**Workshop frontend day 2**

**AngularJS Introduction**

1. Create new project in WebStorm(‘angularhr’);

2. Copy from GitHub project from the folder ‘day2-files’ the following files: ‘bower.json’, ‘Gruntfile.js’ and ‘package.json’;

3. Run from CMD(inside the project folder) the following commands:

* npm install
* bower install

4. Create file ‘index.html’;

5. Create folder ‘scripts’;

6. Index.html should look like this:



7.Create in scripts folder the app.js file where you should define the hrApp angular module.

8.Create in the scripts folder the MainController.js file where you should define the angular controller MainController. MainController should do:

1. Inject $scope and $rootScope.
2. Print the name value from the scope of the MainController.

9. grunt connect:server in the cmd console.

10. Create scripts/DescriptionController.js and populate an angular controller with the same name with the following code:

$scope.title = 'Two Way Binding Demo';

$scope.childtemplate = 'templates/childscope.html';

$scope.resetFirstVariable = function() {

$scope.firstVariable = undefined;

};

$scope.setFirstVariable = function(val) {

$scope.firstVariable = val;

};

11. Add to the index.html a <div></div> which includes a html template controlled by DescriptionController.

12. Complete the html template:

<div class="container">

<div class="jumbotron" ng-show="descriptionShow">

<h1>{{title}}</h1>

<p class="lead">

A simple demo application.

</p>

<p class="lead">

In this application we will use AngularJS framework.

</p>

</div>

<p>

<a class="btn btn-primary btn-lg" role="button" ng-click="toggleDescriptionShow()">Show/Hide Description</a>

</p>

</div>

13. Use the directive ‘ng-hide’ to show/hide the description;

14. Create ScopesController in the scripts folder with the following content:

$scope.title = 'Two Way Binding Demo';

$scope.childtemplate = 'templates/childscope.html';

$scope.resetFirstVariable = function() {

$scope.firstVariable = undefined;

};

$scope.setFirstVariable = function(val) {

$scope.firstVariable = val;

};

15. Create ‘scopes.html’ in the views folder:



16. Use ng-include to include the scopes.html in index.html.

17. Add in ‘menu.html’ a new element in the links list, below the Home link the following element:

<li class="dropdown" ng-controller=”MenuController”>

<a href="" class="dropdown-toggle" data-toggle="dropdown">Demo actions<span class="caret"></span></a>

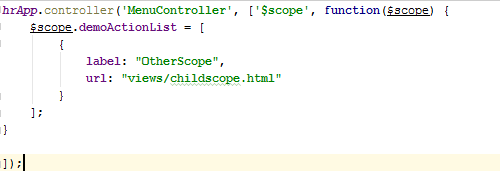
<ul class="dropdown-menu" role="menu">

<li ng-repeat="action in demoActionList"><a href="{{action.url}}">{{action.label}}</a></li>

</ul>

</li>

18. Create MenuController:



19. Add ‘childscope.html’ view:

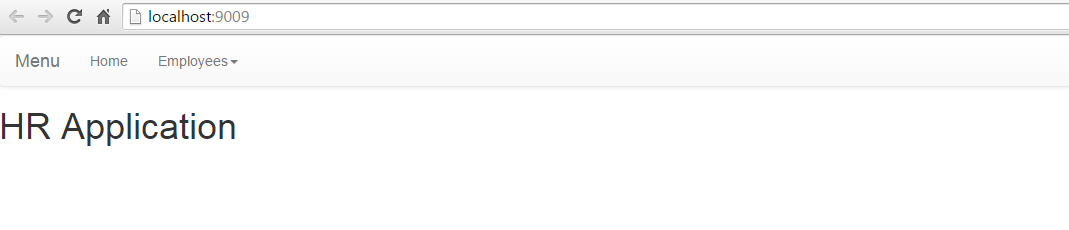
20. Add OtherController where you should implement the setTitle function.

21. Create a new route in your application ‘/demomath.html’.

22. Show the link for this route in the ‘Demo actions’ dropdown from the menu;

23. In this page create two inputs(‘Number A’ and ‘Number B’). Below this inputs show the 4 math operations(addition, subtraction, multiplication, division);

24. Create a new empty project. This should be start of your final application. Create the index.html and the needed controllers in order to obtain a menu like this.



You should use ng-include to include a menu.html template, and a ng-repeat to iterate through the Employees options. The options from the Employees menu should be “List Employees”…

For the menu use the navbar class from bootstrap: <http://getbootstrap.com/components/#navbar>

Exercitii suplimentare:

1. Create a new route in your application ‘/shoppingcart.html’.
2. Show the link for this route in the ‘Demo actions’ dropdown from the menu.

Overview:

The new page should emulate an online shop, where the user can choose products by clicking on their images and add them to a shopping cart. After adding them to the shopping cart, the client should be able to modify the quantity and total price should be automatically updated.

After finishing the order, the client clicks “Finish order” and the order is stored in memory with the current date and displayed in a table when the history of the orders are requested.

1. Download 4 pictures of random objects and add them to the projects in a folder called “images”.
2. Create a <div></div> and add each image in a fixed size rectangle. Also add the name of the product beneath each picture.

Hint: create a Javascript array with objects holding the name, the URL (and other future properties) of the products. Use ng-repeat to display them in page.

1. Using the proper directive, display the price of the object (anywhere on the page) when hovering over an image.
2. When clicking a product (on the picture or the name), the product should be added to the shopping cart. The shopping cart can be viewed/hidden when clicking the button “Go to shopping cart”.

The shopping cart is a html table where each row contains a product with the image, name and price on different columns.

When the shopping cart is displayed, all the products are hidden and viceversa.

Hint: Use AngularJS directives for hiding/displaying HTML.

Hint2: Use ng-repeat to create the data in the table.

1. Under the table, display the total price of the order.
2. Add an extra column in the table called “Quantity”. This extra column should contain an input of type number. After modifying the quantity, the total price of the order should be automatically updated.
3. For each product, define a property called: “availableQuantity”. If the quantity required by the client is greater than the available quantity, an warning message with red text should be displayed in a paragraph under the table.
4. Add a button called “Finish order”. When clicking the button, the current content of the shopping cart should be deleted. Also, you need to save some details about the order in an array: date of the order, total price, total amount of products.

Display/hide the orders history when clicking a button.