردی ایجاد شود که بر روی مسائل ظاهری مسلط تر است
2. امکان نوشتن Unit Test برای View Model به صورت آسانتر، Unit Test نوعی تست خودکار است که برای هر Unit از برنامه شما ایجاد می شود، و با انجام دادن تست های خودکار با فواصل کوتاه ( برای مثال ۵ دقیقه )، برنامه شما عملا زیر تست دائم و خودکار خواهد بود، و به محض نوشتن کدی که قسمت، یا قسمت هایی از برنامه را از کار بیاندازد، شما متوجه اشکال در کارتان خواهید شد.
3. امکان ارائه چندین View برای یک View Model ( لطفا برای اطلاعات بیشتر به پست [کتاب رایگان Telerik](http://dnpars.ir/blog/?p=119) مراجعه کنید تا متوجه زیبایی کار شوید )
4. توسعه و نگهداری راحت تر نرم افزارها ( به خصوص در Java Script و HTML )

**باید ها و نباید ها**

1. طراحی View باید Visual باشد، شامل ظاهر و نهایتا تعدادی Animation و …، نباید منطقی در آن لحاظ شود
2. طرای View Model باید Logical باشد، یعنی فقط روی منطق تمرکز شود و مسائل ظاهری برنامه در آن دیده نشود
3. کد View Model باید تست پذیری خودکار داشته باشد ( در بحث تست خودکار به معیارهای آن خواهیم پرداخت )
4. نگهداری State بر عهده View Model است، و View وظیفه ای در این رابطه ندارد، برای مثال در یک View مربوط به “فرم جستجو”، نگهداری کلمه ای که اپراتور می خواهد روی آن جستجو کند، و تنظیمات جستجو به عهده View Model است، و View تنها Text Box و Check Box و سایر آیتم هایش را به View Model متصل می کند ( با استفاده از Binding ) و دکمه جستجو موجود در View به Command جستجو در View Model متصل است.

در نهایت چنین آیتمی مد نظر است :



**چارچوب های مبتنی بر MVVM**

چارچوب های زیادی بر روی MVVM توسعه یافته اند، که در اینجا ما دو تا از آنها را بررسی می کنیم

۱- MVVM Light Toolkit

2- Knockout.js

[MVVM Light Toolkit](http://mvvmlight.codeplex.com/)، یکی از بهترین فریم ورک های مبتنی بر MVVM است که بر روی WPF، Silverlight، Win-RT، Windows Phone کار می کند.

سادگی کار و امکانات گسترده آن، آن را به گزینه ای محبوب در میان برنامه نویسان حرفه ای بدل کرده است.

[Knockout.js](http://knockoutjs.com/)، یکی از بهترین فریم ورک های مبتنی بر MVVM است که بر روی Java Script و HTML کار می کند

مزیت knockout.js در عدم وابسته بودن آن به سایر فریم ورک های جاوااسکریپتی است، و همچنین حجم بسیار کمی دارد (۱۴ کیلوبایت)، و تمامی مرورگرهای متداول را تحت پوشش خود دارد ( حتی IE 6 )، و دارای مکانیزم اعتبارسنجی داخلی و بسیاری امکانات دیگر است.

همچنین این فریم ورک با سایر فریم ورک های View، مانند Kendo UI و … سازگاری کامل دارد.

همانطور که گفته بودیم، برای پیاده سازی مثال برنامه Chat مان از knockout.js استفاده خواهیم کرد، و سپس برای درک امکاناتی که در Silverlight هست، ولی در جاوا اسکریپت نیست، مثال را با Silverlight بازنویسی خواهیم کرد.

برای نوشتن مثال های مان و نمایش آنها از سایت زیر بهره خواهیم برد

<http://jsfiddle.net/>

در این سایت شما می توانید به فریم ورک های جاوااسکریپتی مد نظرتان Reference بزنید، Java Script، CSS و HTML تان را بنویسید، و آنلاین خروجی را مشاهده کنید و لینک آنرا در اختیار دوستانتان قرار دهید.

