Angular.js Introduction Tutorial

The solution can be pulled down from this URL: <https://www.dropbox.com/sh/f1oquf89ksitfmd/AAAdKklmf0hjePbk8DvOixpja>

This tutorial will walk you through creating an Angular.js application using Visual Studio. The application you will be building will be a simple commerce application. While the type of application has been re-created hundreds of times; I chose this type of application for a few reasons: (1) most people are familiar with the logic that is involved in this application and (2) we can touch many different aspects of angular so you can have a broad sense on how to use angular and how to approach different problems you may encounter.

# Note

Please note that this tutorial is intentionally setup in one file to make it easier to grasp the concepts of what is going on in the tutorial. I will be creating another tutorial that will take the code that is used at the end of this tutorial and we will take it a step further and make it more “real” world like by separating it into modules and making the JavaScript more structured. So just note that while this code works it isn’t necessary the recommended way to structure your angular applications for a real application.

# Part One: Setting up the Application and configuring the routes

1. Open up index.html and add the following line to the body tag :



The ng-app attribute is used to tell Angular where the root element of the application begins. Note the name you give the application here is also the same name that you will give to the module that you will create shortly. This attribute is also where angular begins the “compile” or bootstrap process when you first run the page Angular looks for this tag and then compiles all of the other Angular tags it finds into a JavaScript Object that you will code against. This is also what will setup the $scope object for you that you will use in your controllers later.

1. Next, you will add the ng-view directive to the div with the Id of “view”.



The ng-view directive tells Angular where to load all of the templates that you will create and assign to your routes. So when we navigate to /products; the template that we will specify for this route will be loaded into this div.

1. Open up the “app.js” file in the app folder and add the following lines to create our module and attach to angular.

This name is how we attach our objects to angular and basically define our container inside angular. Angular uses this name to reference objects that we will create. One way to think of Angular is to think of it as a dependency injection container that you register different objects with that Angular knows how to resolve when the different objects are referenced as dependencies.

The ‘ngRoute’ is an Angular module that we are pulling to handle our routing in our application.

1. Next, we need to configure the routes in Angular so that when angular receives a route it knows where to retrieve the template and controller for that route.

To configure the routes we need give a function to config container in Angular. This will instruct Angular in the bootstrap process to run this configuration. This is only run once when the index.html page first loads.



So what we are doing here is telling Angular that when it receives the route in the routeProvider for “/products” it needs to retrieve the “productListingController” and the template from the specified URL and load it in the ng-view container we created in the step # 2. Also, the last line instructs Angular that if it receives a route that it isn’t aware of then to redirect to ‘/products’ which is our default route.

# Part Two: Setting up the Product Listing Controller and Template

1. Next, we will create the productListingController by adding the following code :



What we are doing here is attaching a controller to our module so that the controller can be retrieved by Angular when it is needed. The $scope reference that is being injected is what is going to be used to bind our properties that we will be creating in the controller to our template that we will create.

1. Now that we have our controller setup let’s add a collection that will be used to list out all of the products when the application is loaded. Add the following code to the controller function:

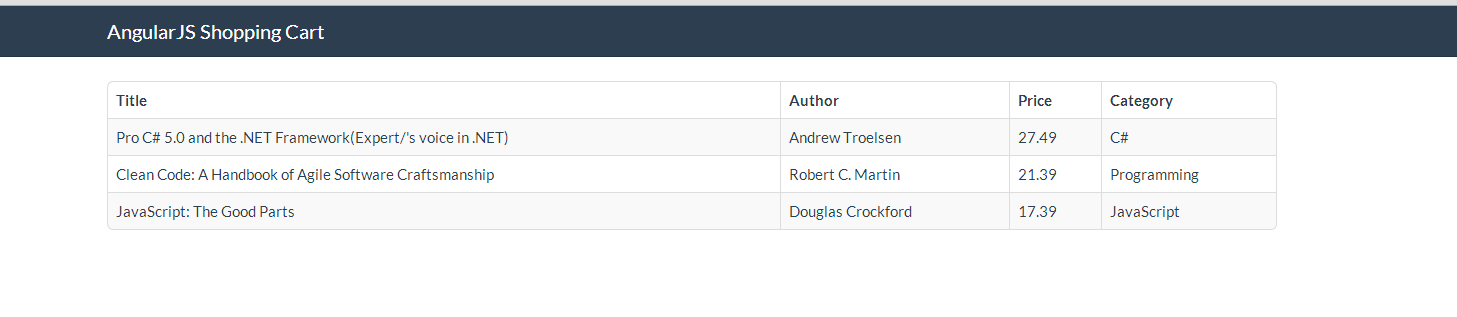


Here we just added a collection of products that share a common object with common properties that will be used to display in our template that we will create next. We will eventually move this code out of the controller into another structure but for now we will leave it like this.

1. Now that we have our controller setup, we need to add the partial that will be used to display the books when we navigate to ‘/products’. In the partials folder in the solution, right click and add a new HTML file named “product-listing”. Once the file is created delete all of the markup contained in the file. Then add the following markup below :



1. Now you can run the application and when you do you will see the following page :



Also note how Angular automatically handled the URL and applied the default URL for us. The URL is <http://localhost:PortNumber/#/products>