

<anglebrackets/>

<anglebrackets/>

POSTCON08: Building Data-Centric Single Page Applications with Durandal, Knockout, Breeze and Web API

Brian Noyes

CTO, Solliance (www.solliance.net)

brian.noyes@solliance.net, @briannoyes

<anglebrackets/>

About Brian Noyes

Solliance (www.solliance.net)
CTO


Microsoft Regional Director

Microsoft MVP

Pluralsight author
www.pluralsight.com

Web API Insider, Windows Azure Insider,
Windows Store App Insider, C#/VB Insider



 brian.noyes@solliance.net

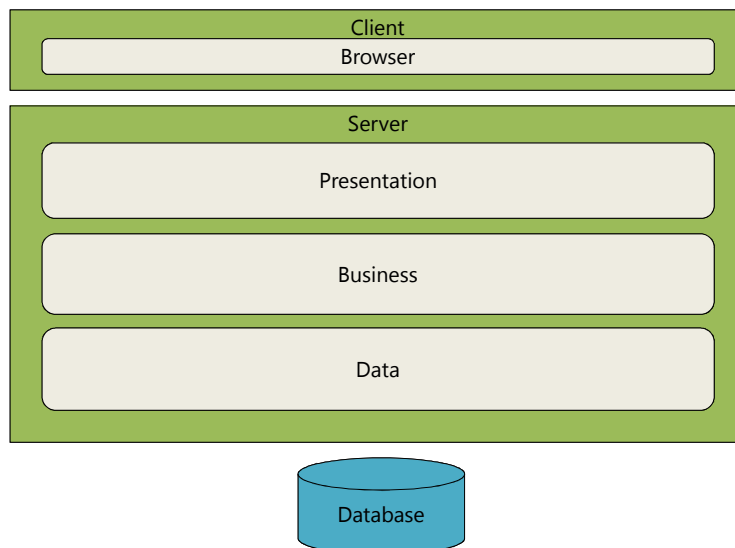
 [@briannoyes](https://twitter.com/briannoyes)

