Angular Line of Business Applications: Hands-On Labs

# PREP: Starting Files

Repository: **https://github.com/DeborahK/AngularU2015**

Folder: **HandsOnLabs**

# LAB 1: Basics

Starting Files: **MovieHunter Take1 – Basics**

#### Key Tasks:

1. index.html
   1. Add the script tags for:
      1. angular.js
      2. app.js
      3. movieListCtrl.js
2. movieListView.html
   1. Add a column to display the movie director.
3. app.js
   1. Create the “movieHunter” module.
4. movieListCtrl.js
   1. Create the controller using the “controller as” syntax.
   2. Use contents of the **dataClipboard.txt** file to hard-code movie data
5. Run the application
   1. The list of movies should appear.
   2. The Hide/Show poster button should work.
   3. The filter feature should work.

#### Extra Tasks (as time permits):

1. Try out the F12 developer tools.
2. Change the sort order for the display of the movies.

# LAB 1 Code

*movieListCtrl.html*

<table ng-if="vm.movies.length">

<thead>

<tr>

<td>

<button type="button"

ng-click="vm.toggleImage()">

**{{**vm.showImage ? "Hide" : "Show"**}}** Poster

</button>

</td>

<td>Title</td>

**<td>Director</td>**

<td>Release Date</td>

<td>Rating</td>

</tr>

</thead>

<tbody>

<tr ng-repeat="movie in vm.movies | filter : {title:listFilter} | orderBy : 'title'">

<td>

<img ng-show="vm.showImage" style="width:50px;margin:2px"

ng-src="**{{**movie.imageurl**}}**" title="**{{**movie.title**}}**">

</td>

<td>

**{{** movie.title**}}**

</td>

**<td>{{ movie.director }}</td>**

<td>**{{** movie.releaseDate | date **}}**</td>

<td>**{{** movie.mpaa | uppercase**}}**</td>

</tr>

</tbody>

</table>

*app.js*

(function () {

"use strict";

var app = angular.module("movieHunter", []);

*}());*

#### movieListCtrl.js

(function () {

"use strict";

angular

.module("movieHunter")

.controller("MovieListCtrl",

[MovieListCtrl]);

function MovieListCtrl() {

var vm = this;

vm.movies = [

{

title: 'The Fellowship of the Ring',

director: 'Jackson',

releaseDate: '2001-12-19',

imageurl: 'http://www.coverwhiz.com/content/The-Lord-Of-The-Rings-The-Fellowship-Of-The-Ring\_small.jpg',

mpaa: 'pg-13'

},

{

title: 'The Two Towers',

director: 'Jackson',

releaseDate: '2002-12-18',

imageurl: 'http://www.coverwhiz.com/content/The-Lord-Of-The-Rings-The-Two-Towers\_small.jpg',

mpaa: 'pg-13'

},

{

title: 'The Point',

director: 'Wolf',

releaseDate: '1971-02-02',

imageurl: '',

mpaa: 'nr'

}

];

vm.title = "Search by Movie Title";

vm.showImage = false;

vm.toggleImage = function () {

vm.showImage = !vm.showImage;

};

}

#### }());

# LAB 2: Twitter Bootstrap - Option 1

Starting Files: **MovieHunter Take2 – Bootstrap**

#### Key Tasks:

1. index.html
   1. Add the link tag for **Content/bootstrap.css**
2. movieListView.html
   1. Examine the **class** attributes added to this file.
3. Run the application
   1. It should look much better!
   2. The features should still all work.

#### Extra Tasks (as time permits):

1. Check out the <http://getbootstrap.com/> site for additional styles.
   1. CSS tab
   2. Components tab
2. Try out some of the other styles.

LAB 2: Twitter Bootstrap - Option 2

Starting Files: **From End of Lab 1**

*Key Tasks:*

1. Download and install Bootstrap.
   1. If using Visual Studio, you can use NuGet Package Manager.
2. index.html
   1. Add the link tag for **Content/bootstrap.css**
3. movieListView.html
   1. Add the **class** attributes to this file as shown in the **styleClipboard.txt** file.
4. Run the application
   1. It should look much better!
   2. The features should still all work.

*Extra Tasks (as time permits):*

1. Check out the <http://getbootstrap.com/> site for additional styles.
2. CSS tab
3. Components tab
4. Try out some of the other styles.

