Vincent Farah

Abstract

https://app.pluralsight.com/player?course=angular-web-api-front-back&author=deborah-kurata&name=angular-web-api-front-back-m2&clip=6&mode=live

Angular front to back with web api

Help keep track of basics and git sample to fork and play with and to use tracking stuff done and keeping a nice summary of important information

# Angular front to back

## Getting started with Angular

<https://app.pluralsight.com/player?course=angular-web-api-front-back&author=deborah-kurata&name=angular-web-api-front-back-m2&clip=3&mode=live>

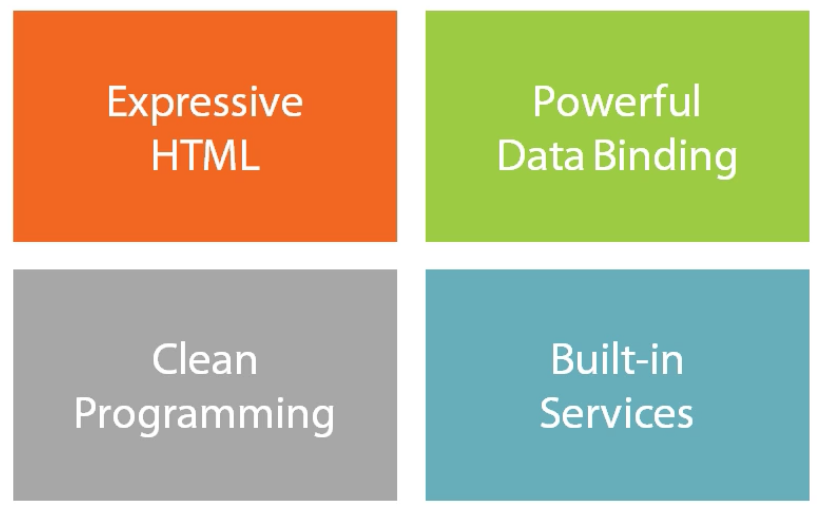
This is a small pluralsight course, will keep docs while doing the course for any good tips and hints

Make a point of doing the basics

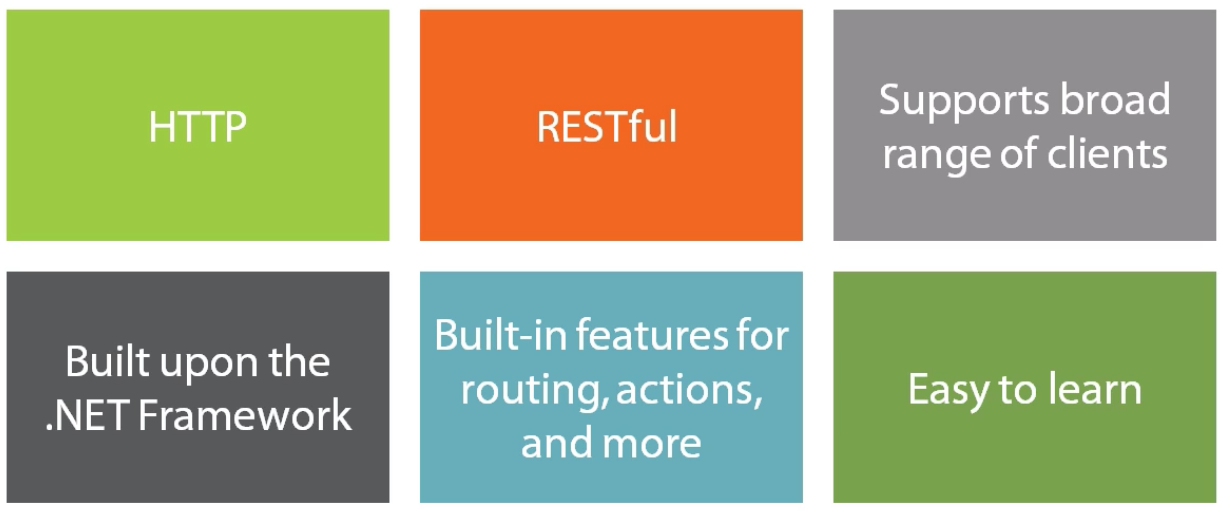
* **Angular courses**
  + AngularJS: Get started
  + Shaping up with Angular JS
  + AngularJS Line of Business Applications
* What this course covers
  + Application Anatomy
  + Retrieving Data
  + CORS and Formatters
  + Passing Parameters
  + Using ODATA Queries
  + Saving Data
  + Action Results, Error Handling & Validation
  + User Authentication in Web API
  + User registration and Login
  + User Authorization



