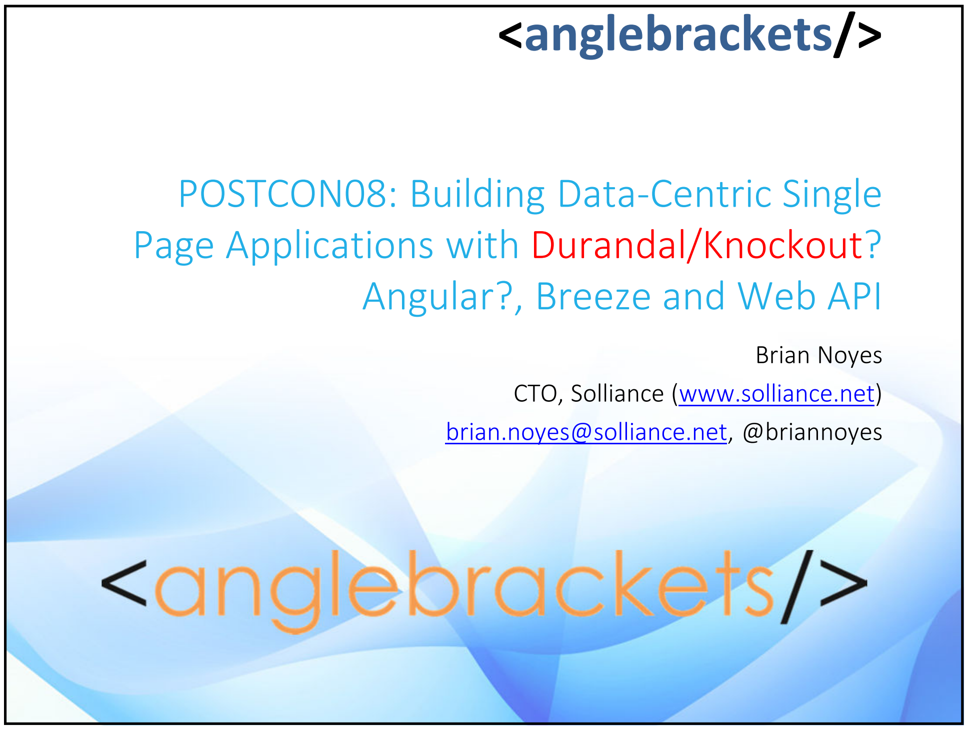
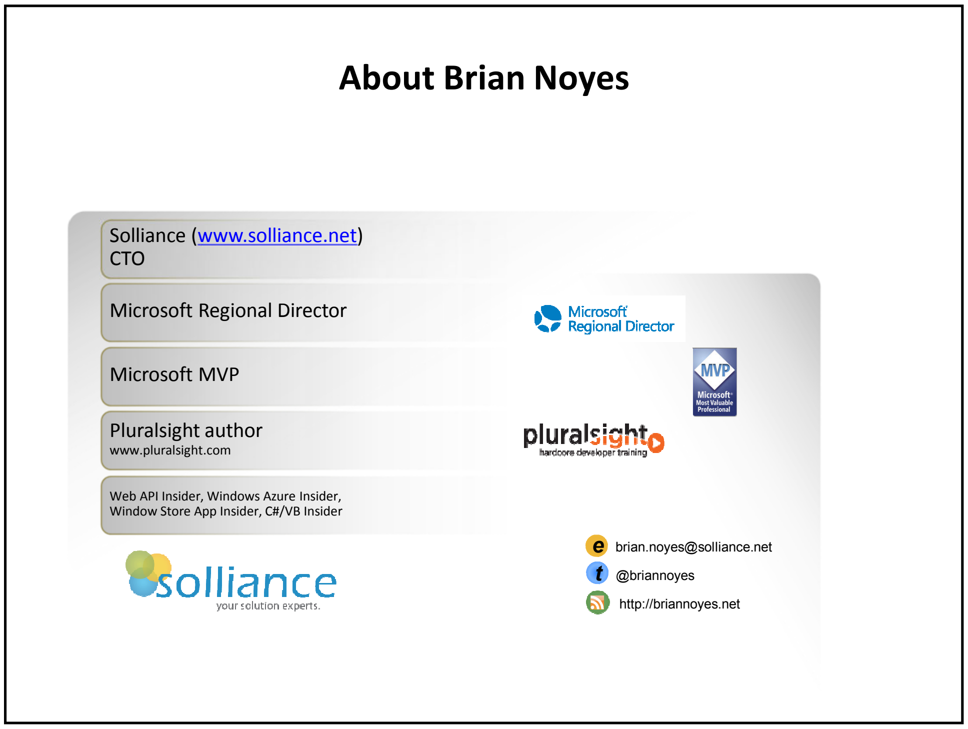


