JAVA SPRING FRAMEWORK

Lab Guides

Document Code	**25e-BM/HR/HDCV/FSOFT**
Version	**1.0**
Effective Date	**01/09/2024**

Hanoi, 08/2024

RECORD OF CHANGES

No	Effective Date	Change Description	Reason	Reviewer	Approver
1	06/08/2024	Create a new Lab	Create new		VinhNV

Contents

FRESHER ACADEMY	CODE:	JSFW_Lab_08_Opt2
	TYPE:	MEDIUM
	LOC:	200
	DURATION:	120 MINUTES

Java Spring Framework Introduction

Objectives:

In this lab, you will learn how to manage users using Spring Boot. You will set up the necessary entities, repositories, services, and controllers to handle user creation, updates, and deactivation. You will also build Thymeleaf views to manage user-related functions.

Lab Specifications:

Trainees are required to:

- Create a User entity.
- Implement CRUD operations using UserRepository.
- Create Thymeleaf views to manage users (create, read, update, and deactivate).

Problem Description:

Trainees are required to:

- Implement user management functionality (add/edit user, deactivate user).

- Use Thymeleaf for the user interface (UI).

Prerequisites:

- Completed **JSFW_Lab_08_Opt1**.

Guidelines:

Step 1: Update User Entity

Update the User class in com.example.model like this:

```
package com.example.model;

import javax.persistence.*;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;

@Entity
@Table(name = "app_user")
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(nullable = false, unique = true)
    @NotBlank(message = "Username is required")
    @Size(min = 3, max = 50, message = "Username must be between 3 and 50
characters")
    private String username;

    @Column(nullable = false)
    @NotBlank(message = "Password is required")
    @Size(min = 6, max = 100, message = "Password must be between 6 and 100
characters")
    private String password;

    // Getters and Setters
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Step 2: Update UserRepository

In com.example.repository, update UserRepository (if needs):

```
package com.example.repository;

import com.example.model.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {
    User findByUsername(String username);
}
```

Step 3: Update UserService

In com.example.service, update the UserService.java (if needs):

```
package com.example.service;

import com.example.model.Subject;
import com.example.model.User;
import com.example.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;
import java.util.Optional;

@Service
```

```
public class UserService {
    @Autowired
    private UserRepository userRepository;

    public User save(User user) {
        return userRepository.save(user);
    }

    public User findByUsername(String username) {
        return userRepository.findByUsername(username);
    }

    public User findById(Long id) {
        return userRepository.findById(id).orElse(null);
    }

    public List<User> findAllUsers() {
        return userRepository.findAll();
    }

    // Deactivate (soft delete) user by removing or deactivating them
    public void deactivateUser(Long id) {
        User user = findById(id);
        if (user != null) {
            // Option 1: Hard delete
            userRepository.delete(user);

            // Option 2: Soft delete (if you add an "active" field to the User
entity)
            // user.setActive(false); // Requires a field like boolean active in
the entity
            // userRepository.save(user);
        }
    }
}
```

Step 4: Update UserController

In com.example.controller, create/update UserController.java:

```
package com.example.controller;

import com.example.model.User;
import com.example.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;

import javax.servlet.http.HttpSession;
import javax.validation.Valid;
import java.util.List;

@Controller
@RequestMapping("/users")
public class UserController {

    @Autowired
    private UserService userService;

    // View logged-in user profile from session
    @GetMapping("/profile")
    public String userProfileFromSession(HttpSession session, Model model) {
        String username = (String) session.getAttribute("username");
        if (username != null) {
            User user = userService.findByUsername(username);
            if (user != null) {
                model.addAttribute("user", user);
                return "profiles/profile";
            }
        }
        return "error";
    }

    // View another user's profile by ID
    @GetMapping("/profile/{id}")
    public String userProfile(@PathVariable("id") Long userId, Model model) {
        User user = userService.findById(userId);
        if (user != null) {
            model.addAttribute("user", user);
            return "profiles/profile";
        } else {
            return "error";
        }
    }

    // Show the form to edit the user's profile
    @GetMapping("/edit/{id}")
    public String editProfileForm(@PathVariable Long id, Model model) {
        User user = userService.findById(id);
        if (user != null) {
            model.addAttribute("user", user);
            return "profiles/edit_profile";
        }
        return "redirect:/error";
    }

    // Process the form to update the user's profile
    @PostMapping("/edit/{id}")
    public String updateProfile(@PathVariable Long id, @ModelAttribute @Valid
User user, BindingResult result, Model model) {
        if (result.hasErrors()) {
            return "profiles/edit_profile";
        }
        user.setId(id); // Ensure ID is set for updating
        userService.save(user);
        return "redirect:/users/profile/" + id;
    }

    // Show the form to add a new user
    @GetMapping("/add")
    public String addUserForm(Model model) {
        model.addAttribute("user", new User());
        return "profiles/add_user";
    }