در قسمت Choose Framework مورد jQuery 1.8 را انتخاب کنید

و در قسمت Add Resources آدرس زیر را اضافه کنید :

<https://github.com/downloads/SteveSanderson/knockout/knockout-2.1.0.debug.js>

این نسخه از Knockout که نسخه ۲٫۱ است، برای برنامه نویسی ایجاد شده است، و حجم نسبتا بالایی دارد، و مزیت آن ارائه خطاهای قابل فهم و عیب یابی آسان کدهای خوانا است، ولی این نسخه فقط برای برنامه نویسان است، برای مشتری نسخه عادی ارسال می شود.

در قسمت بعد به بررسی Binding در Knockout می پردازیم، و سپس به یک نمونه ساده MVVM ای را پیاده سازی می کنیم

موفق و پایدار باشید

From <<http://dnpars.ir/blog/?p=130>>

[تاريخ:۵ آبان ۱۳۹۱] ~ [Java Script](http://dnpars.ir/blog/?cat=21)،[MVVM](http://dnpars.ir/blog/?cat=20)،[مطلب](http://dnpars.ir/blog/?cat=6)

همانطور که در قسمت های قبلی گفتیم، پیش نیاز نوشتن برنامه های مبتنی بر MVVM، تسلط بر Binding و Commanding است.

در این قسمت می خواهیم اولین کد Binding خود را بنویسیم

نمونه ساده برنامه ما، یک مدل است، با نام Person، که دارای نام، و نام خانوادگی است. همچنین دارای یک ویژگی محاسباتی با نام “نام کامل” نیز هست، که با فرمول زیر محاسبه می شود

نام + ” ” + نام خانوادگی

می خواهیم دو Text Box برای نام و نام خانوادگی داشته باشیم،و یک Label برای نمایش نام کامل قرار داهیم

برای این کار ابتدا مدل مان را که Person است تعریف می کنیم

[?](http://dnpars.ir/blog/?p=141)

|  |  |
| --- | --- |
| function Person = ()  {   this.firstName = ko.observable("MVVM");   this.lastName = ko.observable("Knockout");  }; | 1  2  3  4  5 |

همان طور که می بینید، firstName و lastName از نوعی تعریف شده اند با نام ko.observable

جاوااسکریپت از Property پشتیبانی می کند، منتها برخی مرورگرها ( IE های نسخه ۶ و ۷ )، در استفاده از آن مشکل دارند.

استفاده از فیلد هم برای Binding متصور نیست، زیرا هیچ سیستم رهگیری تغییرات ( Change Tracking )، برای فیلد تعبیه نشده است،

و Binding کلا بر روی Change Tracking استوار است، در همین مثال ما، هر گونه تغییر نام و یا نام خانوادگی باید، Label مربوط به “نام کامل” را به روزرسانی کند.

این جا است، که knockout بر اساس Design Pattern ای با نام Observer، اقدام به حل مشکل به روشی نمود که بر روی تمامی مرورگرها ( از جمله IE 6 ) جواب دهد.

نکته دوم این است، که اگر شما اطلاعات مربوط به Person را از سرور به صورت json گرفته اید، هیچ نگران نباشید، چون با یک خط کد می توانید json را به Person مربوطه

Map نمایید که در فرصت مقتضی به آن خواهیم پرداخت.

هم اکنون نوبت به نام کامل می رسد ، که محاسباتی است، برای آن داریم :

[?](http://dnpars.ir/blog/?p=141)

|  |  |
| --- | --- |
| this.fullName = ko.computed(function() {      return this.firstName() + " " + this.lastName();  }, this); | 1  2  3 |

شاید فرمت نوشتن fullName محاسباتی کمی سخت به نظر بیاید، اما به زودی ابزاری را به شما معرفی خواهم کرد، که از روی Person ای که شما آنرا برای #C نوشته اید، معادل Java Script ای آنرا بسازد.

