**Exercise04\_03\_01 – Step 1**



1. Create a new folder for the exercise called ***Exercise04\_03\_01***. Copy all of the files into it from the previous ***Exercise04\_02\_01***.
2. Go to the ***builds/angular/js*** folder and create a new folder called ***partials***. Create a new file in it called ***search.html***. To use multiple controllers, we are going to have to divide our code into more sections. Open up the ***index.html*** file and cut out all of the HTML from directly below the opening ***<body>*** element to directly before the first lower ***<script>*** element. Paste all of this code into the new ***search.html*** file. all of this content will end up being managed by our ***ng-view***.
3. Now, we define a new ***<div>*** directly below the opening ***<body>*** element. Place an ***ng-view*** directive in it as an attribute as follows:  
    ***<div ng-view></div>***
4. We can start to modify the code to use multiple controllers. First, we will remove the ***ng-controller*** directive from the ***<body>*** element:  
   ***<body class="bg-secondary">***
5. We need to load up the AngularJS ***routing*** library. Let’s link it in directly below our previous AngularJS ***<script>*** as follows:  
    <script src="lib/angular/angular.min.js"></script>  
    ***<script src="lib/angular/angular-route.min.js"></script>***
6. We are going to need a new JavaScript file to manage our controllers. In the ***builds/angular/js***, build a new file called ***controllers.js***. While we are in **index.html**, let’s link it in beneath our ***app.js*** link:  
    <script src="js/app.js"></script>  
    ***<script src="js/controllers.js"></script>***
7. Now we will build out our new ***controllers.js*** file. First, remove the entire ***controller()*** constructor from ***app.js*** and paste it into the new file. We will also make it into a new AngularJS module call ***myControllers***:  
   ***var myControllers = angular.module('myControllers', []);  
     
   myApp.controller('MyController', function MyController($scope, $http) {  
    $http.get('js/data.json')  
    .then(function(response) {  
    $scope.artists = response.data;  
    $scope.artistOrder = 'name';  
    });  
   });***
8. Now let’s modify the ***controller()*** method to use the new module object:  
   ***myControllers.controller('SearchController', function   
    MyController($scope, $http) {***
9. We are going to have to modify the way that our app works. Let’s go back into ***app.js***, we are going to need to add some ***dependencies*** to the app. It is going to need both ***ngRoute*** and our new controller module ***myControllers***:  
   ***var myApp = angular.module('myApp', [  
    'ngRoute',  
    'myControllers'  
   ]);***
10. The next move is to define how the ***ngRoute*** library is going to set up the paths for our app. We will use the ***config()*** method of the module in conjunction with the ***$routeProvider*** service, which actually takes an ***array*** as its parameter. Let’s scaffold that out as follows:  
    ***myApp.config(['$routeProvider', function($routeProvider) {  
       
    }]);***
11. Let’s build out the function, using some methods of the ***$routeProvider***. It makes use of its ***.when()*** method to define the ***properties*** of the ***route***, which is specified as it’s first parameter. The second parameter is an object which will actually set its ***View*** and its ***Controller***:  
    myApp.config(['$routeProvider', function($routeProvider) {  
     ***$routeProvider  
     .when('/', {  
     templateUrl: 'js/partials/search.html',  
     controller: 'SearchController'  
     });***}]);  
    Let’s run Gulp and give this a browser test, it should run as before. But there is a subtle difference if we examine the URL in the browser bar. A ***hashtag*** and a ***slash*** ( ***#/*** ) have been appended to the URL. What follows the hashtag is the URL of the ***partial*** which is currently loaded in the ***ng-view***. This will take on more meaning later.