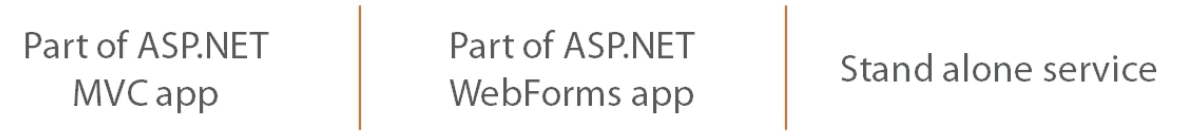
* Why Angular



* Why ASP.NET Web Api



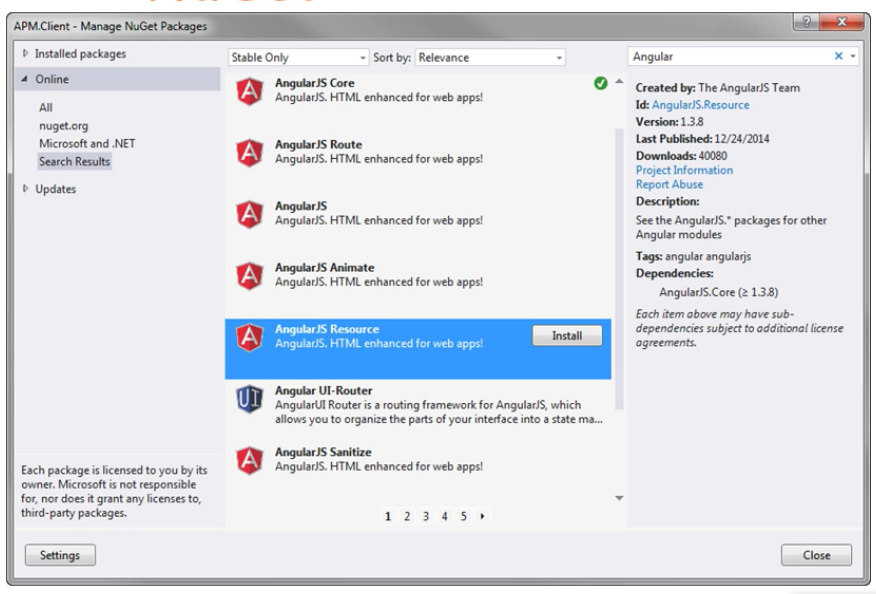
* Building an ASP.NET Web API



* Create a default Web API project
  + Separate project will be to do tests for it
  + Create dependency injection
  + Typescript for all the libraries that will be used
  + Generate MVC strongly typed stuff for Angular
* I changed the default path and moved the solution to the root and as a result Roslyn broke, I resolved this by running this line for Roslyn to work again
  + **update-package Microsoft.CodeDom.Providers.DotNetCompilerPlatform -r**

### Creating the Projects: Angular

* Add an Empty Web project with nothing in it for the web client and add it to the existing web solution
* NuGet



* + AngularJS Core
  + Bootstrap

### Anatomy of an Angular Application

* Application called *productManagement*
  + var app = angular.module("productManagement",
* *Index.html* with the ng angular prefix on the application and the html view
* View files have an associated productListView Controller and manages the model

angular.module("productManagement")

       .controller("productListCtrl",

                     productListCtrl);

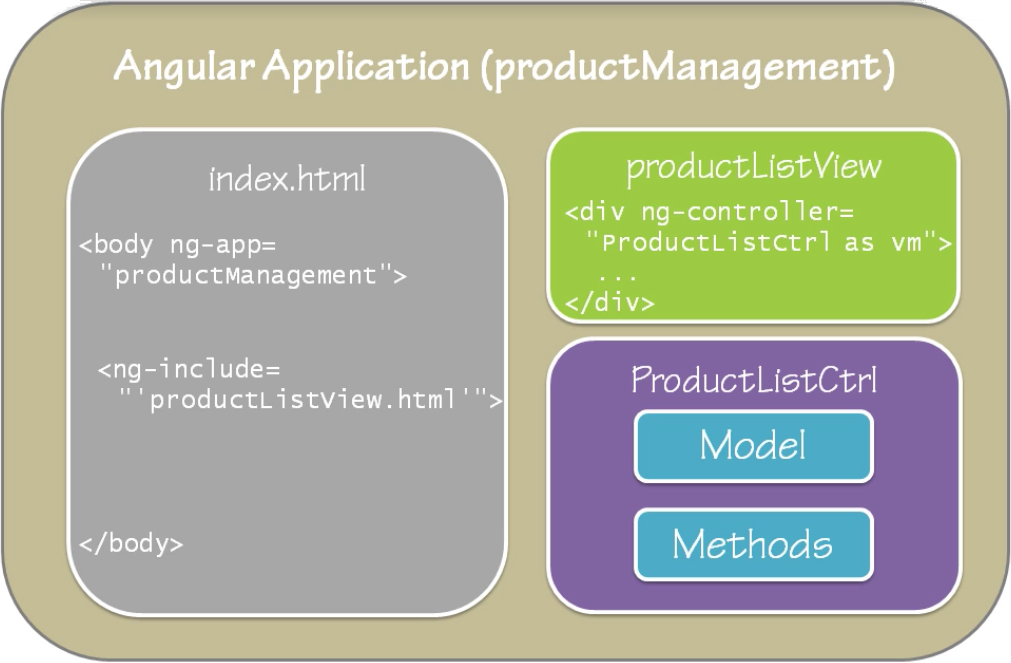
    function productListCtrl() {

        var vm = this;