حال نوبت به فرمی برای نمایش Person می رسد.

در فرم مربوطه دو عدد Text Box می خواهیم و یک عدد Label

برای این کار در HTML داریم :

[?](http://dnpars.ir/blog/?p=141)

|  |  |
| --- | --- |
| First name: <input data-bind="value: firstName" />  <br />  Last name: <input data-bind="value: lastName" />  <br />  Full name: <strong data-bind="text: fullName" /> | 1  2  3  4  5 |

همانطور که می بینید از ویژگی ای استفاده شده است با نام data-bind، که برای binding به کار می رود ( در این مثال text و value ها را Bind کرده ایم )

چه ویژگی هایی پشتیبانی می شود ؟ تمامی ویژگی ها پوشش داده می شود، حتی می توانید عمودی یا افقی بودن منو را هم Bind کنید

چه کنترل هایی پشتیبانی می شوند ؟ عمده کنترل ها، مانند jQuery UI، [Kendo UI](http://dnpars.ir/blog/demos.kendoui.com/web/mvvm/index.html) و [Wijmo](http://wijmo.com/widgets/mvvm-support/) با MVVM Knockout کار می کنند

اصولا هم Kendo UI را توصیه می کنم، چون در این جا ( یعنی MVVM ) و در قسمت های دیگر مربوط به Single Page Application به ما پوشش خوبی می دهد ( البته اگر مشکلات مربوط به زبان های راست به چپ خود را به صورت صد در صدی حل کند )

در نهایت در Form Load برنامه برای فعال شدن برنامه داریم

[?](http://dnpars.ir/blog/?p=141)

|  |  |
| --- | --- |
| ko.applyBindings(new Person()); | 1 |

**برای دیدن کل سورس کد این قسمت و مشاهده نتیجه به** [**این جا**](http://jsfiddle.net/dnpras/9rcVG/) **بروید**

در قسمت بعدی، Binding های پیشرفته تر را بررسی می کنیم

From <<http://dnpars.ir/blog/?p=141>>

[تاريخ:۶ آبان ۱۳۹۱] ~ [Java Script](http://dnpars.ir/blog/?cat=21)،[MVVM](http://dnpars.ir/blog/?cat=20)

در قسمت‌های قبلی آموختیم، که می‌توان ویژگی‌های با قابلیت Bind تعریف کرد (با استفاده از ko.observable )، و سپس به بررسی ویژگی‌های محاسباتی پرداختیم (با استفاده از Computed ).

حال شاید کسی در مثال قبلی بگوید من می‌خواهم به محض تغییر نام و یا نام خانوادگی و به ازای هر بار تایپ شدن یک کارکتر جدید، Binding من به روز شود.

به صورت پیش فرض Binding پس از رفتن از یک کنترل به یک کنترل دیگر به روز می‌شود.

برای انجام این کار داریم در View داریم :

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| First name: <input data-bind="value: firstName, valueUpdate: 'afterkeydown'" />  <br /> | 1  2 |

در این حالت به محض تغییر نام، نام کامل که محاسباتی است نیز به روز خوهد شد.

حال فرض کنید، یک ویژگی نیز با نام Is Activated به Person اضافه کرده‌ایم، تا مادامی که Person ما Is Activated اش False باشد، تمامی کنترل‌های رو صفحه باید Disable شوند، و بالعکس زمانی که Is Enabled برابر True قرار گیرد.

برای این کار داریم :

۱- تعریف is Activated

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| this.isActivated = ko.observable(true); | 1 |

2- سپس enable بودن کنترل هایمان را به isActivated بایند می کنیم

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| Last name: <input data-bind="value: lastName,enable: isActivated" /> | 1 |

حال برای Person یک ویژگی قرار می‌دهیم به نام رتبه، رتبه Person ما باید مقداری بین ۱ الی ۵ باشید.

