

Issue Tacker - 01.03.2017

Issue Tracker is a system that reports bugs. Your task is to build one. You have to use **TomEE**, **Hibernate**, **JSP** and the **custom MVC Framework** for that purpose. Your application needs to have **six** pages.

Views

- **Home**
 - Entry point of the application
- **Issues**
 - All the issues are reported here
 - You can search, add and edit them
- **Log in**
 - Log in with an existing user
- **Create User**
 - Creates a user in the database
- **Add Issue**
 - Add and issue in the database
- **Edit Issue**
 - Edit an existing issue

Data Models

Create the required **entities**. Use the appropriate **data types**.

- **User**
 - Username
 - Full Name
 - Password
 - Role
- **Role**
 - User or Admin
- **Issue**
 - Name
 - Priority
 - Status
 - Creation Date
 - Author – the user that creates the issue
- **Priority**
 - Low, Medium, High
- **Status**
 - New, Solved

Functionality

- **Guests** (anonymous users) can register an account with their own username, email and password.
- **Guests** can login by username and password.
- **Logged-in users** can logout.
- **Guests** can view the home page.
- **Logged-in users** can create new issues.
- **Logged-in users** can edit or delete their own issues.

- **Administrators** can add, edit or delete all issues
- When guest user tries to access a page that is allowed only for logged in user he should be redirected to the login page

1. Design the Database

Design **entity classes** and create a **database** to hold the **users** and the **issues**. Use **Hibernate** implementation of **JPA**.

2. Use DTO

Use appropriate **binding** and **view** models.

3. Use Repositories

Create repositories for your **entities**.

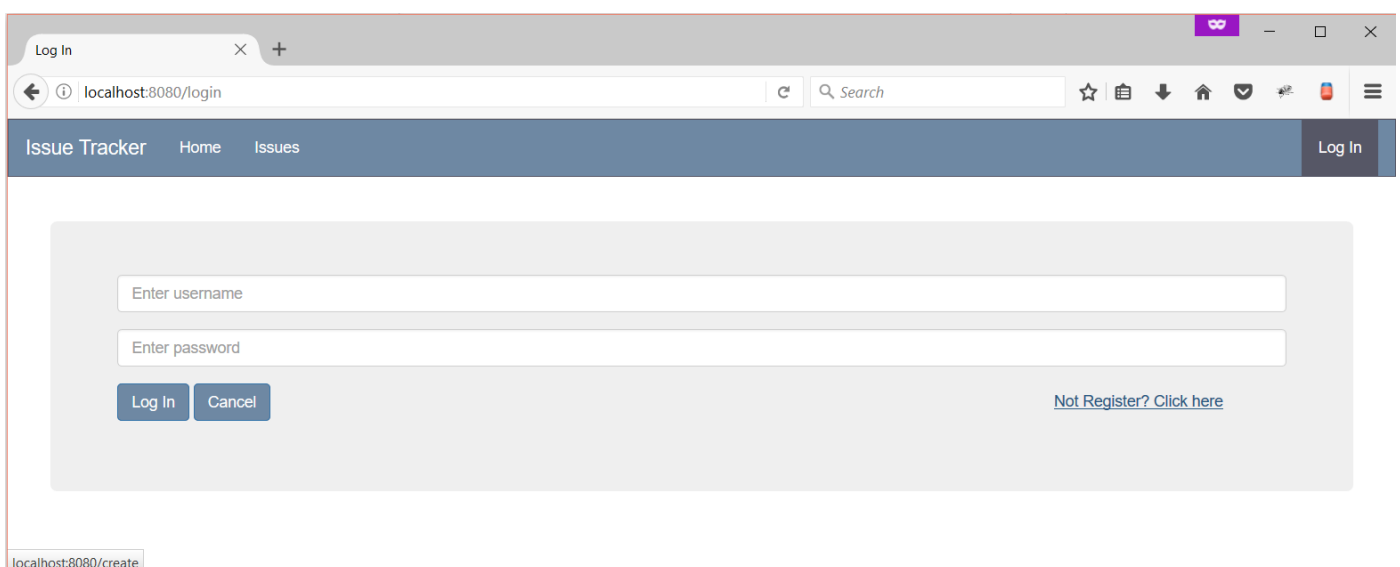
4. Use Services

Create services to hold your **business logic**.

