Spring Boot Thymeleaf

Thymeleaf is a java web template used to develop web app applications in java. It uses html as a front-end language. We can connect our java project to the web with this template. Thymeleaf template does not require spring framework to develop web app applications, but we can use them both for an awesome development experience.

Steps for CRUD application project-

1. Go to https://start.spring.io/ and kick start your project. Add necessary dependencies like Spring Data JPA,SpringWeb, Spring Boot DevTools, MySql Driver. Select as maven and enter.

2. Import as maven project and check pom for dependencies we added.

3. Create some basic packages named as entity (for entity classes), dao (for dao classes), service (for service classes), controller (for controller classes).

4. Create entity class in entity package and write the required code with annotations, getters and setters, toString() and constructors.

5. Update application.properties in resources folder with below code.
   ```
   spring.datasource.url=jdbc:mysql://localhost:3306/employee_directory?useSSL=false
   spring.datasource.username=root
   spring.datasource.password=root
   spring.datasource.hikari.maximum-pool-size=100
   spring.datasource.hikari.minimum-idle=10
   spring.jpa.show-sql=true
   logging.level.web=DEBUG
   ```

6. Create a JpaRepository interface extending "org.springframework.data.jpa.repository.JpaRepository<T, ID>" extension. After this just replace <T, ID> with <Entity_name, Integer>, Add @RepositoryRestResource(path="members") annotation at top of Repository interface.

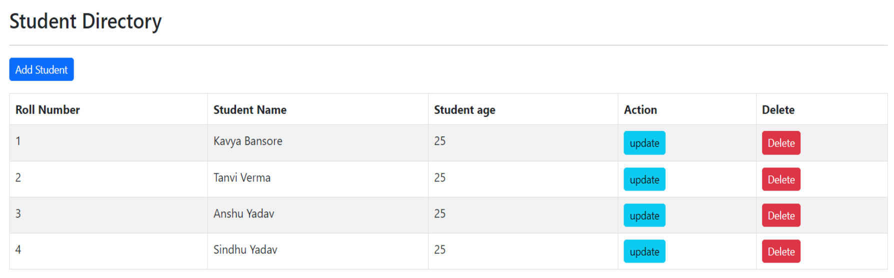
7. Create an interface in the service package with the name service.java and declare the necessary methods for your project and then create a class with name ServiceImpl in the same package with the extension of service interface we just created to define the methods declared in service class. Use @Service before ServiceImpl class name, Inject JpaRepository with an object, @Autowired constructor injection and define JpaRepository with tempJpaObject in the parenthesis and initialize the main JpaObject with tempJpaObject in the constructor body. Now one by one write codes for different methods like you want them to function in your application. Basically, in ServiceImpl all you need is to return the methods with the JpaRepository.

8. Now create a controller class in the rest package to control your serviceImpl class functions. First define @RestController and @RequestMapping("/api") before controller class name, inject service Impl with its object, @Autowired a constructor injection and define serviceImpl with tempImplObject in the parenthesis and initialize the main ImplObject with tempImplObject in the constructor body. Map with annotations before all the functions. For a web app project we need to map the controller with the html file and do the particular function in a specific html file. Use @GetMapping () to map with the server and return "html-filename" to return the result we want to display in the browser.

9. After doing the necessary coding in the above classes, run the project as an application and after it is up in IDE, check localhost:8080/students/list in Browser.

10. Find Screenshots of the UI results for List, Add, Update and delete Students respectively after running the application in IDE:

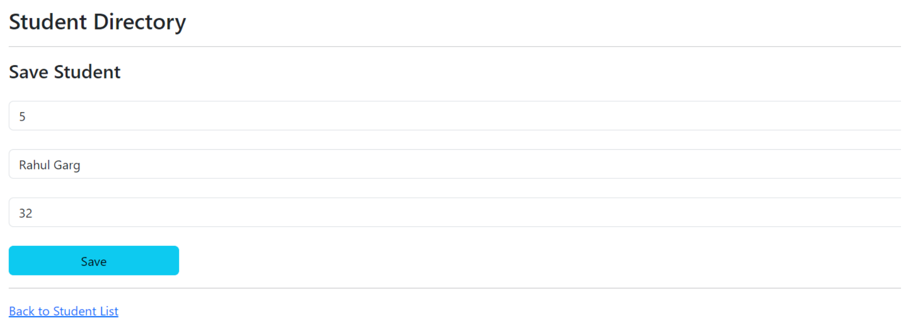
Spring Boot Thymeleaf


Student Directory

Add Student

Roll Number	Student Name	Student age	Action	Delete
1	Kavya Bansore	25	update	Delete
2	Tanvi Verma	25	update	Delete
3	Anshu Yadav	25	update	Delete
4	Sindhu Yadav	25	update	Delete