 <http://briannoyes.net>

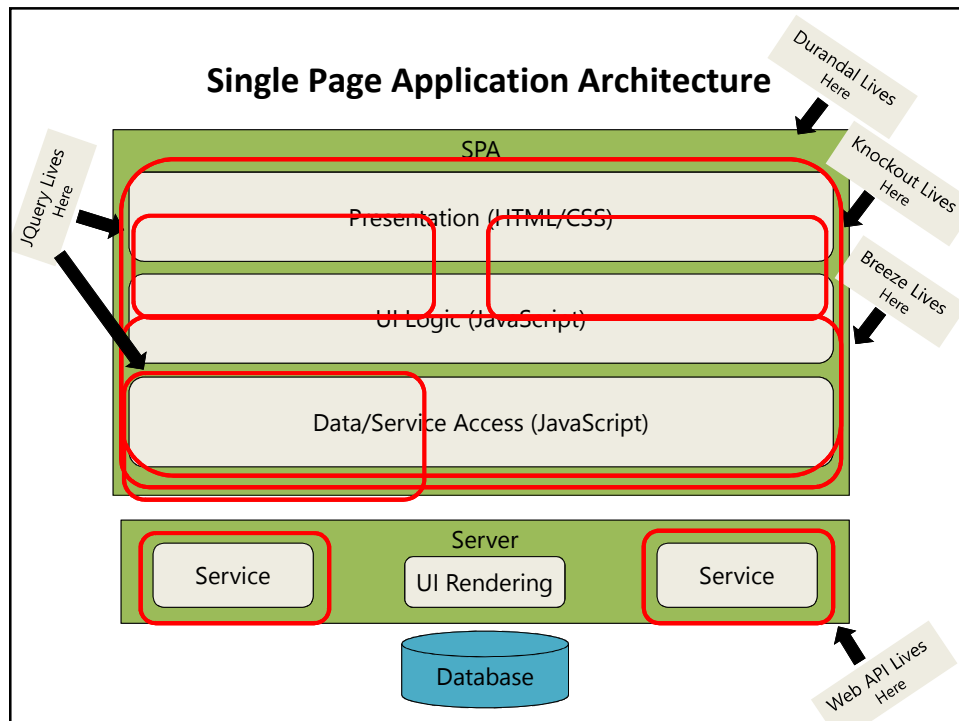
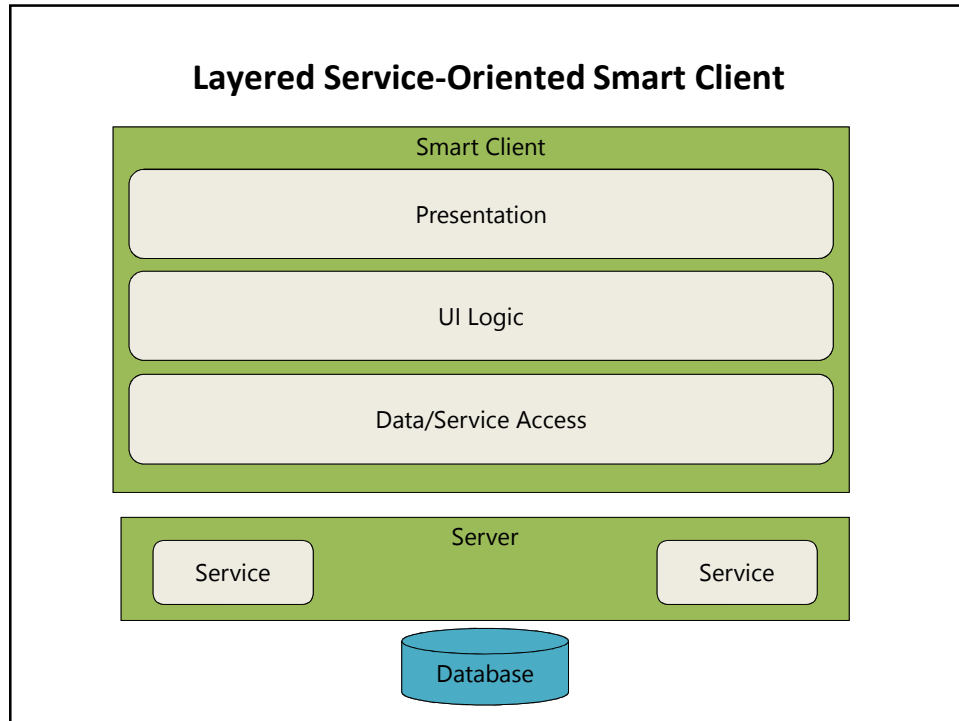
Agenda

- **HTML / SPA client architecture**
- **JavaScript libraries / frameworks**
- **JQuery**
- **Knockout**
- **ASP.NET Web API**
- **Breeze**
- **Durandal**

Browser App Architecture



<anglebrackets/>



Single Page Applications (SPAs)

- **Web pages**
 - Do not have to be the entire “application”
- **More user interactivity than scrolling or filling in a few fields and submitting**
- **Alternative to server post-back model for web page development**
- **Evolution driven by AJAX + maturing of JavaScript execution and libraries**
- **Can be built with any HTML technology stack**
- **Can be deployed to a web site**
- **Can be packaged as a mobile application**
 - Cordova / PhoneGap / Icenium / DXtreme

SPA / HTML Architecture

- **HTML is just structure of the view**
- **CSS drives appearance of the structural elements**
- **JavaScript for all the client side logic**

Keeping SPAs Maintainable

- Separation of concerns
- Layered architecture
- UI separation patterns
- Modular JavaScript

HTML Browser Clients

- Could be straight HTML
 - Could be ASP.NET MVC
 - Could be ASP.NET Web Forms
 - Could be JSP
-
- For this workshop – it all produces client side HTML / CSS / JS rendered from a web server

HTML Mobile Apps

- **Could be mobile web pages**
- **Could be packaged mobile app**
 - Developed as a SPA
 - Packaged with Cordova or derivatives
 - Deployed through an app store
 - Can access native features of the device / platform

