Module 5 & 6

1. **Write a program to create a simple Spring Boot application that prints a message.**

**MessageController.java**

**package** com.example.demo;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.GetMapping; @Controller

**public class** MessageControlleer { @GetMapping("/message")

**public** String getmessage(Model m) { m.addAttribute("message", "hello world ");

**return** "message";

}

}

**DemoApplication.java:-**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication public class DemoApplication {

public static void main(String[] args) { SpringApplication.run(DemoApplication.class, args);

}

}

**Message.html:-**

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<h1 th:text=*"${message}"*></h1>

</body>

</html>

**Pom.Xml:-**

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="<http://maven.apache.org/POM/4.0.0>" xmlns:xsi="<http://www.w3.org/2001/XMLSchema-instance>"

xsi:schemaLocation="<http://maven.apache.org/POM/4.0.0> https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.4.0</version>

<relativePath/> <!-- lookup parent from repository -->

</parent>

<groupId>com.hw</groupId>

<artifactId>demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>hw</name>

<description>Demo project for Spring Boot</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

<groupId>log4j</groupId>

<artifactId>log4j</artifactId>

<version>1.2.17</version> <!-- Latest stable version of Log4j 1.x -->

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-logging</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-thymeleaf</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

</build>

</project>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

**application.properties:-** spring.application.name=demo server.port=8081

**Output:-**

1. **Write a program to demonstrate RESTful Web Services with spring boot.(Product Application)**

**RestDemoApplication.java:-**

package com.example.rest\_demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class RestDemoApplication {

public static void main(String[] args) { SpringApplication.run(RestDemoApplication.class, args);

}

}

**ProductServiceController.java:-**

package com.example.rest\_demo;

import org.springframework.http.HttpStatus; import org.springframework.http.ResponseEntity; import org.springframework.web.bind.annotation.\*;

import java.util.HashMap; import java.util.Map;

@RestController

public class ProductServiceController {

private static Map<String, Product> products = new HashMap<>(); static {

Product honey = new Product(); honey.setId("1"); honey.setName("Honey"); products.put(honey.getId(), honey);

}

@RequestMapping(value = "/products", method = RequestMethod.GET) public ResponseEntity<Object> getProducts() {

return new ResponseEntity<>(products.values(), HttpStatus.OK);

}

@RequestMapping(value = "/product", method = RequestMethod.POST) public ResponseEntity<Object> addProduct(@RequestBody Product product) {

products.put(product.getId(), product);

return new ResponseEntity<>("Product added successfully", HttpStatus.OK);

}

@RequestMapping(value = "/product/{id}", method = RequestMethod.PUT) public ResponseEntity<Object> updateProduct(@PathVariable("id") String id,

@RequestBody Product product) { products.remove(id); products.put(id, product);

return new ResponseEntity<>("Product updated successfully", HttpStatus.OK);

}

@RequestMapping(value = "/product/{id}", method = RequestMethod.DELETE) public ResponseEntity<Object> deleteProduct(@PathVariable("id") String id) {

products.remove(id);

return new ResponseEntity<>("Product is deleted successfully", HttpStatus.OK);

