### **Building a Employee Management Application**

### **With Spring Boot, HTML, CSS, and Thymleaf**

## Description:

This application allows users to add, view, update, and delete user information, including first name, last name, and email address. It is built using Spring Boot and Thymeleaf for the frontend.

## Setup Instructions

### **Prerequisites**

* Java 11 or higher (I use java 17)
* Maven
* IDE (IntelliJ IDEA, Eclipse, etc.)
* MySQL

### **Steps**

1. **Clone the Repository**
   * Clone the project repository and navigate to the project directory.
2. **Import the Project**
   * Open the project in your preferred IDE.
3. **Configure the Database**
   * Use H2 in-memory database by setting the necessary properties in application.properties.
4. **Run the Application**
   * Use Maven to run the Spring Boot application.

# Functionality

## Employee Management

**View Employees**

* A list of all employees is displayed on the home page **(index.html)**, showing their first name, last name, and email address.

**Add Employee**

* A form **(form.html**) allows users to enter a new employee's details, including first name, last name, and email address. Upon submission, the employee is added to the database and displayed in the employee list.

**Update Employee**

* Users can update an employee's details by clicking the "Update" button next to an employee's entry.

**Delete Employee**

* Users can delete an employee's entry by clicking the "Delete" button next to the employee.

**View Employee**

* Users can view detailed information about an employee by clicking the "View" button next to the employee's entry.

### **User Interface**

The application uses Thymeleaf templates for rendering HTML pages. Bootstrap is used for styling to create a responsive and user-friendly interface.

### **Templates**

The main templates used in the application are:

* **index.html**: Displays the list of employees and provides options to add, update, view, or delete employees.
* **form.html**: Provides a form for adding and updating employee details.

### **Database Setup**

### **Configuration in application.properties**

The application.properties file contains the necessary configuration settings for the H2 database. These settings include the database URL, driver class name, username, and password, as well as JPA settings.

spring.datasource.url=jdbc:mysql://localhost:3306/DB name  
spring.datasource.username=root  
spring.datasource.password=\*\*\*\*\*\*\*\*  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true

### **Application Flow**

**Home Page (index.html)**

* The home page displays a list of all employees stored in the database.
* Provides options to add a new employee, update existing employee details, delete an employee, or view detailed information about an employee.

**Add/Edit Employee (form.html)**

* The form page allows users to enter or update employee details such as first name, last name, and email address.
* On submission, the form data is sent to the server to add a new employee or update an existing employee's details.

**Controller Layer**

* The EmployeeController handles HTTP requests and responses. It manages the CRUD operations by interacting with the service layer and the repository layer.
* The controller maps URL paths to methods that handle the respective operations, such as displaying the employee list, showing the form for adding/updating employees, saving employee details, and deleting an employee.

**Service Layer**

* The EmployeeService contains the business logic for managing employees.
* It acts as an intermediary between the controller layer and the repository layer, ensuring that the controller does not directly interact with the database.

**Repository Layer**

* The EmployeeRepository extends the JpaRepository and provides methods for CRUD operations on the employee entity.
* It interacts with the database to perform operations such as saving, updating, deleting, and fetching employee data.

**Entity Layer**

* The EmployeeEntity represents the employee data model.
* It is annotated with JPA annotations to map the class to a database table and its fields to the corresponding columns.

