**Workshop 3 – AngularJS**

Please get the ‘FrontendCoding’ project from github. Run the necessary commands for node packages and bower dependencies.

**Part 0. Find the bug!**

**Part I. Routes**

1. Include angular-route.js in index.html (TODO #1):

<script type="text/javascript" src="bower\_components/angular-route/angular-route.js"></script>

2. Load the module in the application by adding it as a dependent module in app.js file (TODO #2):

var hrApp = angular.module('hrApp', ['ngRoute']);

3. Add the default route pointing to the main page (main.html, MainController.js) (TODO #3):

4. Configure a new route for the following scenario: when the user accesses the ‘#/numbers’ url from the browser, he should be redirected to the ‘/math’ route. (TODO #4)

5. Run grunt connect:server. Try to access the ‘#/numbers’ route. You should be redirected to the main page. If not, try to solve the problem. (Hint: TODO #5)

6. Add the following route (TODO #6):

$routeProvider.when('/math', {

templateUrl: 'views/demo/math.html',

controller: 'MathController'

})

7. Open math.html file. Add two input fields of type number and a button that will call the calculate() function when pressed. (TODO #7)

8. Open MathController.js and complete it with the necessary lines to calculate the four values from the lower part of the html template. (TODO #8)

9. Access ‘#/numbers’ to check your work.

**Part II. HTTP Request**

Check that you can access the backend application deployed in Docker:

[**http://localhost:8080/datamodel/employees/findAll**](http://localhost:8080/datamodel/employees/findAll)

(If you have problems with your Docker, you can use the following URL:

<http://hrapp-zth.rhcloud.com/hrapp>/employees)

10. Add the following route in app.js (TODO #9):

.when('/demoRequest', {

templateUrl: 'views/demo/request.html',

controller: 'RequestController'

})

11. Complete RequestController.js by using $http to make an AJAX request:

url: http://localhost:8080/datamodel/jobs/findAll

(url for those without Docker: http://hrapp-zth.rhcloud.com/hrapp/jobs)

method: GET

Assign the response data to $scope.jobList. (TODO #10)

12. Check your work in the browser.

**Part III. Services**

13. Check the Employees menu in the browser.

14. Open MenuService.js and complete it with the following items for the Employee menu (TODO #11):

{

label: "Employee list",

url: "#/employeeslist"

},

{

label: "Add employee",

url: "#/employeeadd"

}

15. Open MenuController.js and populate the employeeActionList using the Value defined at the previous step. (TODO #12)

16. Check again the Employees menu in the browser.

17. Open MathController.js. Refactor your calculate() function using the MathService service by injecting it into the controller and calling its functions for: add, substract, multiply and divide.(TODO #13)

Two of these methods are missing. Complete MathService. (TODO #14)

18. Check your index.html file to be sure that all of the used resources are added.

19. Check your work in the browser.

**HR App:**

At the end of this workshop we will have a listing of all employees and the possibility to view the employee’s details.

1. Add routes for Employee List page and Employee View page (TODO #HR1):

.when('/employeeslist', {

templateUrl: 'views/employeelist.html',

controller: 'EmployeeListController'

})

.when('/employeeview/:employeeid', {

templateUrl: 'views/employeeview.html',

controller: 'EmployeeViewController'

})

2. Let’s start with the Employee list page.

We have the template and controller for the Employees list page. Complete the controller using the $http service to get the employee list from the server.

First inject commonResourcesFactory – a factory you can use to retrieve URLs for the AJAX calls. (TODO #HR2) (or commonResourcesFactoryBackup for those without Docker)

Use commonResourcesFactory.findAllEmployeesUrl to load the data from the server. (TODO #HR3)

3. Open the browser to see the Employee List page (you can get to this page from the menu bar).

4. Add a column for actions in employeelist.html and add the ‘View’ button in it for every row of the table. When clicked the viewEmployee() function should be called with the id of the employee as parameter. (TODO #HR4, TODO #HR5)

5. The viewEmployee(employeeId) has already been defined in EmployeeListController. Take a look at it!

6. Let’s continue with the Employee View page.

The template html file has been created for you. Take a look at it!

Open EmployeeListController.js. You have an empty employee object on the $scope. You will have to make an http request to get the employee entity for the id from the URL.

$http and $routeParams services has been injected in the controller. Try to take the employeeid parameter and use it to construct the url for the HTTP GET call. (TODO #HR6)

7. Refresh the page in your browser. Navigate to the Employee View page and check that everything works as it should.

8. The Employee View page has a Back button. Implement the back() function so that it takes you back to the Employee List page. Use $location service.

**Optional:**

Create a new route: ‘#/user’.

Add multiple input fields for user details: firstName, lastName, id (cnp), age, etc.

Add the following buttons:

- Back – pressing the button will call a function defined in the page’s controller that will redirect the user to the main page

- Reset – pressing the button will reset the fields in the page

- Save – pressing the button will save the user’s details in a list stored in a service (UserService) and notify the user that the data has been saved (using an alert)

- Show/Hide User List – pressing the button will show/hide the list stored in the service and render it in the page (using ng-if and ng-repeat)