5. Implement User Registration, Login and Logout

The guest users can register and log in the forum. He can provide to:

- **Register**
 - **Username** - must be between **5** and **30** symbols
 - **Full Name** - must be at least **5** symbols
 - **Password** - at least **8** symbols. It should contain a **capital letter**, a **number** and one of the following **signs**: **[!@#\$\$%^&*,.]**
 - **Confirm Password** - must **match** the provided password
- **Login**
 - User can log in with **username and password**
- **Logout**
 - When logged in the user should have option to **log out**



The screenshot shows a web browser window with the address bar at 'localhost:8080/login'. The page has a dark blue header with 'Issue Tracker' on the left, 'Home' and 'Issues' in the middle, and a 'Log In' button on the right. The main content area is a light gray box containing a login form. The form has two input fields: 'Enter username' and 'Enter password'. Below the fields are two buttons: 'Log In' and 'Cancel'. To the right of the 'Cancel' button is a link that says 'Not Register? Click here'. The browser's status bar at the bottom shows 'localhost:8080/create'.

Create User

localhost:8080/create

Issue Tracker Home Issues Log In

t.dimitrov@softuni.bg

Teodor Dimitrov

.....

.....

Create Cancel

6. Implement Validations

If any of the constraints fails **alert** a message.

Create User

localhost:8080/create

Issue Tracker Home Issues Log In

Error! The username should be between 5 and 30 symbols

Error! The password should contain a capital letter, a number and a sign [!@#\$\$%^&*,.]