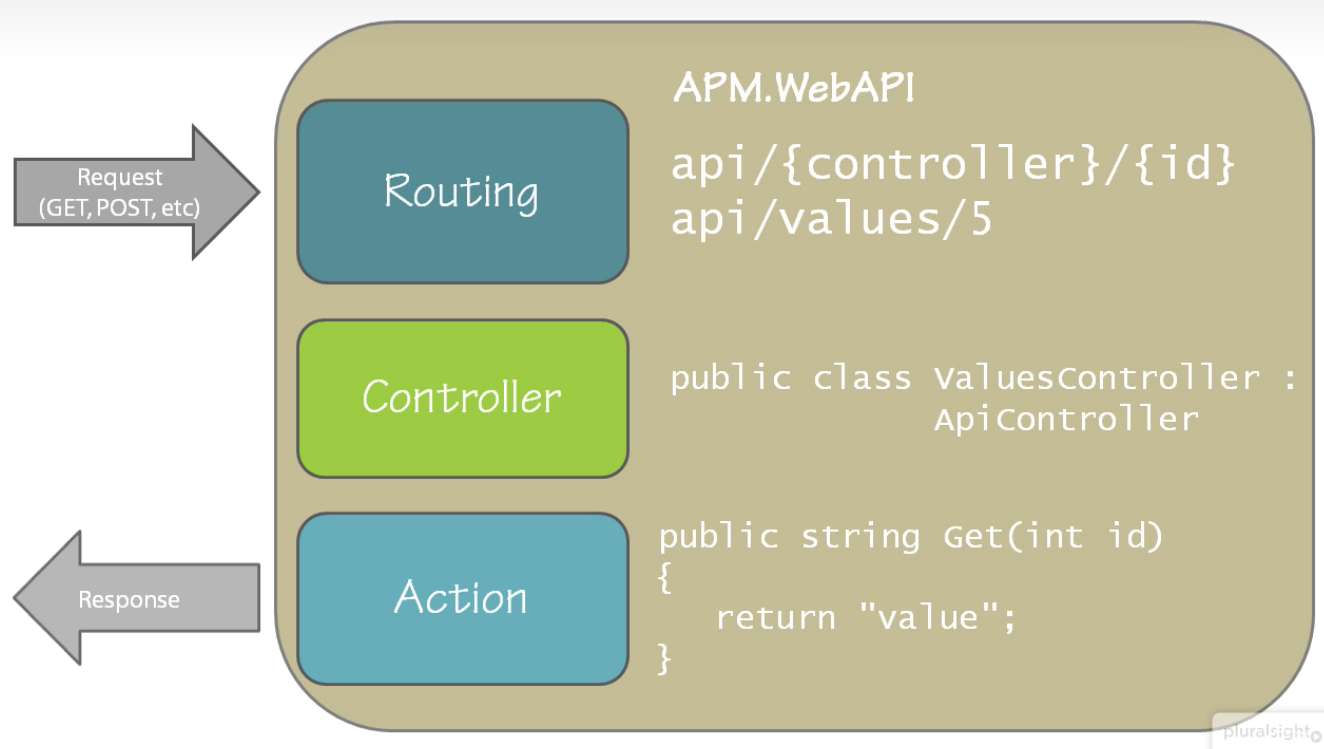
        vm.products = [

        {

            "productId": 1, …

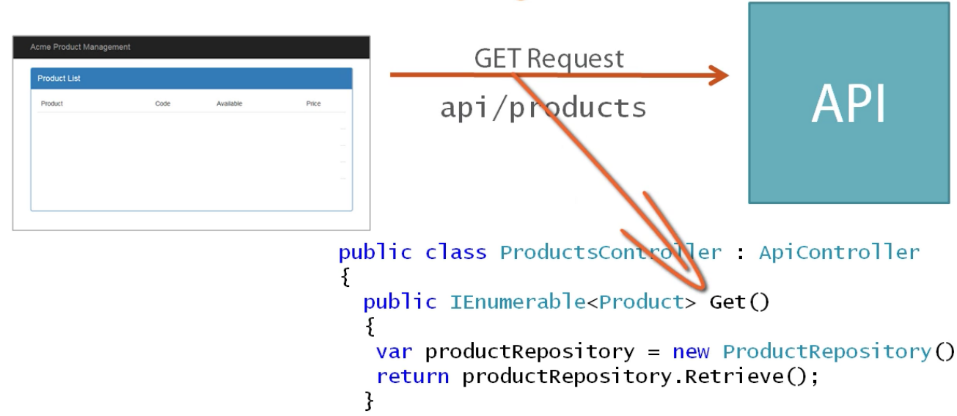


### Anatomy of an ASP.NET Web API Service

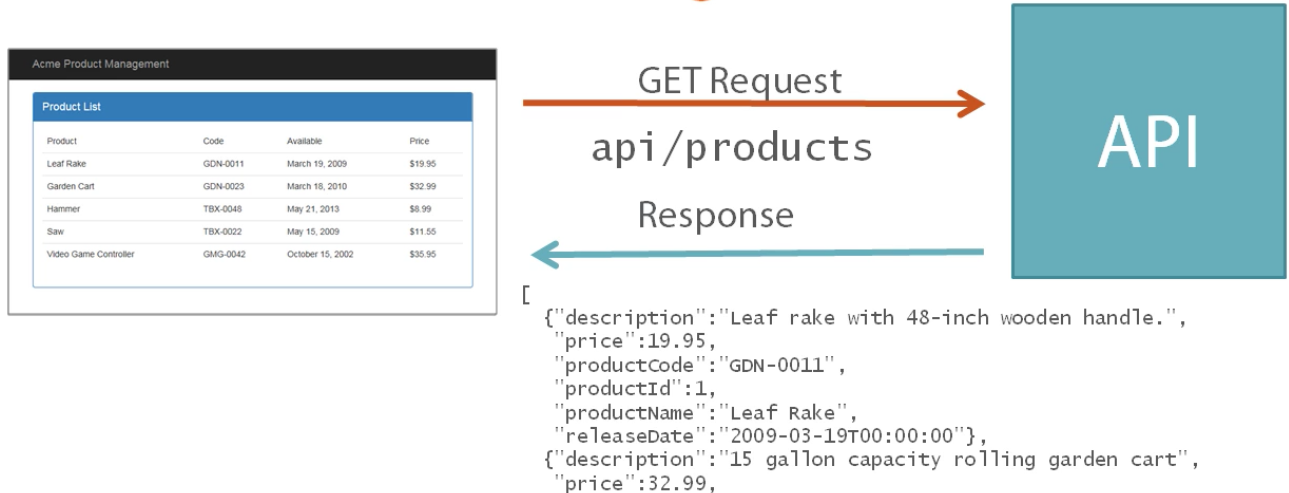


## Retrieving the Data

* Makes a request that gets routed via the Angular controller to the Web Api controller to retrieve the products



