Date : 26/02/2021
Spring Boot 9AM
Mr. RAGHU

Spring Boot Mini Project
Web Mvc + Data Jpa
Thymeleaf + MYSQL
CRUD App

----

*) Application are implemented using Layers Design.
Layer indicates 'code used for one concept'.

Layers(4):

a. PL = Presentation Layer
    [Read data from UI, Write Data to UI] (MVC)

b. SL = Service Layer
    [Business Logics , calculations/operations...etc]

c. DAL = Data Access Layer
    [DB Operations SELECT/NO-SELECT]

*) Within Layer use IS-A Relation, between layers
using HAS-A Relation (Class--<>Interface)

-------------Stage Coding---------------------------------

Stage#1 Register Employee
Stage#2 Display Data
Stage#3 Delete By Id
Stage#4 Edit Page and Update Data
Stage#5 UI BootStrap Design
Stege#6 UI-JQuery Validation
Stage#7 AJAX validation
Stage#8 Error Page Handling
Stage#9 Exception Handling
Stage#10 Log Implementation

____

Project Setup

#1. Create Starter Project
Name: SpringBoot2WebMvcMySQLCrud
Dep : Web, DevTools, Lombok, Thymeleaf, MySQL, Data Jpa

#2 application.properties

```
# Server details
server.port=9292

# DataSource
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/boot9am
spring.datasource.username=root
spring.datasource.password=root

# Data JPA
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect
```

```
#3. Model class
package in.nareshit.raghu.model;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Table;

import lombok.Data;

@Data
@Entity
@Table(name="employee_tab")
public class Employee {
        @Id
        @Column(name="emp_id_col")
        @GeneratedValue
        private Integer empId;

        @Column(name="emp_name_col")
        private String empName;

        @Column(name="emp_sal_col")
        private Double empSal;

        @Column(name="emp_dept_col")
        private String empDept;

        @Column(name="emp_addr_col")
        private String empAddr;

}

#4 Layers Setup

a. Repository Interface

package in.nareshit.raghu.repo;
import org.springframework.data.jpa.repository.JpaRepository;
import in.nareshit.raghu.model.Employee;
public interface EmployeeRepository
        extends JpaRepository<Employee, Integer> {

}
-------
 b. Service Interface

package in.nareshit.raghu.service;
public interface IEmployeeService {

}
----------
 c. Service Impl class

package in.nareshit.raghu.service.impl;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import in.nareshit.raghu.repo.EmployeeRepository;
import in.nareshit.raghu.service.IEmployeeService;

@Service // = @Component + Logics/cal + TxManagement
public class EmployeeServiceImpl
        implements IEmployeeService
{

        @Autowired
        private EmployeeRepository repo; //HAS-A
}
--------------------------
 d. Controller

package in.nareshit.raghu.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import in.nareshit.raghu.service.IEmployeeService;

@Controller
@RequestMapping("/employee")
public class EmployeeController {

        @Autowired
        private IEmployeeService service;//HAS-A
}
_____________________________________________________
            Stage#1  Register Employee

a. Create Register Page under templates folder

--EmployeeRegister.html--
<!DOCTYPE html>
<html xmlns:th="https://www.thymeleaf.org/">
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>EMPLOYEE REGISTER PAGE</h2>
<form action="#" method="POST">
<pre>
NAME   : <input type="text" name="empName"/>
SALARY : <input type="text" name="empSal"/>
DEPT   : <select name="empDept">
                        <option value="">-SELECT-</option>
                        <option value="DEV">DEV</option>
                        <option value="QA">QA</option>
                        <option value="BA">BA</option>
                        <option value="MS">MS</option>
                </select>
```

```
ADDRESS: <textarea name="empAddr"></textarea>
                <input type="submit" value="Add Employee"/>
</pre>
</form>
</body>
</html>

-----------------------------------------------------
```

b. Write method in controller to show register page

```
--EmployeeController.java(method code only)--
        //1. To Display Register Page
        @GetMapping("/register")
        public String showRegPage() {
                return "EmployeeRegister";
        }
-----------------
```

*) Run your application and enter URL
http://localhost:9292/employee/register

ctrl+shift+T Open type [pre-defined code]
ctrl+shift+R Open Resource [programmer-defined code]
*) Note: Repository code is pre-defined (generated class)
   So, start code from Service Layer in Boot.
----

c. add one abstract method in Service interface

```
---IEmployeeService.java---
  Integer saveEmployee(Employee e);
```

d. Implement method in Service Impl

```
---EmployeeServiceImpl.java(method code only)---
        public Integer saveEmployee(Employee e) {
                //JDK 10# Local Variable Type Inference
                //the best datatype is selected at compile time
                //---calculations--
                var sal = e.getEmpSal();
                var hra = sal * 12/100;
                var ta = sal * 3/100;

                //set data to model cls obj
                e.setEmpHra(hra);
                e.setEmpTa(ta);

                //save data in db
                //this method again returns same object
                // with PK updated value
                e = repo.save(e);

                //PK
                Integer empId = e.getEmpId();
                return empId;
        }
---------------------------------
e. Define method in Controller
  -> Read Form On click Submit
  -> call service for save operation
  -> Read PK(ID) back to Controller
  -> Create one String message
  -> Send Message to UI using Model
  -> Link with Path /save with Method POST

---EmployeeController.java---
        @PostMapping("/save")
        public String saveEmp(
                        @ModelAttribute Employee employee,
                        Model model
                        )
        {
                Integer id = service.saveEmployee(employee);
                String msg = "Employee '"+id+"' saved";
                model.addAttribute("message", msg);

                return "EmployeeRegister";
        }
--------------------------

f. Display message at UI using code After Form tag
<span th:text="${message}"></span>

*) Run app : http://localhost:9292/employee/register
-------------------------------------------------------
Q) What is @ModelAttribute?
A) To Read Form Data to Controller use this annotation

Q) What is diff b/w below annotations ?

@Component   = creating obejct to class
@Controller  = @Component + HTTP protocol support(MVC)
@Service     = @Component + calc/logic/opr + TxManage
@Repository  = @Component + Db Operations/DB Exception


Q) What is the diff b/w below class types?
 entity : A clas mapped with database table only
 pojo   : A Simple class with variable and set/get methods
          No Logical/operational methods

  model : a class behaves like Data Transfer b/w UI and
          Database can be entity also.

Q) What are prefix/suffix here?
A) Defulats are given by Thymeleaf only.
   prefix = templates folder
   suffix = .html

Q) What is default DataSource in Spring Boot?
A) HikariDatabase is default in Spring boot.
```