Seminar #4

Miroslav Blaško

1 Goal

You will get experience with subset of Spring related to web applications development.

2 Getting Ready

- Ensure you have available the software stack installed during the first lab (PostgreSQL server, PgAdmin, Netbeans 8.2).[1]
- Clone the GIT repo from `https://gitlab.fel.cvut.cz/ear/reporting-tool-seminars`, switch to the branch `WS2017-seminar-4-problem`[2].
- Open the project in Netbeans 8.2 and resolve any warnings/errors in configuration reported by Netbeans.[3]

At the end of the current week, the branch `WS2017-seminar-4-solution` becomes available where you can check your solutions.

3 Familiarization with the Project

The project is a small and JPA-ized excerpt of a bigger one developed for safety management in the czech aviation industry. The system allows to create safety reports and classify them. The system is based on JPA, Spring, and ReactJS. First, clean and build the project.

Before proceeding, go through the structure of pom.xml, Java sources at `src/main/java` and resources at `src/main/resource`.

[1] Due to compatibility issues JDK 9 cannot be used to run reporting tool project.

[2] Run `git checkout WS2017-seminar-3-problem`

[3] In case you are using Intellij Idea, you can install Lombok Plugin from `https://plugins.jetbrains.com/plugin/6317-lombok-plugin` to resolve errors due to missing getters and setters within the model package. Note, that getters and setters should be generated automatically due to Lombok annotations `@Getter` and `@Setter`.

4 (0.5pt) Task 1: Initialize Application With Default Tags

The goal of this task is to implement system initialization service which populates application's database with default tags such as "partial", "outdated", "needs-double-check", "double-checked". Due to database integrity constraints related to author of a tag it is necessary to create and persist a system user as well (pick an username, a password, a first name etc.). Default tags should be entered into database on deploy of the application into a web server[4]. Run all tests[5] to find out failing ones. Fix failing tests by implementing the system initialization service.

5 (0.5pt) Task 2: Create User Preferences Service

Implement user preferences service that stores user's selection of a time zone (see class `UserPreferencesService`)[6]. Note usage of this service within the rest controller class `UserPreferencesController`[7].

6 (optional) Task 3: Migrate Spring Annotations into XML

Spring's XML Schema-based configuration[8] is an alternative to Spring's annotation programming model. The goal of this task is to migrate some parts of Spring's configuration within the application from the annotation programming model into XML configuration.

[4] Hint: there are multiple ways how to listen to a deploy event. We suggest to use `@javax.annotation.PostConstruct` annotation of a method within a singleton service bean. Such method is executed when spring bean is registered within an application context, which should be sufficient for the purpose. In case you need to use transactions within the "post-construct method", programmatic transaction management must be utilized as explained in `https://stackoverflow.com/questions/17346679/transactional-on-postconstruct-method`.

[5] Hint: Your IDE most-likely has option to run all JUnit tests within the project. Use it instead of execution of maven command manually (e.g. by 'mvn test') as it usually provides easier way to find failing tests and execute them again.

[6] Hint: Possible issues are discussed at `http://ankursinghal86.blogspot.cz/2014/07/injecting-prototypesession-bean-into.html`

[7] Examples of valid time zone ids are "Europe/Prague", "Etc/GMT+2", "US/Pacific", "HST"

[8] `https://docs.spring.io/spring/docs/5.0.0.RELEASE/spring-framework-reference/appendix.html#xsd-configuration`