**Exercise04\_03\_01 – Step 2**



1. We will create another partial to provide us a details page for the different artists. Move the file details.html into the ***builds/angular/js/partials*** folder. Open the file and examine its structure, taking a look at the AngularJS ***expressions*** that are used to display the information for each artist. Notice that we are using a variable as an ***array*** ***subscript*** to pick off an artist from the array of artists.
2. We will need to build another Controller for this file, so let’s open up ***controllers.js***. Let’s copy the first ***controller()*** method down, and we will modify it to handle our new partial. We can change it’s name to ***DetailsController***, and we do not need the ***$scope.artistOrder*** assignment:  
   ***myControllers.controller('DetailsController', function   
    MyController($scope, $http) {  
    $http.get('js/data.json')  
    .then(function(response) {  
    $scope.artists = response.data;  
    });  
   });***
3. Now let’s add another property to ***$scope*** called ***whichItem***. It will hold the ***array*** ***index*** of the artist whose details we want to display, which we will initially set to ***0***:  
    $scope.artists = response.data;  
    ***$scope.whichItem = 0;***
4. Let’s take care of the ***routing*** for the new partial. Open up ***app.js*** and copy the ***.when()*** method below itself, being careful to remove the ***semicolon*** between them. We can then modify the properties of its object to ***route*** the new partial, its ***route***, its ***View***, and its ***Controller***:  
    .when('/', {  
    templateUrl: 'js/partials/search.html',  
    controller: 'SearchController'  
    })  
    ***.when('/details', {  
    templateUrl: 'js/partials/details.html',  
    controller: 'DetailsController'  
    });***Go to the browser and look at the app. Place ***details*** at the end of the URL in the browser bar. We should be able to see the details for a single artist, the one that is ***indexed*** in the array by ***0***, which is what we set in our controller.

**Exercise04\_03\_01 – Step 3**



1. In this section we are going to wire our artists list to our details page, so that we can get information on any artist that we click, along with navigation between artists from the details page.
2. First let’s go to ***app.js*** and modify the ***/details*** route to take a parameter as follows:  
    ***.when('/details/:itemId', {***
3. Let’s open up ***controllers.js*** to modify the ***DetailsController*** to use the ***$routeParams*** service:   
   myControllers.controller('DetailsController', function   
    MyController($scope, $http***, $routeParams***) {
4. We can now use the service to modify ***$scope.whichItem*** to hold a variable for the ***array*** ***index*** of the artist whose details we want to display, which we initially set to ***0***:  
    $scope.artists = response.data;  
    ***$scope.whichItem = $routeParams.itemId;***Go to the browser and look at the app. Place ***details/1*** at the end of the URL in the browser bar. We should be able to see the details for another single artist, the one that is ***indexed*** in the array by ***1***. Remember the array is not sorted in our screen order, we will fix that later.
5. It is time to wire up the individual artists on the lists to their details to automate the URL posting. Let’s open up ***search.html***. We can add a hyperlink to each ***<li>*** element in our ***ng-repeat*** loop. It will use an AngularJS ***expression*** to insert the index of the artist array element. It will also enclose its entire content. Right below the ***<li>*** element with the ***ng-repeat***, add the following:  
   ***<a href="#/details/{{artists.indexOf(artist)}}">*** <div class="media d-flex align-items-center">  
    <img class="rounded-circle mr-3" ng-  
    src="images/{{artist.shortname}}\_tn.jpg"   
    alt="Photo of {{artist.name}}">  
    <div class="media-body">  
    <h5 class="my-0 text-dark">{{artist.name}}</h5>  
    <div class="text-secondary font-italic">  
    {{artist.reknown}}</div> </div>  
    <!-- media-body -->  
    </div>  
    <!-- media -->  
   ***</a>***Go to the browser and look at the app. Move down the list, and we should now see a ***hover*** happening. Click on various artists, and we should be able to get their details. Clicking on the middle dot of the navigation symbols on the details page should return us to the list.
6. We can now wire up the navigation arrows on the details page. Open up ***details.html***. Go down to the hyperlinks that implement the arrows, and we will first modify the one that displays the ***&lt;*** arrow. We will change the ***href*** to go to the ***/details*** page and use an AngularJS ***expression*** with a variable:  
    ***href="#/details/{{prevItem}}">&lt;</a>***
7. Now let’s modify the one that displays the ***&gt;*** arrow. We will change the ***href*** to go to the ***/details*** page and use an AngularJS ***expression*** with a variable:  
    ***href="#/details/{{nextItem}}">&gt;</a>***
8. Now we can put some JavaScript code into ***controllers.js*** that will manage the current array index as follows:  
    $scope.whichItem = $routeParams.itemId;  
      
    ***if ($routeParams.itemId > 0) {  
    $scope.prevItem = Number($routeParams.itemId) - 1;  
    }  
    else {  
    $scope.prevItem = $scope.artists.length - 1;  
    }  
    if ($routeParams.itemId < $scope.artists.length - 1) {  
    $scope.nextItem = Number($routeParams.itemId) + 1;  
    }  
    else {  
    $scope.nextItem = 0;  
    }***Let’s go to the browser and get any details page. Test the forward and back arrows. This project is a wrap. In the next project we will examine the AngularJS great features for form validation.