<anglebrackets/>

POSTCON08: Building Data-Centric Single Page Applications with Durandal/Knockout? Angular?, 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, Window Store App Insider, C#/VB Insider

solliance
your solution experts.

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

Agenda

- Durandal vs Angular
- HTML / SPA client architecture
- JavaScript libraries / frameworks
- Angular Basics
- Angular Data Binding
- Angular Routing and Composition
- ASP.NET Web API
- Breeze

Durandal vs Angular

- **Angular has been "winning the war" for a while now**
  - Adoption
  - Mindshare
  - Resources
- **Durandal (+Knockout) was more popular for a while in Microsoft camps**
  - Knockout included in ASP.NET templates
  - Durandal composed nicely on top of Knockout, RequireJS, and Jquery
- **Durandal had some specific things it did "better" (depends on perspective) than Angular**
- **Both have same core capabilities, apply same design patterns**
- **Some differences under the covers, and differences in syntax**
- **Breaking news:**
  - Rob Eisenberg (creator of Durandal) is now part of the AngularJS team
  - Durandal 2.x will continue to be supported
  - There will not be any significant new versions of Durandal

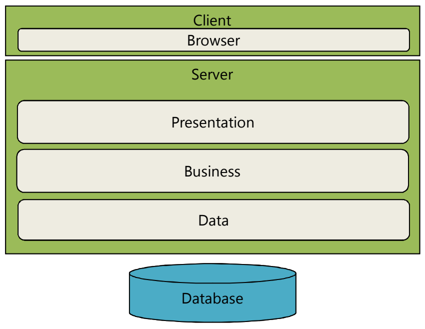
Durandal vs Angular

- **If you are already deeply invested in Durandal from a code perspective – stay put for now**
  - No reason to stop using it immediately
  - Migration guidance will be forthcoming
- **If you are just getting started or not too deep**
  - Go Angular FTW! :P

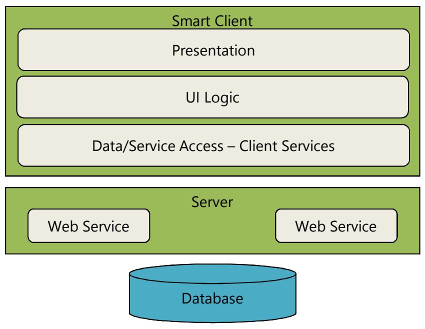
Agenda

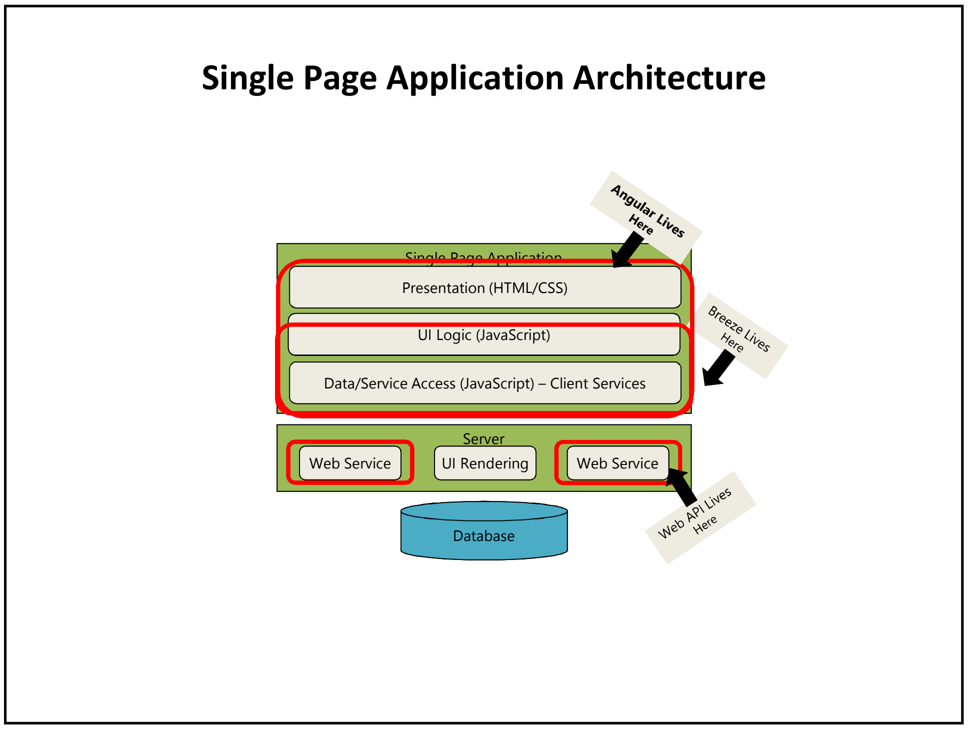
- **Durandal vs Angular**
- **HTML / SPA client architecture**
- **JavaScript libraries / frameworks**
- **Angular Basics**
- **Angular Data Binding**
- **Angular Routing and Composition**
- **ASP.NET Web API**
- **Breeze**