Agenda

- **HTML / SPA client architecture**
- **JavaScript libraries / frameworks**
- **JQuery**
- **Knockout**
- **ASP.NET Web API**
- **Breeze**
- **Durandal**

JavaScript Libraries / Frameworks

- **One of the biggest challenges in HTML client development today**
 - TOO MANY CHOICES
- **Lots of little libraries that do one thing**
- **Several frameworks that drive the structure and patterns of your app and do lots of things**
- **One big framework vs composition of libraries to form a framework**

JavaScript Libraries / Frameworks

- **JQuery**
 - DOM manipulation and Web API service calls
- **Knockout**
 - Data binding and observables
- **Breeze**
 - CRUD data service calls, change tracking, validation
- **Twitter Bootstrap**
 - CSS styling and widgets
- **Durandal**
 - Full JavaScript application framework
 - Ties together JQuery, Knockout, and Require.js
 - Adds many additional capabilities

JavaScript Libraries / Frameworks

- **What about?**
 - Angular
 - Ember
 - Backbone
 - Foundation
 - etc
- **Alternative approaches**
- **If you learn one, it is easy to switch to another**
 - From a skills perspective
- **Have to decide which you like best**
 - But learning the architecture and the approach is more important than the low level syntax

Apples to Apples

- **Durandal vs Angular**
 - Not Knockout vs Angular
 - Data binding
 - Dependency injection
 - MV* composition
 - Navigation
 - Templating
 - Animation/transitions
 - Plug-ins
- **Either stack composes well with Breeze and JQuery**

What about TypeScript?

- **Learn fundamentals of framework / library in JavaScript**
- **Then leverage TypeScript support, if available**
 - Better productivity
 - Better maintainability
 - For some...

Valuable Development Tools

- NuGet
- Chrome / IE developer tools (F12)
- Fiddler
- Postman
- Knockout Context Debugger
- jsFiddle
- Web Essentials
- CodeRush / Resharper
- Productivity Power Tools – 2012
- SublimeText / WebStorm
- Grunt / Gulp / Mimosa
- QUnit / Jasmine / Mocha / Sinon

Agenda

- HTML / SPA client architecture
- JavaScript libraries / frameworks
- **JQuery**
- Knockout
- ASP.NET Web API
- Breeze
- Durandal

JQuery

- Great library for rich DOM manipulation
- Normalizes the API for working with the DOM on many browsers
- Widespread adoption / lots of resources
- Depended on by many libraries
- Becomes significantly less important when using a data binding JS library like Knockout or Angular
 - DOM manipulation should be rare and encapsulated in custom bindings / directives

JQuery Usage in SPA

- **Raw Web API service calls (AJAX)**
 - Breeze can take care of CRUD service calls
- **Animations**
- **Occasional workarounds for complex interaction scenarios**
- **Both Durandal and Angular work with JQuery**
 - Durandal requires
 - Angular can use its own jqLite language if not present

Agenda

- **HTML / SPA client architecture**
- **JavaScript libraries / frameworks**
- **JQuery**
- **Knockout**
- **ASP.NET Web API**
- **Breeze**
- **Durandal**

Knockout Overview

- **Data binding**
 - As a means for separation of concerns
- **Observables**
 - Objects that raise events when their properties change
 - Like INotifyPropertyChanged or DependencyProperties in .NET
- **UI Templating**

Knockout Overview

- **Works on any mainstream browser**



- **Fairly lightweight**
 - 13kb gzip
- **No dependencies on other libraries**

A World With No Data Binding

- Logic code pushes discrete values from data object properties into UI element properties
- Logic code pulls modified values out of UI element properties and puts them into data object properties
- Need explicit triggers for when to push and pull
- JQuery works well for this task

Data binding

- **Declarative approach**
- **Associates UI element properties with data object properties**
- **Two-way**
 - Automatically retrieves data object properties into element for presentation
 - Automatically pushes changed values in element into underlying data object
- **Relies on property change notifications to keep UI in sync**
 - Observables

Model-View-ViewModel (MVVM)

- UI separation pattern
- Derivative from MVC and MVP
- Designed to fully leverage data binding technologies
- Initially developed for XAML apps