# LAB 2 Code

#### movieListView.html

<div **class="panel panel-primary"** ng-controller="MovieListCtrl as vm">

<div **class="panel-heading"**>

<h2 **class="title"**>**{{**vm.title**}}**</h2>

</div>

<div **class="panel-body"**>

<!-- Filter the Title -->

<div **class="row"**>

<div **class="col-md-2"**>Filter by:</div>

<div **class="col-md-4"**><input type="text" ng-model="listFilter" /></div>

</div>

<div **class="row"** ng-show="listFilter">

<h3>Movies filtered by: **{{**listFilter**}}**</h3>

</div>

<div **class="table-responsive"**>

<table **class="table"** ng-if="vm.movies.length">

<thead>

<tr>

<td>

<button type="button"

**class="btn btn-info"**

ng-click="vm.toggleImage()">

**{{**vm.showImage ? "Hide" : "Show"**}}** Poster

</button>

</td>

<td>Title</td>

<td>Director</td>

<td>Release Date</td>

<td>Rating</td>

</tr>

</thead>

<tbody>

<tr ng-repeat="movie in vm.movies | filter : {title:listFilter}   
 | orderBy : 'title'">

<td>

<img ng-show="vm.showImage" style="width:50px;margin:2px"

ng-src="**{{**movie.imageurl**}}**" title="**{{**movie.title**}}**">

</td>

<td>

**{{** movie.title**}}**

</td>

<td>**{{** movie.director **}}**</td>

<td>**{{** movie.releaseDate | date **}}**</td>

<td>**{{** movie.mpaa | uppercase**}}**</td>

</tr>

</tbody>

</table>

</div>

</div>

</div>

LAB 3: Routing

Starting Files: **MovieHunter Take3 – Routing**

*Key Tasks:*

1. index.html
   1. Add the script tag for **Scripts/angular-route.js**
   2. Add the script tag for the new detail controller: **app/movies/movieDetailCtrl.js**
   3. Notice the <nav></nav> Bootstrap styles added to display the menu.
   4. Replace the ng-include directive with the ng-view directive.
2. app.js
   1. Add **ngRoute** as a dependency when defining the movieHunter module.
   2. Examine the routing code.
3. movieDetailView.html
   1. Review the code in this new file.
   2. Notice the use of the Bootstrap grid columns.
   3. Modify the back button to add the ng-href to activate the **#searchByTitle** route.
4. movieDetailCtrl.js
   1. Review the code in this new file.
5. movieListView.html
   1. Example the anchor tag used to navigate to the #showDetail route.
6. Run the application
   1. The main page should appear with a menu and welcome icon.
   2. Selecting the Search by Title menu option should display the list of movies.
   3. Clicking on a movie should display the movie detail.
      1. NOTE: The one hard-coded movie define in the movieDetailCtrl file will always appear here regardless of the movie selected on the Search by Title page.

*Extra Tasks (as time permits):*

1. Add another route to the list of routes for an “about” view (with no controller).
2. Rough out an HTML fragment for this view (about.html).
3. Add the route to **app.js**
4. Add a menu option for the route to **index.html**.
5. Run the application and ensure that your “about” page appears when selected.

# LAB 3 Code

#### index.html

<div class="container">

<div ng-view></div>

</div>

*app.js*

(function () {

"use strict";

var app = angular.module("movieHunter", [**"ngRoute"**]);

app.config(["$routeProvider",

function ($routeProvider) {

$routeProvider

.when("/", {

templateUrl : "app/welcomeView.html"

})

.when("/searchByTitle", {

templateUrl : "app/movies/movieListView.html",

controller : "MovieListCtrl as vm"

})

.when("/showDetail/:movieId", {

templateUrl : "app/movies/movieDetailView.html",

controller : "MovieDetailCtrl as vm"

})

.otherwise("/");

}]);

}());