Browser App Architecture

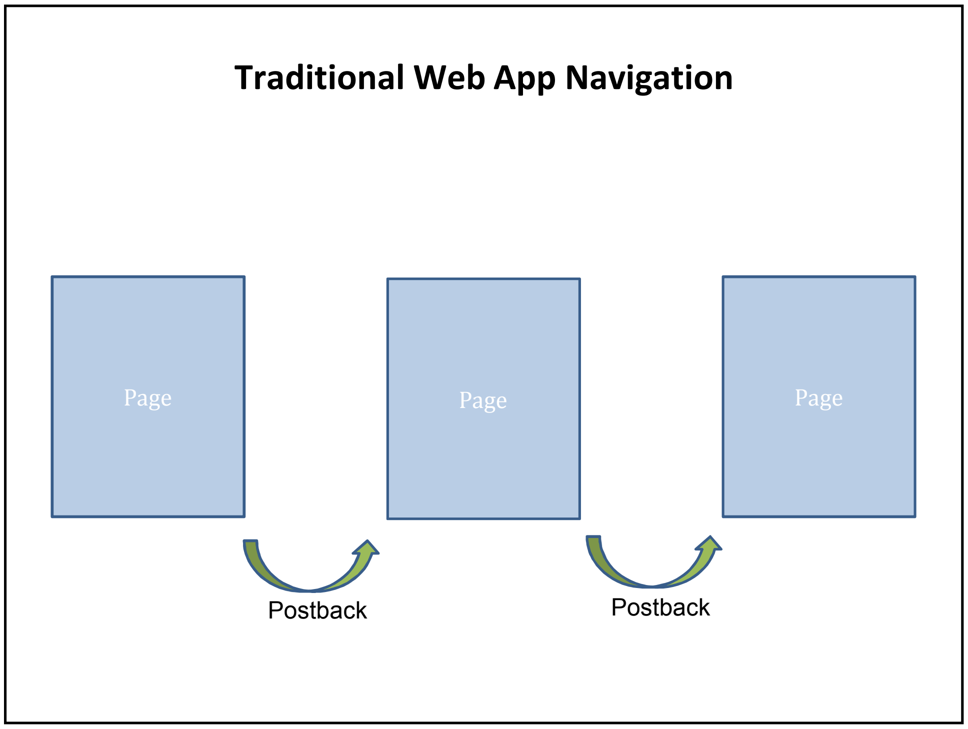


Layered Service-Oriented Smart Client



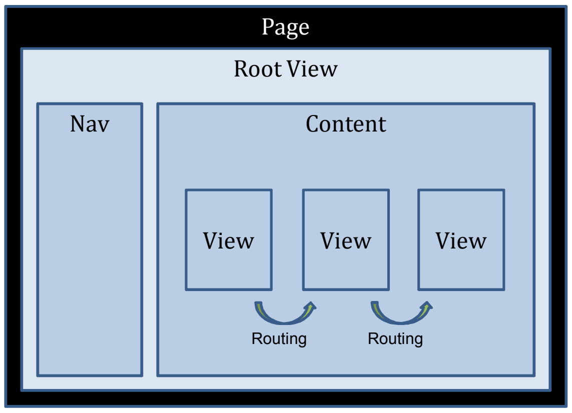


Single Page Application Architecture

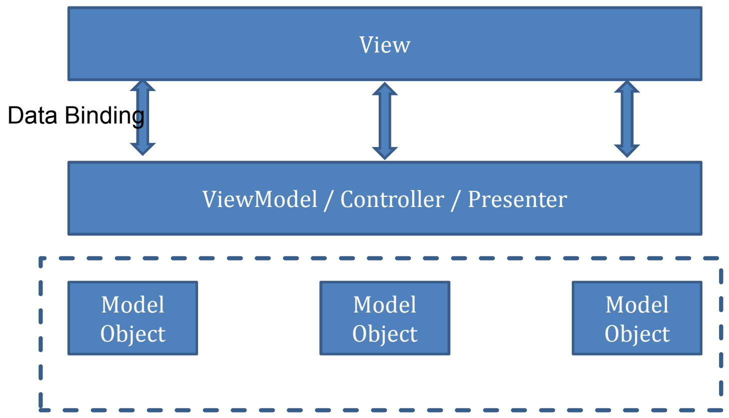


Traditional Web App Navigation

SPA Navigation



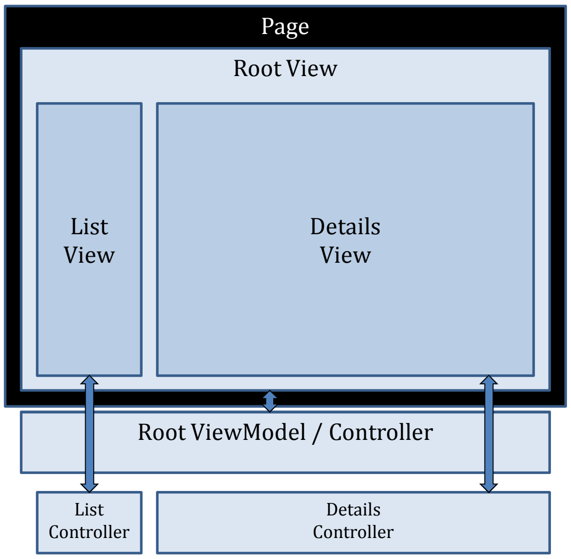
MV* Structuring



MVVM / MVC 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
- **Controller (JS)**
  - Similar responsibility as ViewModel
  - May use intermediary ViewModel object ($scope) that View talks to directly

Composite Views



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**
- **Angular Basics**
- **Angular Data Binding**
- **Angular Routing and Composition**
- **ASP.NET Web API**
- **Breeze**

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
- **Angular**
  - Data binding, dependency injection, routing, services, directives
- **Breeze**
  - CRUD data service calls, change tracking, validation
- **Twitter Bootstrap**
  - CSS styling and widgets

JavaScript Libraries / Frameworks

- **What about?**
  - Durandal
  - 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
  - Angular ~~is winning~~ has won the war…

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…

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

Important Development Tools

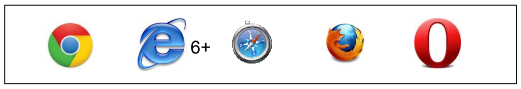
- **NuGet**
- **Chrome / IE developer tools (F12)**
- **Fiddler**
- **Postman**
- **SideWaffle**
- **HotTowel Angular**
- **jsFiddle**
- **Web Essentials**
- **CodeRush / Resharper**
- **Productivity Power Tools – 2013**
- **SublimeText / WebStorm**
- **Grunt / Gulp / Mimosa**
- **QUnit / Jasmine / Mocha / Sinon**

Agenda

- HTML / SPA client architecture
- JavaScript libraries / frameworks
- Angular Basics
- Angular Data Binding
- Angular Routing and Composition
- ASP.NET Web API
- Breeze

Angular Basics

- **End-to-end client JavaScript application framework**
- **Open source / Google team**
- **Current version: 1.2.X**