MVVM Principals

- **Model (JS)**
 - Data structures and logic to support the presentation
- **View (HTML / CSS)**
 - Just the structure of what the user sees on the screen
- **ViewModel (JS)**
 - Provides data to the view for binding / presentation
 - Interaction logic

Knockout Basics

- JavaScript object to bind to

```
var customer = {  
  Name: "Brian"  
};
```

- Data binding attribute on element

```
<input type="text" data-bind="value: Name"/>
```

- applyBindings to marry them together

```
ko.applyBindings(customer);
```

Binding syntax

- name: value
- Can include more than one, comma separated
- value =
 - Variable, literal, expression
 - Variables and expressions based on the current data context

Observables

- **Bound object properties can change:**
 - Because of input from bound control
 - Because JS logic sets value
- **Observables make sure bound elements update when the value changes**
- **Can be subscribed to in JS code for reacting to changes**

Observable syntax

- **Observables are function values**

```
function customer() {  
    this.FirstName = ko.observable();  
    this.LastName = ko.observable();  
    this.Orders = ko.observableArray([]);  
};
```

- **Get/Set through function calls**

```
var cust = new customer();  
var custName = cust.FirstName(); // Not  
cust.FirstName  
cust.FirstName("Brian"); // Not cust.FirstName =  
"Brian"
```

- **Empty observables are "undefined"**

Computed Observables

- Allows you to compose a new observable from an expression containing one or more other observables
- Computed observable changes any time one of the contained observables change

```
function customer() {  
  this.FirstName = ko.observable();  
  this.LastName = ko.observable();  
  this.Orders = ko.observableArray([]);  
  this.FullName = ko.computed(function () {  
    return this.FirstName() + " " + this.LastName();  
  }, this);  
};
```

Knockout Bindings

- Text and appearance
- Control flow
- Form data

Custom Bindings

- **Define off of ko.bindingHandlers**
- **Object literal containing init and update functions**
 - Each passed five (optional) parameters:
 - element
 - valueAccessor
 - allBindingsAccessor
 - viewModel
 - bindingContext
- **init called when binding discovered**
- **update called on discovery and every time source object changes**

Binding Data Context

- **Top level: what is passed to applyBindings**
- **Hierarchical:**
 - Current context can be set to child objects through “foreach” or “with” binding
- **Accessing scope:**
 - \$data – current
 - \$parent – direct parent context
 - \$parents – flattened list of all parents up the chain
 - \$root – context set by applyBindings

Templates

- Predefined fragments of HTML
- Presented dynamically based on control flow bindings
- Named / unnamed
- Script or div
- <!-- ko --> containerless templates/bindings

Agenda

- HTML / SPA client architecture
- JavaScript libraries / frameworks
- JQuery
- Knockout
- **ASP.NET Web API**
- Breeze
- Durandal

ASP.NET Web API Overview

- **New platform for building HTTP web services (Web APIs)**
- **Built on top of ASP.NET MVC 4 framework**
 - Released with .NET 4.5
 - Compatible with .NET 4.0
 - Web API 2 released with Visual Studio 2013 / .NET 4.5.1
- **Makes it easy to build services for consumption from multi-platform clients**
 - Simple RPC services
 - CRUD services
 - REST services
 - OData services

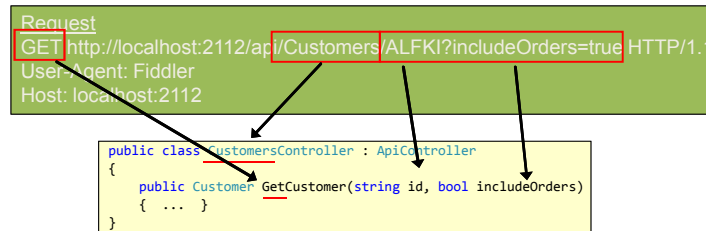
ASP.NET Web API Overview

- **Services are Controllers**
 - ApiController class
- **Leverages MVC features**
 - Routing
 - Model binding
 - Action filters

ASP.NET Web API Overview

Convention over configuration

- Maps URIs to controllers
- Maps HTTP verbs to methods / actions
- Maps URI / query string parameters to method parameters