*movieDetailView.html*

<div class="panel-footer">

<a class="btn btn-default"

ng-href="#searchByTitle"

style="width:80px">

<i class="glyphicon glyphicon-chevron-left"></i>

Back

</a>

</div>

LAB 4: Data Access

Starting Files: **MovieHunter Take4 – Data**

*Key Tasks:*

1. index.html
   1. Add the script tag for **Scripts/angular-resource.js**
   2. Add the script tag for **Scripts/angular-mocks.js**
   3. Add the script tag for the new common services module: **common/common.services.js**
   4. Add the script tag for the new service: **common/movieResource.js**
   5. Add the script tag for the mocking module: **common/movieResourceMock.js**
2. app.js
   1. Add **common.services** as a dependency when defining the movieHunter module.
   2. Add **movieResourceMock** as a dependency when defining the movieHunter module.
3. common.services.js
   1. Examine the code for this new module.
4. movieResource.js
   1. Example the code for this custom service.
5. movieListCtrl.js
   1. Inject **movieResource** as a dependency
      1. Be sure to add it to the controller method array as well
   2. Add the code to call the **movieResource** **query** method.
6. Run the application
   1. The main page should appear with a menu and welcome icon.
   2. Selecting the Search by Title menu option should display the list of movies.
      1. This should be different from the hard-coded list of movies.
   3. Clicking on a movie still displays the hard-coded movie.
7. movieDetailCtrl.js
   1. Inject **movieResource** as a dependency.
      1. Be sure to add it to the controller method array as well.
   2. Add the code to call the **movieResource** **get** method and pass in the movie Id.
8. Run the application
   1. The main page should appear with a menu and welcome icon.
   2. Selecting the Search by Title menu option should display the list of movies.
      1. This should be different from the hard-coded list of movies.
   3. Clicking on a movie should now display the appropriate movie.

# LAB 4 Code

*app.js*

var app = angular.module("movieHunter", ["ngRoute",

"common.services",

"movieResourceMock"]);

*movieListCtrl.js*

angular

.module("movieHunter")

.controller("MovieListCtrl",

["movieResource",

MovieListCtrl]);

function MovieListCtrl(movieResource) {

...

movieResource.query(

function (data) {

vm.movies = data;

});

}

}());

*movieDetailCtrl.js*

angular

.module("movieHunter")

.controller("MovieDetailCtrl",

["movieResource" ,

"$routeParams",

MovieDetailCtrl]);

function MovieDetailCtrl(movieResource, $routeParams) {

...

movieResource.get({ id: $routeParams.movieId },

function (data) {

vm.movie = data;

});

}

}());

LAB 5: Forms & Validation: Part 1

Starting Files: **MovieHunter Take5 – Forms**

*Key Tasks:*

1. index.html
   1. Add the script tag for **app/movies/movieEditCtrl.js**
2. app.js
   1. Add an **editMovie** route that routes to the movieEditView.html file using the MovieEditCtrl.
      1. NOTE: This should look very similar to the showDetail route.
3. movieEditView.html
   1. Example the code for the view.
4. movieEditCtrl.js
   1. Example the code for the controller.
   2. Notice the code both to get the movie and to save the movie on submit.
5. Run the application
   1. The main page should appear with a menu and welcome icon.
   2. Selecting the Search by Title menu option should display the list of movies.
   3. Clicking the edit button should display the edit page for the appropriate movie.
6. movieEditView.html
   1. Add a label and input box to the form for the director.
7. Run the application
   1. The director should now appear on the edit page.

LAB 5: Part 1 Code

*app.js*

.when("/editMovie/:movieId", {

templateUrl: "app/movies/movieEditView.html",

controller: "MovieEditCtrl as vm"

})

*movieEditView.html*

<div class="form-group">

<label class="col-md-2 control-label"

for="inputDirector">Director</label>

<div class="col-md-8">

<input class="form-control"

id="inputDirector"

name="inputDirector"

type="text"

placeholder="Director (required)"

ng-model="vm.movie.director" />

</div>

</div>

LAB 5: Forms & Validation: Part 2

Starting Files: **Lab 5 Part 1**

*Key Tasks:*

1. movieEditView.html
   1. Add validation attributes to the movie **title** input element.
   2. Add the **ng-class** attribute to the movie **title** form group.
2. Run the application
   1. The main page should appear with a menu and welcome icon.
   2. Selecting the Search by Title menu option should display the list of movies.
   3. Clicking the edit button should display the edit page.
   4. Make an edit to the movie title that breaks a validation rule.
   5. The label and field should turn red.
   6. Repeat these last two steps for several different validation rules.
3. movieEditView.html
   1. Add validation attributes to the **director** input element.
   2. Add the **ng-class** attribute to the **director** form group.
4. Run the application
   1. The main page should appear with a menu and welcome icon.
   2. Selecting the Search by Title menu option should display the list of movies.
   3. Clicking the edit button should display the edit page.
   4. Make an edit to the director that breaks a validation rule.
   5. The label and field should turn red.
   6. Repeat these last two steps for several different validation rules.