}

}

**Product.java:-**

**package** com.example.rest\_demo;

**public class** Product {

**private** String id;

**private** String name;

**public** String getId() {

**return** id;

}

**public void** setId(String id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public void** setName(String name) {

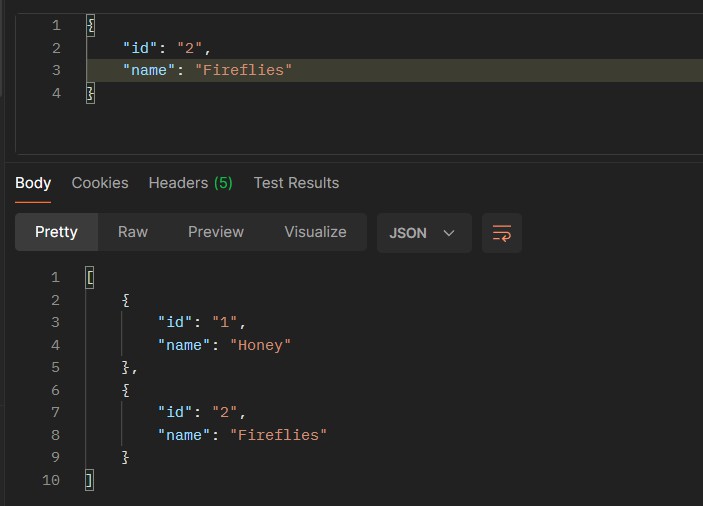
**this**.name = name;

}

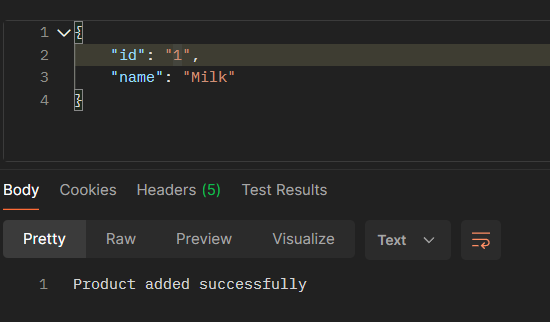
}

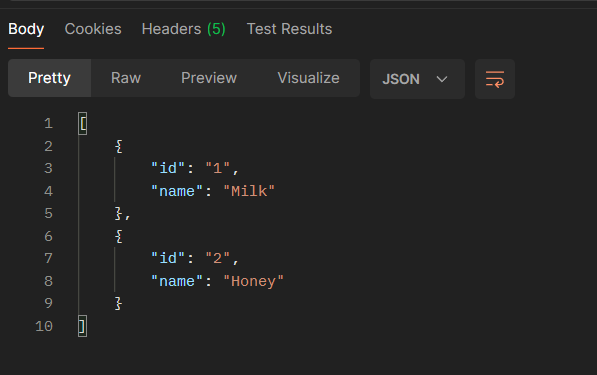
**Output:- Get:-**

**Post:-**

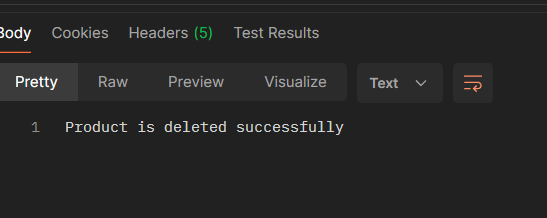
****

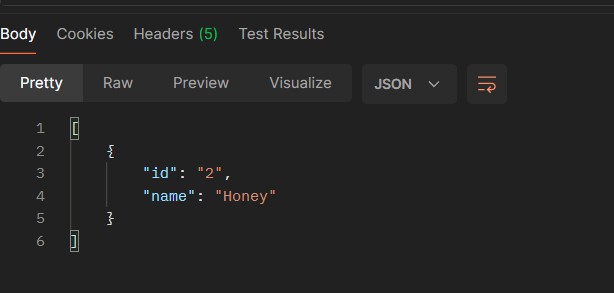
**Put:-**

****



**Delete:-**

****



1. **Write a program demonstrating how to set up a CRUD (Create, Read, Update, Delete) application using Spring Framework and Hibernate ORM with annotations for entity mapping and dependency injection.**

**Application.properties**

spring.application.name=Hibernate\_1

spring.datasource.url=jdbc:mysql://localhost:3306/college spring.datasource.username=root spring.datasource.password=root spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**Employee.java**

**package** com.example.Hibernate\_1;

**import** jakarta.persistence.Column;

**import** jakarta.persistence.Entity;

**import** jakarta.persistence.GeneratedValue; **import** jakarta.persistence.GenerationType; **import** jakarta.persistence.Id;

**import** jakarta.persistence.Table;

@Entity @Table(name="employee") **public class** Employee {

@Id @GeneratedValue(strategy=GenerationType.***IDENTITY***) **private** Long id;