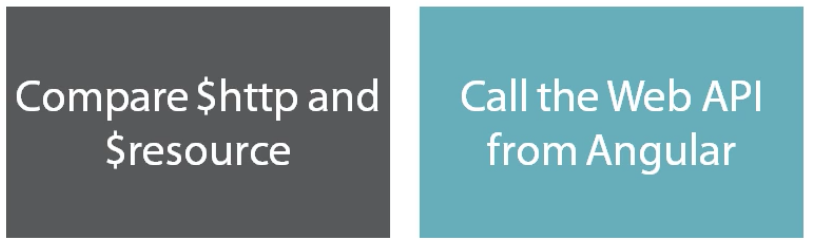
This in turn responds with JSON data to load into the page with Angular binding



Web API objectives



Angular Objectives



### Building the Web API

* What is a Web API Model?
  + Model the data exposed by Web API, specifying the shape and elements of that data
  + Define the return type for the get operations

public class Product

{

    public string Description { get; set; }

    public decimal Price { get; set; }

    public string ProductCode { get; set; }

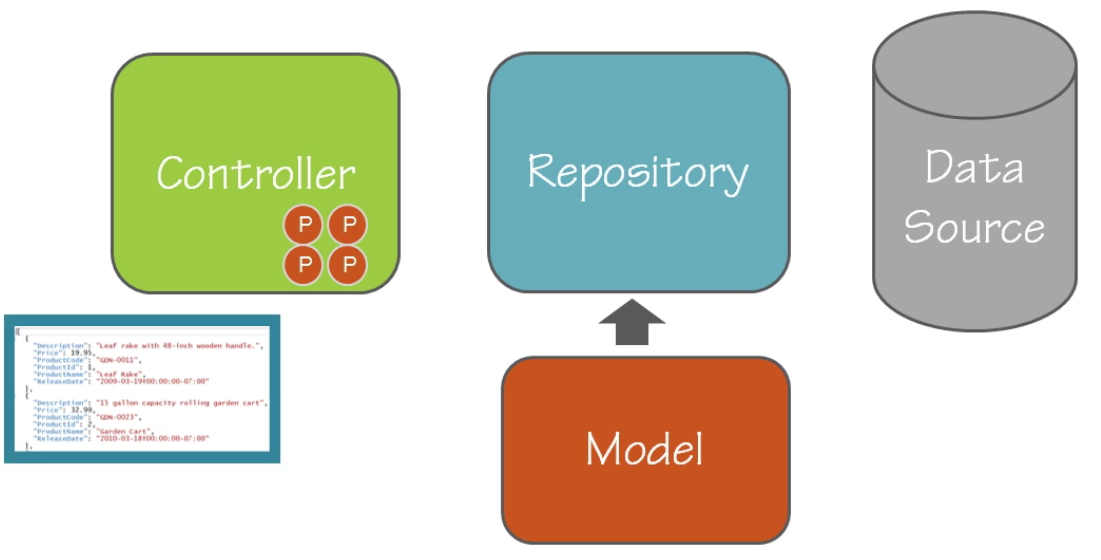
    public int ProductId { get; set; }

    public DateTime ReleaseDate { get; set; }

}

### Building the repository

* What is a repository
  + The data access logic is separated from the data
  + The data could be obtained from a db, JSON, No SQL or any other data source



* The repository should retrieve and create/update data
* Create a repository abstraction and a simple concrete type to implement the data

public interface IProductRepository

{

    Product Create();

    IList<Product> Retrieve();

    Product Save(Product product);

}

public class ProductTextRepository : IProductRepository …

* Hooking this up in the Web API controller with very simple patterns

public class ProductsController : ApiController

{

    private readonly IProductRepository productRepository;

    public ProductsController()

    {

        this.productRepository = new ProductRepository();

    }

    // GET: api/Products

    public IEnumerable<Product> Get()

    {

        return productRepository.Retrieve();

    }

    // GET: api/Products/5

    public Product Get(int id)

    {

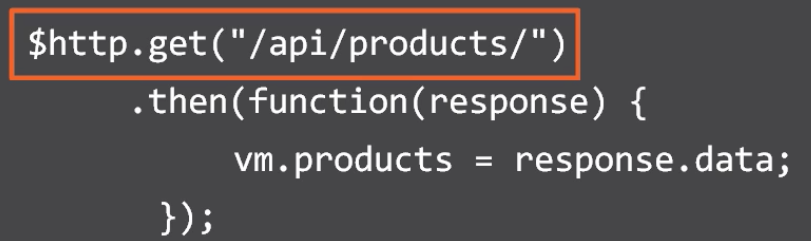
        return productRepository.FindProductId(id);