*Extra Tasks (as time permits):*

1. movieEditView.html
   1. Add the appropriate attributes for the other fields on the form.
2. Run the application and try out the new validation.

LAB 5: Part 2 Code

*movieEditView.html*

<div class="form-group"

ng-class="{'has-error':movieForm.inputMovieTitle.$invalid &&

movieForm.inputMovieTitle.$dirty}">

<label class="col-md-2 control-label"

for="inputMovieTitle">Movie Title</label>

<div class="col-md-8">

<input class="form-control"

id="inputMovieTitle"

name="inputMovieTitle"

type="text"

placeholder="Title (required)"

ng-model="vm.movie.title"

required

ng-minlength="3"

ng-maxlength="50" />

</div>

</div>

*movieEditView.html*

<div class="form-group"

ng-class="{'has-error':movieForm.inputDirector.$invalid &&

movieForm.inputDirector.$dirty}">

<label class="col-md-2 control-label"

for="inputDirector">Director</label>

<div class="col-md-8">

<input class="form-control"

id="inputDirector"

name="inputDirector"

type="text"

placeholder="Director (required)"

ng-model="vm.movie.director"

required

ng-minlength="3"

ng-maxlength="50" />

</div>

</div>

LAB 5: Forms & Validation: Part 3

Starting Files: **Lab 5 Part 2**

*Key Tasks:*

1. index.html
   1. Add the script tag for **Scripts/angular-messages.js**
2. app.js
   1. Add **ngMessages** as a dependency when defining the movieHunter module.
3. movieEditView.html (see clipboard.txt)
   1. Add validation messages for the movie title.
   2. Add validation messages for the director.
4. Run the application
   1. The main page should appear with a menu and welcome icon.
   2. Selecting the Search by Title menu option should display the list of movies.
   3. Clicking the edit button should display the edit page.
   4. Make an edit that breaks a validation rule.
   5. The appropriate validation message should appear.
   6. Repeat these last two steps for several different validation rules.

*Extra Tasks (as time permits):*

1. movieEditView.html (see clipboard.txt)
   1. Add validation message for the other fields on the form.
2. Run the application and try out the new validation.

# LAB 5: Part 3 Code

*app.js*

var app = angular.module("movieHunter", ["ngRoute",

"ngMessages",

"common.services",

"movieResourceMock"]);

*movieEditView.html*

<div class="form-group"

ng-class="{'has-error':movieForm.inputMovieTitle.$invalid &&

movieForm.inputMovieTitle.$dirty}">

<label class="col-md-2 control-label"

for="inputMovieTitle">Movie Title</label>

<div class="col-md-8">

<input class="form-control"

id="inputMovieTitle"

name="inputMovieTitle"

type="text"

placeholder="Title (required)"

ng-model="vm.movie.title"

required

ng-minlength="3"

ng-maxlength="50" />

</div>

<span class="help-block"

ng-if="movieForm.inputMovieTitle.$dirty">

<span ng-messages="movieForm.inputMovieTitle.$error">

<span ng-message="required">Movie title is required</span>

<span ng-message="minlength">Movie title must be at least 3 characters </span>

<span ng-message="maxlength">Movie title cannot exceed 50 characters </span>

</span>

</span>

</div>

*movieEditView.html*

<div class="form-group"

ng-class="{'has-error':movieForm.inputDirector.$invalid &&

movieForm.inputDirector.$dirty}">

<label class="col-md-2 control-label"

for="inputDirector">Director</label>

<div class="col-md-8">

<input class="form-control"

id="inputDirector"

name="inputDirector"

type="text"

placeholder="Director (required)"

ng-model="vm.movie.director"

required

ng-minlength="3"

ng-maxlength="50" />

</div>

<span class="help-block"

ng-if="movieForm.inputDirector.$dirty">

<span ng-messages="movieForm.inputDirector.$error">

<span ng-message="required">Director is required</span>

<span ng-message="minlength">Director must be at least 3 characters </span>

<span ng-message="maxlength">Director cannot exceed 50 characters </span>

</span>

</span>

</div>