- **Coming soon: 1.3**
  - Dropping support for IE8 and prior
- **In work: 2.0**
  - Evergreen browsers only
  - Focus on mobile first

Angular Features

- **Data binding**
- **Dependency injection**
- **Modular composition**
- **MV* structuring**
- **Navigation / routing**
- **Built-in / Custom directives**
- **Templating**
- **Services**
- **Animations**
- **Filters**
- **ETE Testing**

Modules

- **Logical container for a set of functionality**
  - Controllers, directives, services, etc.
  - Defined in their own individual JavaScript files
- **Always at least one root application module**
- **Can define as many modules as you like for factoring**
  - i.e. common module for reusable functionality
- **Used to create the constructs that are contained within it**

Directives

- **Can be elements, attributes, or comments**
- **Built-in Angular directives named ng-**
- **Can prefix with data-**
  - Preferred for HTML 5 validators
- **Naming conventions**
  - ngCamelCased in JavaScript
  - ng-lower-cased in HTML

Controllers

- **MVC-based controllers**
- **Can be treated as ViewModels if more familiar / prefer MVVM**
- **Contain the interaction logic and data manipulation for a view**
- **Registered with the application module in Angular**
- **Expose properties and functions for the view to call through directives and data binding**

Services

- **Client side construct**
  - Not “web services”
- **Shared code across controllers, directives, or other services**
- **Typically a singleton instancing model**
- **Defined through the module with .factory() method**

Getting Started with Angular

- **Include core Angular script in the page**
- **Include ng-app directive on root element of page**
- **Declare Angular module for the app**
  - Container for all the parts of your app
    - Controllers, services, directives, filters, etc
- **Define controller for each view**
- **Tie controller in with ng-controller directive**
  - Or through routing
- **Use data binding and other directives to drive behavior and presentation**

Dependency Injection

- **Used to manage complex graphs of dependencies**
- **Decouples dependent code from details of**
  - Where is dependency defined
  - How is its object lifetime managed (singleton vs not)
  - What dependencies does the dependency have

Dependency Injection in Angular

- **Second argument to .controller, .directive, .factory, etc.**
  - Array of dependency names (id's)

```
var controllerId = 'dashboard';
angular.module('app').controller(controllerId,
     ['common', 'datacontext', dashboard]);

function dashboard(common, datacontext) {
    // use common, datacontext dependencies
}
```

- **Function is last argument in array**
- **Function takes dependencies as arguments, in order declared**

Agenda

- HTML / SPA client architecture
- JavaScript libraries / frameworks
- Angular Basics
- Angular Data Binding
- Angular Routing and Composition
- ASP.NET Web API
- Breeze

A World With No Data Binding

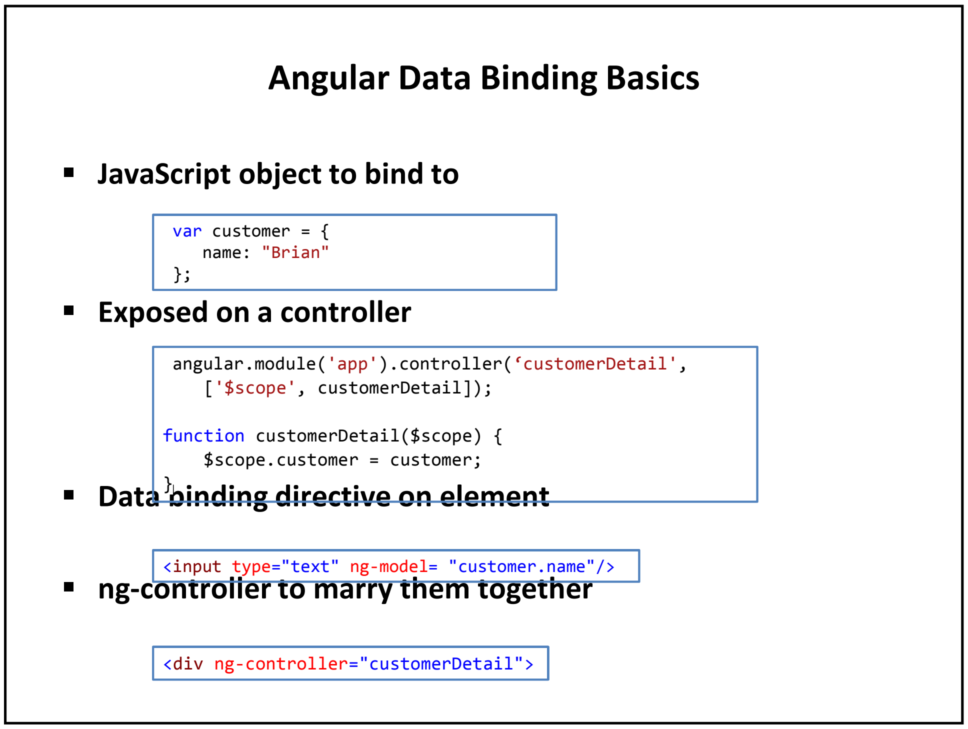
- Logic code pushes discrete values from data object properties into UI element properties
- Logic code pulls modified values out of UI element properties and puts them into data object properties
- Need explicit triggers for when to push and pull
- JQuery works well for this task