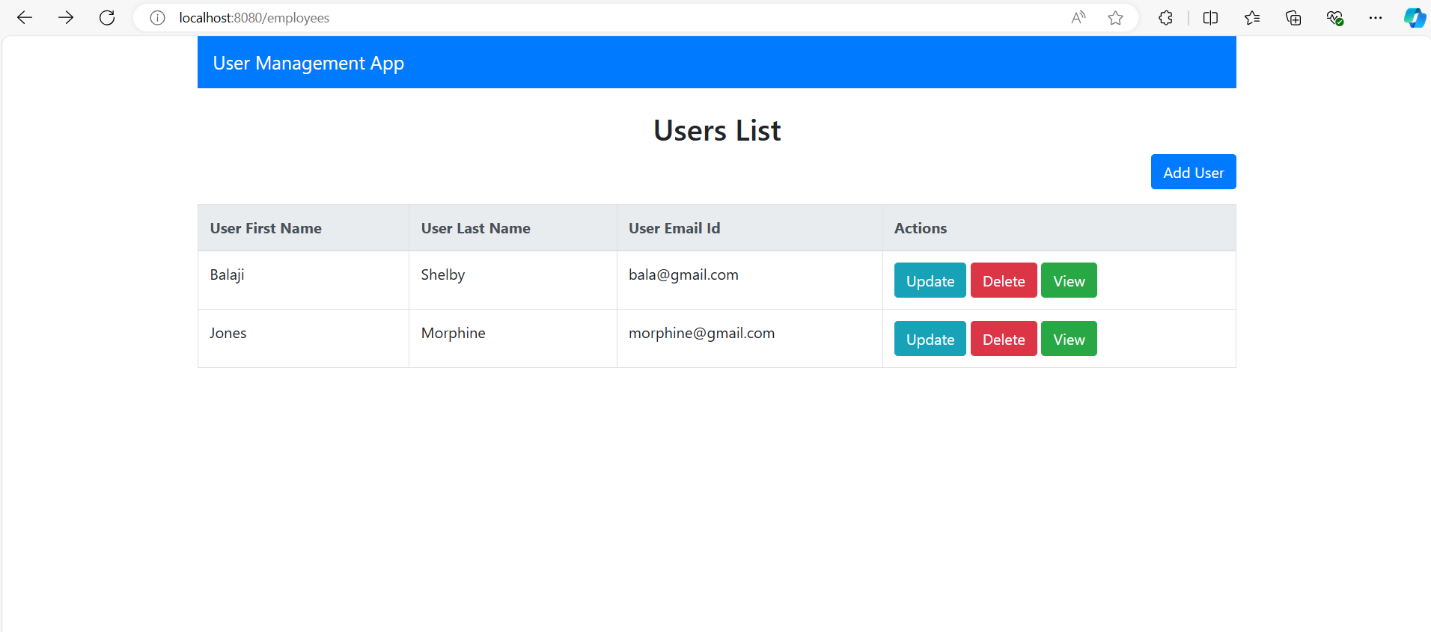
## **How to Use**

## **Accessing the Application**

* **Home Page**: Navigate to <http://localhost:8080/employees> to view the list of employees.
* **Add Employee**: Navigate to <http://localhost:8080/employees/new> to access the form for adding a new employee.