حال می‌خواهیم یک دکمه قرار دهیم، که با هر بار کلیک بر روی آن، مقدار رتبه یک عدد افزایش پیدا کند.

افزایش رتبه می‌شود کار دکمه ما، و این کار یک شرط دارد، رتبه نباید بیشتر از ۵ شود.

در MVVM ویژگی‌ای وجود دارد به نام Command. یک Command از دو قسمت تشکیل شده است، execute و can execute، execute به کاری که باید انجام شود اشاره می‌کند، و can execute امکان اجرای آن را توضیح می‌دهد.

در WPF، Silverlight، Win RT و Windows Phone، یک interface به وجود دارد، با نام ICommand که مسئول این دو خاصیت execute و can execute است.

شما یک Command را در View Model تعریف می‌کنید، و سپس کلیک یک دکمه را (یا رویداد هر کنترل دیگری را)، را به آن وصل می‌کنید.

دکمه تا مادامی که can execute مقدار False را برگرداند، غیر فعال می‌ماند و با کلیک بر روی آن در مواقعی که فعال است، execute اجرا می‌شود.

برای تعریف یک Command در MVVM Knockout با اندکی تفاوت داریم :

۱- ابتدا تعریف rate :

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| this.rate = ko.observable(1); | 1 |

سپس داریم تعریف execute

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| this.increaseRate = function() {       this.rate(this.rate() + ۱);  }; | 1  2  3 |

همانطور که می‌بینید، هیچ وابستگی به View در کدنویسی View Model وجود ندارد، و همین عامل کدنویسی آنرا بسیار آسان می‌سازد.

سپس دکمه را به روش زیر به آن execute متصل می کنیم.

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| <button data-bind="click: increaseRate">Increse Rate</button> | 1 |

می توان هر رویداد دیگری را به آن execute متصل نمود، و یا حتی Hotkey قرار داد، که در مثال های دیگر به آنها می رسیم.

برای can execute دو راه داریم (البته راه های دیگری نیز وجود دارد)، اول این که یک Computed را پیاده سازی کنیم به شرح زیر

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| this.canIncreaseRate = ko.computed(function(){       return this.isActivated() && this.rate() < ۵;  } ,this); | 1  2  3 |

همانطور که می بینید، این جا می توانید خیلی با دست باز حرکت کنید، و شرط را برابر آن قرار دهید که هم Person ما Active باشد، و هم Rate آن، کمتر از ۵ باشد

سپس enabled بودن دکمه را به آن وصل می کنیم.

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| <button data-bind="click: increaseRate, enable: canIncreaseRate">Increse Rate</button> | 1 |

و یا در View، بر روی دکمه این کد را بنویسیم ( یعنی از Computed استفاده نکنیم )

[?](http://dnpars.ir/blog/?p=165)

|  |  |
| --- | --- |
| <button data-bind="click: increaseRate, enable: (rate() < 5 && isActivated) ">Another Increse Rate</button> | 1 |

همانطور که می‌بینید، می‌توانید قسمتی از مسائل Binding را در View مدیریت کنید، که به آن Value Convertor گفته می‌شود.

در این جا مقدار رتبه Person که عددی است، با یک if به Boolean ای تبدیل شده است، که Is Enabled بودن یا نبودن دکمه را مشخص می‌کند.

بدیهی است، همیشه Binding مستقیم نیست، و بعضی تبدیلات نیز نیاز است.

با این روش شما می‌توانید هر چیزی را به هر چیز دیگری Bind کنید.

در قسمت بعدی به Binding مجموعه‌ها می‌پردازیم، تا پس از آن بتوانیم به مسائل حرفه ای تر همچون ارتباط با سرور و ذخیره و بازیابی اطلاعات بپردازیم.

[**برای دیدن سورس تا این لحظه کلیک کنید**](http://jsfiddle.net/dnpras/Twfaa/5/)

موفق و پایدار باشید

From <<http://dnpars.ir/blog/?p=165>>