Data binding

- **Declarative approach**
- **Associates UI element properties with data object properties**
- **Can be two-way**
  - Automatically retrieves data object properties into element for presentation
  - Automatically pushes changed values in element into underlying data object
- **Driven by directives in Angular**

Data-bound objects

- **Just Plain-Old-JavaScript-Objects (POJOs) with Angular**
- **No special requirements**
  - i.e. “observables” in Knockout



Angular Data Binding Basics

- JavaScript object to bind to

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

- Exposed on a controller

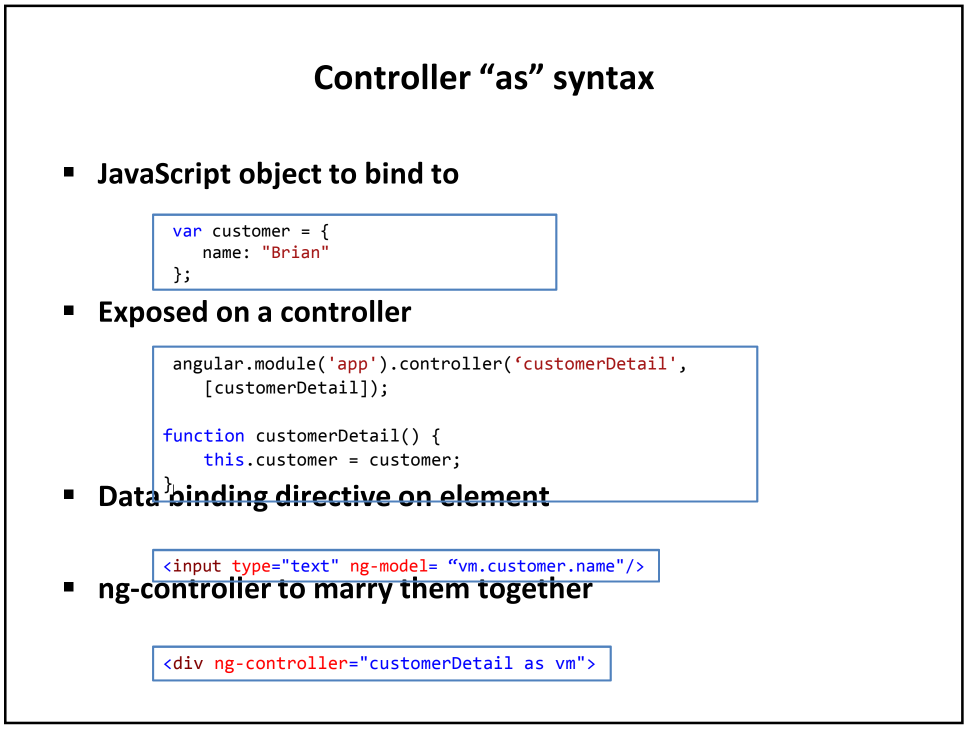
```
 angular.module('app').controller(‘customerDetail',
    ['$scope', customerDetail]);

function customerDetail($scope) {
    $scope.customer = customer;
}
```

- Data binding directive on element

```
<input type="text" ng-model= "customer.name"/>
```

- ng-controller to marry them together

```
<div ng-controller="customerDetail">
```



