MVC 1.0 HANDS-ON LAB

Global Prerequisite:

- Laptop
- Java 8
- Glassfish Nightly downloads from
  - http://download.oracle.com/glassfish/4.1/nightly/index.html
- Eclipse or IntelliJ Idea or even NetBeans :) and configure the just downloaded glassfish.
- Maven

Project repo: https://github.com/trance1st/mvc-lab

1. **Project setup & Hello World**
   Checkout the master branch of the project and import it into your IDE.
   Build & Run the project.

> *i) Start the JavaDB database process:*
> ***/home/bogdan/Documents/glassfish4/javadb/bin/startNetworkServer***
> *(on Windows add the option –noSecurityManager)*
>
> *ii) If you don't have the GlasshFish server integrated in the IDE you can manually start the server by running:*
> ***./home/bogdan/Documents/glassfish4/glassfish/bin /startserv***
> *To manually deploy the application go to GlashFish Administration console (http://localhost:4848/common/index.jsf) and deploy the app*

   Familiarize with the project.

2. **Create a login page**

   Hints:

   - Create a controller with two methods: one that returns the jsp login page and other method that handles the form submission.

- Use the following “bussines objects”:
    - UserContext – holds the current logged user
    - UserManager – all that you need to interact with users
- You can handle form submits in two ways
    i) Using JAX-RS @FormParam annotation
    ii) Annotate with @FormParam fields of a bean Model

    See: http://www.bennet-schulz.com/2015/11/mvc-10-in-java-ee-8-form-validation.html

3. Display all the sessions as well as the session by the currently logged in user

Hints:

- Create a controller that puts into the model the sessions and returns sessions.jsp
- Maybe you need two separate methods in the controller that are listening to two different paths
- If you want to get the current logged in user, create the following field:
    @Inject
    @LoggedIn
    private User currentUser
- If you want to do anything with sessions, inject and use SessionManager

4. Submit a proposal and validate the input

Hints:

- You should create a controller (or reuse existing) again with a couple of methods: one for showing the form (GET) and another one for handling its submission (POST)
- For accessing the validation result inject the class BindingResult into your controller.
    Use the following methods:
    getAllViolations() - Returns an immutable set of all constraint violations detected.
    isFailed() - Returns true if there is at least one binding error or constraint violation.
- The method that handles the form submit must be annotated with
    @ValidateOnExecution(type = ExecutableType.NONE)

- Consider creating another @Model bean that holds the validation error messages and can be accessed from the JSP
- You can handle form submits in two ways
    iii) Using JAX-RS @FormParam annotation
    iv) Annotate with @FormParam fields of a bean Model

See: http://www.bennet-schulz.com/2015/11/mvc-10-in-java-ee-8-form-validation.html

- You can you the following validation annotations: @Size(min = 8, max = 100)

5. TODO