ASP.NET Web API Overview

Content negotiation

- Based off HTTP Accept / Content-Type headers
- JSON / XML formatters out of the box
- OData formatter through NuGet
- Can plug in custom formatters



ASP.NET Web API Configuration

- **No config file settings needed**
- **HttpConfiguration class**
 - Associated with the ASP.NET web application instance
 - Accessible from Global.asax code behind
 - Calls WebApiConfig.Register
 - Defaults are good enough for resource-oriented basic Web APIs
 - Can plug in formatters, filters, message handlers and other custom extensibility objects through this class

ASP.NET Web API Configuration

- **WebAPIConfig**

```
public static class WebApiConfig
{
    public static void Register(HttpConfiguration config)
    {
        config.Routes.MapHttpRoute(
            name: "DefaultApi",
            routeTemplate: "api/{controller}/{id}",
            defaults: new { id = RouteParameter.Optional }
        );
    }
}
```

ASP.NET Web API Configuration

- **Overriding conventions**
 - Routing
 - Can add custom routes – i.e. action-based
 - Method invocation
 - Can use query string parameters
 - Method names
 - Http<Verb> attributes

Agenda

- **HTML / SPA client architecture**
- **JavaScript libraries / frameworks**
- **JQuery**
- **Knockout**
- **ASP.NET Web API**
- **Breeze**
- **Durandal**

Breeze Overview

- **Acts as a data layer / repository for the JavaScript client code**
- **Dispatches service calls to a CRUD Web API for you**
- **Focused on CRUD calls and working with data**
- **Primarily a client side technology**
 - But has server support for ASP.NET Web API as well

Breeze Capabilities

- **Retrieve / query data (entities) via web service calls**
 - Filter, page, sort from the client
- **Cache data on the client side**
- **Track changes to modified entities**
 - Added / Edited / Deleted
 - Observable changes for data binding support
- **Save changes via web service calls**
- **Validate modified entities**
- **Export / Import data on the client for offline storage**
- **Simplify implementing Web API data services**
- **Work with OData services**
- **Work well with data binding frameworks**
 - Knockout, Angular, Backbone, Ember, etc.

Breeze Overview

- Create queries with Breeze EntityQuery
- Execute queries with Breeze EntityManager
- Breeze caches retrieved entities and tracks changes on them
- Persist changes through calls on EntityManager
- EntityManager issues the service calls to query and update
- EntityManager depends on service metadata to define the client side entities and manage relationships between them

Breeze and Other JavaScript Libs

- Breeze designed to work with other JavaScript libraries
 - But not depend on them
- Needs observable support
 - Knockout, Angular, Backbone support out of the box
 - Extensible for other libraries
- Works with Require.js for module dependency management
- Uses q.js for promises
- TypeScript compilation checking
- Visual Studio Intellisense support

BreezeControllers

- **Simplifies development of a CRUD data service with ASP.NET Web API**
- **Automates CRUD patterns**
- **Automatically generates metadata about entities from server side model**

BreezeController Methods

- **Metadata()**
 - Called first by Breeze to retrieve the metadata for the service model
- **<Collection>()**
 - Retrieves collection of some entity type
 - Generally want to name for the collection it returns
- **SaveChanges()**
 - Takes a batch payload and persists all the changes in it (Create/Update/Delete)
- **Other**
 - Can expose arbitrary methods as well

EFContextProvider

- **The brains of the Breeze Web API support**
- **Wraps an EntityFramework DbContext orObjectContext**
- **Dispatches queries through EF**
- **Handles SaveChanges JSON payload**
 - Executes individual CUD changes in proper order based on relationships in the model
- **Can extend to implement custom validation / business logic**

BreezeController Routing

- **Breeze auto registers custom route**
 - /breeze/{controller}/{action}
- **Allows side by side “normal” and OData Web APIs with BreezeControllers**

Extending EFContextProvider

- **Can derive from EFContextProvider**
 - Override BeforeSaveEntity / BeforeSaveEntities
- **Better: Delegate**
 - BeforeSaveEntityDelegate / BeforeSaveEntitiesDelegate