Controller “as” syntax

- JavaScript object to bind to

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

- Exposed on a controller

```
 angular.module('app').controller(‘customerDetail',
    [customerDetail]);

function customerDetail() {
    this.customer = customer;
}
```

- Data binding directive on element

```
<input type="text" ng-model= “vm.customer.name"/>
```

- ng-controller to marry them together

```
<div ng-controller="customerDetail as vm">
```

Two way data sync

- **Between View and Controller**
- **Digest cycle**
  - Keep track of bindings when initial view is parsed
  - When any bound property/function on the controller changes – re-evaluate all
  - Less efficient than observables
    - But much less obtrusive than observables
  - Sometimes have to trigger manually
    - $scope.$apply();

Angular Data Binding Directives / Syntax

- **Value binding for content:**

```
<div>{{vm.customer.name}}</div>
```

- **Equivalent directive:**

```
<div ng-bind="vm.customer.name"></div>
```

- **Two-way data binding on input elements:**

```
<input type="text" ng-model= “vm.customer.name"/>
```

Angular Bindings

- Text
- Input
- Appearance
- Control flow

Data Binding Scope

- $scope always ties to the current controller
  - Even when using the “as” syntax
- ng-controller establishes new scope
  - Child scope of parent
  - Can get to parent scope
- Can be nested N-levels deep

Custom Directives

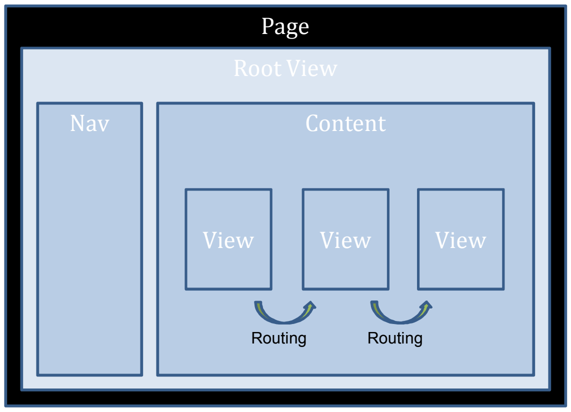
- Can define your own custom directives

```
angular.module('app').directive('myImageSource',
[myImageSource]);
function myImageSource () {
    var directive = {
        link: link,
        restrict: 'A'
    };
    return directive;

    function link(scope, element, attrs) {
        // implementation...
    }
}
```

Agenda

- HTML / SPA client architecture
- JavaScript libraries / frameworks
- Angular Basics
- Angular Data Binding
- Angular Routing and Composition
- ASP.NET Web API
- Breeze

Need something to do this…



Setting up routing

- **Define routes against $routeProvider**

```
$routeProvider.when('/customerDetail/:customerId',
{
    templateUrl: 'customers/customerDetail.html',
    controller: 'customerDetail'
});
```

- **Set up container element where views will navigate**
    - ng-view attribute or element

Navigating

- **Address bar / anchor tags**
  - URL with relative #/address
    - Will match against routes
- **$location.url**
  - Programmatic invocation

Access parameters

- **URL parameters**
  - Defined with names as part of route template
  - Use $routeParams service to access
    - Named properties on service object
- **Query string parameters**
  - $location.search() returns JS object with named properties based on parameter names

Agenda

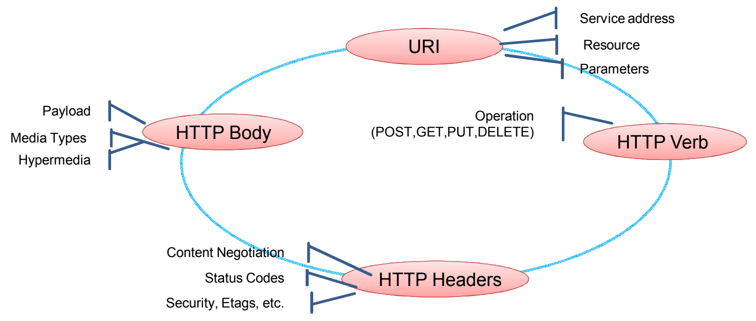
- **HTML / SPA client architecture**
- **JavaScript libraries / frameworks**
- **Angular Basics**
- **Angular Data Binding**
- **Angular Routing and Composition**
- **ASP.NET Web API**
- **Breeze**