Error! The full name should be at least 5 symbols

Enter username

Enter full name

Enter password

Repeat password

Create Cancel

Create User

localhost:8080/create

Issue Tracker Home Issues Log In

Error! Passwords are not matching

Enter username

Enter full name

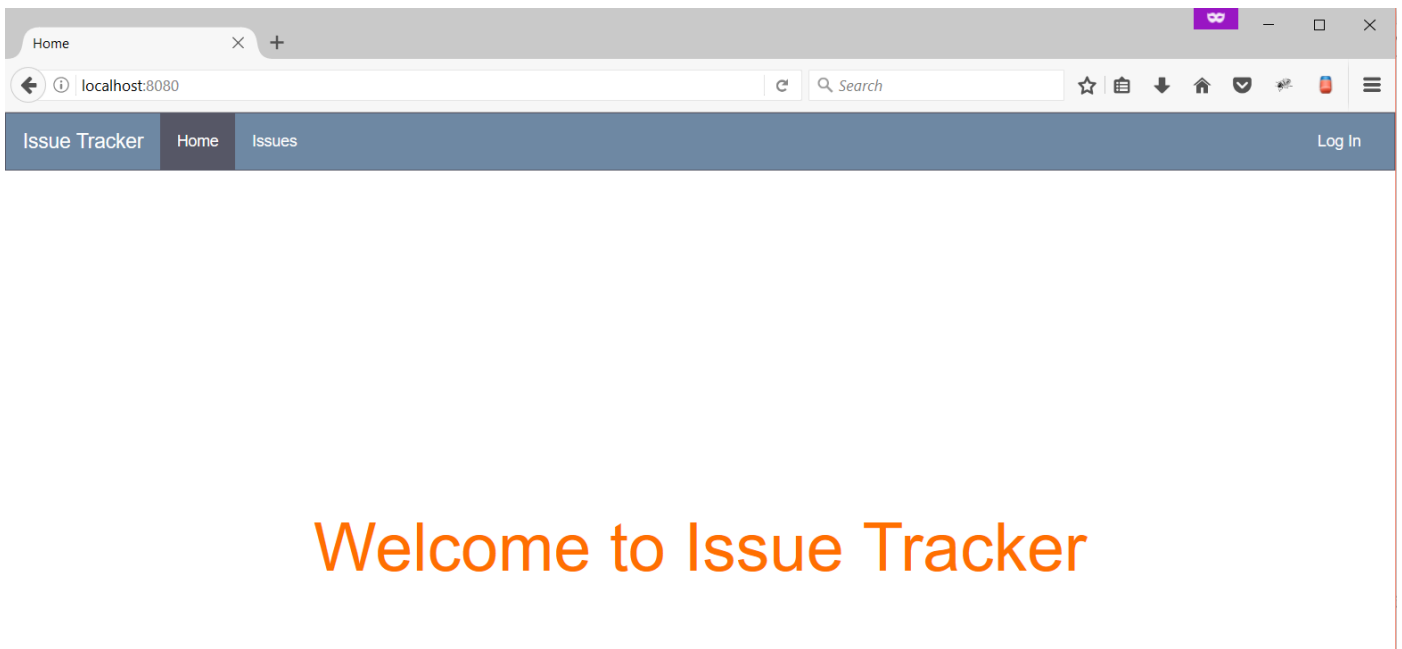
Enter password

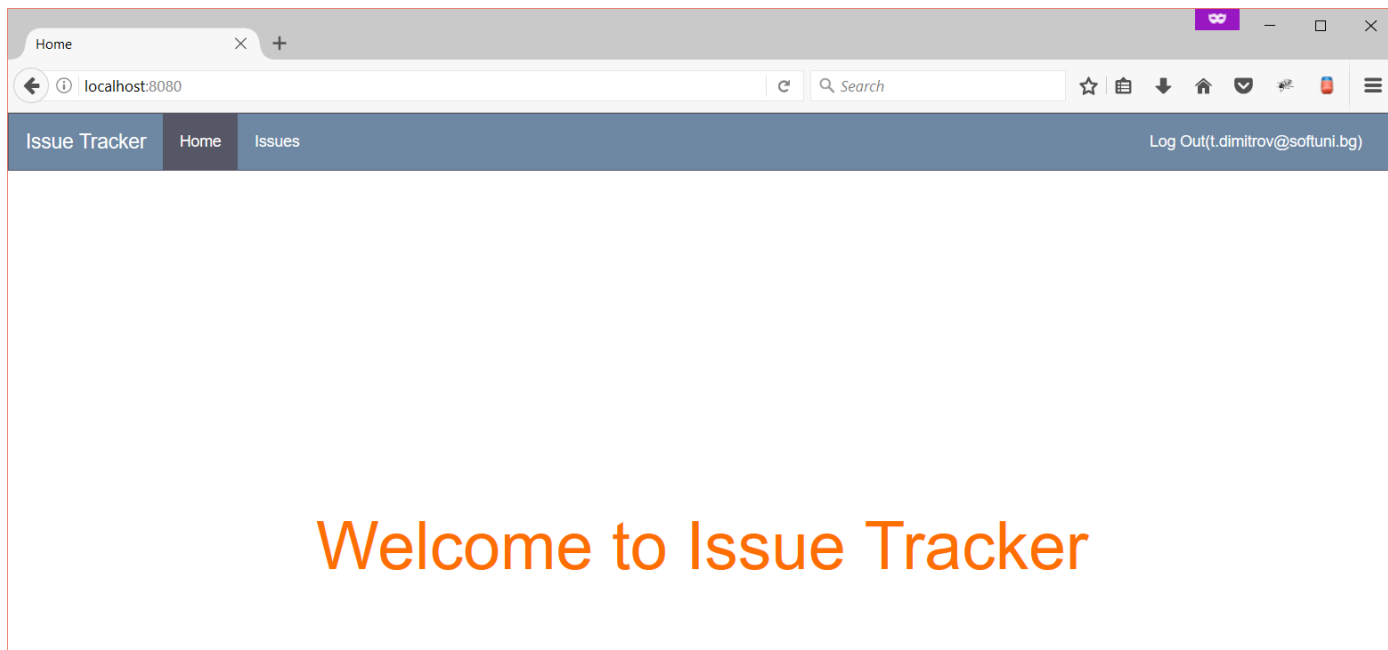
Repeat password

Create Cancel

7. Implement Home Page

The home page is available to **guests** and should have a small **menu**. If the user is logged in a **logout** option should appear. **Edit menu.jsp** to achieve this.