Homepage


Student Directory

Save Student

5

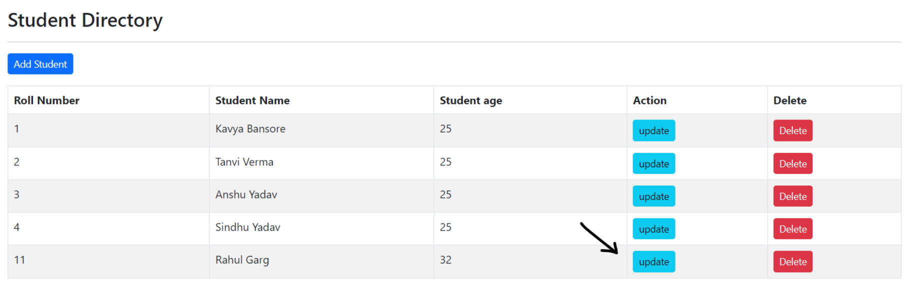
Rahul Garg

32

Save

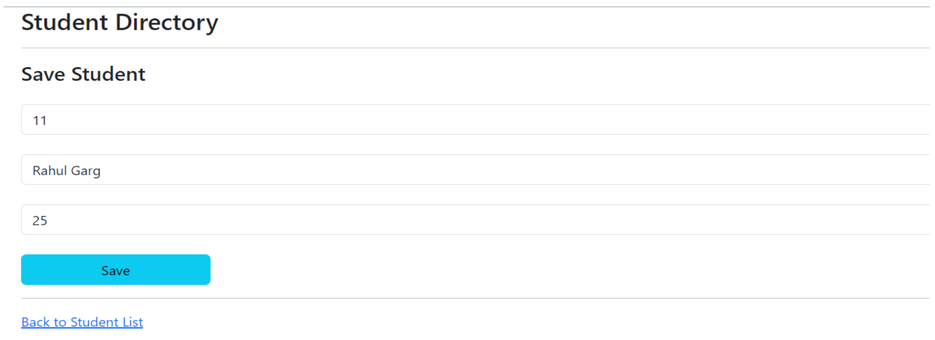
Back to Student List

Add New Data


Student Directory

Add Student

Roll Number	Student Name	Student age	Action	Delete
1	Kavya Bansore	25	update	Delete
2	Tanvi Verma	25	update	Delete
3	Anshu Yadav	25	update	Delete
4	Sindhu Yadav	25	update	Delete
11	Rahul Garg	32	update	Delete


Student Directory

Save Student

11

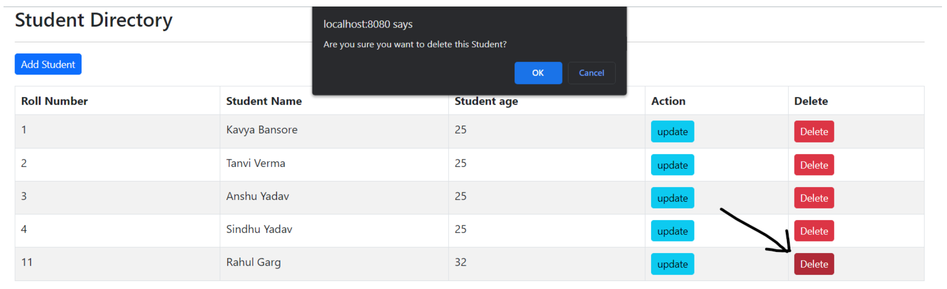
Rahul Garg

25

Save

Back to Student List

Update Existing Data


Student Directory

localhost:8080 says

Are you sure you want to delete this Student?

OK Cancel

Add Student

Roll Number	Student Name	Student age	Action	Delete
1	Kavya Bansore	25	update	Delete
2	Tanvi Verma	25	update	Delete
3	Anshu Yadav	25	update	Delete
4	Sindhu Yadav	25	update	Delete
11	Rahul Garg	32	update	Delete

Delete the data

11. Find the code and SQL query here: https://github.com/Leaf-Co-Kb/Student-Management-Tymeleaf.