

# Convert MVC Tables to Angular Templates Using Data Binding

---



**Paul D. Sheriff**

PRESIDENT, PDSA, INC.

@pdsainc [www.pdsa.com](http://www.pdsa.com) [psheff@pdsa.com](mailto:psheff@pdsa.com)



# Goals



**Add Angular to your project**

**Call Web API**

**Data binding in Angular**

**Angular filters**



# Add AngularJS to MVC

---



# Key Pieces of an Angular Application

**Module**

**app.module.js**

**Controller**

**product.controller.js**



# Module Is Like a Namespace in C#

## Wrapper Around...

**Controller(s)**

**Directives**

**Filters**



# Controller Is Like a Class in C#

**It is a JavaScript object**

**Contains data and functions**



```
(function () {  
  'use strict';  
  angular.module('app', []);  
})();
```



```
<div ng-app="app">
```

## Create app.module.js file

## Use an IIFE

## angular.module creates scope

- You supply module name 'app'
- Pass an empty array to create new module

## Link to the DOM using ng-app directive

- Add directive to HTML element
- html, body, div, etc

# Define Controller Two Ways

**ng-controller="Controller"**

**ng-controller="Controller as vm"**





```
<div ng-app="app"  
  ng-controller="ProductController">
```

```
(function () {  
  'use strict';  
  
  angular.module('app')  
    .controller('ProductController', ProductController);  
  
  function ProductController($scope) {  
    $scope.product = {  
      ProductId: 1,  
      ProductName: 'Video Training'  
    };  
  }  
})();
```

Add ng-controller on HTML element

Create controller on 'app' module

Angular passes \$scope variable to controller

\$scope is glue between view and model

Allows data binding

Add to \$scope...

- Variables (think properties)
- Functions (think methods)



```
<div ng-app="app"  
  ng-controller="ProductController as vm">
```

```
(function () {  
  'use strict';  
  
  angular.module('app')  
    .controller('ProductController',  
      ProductController);  
  
  function ProductController() {  
    var vm = this;  
  
    vm.product = {  
      ProductId: 1,  
      ProductName: 'Video Training'  
    };  
  }  
})();
```

Use “Controller as” syntax

Makes controller the “model”

Can use “this” instead of \$scope

- A normal JavaScript object

Define variables on the “model”



```
<div ng-app="app"  
      ng-controller="ProductController as vm">  
  
  <span>{{vm.product.ProductName}}</span>  
  
</div>
```

```
function ProductController() {  
  var vm = this;  
  
  vm.product = {  
    ProductId: 1,  
    ProductName: 'Video Training'  
  };  
}
```

---

Use {{}} for data binding

Bind to any variables defined on scope

“Controller as” syntax specifies which controller is being accessed



# Demo



## Add Angular to MVC application



# Build Web API

---



[HttpGet()]

0 references

```
public IActionResult Get() {  
    IActionResult ret = null;  
    PTCViewModel vm = new PTCViewModel();  
  
    // Get all Products  
    vm.Get();  
    if (vm.Products.Count > 0) {  
        ret = Ok(vm.Products);  
    }  
    else {  
        ret = NotFound();  
    }  
  
    return ret;  
}
```

Add Web API to MVC project

Web API uses our view model

View model gets data

Return Ok() status with products



# Demo



## Add Web API



# Data Access and Build HTML Table

---





```
function ProductController($http) {  
  var vm = this;  
  var dataService = $http;  
  
  vm.products = [];  
}
```

**Data access in Angular uses a data service**

**Data service is thru Dependency Injection**

- \$http is passed to controller
- Makes Web API calls for you

**Map \$http variable to a local variable**

- Allows flexibility later

**Initialize empty products array on scope**

- Fill array using Web API



```
function productList() {  
    dataService.get("/api/Product")  
        .then(function (result) {  
            vm.products = result.data;  
        }, function (error) {  
            handleException(error);  
        });  
}  
  
function handleException(error) {  
    alert(error.data.ExceptionMessage);  
}
```

## Build a productList function

- Get data from our Web API

**\$http data service is simpler than \$.ajax**

- Takes extra code to make it work

## JavaScript “promise”

Pass two parameters to .then() function

First parameter is function to retrieve data

Second parameter is function to handle exceptions











# Demo



## Consume Web API in Angular



Edit	Product Name	Introduction Date	Url	Price	Delete
	A New Product	06/08/2016	http://www.pdsa.com	\$22.00	
	Build an HTML Helper Library for ASP.NET MVC 5	11/05/2015	http://bit.ly/1myXBwj	\$499.00	
	Build your own Bootstrap Business Application Template in MVC	01/29/2015	http://bit.ly/1i8ZqZg	\$499.00	
	Building Mobile Web Sites Using Web Forms, Bootstrap, and HTML5	08/28/2014	http://bit.ly/1J2dcrl	\$499.00	

```
<tr ng-repeat="product in vm.products">
```

```

<td>{{product.ProductName}}</td>
<td>{{product.IntroductionDate}}</td>
<td>{{product.Url}}</td>
<td>{{product.Price}}</td>

```

```
</tr>
```

## Build HTML table

## Use ng-repeat directive

- “product in vm.products”
- “product” = local variable name
- “vm.products” = array in controller

## Use data binding

- {{name}} = variable name from repeat



```
<td>{{product.ProductName}}</td>  
<td>{{product.IntroductionDate | date: 'MM/dd/yyyy'}}</td>  
<td>{{product.Url}}</td>  
<td>{{product.Price | currency: "$"}}</td>
```

---

## Angular “filters” transform data

- ‘date’ format a date to a specific format
- ‘currency’ formats number as currency

## Others

- ‘lowercase’ transforms string to all lower case
- ‘uppercase’ transforms string to all upper case



# Demo



**Build product HTML table**



# Summary



Added Angular to project

Learned about modules and controllers

Created and consumed Web API

Worked with data binding

Filters for transforming data

Coming up in the next module...

- Build <select> list
- Bind input fields to controller
- Search for data

