

JBoss Community

Web Technologies in Java EE Hands-On Lab

JAX-RS, JSON-P, Angular.js

Agenda

- Look at what we have
- Expose REST endpoints and setup Angular JS frontend
- Call REST endpoints from new Angular controllers
- Implement custom directive in Angular JS

Application

- Very simple message/blog application
- Consumes JSON via REST

web-00 : what we have?

- Standard Maven project structure
 - pom.xml
 - Java EE7 dependency
 - webapp/resource/js/libs
 - JavaScript sources and libs prepared

web-01 : REST endpoints

- Expose following endpoints:
 - GET /messages
 - POST /messages
- Produce / consume JSON
- MessageEndpoint JAX-RS endpoint
 - /api/v1/messages
 - 1. @ApplicationPath("api/v1")
 - 2. resource: @Path("messages")
 - 3. metody @GET @POST

web-01 : Connect Angular module

- Use **ng-app** directive to define “simpleChat” module

web-02 : Angular JS services

- Introduce new service in `services.js` using `factory` method and `$resource` object (needs dependency to `ngResource`)
- Usage
 - `$resource(url, [paramDefaults], [actions], options);`
- Define

```
return $resource("./api/v1/messages", {}, {
  get: {method: 'GET', cache: false, isArray: isTrue},
  save: {method: 'POST', cache: false, isArray: false}
```

web-02 : Angular JS controller

- Introduce new controller in `controllers.js` using `controller` method
- `<your_module_name>.controller('MyNewController', [dependencies, function MyNewController(){
...
}]);`
- This controller needs to use (new dependency) your new service to call “save” (post) request

web-02 : Angular JS controller

- In the controller introduce new global (`$scope`) available function call “`submit`” which will accept new message and save it
- We want to have the form clear after `submit!!!`
- Directives needed:
 - `ng-submit`
 - `ng-controller`
 - `ng-model`

web-03 : Angular JS controller

- Introduce new controller to get a list of messages. Definition of a new controller is similar to previous task
- You need to use your service to get all messages again
- e.g `MessageService.get({}, ...)`
- Use **ng-repeat** to display messages in HTML

web-04: new attribute at Message and Angular filter

- Add new attribute of type Date in the Message model class
- In controller (where you're adding new message) initialize this new attribute with `Date.now()` js function
- Display this new date attribute in HTML in following format: 24/12 14:36:51

web-05: Angular custom directive

- Create new custom directive in `directives.js` (don't forget to include this script and add new dependency to your app) using `directive` method
- Restrict this new directive type to 'E' as element and link the template defining the HTML
- Replace the original code in HTML with your new directive