### **Interacting with the Application**

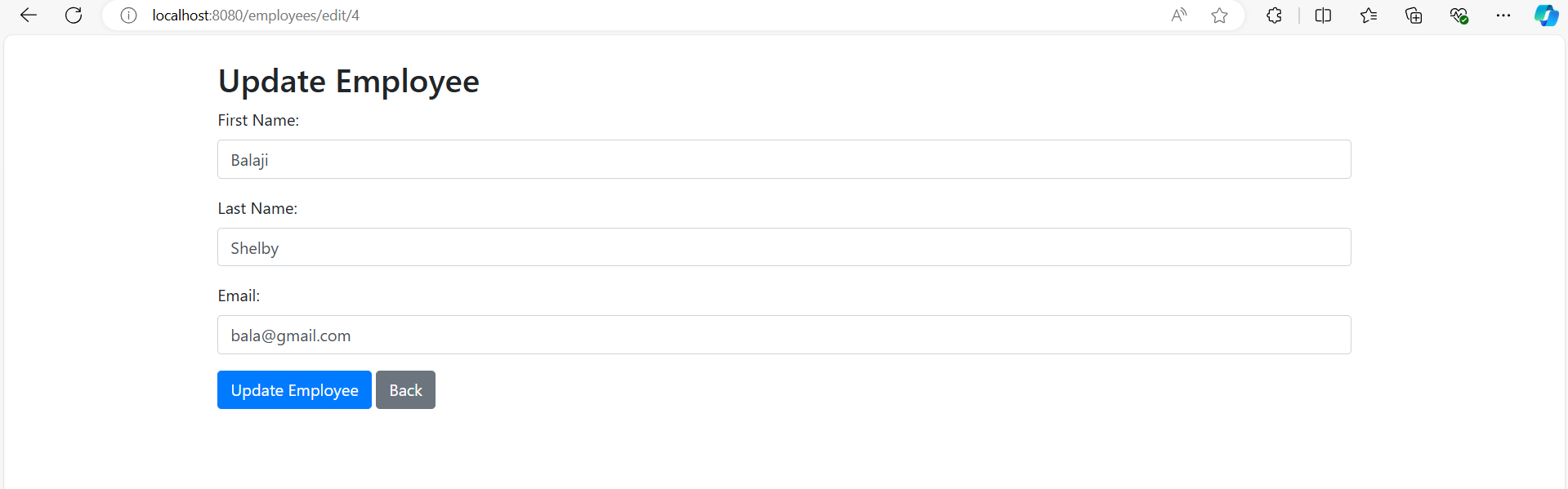
* **Add a New Employee**: Fill out the form with the employee's first name, last name, and email address, then click "Add Employee".
* **View Employee List**: The home page displays a table of all employees.
* **Update Employee**: Click "Update" next to an employee, modify their details, and save.
* **Delete Employee**: Click "Delete" next to an employee to remove them from the list.
* **View Employee Details**: Click "View" next to an employee to see detailed information.
* <http://localhost:8080/employees>



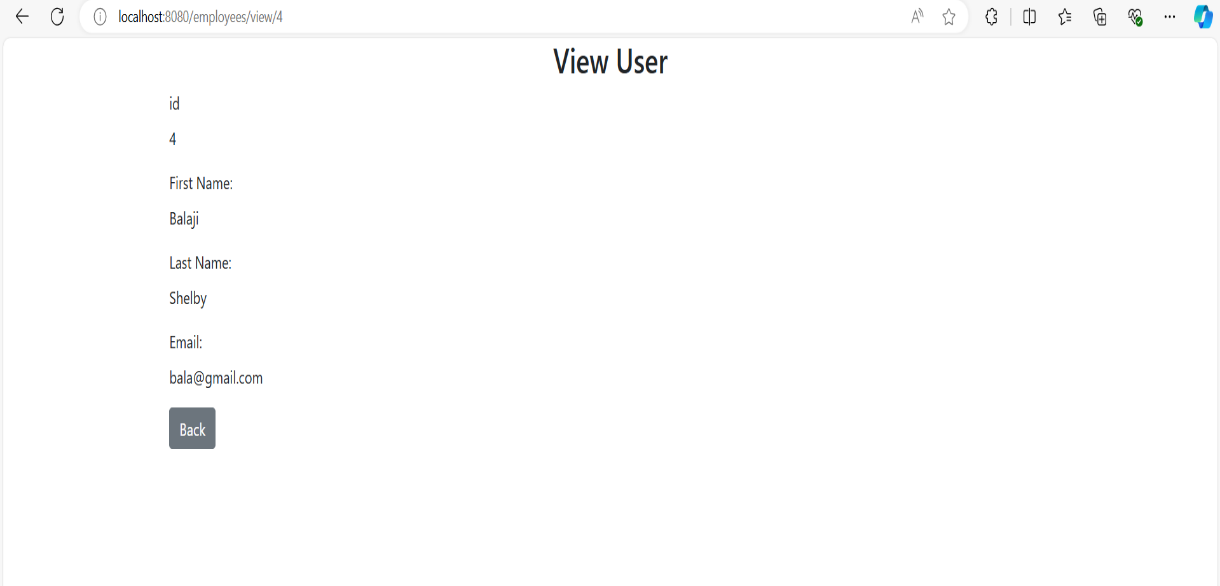
* <http://localhost:8080/employees/new>



* <http://localhost:8080/employees/edit/4>



* http://localhost:8080/employees/view/4



### **Thanks for watching**

### Balaganapathi