    // Process the form to add a new user
    @PostMapping("/add")
    public String addUser(@ModelAttribute @Valid User user, BindingResult
result, Model model) {
        if (result.hasErrors()) {
            return "profiles/add_user";
        }
        userService.save(user);
        return "redirect:/users/profile/" + user.getId();
    }

    // Show a list of users
    @GetMapping
    public String listUsers(Model model) {
        List<User> users = userService.findAllUsers();
        model.addAttribute("users", users);
        return "profiles/user_list";
    }

    // Deactivate (delete) a user by ID
    @PostMapping("/deactivate/{id}")
    public String deactivateUser(@PathVariable Long id) {
        userService.deactivateUser(id);
        return "redirect:/users/users";
    }
}
```

Step 5: Create Thymeleaf Templates

Create the following templates in src/main/resources/templates/profiles:

- **add_user.html**: Form for adding a new user.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Add User</title>
    <!-- Bootstrap CSS -->
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card">
                <div class="card-header bg-primary text-white">
                    <h4 class="card-title mb-0">Add New User</h4>
                </div>
                <div class="card-body">
                    <!-- Form for adding a new user -->
                    <form action="/users/add" method="post">
                        <!-- Username Field -->
                        <div class="form-group">
                            <label for="username">Username</label>
                            <input type="text" class="form-control"
id="username" name="username" placeholder="Enter username" required>
                        </div>

                        <!-- Password Field -->
                        <div class="form-group">
                            <label for="password">Password</label>
                            <input type="password" class="form-control"
id="password" name="password" placeholder="Enter password" required>
                        </div>

                        <!-- Submit Button -->
                        <button type="submit" class="btn btn-primary">Add
```

```
User</button>
                        <a href="/users" class="btn btn-secondary">Cancel</a>
                    </form>
                </div>
            </div>
        </div>
    </div>
</div>

<!-- Optional JavaScript -->
<!-- jQuery and Bootstrap Bundle (includes Popper) -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.bundle.min.j
s"></script>

</body>
</html>
```

- **edit_user.html**: Form for editing an existing user.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Edit Profile</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

<div class="container mt-5">
    <h1 class="mb-4">Edit Profile</h1>
    <form th:action="@{/users/edit/{id}(id=${user.id})}" method="post"
th:object="${user}">
        <div class="form-group">
            <label for="username">Username:</label>
            <input type="text" id="username" name="username" class="form-
control" th:field="*{username}" />
            <div class="invalid-feedback"
th:if="${#fields.hasErrors('username')}" th:errors="*{username}"></div>
        </div>
        <div class="form-group">
            <label for="password">Password:</label>
            <input type="password" id="password" name="password" class="form-
control" th:field="*{password}" />
            <div class="invalid-feedback"
th:if="${#fields.hasErrors('password')}" th:errors="*{password}"></div>
        </div>
        <button type="submit" class="btn btn-primary">Update Profile</button>
        <a class="btn btn-secondary ml-2"
th:href="@{/users/profile/{id}(id=${user.id})}">Cancel</a>
    </form>
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"><
/script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></s
cript>
```

```
</body>
</html>
```

- **user_list.html**: List of all users.

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.w3.org/1999/xhtml">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User List</title>
    <!-- Bootstrap CSS -->
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

<div class="container mt-5">
    <h2 class="mb-4">User List</h2>
    <a href="/users/add" class="btn btn-primary">Add New User</a>
    <a href="/dashboard" class="btn btn-secondary">Back to Dashboard</a>
    <table class="table table-bordered">
        <thead>
        <tr>
            <th>ID</th>
            <th>Username</th>
            <th>Actions</th>
        </tr>
        </thead>
        <tbody>
        <tr th:each="user : ${users}">
            <td th:text="${user.id}"></td>
            <td th:text="${user.username}"></td>
            <td>
                <a th:href="@{/users/profile/{id}(id=${user.id})}" class="btn
btn-info btn-sm">View</a>
                <a th:href="@{/users/edit/{id}(id=${user.id})}" class="btn btn-
warning btn-sm">Edit</a>
                <form action="#"
th:action="@{/users/deactivate/{id}(id=${user.id})}" method="post"
style="display:inline;">
                    <button type="submit" class="btn btn-danger btn-
sm">Deactivate</button>
                </form>
            </td>
        </tr>
        </tbody>
    </table>

</div>

<!-- Optional JavaScript -->
<!-- jQuery and Bootstrap Bundle (includes Popper) -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@4.5.2/dist/js/bootstrap.bundle.min.j
s"></script>

</body>
</html>
```

- **profile.html**: Display user profile details.