    }

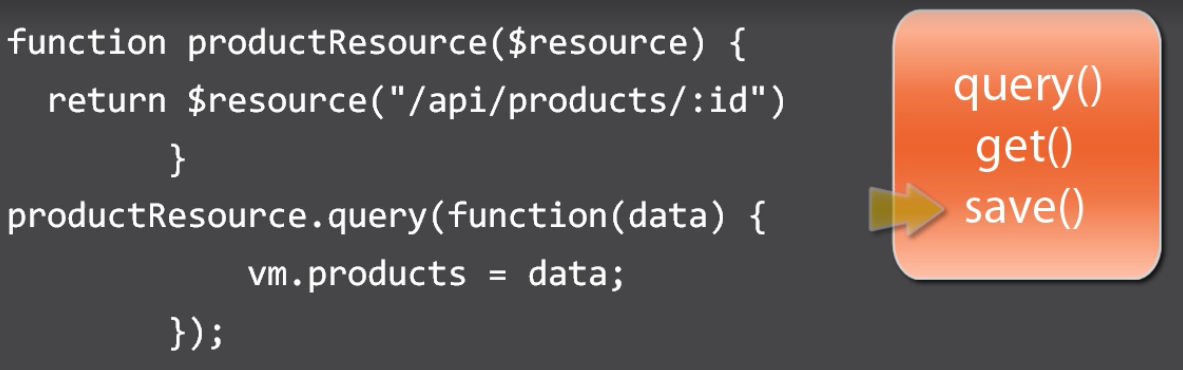
…

### Angular Services for calling a web service

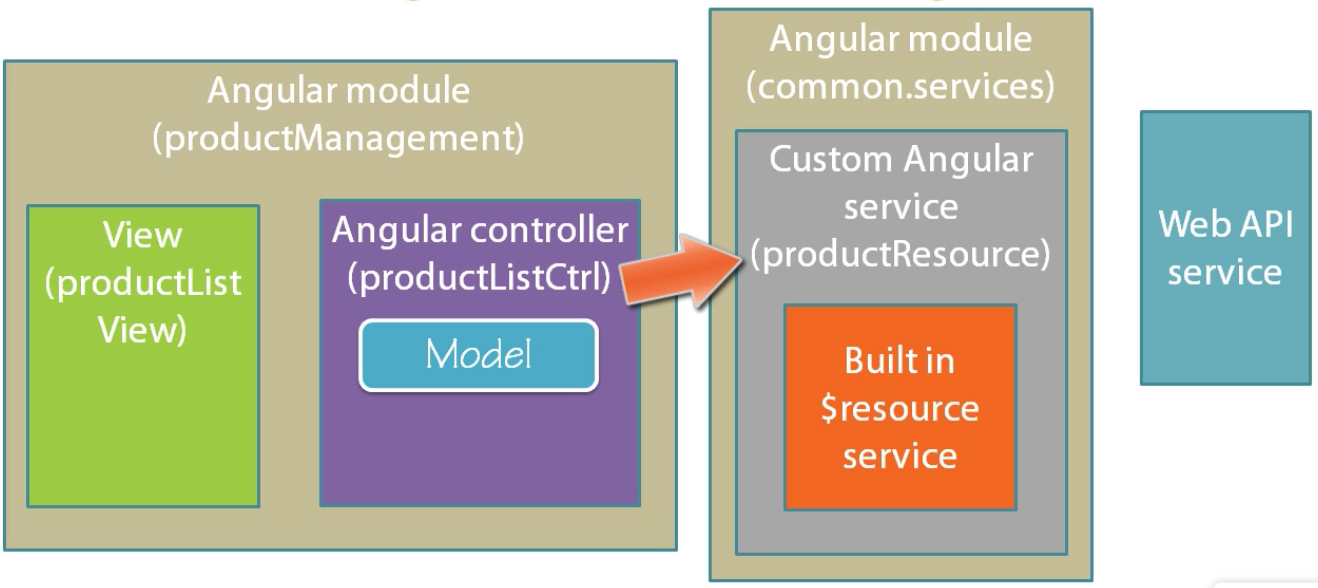
* **REST** requests utilise a standardize on a set of HTTP verbs, GET | POST | PUT |DELETE
* **$http** is part of the Angular Core through a JS promise object



* **$resource** service can communicate best with REST, an abstraction on top of $http for calling RESTful services. Requiring less code and does more in the one line



* Build common services to get *productResource*



**(function () {**

**"use strict";**

**angular.module("common.services", ["ngResource"])**

**.constant("appSettings",**

**{**

**serverPath: 'http://localhost:51735'**

**});**

**}());**

* Build a product resource that uses common service

**(function() {**

**"use strict";**

**angular**

**.module("common.services")**

**.factory("productResource",**

**["$resource", "appSettings", productResource]);**

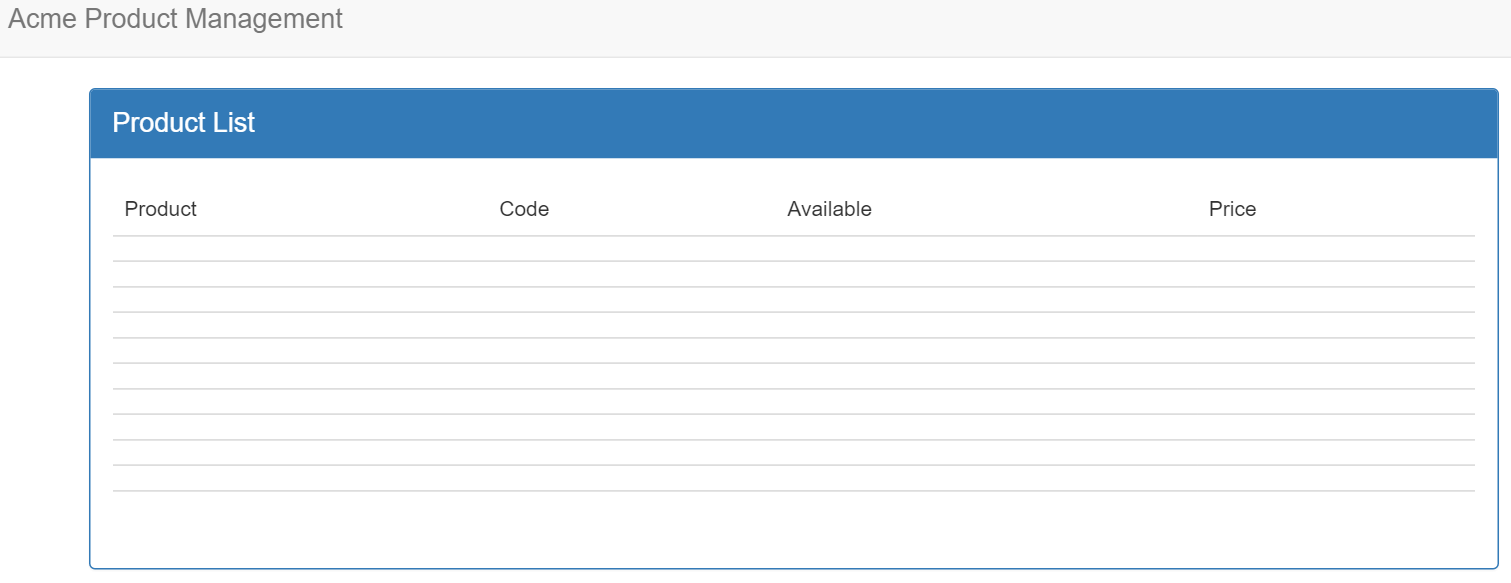
**function productResource($resource, appSettings) {**