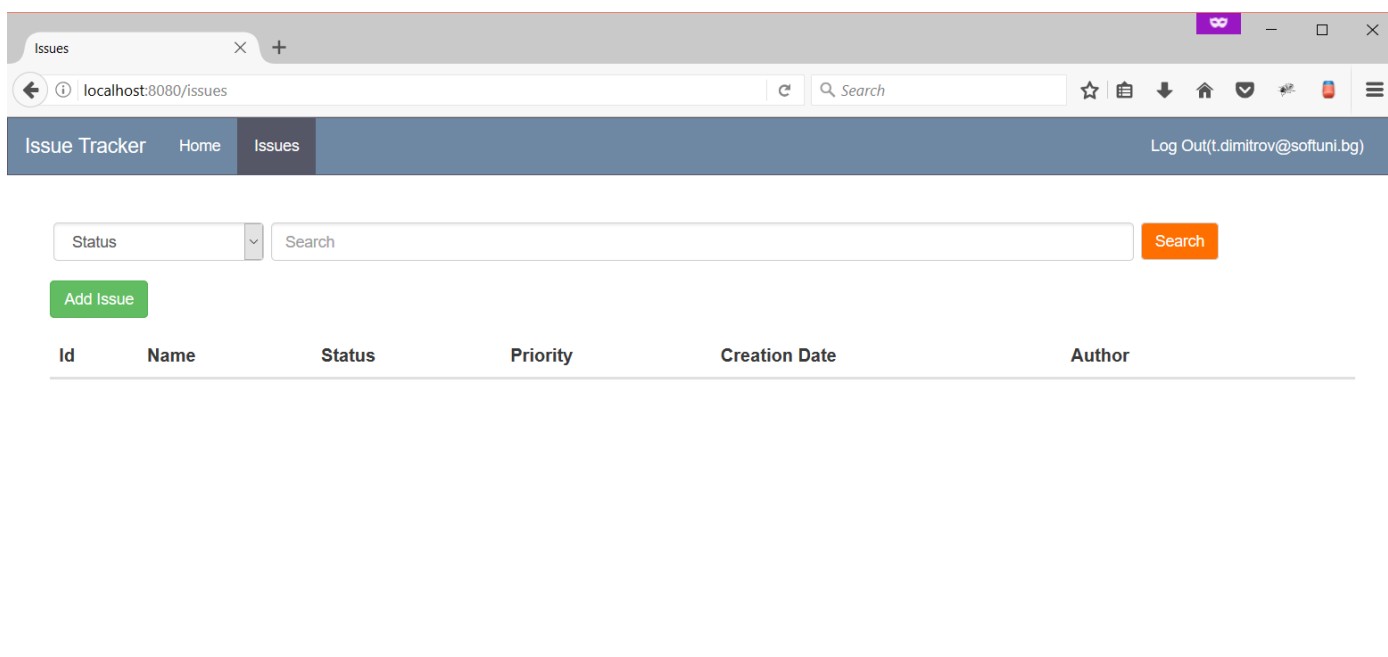




8. Implement Issues Page

You should access this page **only if you are logged in**. Otherwise, a redirect to **/login** should happen. Here all **the issues** are listed. If you are the **owner of the issue** you can edit and delete your own issues. If you are **admin** you can edit and delete any issue. Logged in users can create issues. **Edit issues.jsp**. Implement the **search functionality**.

Default view



Logged In View

The screenshot shows a web browser window with the URL `localhost:8080/issues`. The page has a dark blue header with "Issue Tracker" and navigation links "Home" and "Issues". A "Log Out(t.dimitrov@softuni.bg)" link is on the right. Below the header, there is a search bar with a "Status" dropdown and a "Search" button. A green "Add Issue" button is on the left. The main content is a table with the following data:

Id	Name	Status	Priority	Creation Date	Author
7	IntelliJ plugin bug	SOLVED	LOW	2017-02-26 03:01:10.0	otheruser@abv.bg
8	TomEE doesn't deploy properly	NEW	MEDIUM	2017-02-26 03:01:46.0	t.dimitrov@softuni.bg

Each row has "Edit" and "Delete" buttons. The "Delete" button is only visible for the second row.

Admin View

The screenshot shows the same web browser window as the Logged In View. The table data is identical, but the "Delete" button is visible for both rows. The "Add Issue" button is still present on the left.

Id	Name	Status	Priority	Creation Date	Author
7	IntelliJ plugin bug	SOLVED	LOW	2017-02-26 03:01:10.0	otheruser@abv.bg
8	TomEE doesn't deploy properly	NEW	MEDIUM	2017-02-26 03:01:46.0	t.dimitrov@softuni.bg

9. Implement Add Issue Page

On that page a **new issue** should be added. **Edit add-issue.jsp**

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/issues/add'. The page has a dark blue header with 'Issue Tracker' on the left and 'Log Out(t.dimitrov@softuni.bg)' on the right. Below the header, there are three tabs: 'Home', 'Issues', and 'Add'. The 'Add' tab is active. The main content area is a light gray box containing a form with three input fields: the first contains 'TomEE doesn't deploy properly', the second contains 'New', and the third contains 'Medium'. Below these fields are two buttons: 'Add' and 'Cancel'.

10. Implement Edit Issue Page

On that page **any current issue** should be edited. **Edit edit-issue.jsp**

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/issues/edit/8'. The page has a dark blue header with 'Issue Tracker' on the left and 'Log Out(t.dimitrov@softuni.bg)' on the right. Below the header, there are three tabs: 'Home', 'Issues', and 'Edit'. The 'Edit' tab is active. The main content area is a light gray box containing a form with three input fields: the first contains 'TomEE doesn't deploy properly', the second contains 'NEW', and the third contains 'MEDIUM'. Below these fields are two buttons: 'Edit' and 'Cancel'.

11. Implement Delete Issue Functionality

You should be able to **delete** an issue if that is **permitted**.

12. Project Infrastructure Bonus

Bonus points code quality / good application structure / additional effort.

- Bonus points for using **Web Filter**.
- Bonus points for using **ModelMapper**.
- Bonus points for **quality custom extension** of the application.