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

REST

- **REpresentational State Transfer**
- **REST is an architectural style, SOAP is a protocol**
  - Based on Ph.D. thesis: Roy Fielding
- **REST fully embraces HTTP**

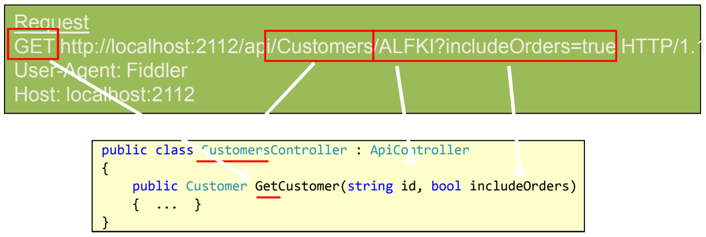


ASP.NET Web API Overview

- **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

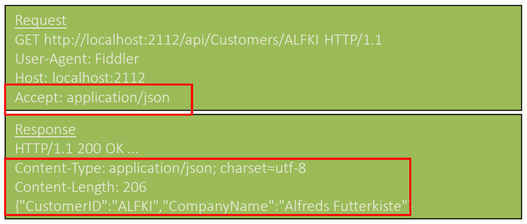


```
Request
GET http://localhost:2112/api/Customers/ALFKI?includeOrders=true HTTP/1.1
User-Agent: Fiddler
Host: localhost:2112
```

```
public class CustomersController : ApiController
{
    public Customer GetCustomer(string id, bool includeOrders)
    { ... }
}
```

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



```
Request
GET http://localhost:2112/api/Customers/ALFKI HTTP/1.1
User-Agent: Fiddler
Host: localhost:2112
Accept: application/json
```

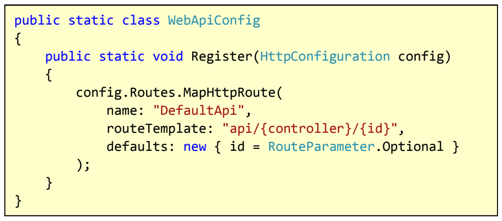
```
Response
HTTP/1.1 200 OK ...
Content-Type: application/json; charset=utf-8
Content-Length: 206
{"CustomerID":"ALFKI","CompanyName":"Alfreds Futterkiste"
```

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**
- **Angular Basics**
- **Angular Data Binding**
- **Angular Routing and Composition**
- **ASP.NET Web API**
- **Breeze**

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 or ObjectContext**
- **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**

Breeze and Angular

- **Breeze uses by default:**
  - q library for promises
  - JQuery for AJAX service calls
- **Angular has its own promises**
  - $q
  - Digest Cycle depends on $q
  - Need to use $q to complete Breeze async calls so digest cycle gets triggered
- **Angular has its own AJAX service object**
  - $http
  - Preferable to use that over JQuery

Breeze and Angular

- **Old way: .to$q()**
- **New way: Breeze Angular Service**
  - Pull in NuGet
  - Include breeze.angular.js in page
  - Replaces use of JQuery and q with Angular's $q and $http
- **Currently: Does not work with IE8 and prior**
  - Can't define properties on objects in way that is compatible with digest cycle

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**

Breeze and Angular Validation

- **Breeze Directives NuGet / script**
- **Contains custom ng-z-validate directive**
- **Knows how to get to / monitor validation info on entityAspect**
- **Ties it in with HTML 5 validation and popups to show validation errors**

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
  - Angular, Breeze as the primary client side libraries
  - ASP.NET Web API (+ BreezeControllers) for the server side
  - HTML/CSS/JavaScript debugging

Resources

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

- **Pluralsight courses:**
  - Breeze – Brian Noyes
    - http://pluralsight.com/training/Courses/TableOfContents/building-single-page-applications-breeze
  - Breeze / Angular – John Papa
    http://pluralsight.com/training/courses/TableOfContents?courseName=build-apps-angular-breeze
    http://pluralsight.com/training/courses/TableOfContents?courseName=build-apps-angular-breeze-part2
- **Angular: http://www.angularjs.org**
- **Breeze: http://breezejs.com**
- **ASP.NET Web API: http://www.asp.net/web-api**

Questions?

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

Thank you!

<anglebrackets/>