**return $resource(appSettings.serverPath + "/api/products/:id");**

**}**

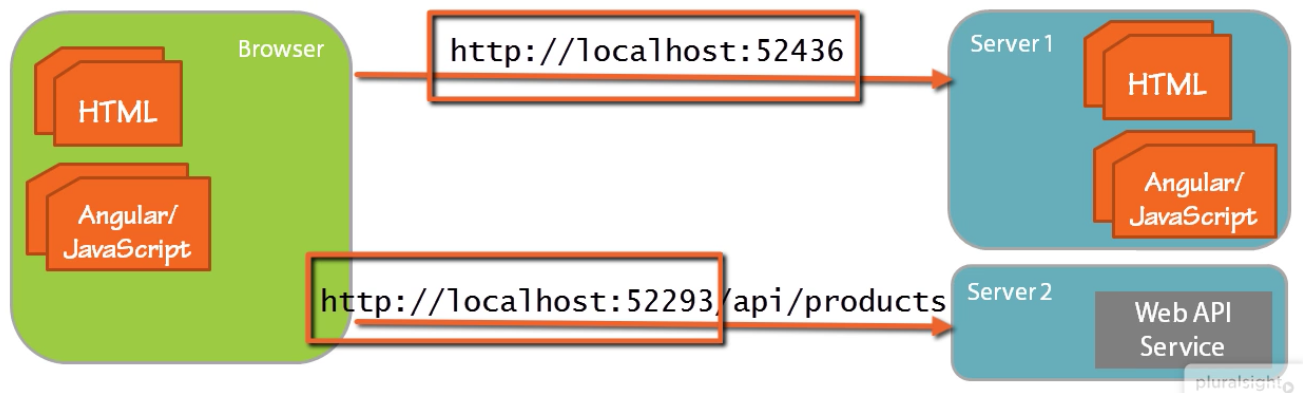
**}());**

* Showing no data … serialization issue to be tackled in the next chapter



## CORS and Formatters

* “CORS … allows many resources … on a web page to be requested from another domain from the resource originated”
* The same origin policy is not adhered within the following because the **port** is different, to prevent a malicious site from seeing information

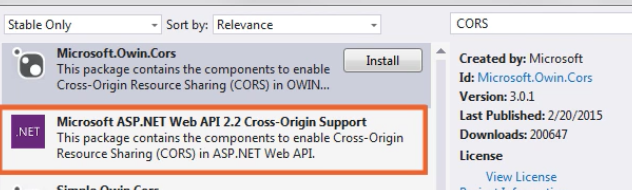


* JSONP is another mechanism for getting round the issue

### Enabling CORS in a Web API Service

* Download the CORS package
* Call the **EnableCors** method as part of the configuration setup





### Configuring Serialization Formatters

* Serialization formatters for both JSON and XML

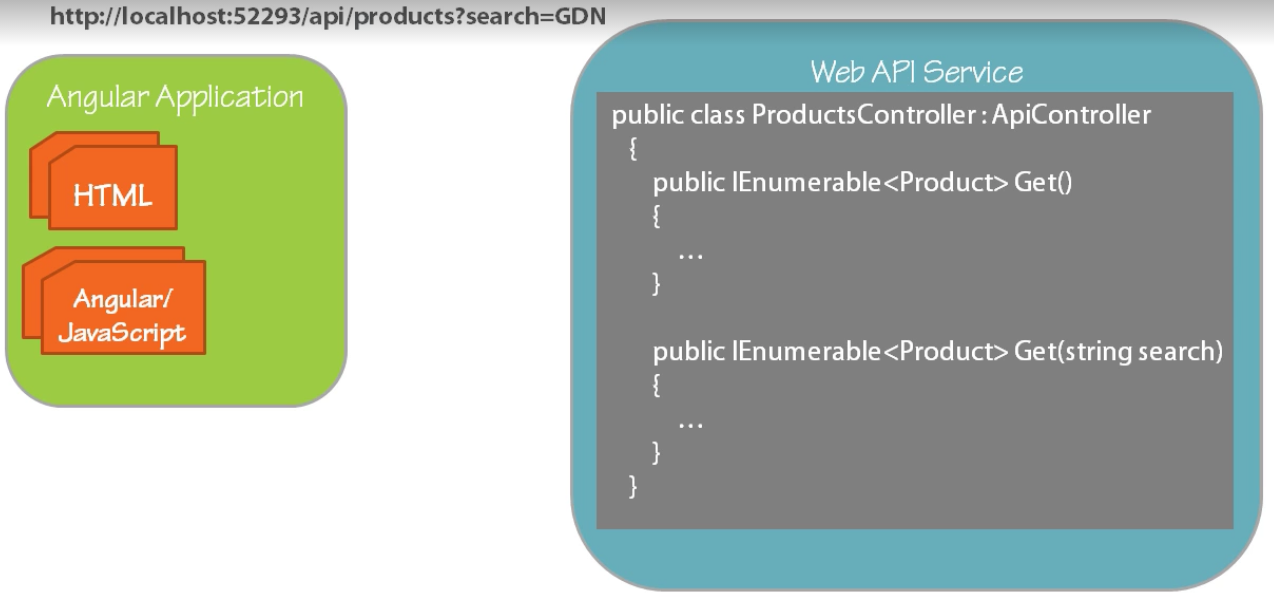
//Setting the Serialization format for camel case