Getting Started with Breeze.js

- **Create an EntityManager**
 - Passing it service address
- **Execute queries to retrieve entities**
- **Modify entities in client JS or through data binding**
- **Save changes through EntityManager**

Querying with Breeze

- **Create EntityQuery object**
- **Use fluent API on it to shape query**
 - from
 - orderby
 - skip
 - take
- **Call EntityManager.executeQuery passing query object**
 - It calls service

Querying with Breeze

- **On first query, Breeze makes metadata GET call**
- **Then issues the query**
- **Subsequent queries go straight through**
- **Breeze caches entity references in EntityManager**
- **Entities are created as observables**
- **Can query local cache to avoid service calls**

Query results shaping

- **expand**
 - Allows you to retrieve related entities (child collections or related objects) based on navigation properties
- **select**
 - Allows you to “project” the results of a query into a new object with a set of properties you control

EntityAspect

- **Contains all the information Breeze needs to track and manage the state of the entity**
 - Entity state
 - Change tracking
 - Validation

Editing Data with Breeze

- **Create**
 - EntityManager.createEntity
 - Don't use "new"
 - Needs to be based on the model metadata
 - Breeze manages key properties that are server populated
- **Update**
 - Just make changes to the properties of the entities returned from a Breeze query
- **Delete**
 - entityAspect.setDeleted()

Saving Changes

- **EntityManager.SaveChanges**
- **Knows what entities have been added, modified, deleted**
 - entityAspect state
- **Makes service call**
- **Gets entities back**
- **Merges the state of returned entities with client side entities**
 - Server computed properties and keys

Breeze Validation

- **Allows you to define validation rules on the data model**
- **Can be driven by ASP.NET and DataAnnotations**
- **Breeze invokes validation rules when:**
 - Entities are added to cache
 - Entities in cache are modified
 - saveChanges is called

Breeze Validation

- **Can manually validate**
 - Whole entity
 - Individual properties
- **Breeze auto-created stock validators**
 - Data type
 - Required
 - MaxLength
- **Breeze additional stock validators**
 - regEx
 - emailAddress, phone, creditCard, url
- **Can write custom validators**

Agenda

- **HTML / SPA client architecture**
- **JavaScript libraries / frameworks**
- **JQuery**
- **Knockout**
- **ASP.NET Web API**
- **Breeze**
- **Durandal**

Durandal Overview




- **JavaScript MV* application framework**
- **Depends on JQuery, Knockout, RequireJS**
- **Provides**
 - Dependency injection / composition
 - Including View/ViewModel composition
 - Routing / navigation
 - Messaging / pub-sub events
 - Dialogs, animations/transitions

Wrapping up...

- **To build a SPA or rich HTML data application, you need to tie together a stack of technologies**
- **You learned about:**
 - SPA architecture
 - JQuery, Knockout, Breeze as the primary client side libraries
 - ASP.NET Web API (+ BreezeControllers) for the server side
 - HTML/CSS/JavaScript debugging

Resources

- **Pluralsight courses:**
 - Breeze – Brian Noyes
 - <http://pluralsight.com/training/Courses/TableOfContents/building-single-page-applications-breeze>
 - Knockout / MVVM / SPAs / Angular – John Papa
 - <http://pluralsight.com/training/Courses/TableOfContents/knockout-mvvm>
 - <http://pluralsight.com/training/Courses/TableOfContents/single-page-apps-jumpstart>
 - <http://pluralsight.com/training/Courses/TableOfContents/spa>
 - <http://pluralsight.com/training/courses/TableOfContents?courseName=build-apps-angular-breeze>
 - Coming Soon: Durandal – Michael Dudley
- **Knockout:** <http://knockoutjs.com>
- **Breeze:** <http://breezejs.com>
- **ASP.NET Web API:** <http://www.asp.net/web-api>
- **Durandal:** <http://durandaljs.com>

 brian.noyes@solliance.net
 @briannoyes
 <http://briannoyes.net>

<anglebrackets/>

Questions?

**Don't forget to enter your evaluation
of this session using EventBoard!**

Thank you!

<anglebrackets/>