@Column(name="name",nullable=**false**)

**private** String name;

@Column(name="department") **private** String department; @Column(name="salary") **private** String salary;

**public** Long getId() {

**return** id;

}

**public void** setId(Long id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public void** setName(String name) {

**this**.name = name;

}

**public** String getDepartment() {

**return** department;

}

**public void** setDepartment(String department) {

**this**.department = department;

}

**public** String getSalary() {

**return** salary;

}

**public void** setSalary(String salary) {

**this**.salary = salary;

}

}

**EmployeeRepository.java**

**package** com.example.Hibernate\_1;

**import** java.util.List;

**import** org.springframework.stereotype.Repository;

**import** jakarta.persistence.EntityManager; **import** jakarta.persistence.PersistenceContext; **import** jakarta.transaction.Transactional;

@Repository @Transactional

**public class** EmployeeRepository {

@PersistenceContext

**private** EntityManager em;

//create and update

**public** Employee save(Employee emp) {

**if**(emp.getId()==**null**) {

em.persist(emp);

}

**else** {

em.merge(emp);

}

**return** emp;

}

//get value based on id

**public** Employee findById(Long id) {

**return** em.find(Employee.**class**, id);

}

//get all employee

**public** List<Employee>findAll(){

**return** em.createQuery("from Employee",Employee.**class**).getResultList();

}

//delete

**public void** deleteById(Long id) {

Employee employee=em.find(Employee.**class**,id);

**if**(employee!=**null**) {

em.remove(employee);

}

}

}

**EmployeeService.java**

**package** com.example.Hibernate\_1;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

@Service

**public class** EmployeeService {

@Autowired

**private** EmployeeRepository er;

**public** Employee saveEmployee(Employee employee) {

**return** er.save(employee);

}

**public** Employee getEmployeeById(Long id) {

**return** er.findById(id);

}

**public** List<Employee> getAllEmployee(){

**return** er.findAll();

}

**public void** deleteEmployee(Long id) { er.deleteById(id);

}

}

**EmployeeController.java**

**package** com.example.Hibernate\_1;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping; **import** org.springframework.web.bind.annotation.GetMapping; **import** org.springframework.web.bind.annotation.PathVariable; **import** org.springframework.web.bind.annotation.PostMapping; **import** org.springframework.web.bind.annotation.RequestBody; **import** org.springframework.web.bind.annotation.RequestMapping; **import** org.springframework.web.bind.annotation.RestController;

@RestController @RequestMapping("/api/employees") **public class** EmployeeController {

@Autowired

**private** EmployeeService es;

@PostMapping

**public** ResponseEntity<Employee> saveEmployee(@RequestBody Employee employee){

**return** ResponseEntity.*ok*(es.saveEmployee(employee));

}

@GetMapping("/{id}")

**public** ResponseEntity<Employee> getEmployeeById(@PathVariable Long id){ Employee emp=es.getEmployeeById(id);

**if**(emp!=**null**) {

**return** ResponseEntity.*ok*(emp);

}**else** {

}

}

**return** ResponseEntity.*notFound*().build();

@GetMapping

**public** ResponseEntity<List> getAllEmployee(){

**return** ResponseEntity.*ok*(es.getAllEmployee());

}

@DeleteMapping("/{id}")

**public** ResponseEntity<Void> deleteEmployee(@PathVariable Long id){ es.deleteEmployee(id);

**return** ResponseEntity.*noContent*().build();

}

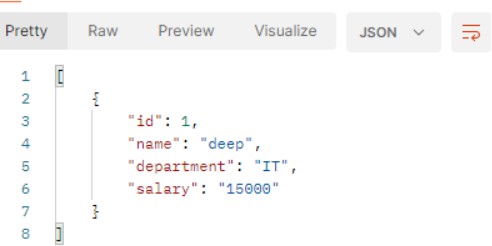
}

**Output:**

**Create**

****

**delete**

****

**update**

****