config.Formatters.JsonFormatter.SerializerSettings.ContractResolver

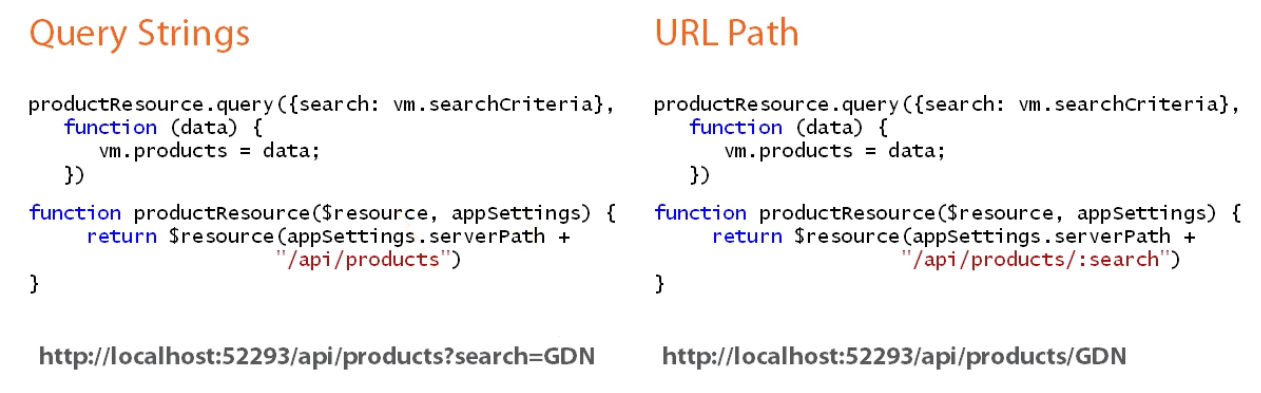
= new CamelCasePropertyNamesContractResolver();

## Passing Parameters

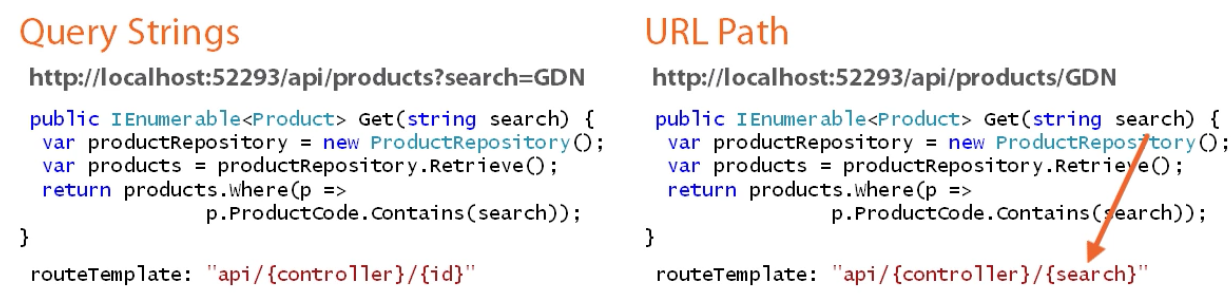
* “A query string is part of the UL containing data that does not fit conveniently into a hierarchical path structure”



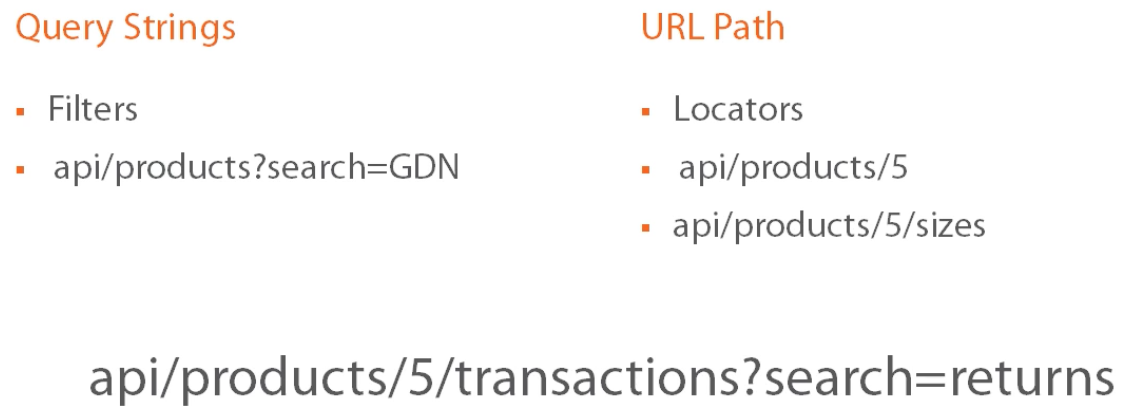
* Angular code for processing the query



* Web API code



* When should we do one or the other?



