**Spring Boot Project**

Created a Spring Boot project using Spring Initializr and included the following dependencies:

1. Spring Web
2. Spring Data JPA
3. Thymeleaf (for the view layer)
4. H2 Database

**Step 2: Created Entity Class in Java (Eclipse)**

Created an Employee entity class with attributes like id, firstName, lastName, and email.

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "first\_name")

private String firstName;

@Column(name = "last\_name")

private String lastName;

private String email;

// getters and setters

}

**Step 3: Created a Repository in Java (Eclipse)**

Created a repository interface that extends JpaRepository to perform CRUD operations on the Employee entity.

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

}

**Step 4: Created a Controller**

Create a controller class with appropriate mappings for your CRUD operations.

@Controller

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@GetMapping

public String listEmployees(Model model) {

List<Employee> employees = employeeRepository.findAll();

model.addAttribute("employees", employees);

return "employee/list";

}

@GetMapping("/create")

public String createEmployeeForm(Employee employee) {

return "employee/create";

}

@PostMapping("/create")

public String createEmployee(Employee employee) {

employeeRepository.save(employee);

return "redirect:/employees";

}

@GetMapping("/edit/{id}")

public String editEmployeeForm(@PathVariable Long id, Model model) {

Employee employee = employeeRepository.findById(id).orElse(null);

model.addAttribute("employee", employee);

return "employee/edit";

}

@PostMapping("/edit/{id}")

public String editEmployee(@PathVariable Long id, Employee updatedEmployee) {

updatedEmployee.setId(id);

employeeRepository.save(updatedEmployee);

return "redirect:/employees";

}

@GetMapping("/delete/{id}")

public String deleteEmployee(@PathVariable Long id) {

employeeRepository.deleteById(id);

return "redirect:/employees";

}

}

**Step 5: Created HTML Templates**

Created Thymeleaf templates for displaying the views. Added templates such as list.html, create.html and editing employees (edit.html).

**List.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Employee List</title>

</head>

<body>

<h1>Employee List</h1>

<table>

<thead>

<tr>

<th>ID</th>

<th>First Name</th>

<th>Last Name</th>

<th>Email</th>

<th>Action</th>

</tr>

</thead>

<tbody>

<tr th:each="employee : ${employeeList}">

<td th:text="${employee.id}"></td>

<td th:text="${employee.firstName}"></td>

<td th:text="${employee.lastName}"></td>

<td th:text="${employee.email}"></td>

<td>

<a th:href="@{/edit/{id}(id=${employee.id})}">Edit</a>

<a th:href="@{/delete/{id}(id=${employee.id})}">Delete</a>

</td>

</tr>

</tbody>

</table>

<a th:href="@{/create}">Create Employee</a>

</body>

</html>

**Create.html – creating/adding employee**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Create Employee</title>

</head>

<body>

<h1>Create Employee</h1>

<form th:action="@{/save}" method="post" th:object="${employee}">

<label for="firstName">First Name:</label>

<input type="text" id="firstName" name="firstName" th:field="\*{firstName}" required><br><br>

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" name="lastName" th:field="\*{lastName}" required><br><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" th:field="\*{email}" required><br><br>

<button type="submit">Save</button>

</form>

<a th:href="@{/}">Back to Employee List</a>

</body>

</html>

**edit.html – form for editing employee data**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<head>

<meta charset="UTF-8">

<title>Edit Employee</title>

</head>

<body>

<h1>Edit Employee</h1>

<form th:action="@{/update}" method="post" th:object="${employee}">

<input type="hidden" th:field="\*{id}">

<label for="firstName">First Name:</label>

<input type="text" id="firstName" name="firstName" th:field="\*{firstName}" required><br><br>

<label for="lastName">Last Name:</label>

<input type="text" id="lastName" name="lastName" th:field="\*{lastName}" required><br><br>

<label for="email">Email:</label>

<input type="email" id="email" name="email" th:field="\*{email}" required><br><br>

<button type="submit">Update</button>

</form>

<a th:href="@{/}">Back to Employee List</a>

</body>

</html>

**Step 6: Configuring Database**

Configured database connection in the application.properties

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=MYSQLdb

spring.datasource.password=Stubborn@16

spring.jpa.hibernate.ddl-auto=update

Navigated to the specified URLs (e.g., /employees) in web browser to access the CRUD operations.