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>User Profile</title>
    <link
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>

<div class="container mt-5">
    <h1 class="mb-4">User Profile</h1>

    <div class="card">
        <div class="card-body">
            <h5 class="card-title">Username: <span
th:text="${user.username}"></span></h5>
            <p class="card-text">Password: <span
th:text="${user.password}"></span></p>
            <!-- Add more fields if necessary -->

            <a th:href="@{/users/edit(id=${user.id})}" class="btn btn-
primary">Edit Profile</a>
            <a th:href="@{/dashboard}"  class="btn btn-secondary">Back to
Dashboard</a>
        </div>
    </div>
</div>

<!-- Bootstrap JS and dependencies -->
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.0.7/dist/umd/popper.min.js"><
/script>
<script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></s
cript>

</body>
</html>
```

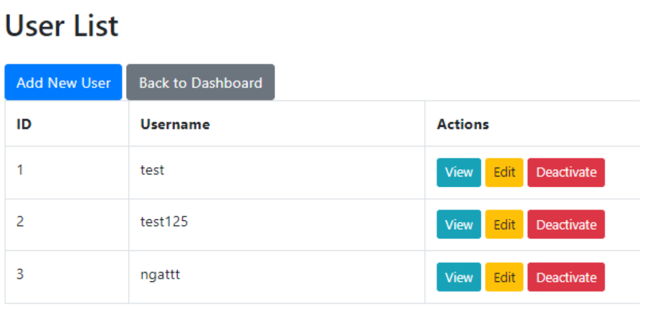
Step 6: Update dashboard.html

- Add a User Management function to dashboard.html:

```
<!-- User Management Card -->
<div class="col-md-3">
    <div class="card">
        <img src="https://cdn-icons-png.flaticon.com/512/149/149071.png"
class="card-img-top" alt="Users">
        <div class="card-body">
            <h5 class="card-title">Users</h5>
            <p class="card-text">Manage your users - Add, Edit, or Deactivate
users.</p>
            <a href="/users" class="btn btn-primary">Go to Users</a>
        </div>
    </div>
</div>
```

Step 8: Run and Test

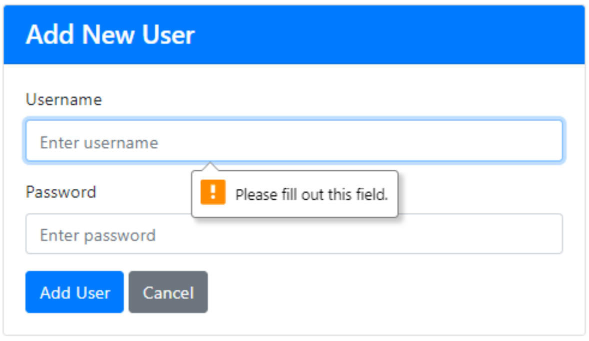
- Run the Spring Boot application, and here are some testing steps:
- Go to [User Management] function: http://localhost:8080/users.



User List

ID	Username	Actions
1	test	View Edit Deactivate
2	test125	View Edit Deactivate
3	ngattt	View Edit Deactivate

- Click on [Add New User] to add a user:
    - If username or password is missing, validation errors will show.



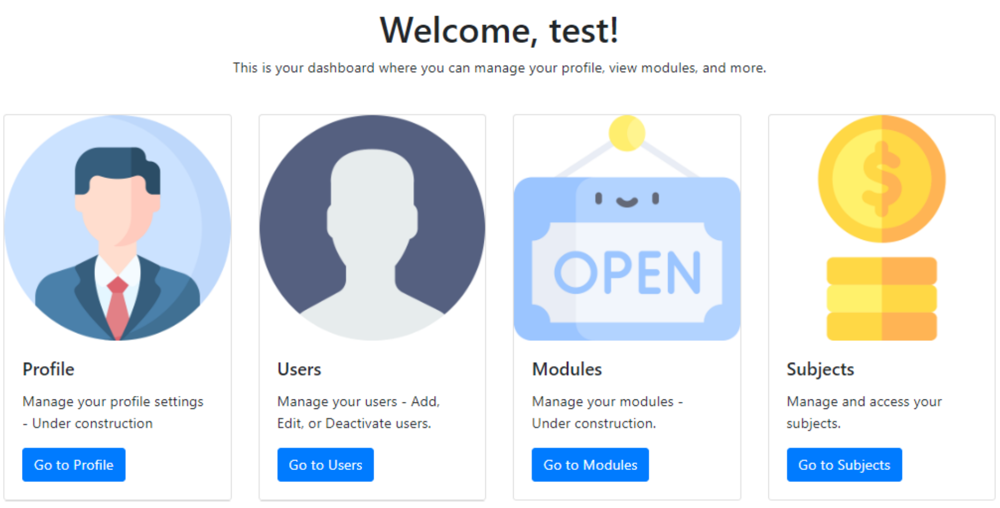
Add New User

Username: Enter username

Please fill out this field.

Password: Enter password

- After adding a user, view the list of users, or edit/deactivate a user.
- Check the user management functions on the dashboard: http://localhost:8080/dashboard.



Welcome, test!

This is your dashboard where you can manage your profile, view modules, and more.

Profile — Manage your profile settings - Under construction

Users — Manage your users - Add, Edit, or Deactivate users.

Modules — Manage your modules - Under construction.

Subjects — Manage and access your subjects.

----o0o-----

THE END