## Using ODATA Queries

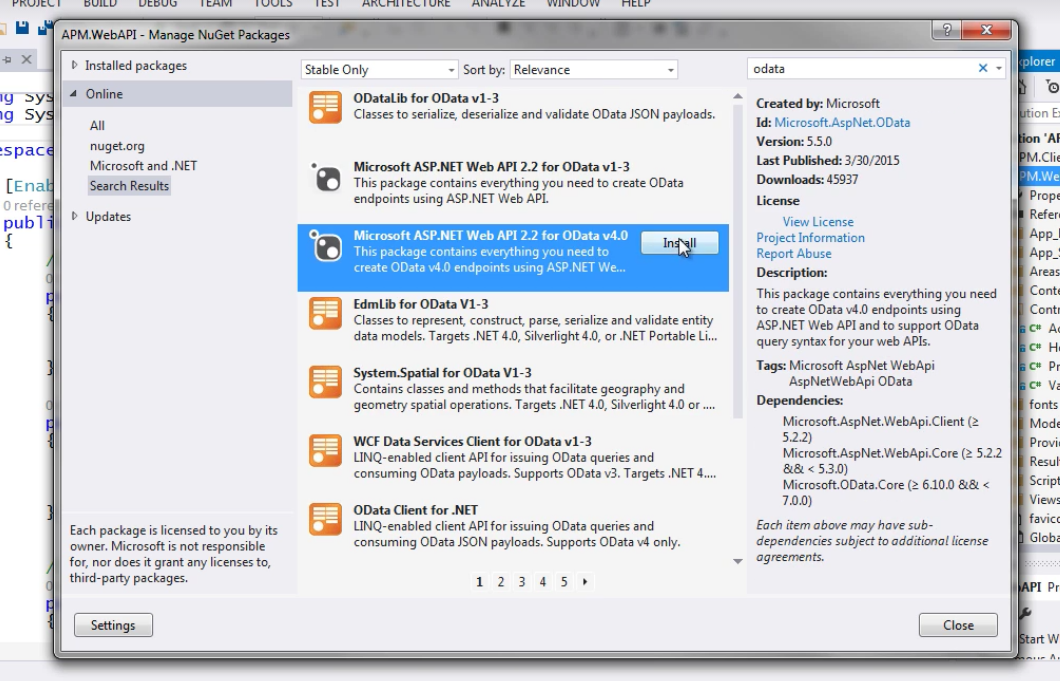
* For more advanced filtering, sorting etc.
* Open Data Protocol
* Standard way to access data using HTTP
* Protocol largely follows the REST convention

### OData Queries

* Allow construction of complex data queries
* Filter, shape, sort, and select data e.g.
  + **$filter**: contains(ProductCode, ‘GDN’) and Price gt 5 and price lt 20,
  + **$orderby**: Price desc
* Web API supports OData Queries with no code

### Enabling ODATA Queries in Web API

* Install OData MS Web API library



* Add **[EnableQueryAttribute()]**to the Get method
* Change IEnumerable to IQueryable
* Any repository data returning stuff should return IQueryable

// GET: api/Products and ODATA stuff now

[EnableQuery()]

public IQueryable<Product> Get()

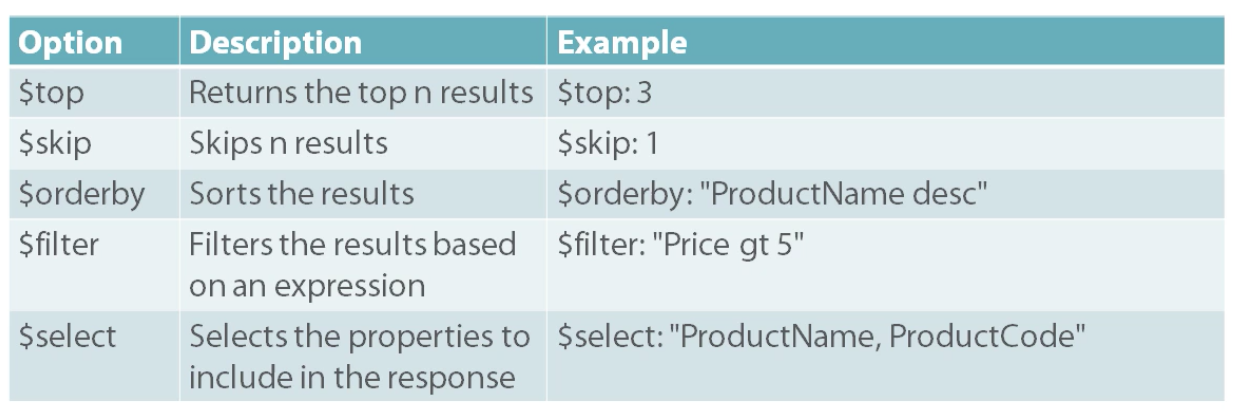
{

    return productRepository.Retrieve().AsQueryable();

}

### OData Query Options

* [http://localhost:51735/api/products?**$filter**=Price gt 19 & **$orderby**=Price desc](http://localhost:51735/api/products?$filter=Price%20gt%2019%20&%20$orderby=Price%20desc)



### Using OData in Angular

TODO: <https://app.pluralsight.com/player?course=angular-web-api-front-back&author=deborah-kurata&name=angular-web-api-front-back-m2&clip=6&mode=live>