Name – Nafreen Anjum

Roll No - 2201330130120

Branch – Information Technology

Section - A

Workshop Lab Number – 203 D

QUESTION 1: Create REST API to print “Welcome to Spring Boot” on Postman. (CO4&K3) Instructions: 1) Create a REST Controller with name HomeController 2) Method name hello() 3) url will be localhost:8080/home/api

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>3.4.5</version>

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

</parent>

<groupId>com.example</groupId>

<artifactId>firstspringbootproject</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>firstspringbootproject</name>

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

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<!-- <dependency>

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

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

--> <dependency>

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

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

</dependency>

<!--<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

<scope>runtime</scope>

</dependency>-->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

application.properties

spring.application.name=firstspringbootproject

server.port = 4206

HomeController.java

package com.example;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RestController;

*@RestController*

public class HomeController {

*@GetMapping*("/home/api")

public String hello() {

return "Welcome to Spring Boot";

}

}

FirstspringbootprojectApplication.java

package com.example;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

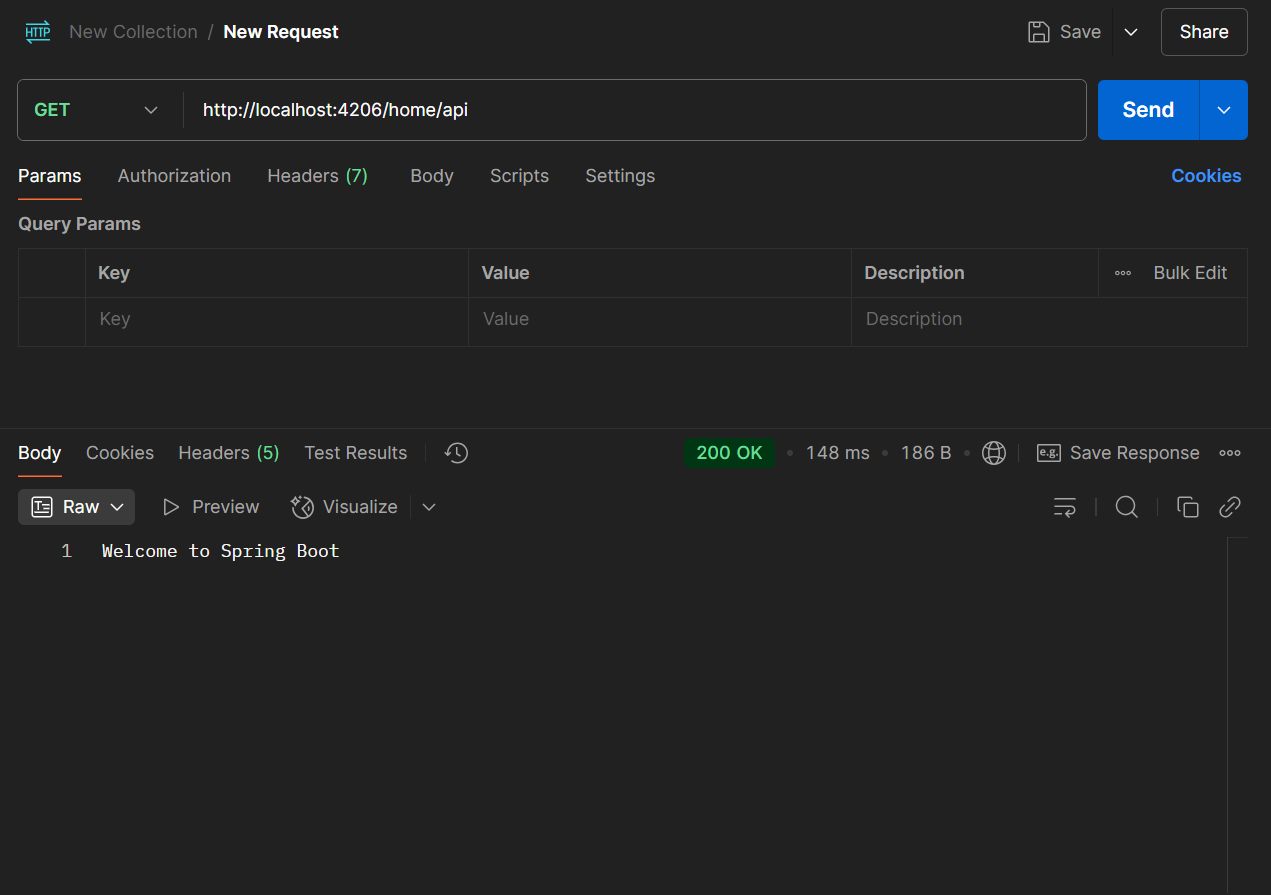
public class FirstspringbootprojectApplication {

public static void main(String[] args) {

SpringApplication.*run*(FirstspringbootprojectApplication.class, args);

}

}



**QUESTION 2 - Create REST API to perform CURD Operation on the Employee table with columns: ID, NAME, DEPARTMENT, and SALARY. (CO4,CO5&K5)**

**EmployeeController.java**

package com.rest.curd.controller;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.\*;

import com.rest.curd.entity.Employee;

import com.rest.curd.service.EmployeeService;

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

private final EmployeeService employeeService;

@Autowired

public EmployeeController(EmployeeService employeeService) {

this.employeeService = employeeService;

}

@GetMapping

public ResponseEntity<List<Employee>> getAllEmployees() {

List<Employee> employees = employeeService.getAllEmployees();

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

}

@GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable Long id) {

Optional<Employee> employee = employeeService.getEmployeeById(id);

return employee.map(value -> new ResponseEntity<>(value, HttpStatus.OK))

.orElseGet(() -> new ResponseEntity<>(HttpStatus.NOT\_FOUND));

}

@PostMapping

public ResponseEntity<Employee> createEmployee(@RequestBody Employee employee) {

Employee createdEmployee = employeeService.createEmployee(employee);

return new ResponseEntity<>(createdEmployee, HttpStatus.CREATED);

}

@PutMapping("/{id}")

public ResponseEntity<Employee> updateEmployee(@PathVariable Long id, @RequestBody Employee employee) {

Employee updatedEmployee = employeeService.updateEmployee(id, employee);

if (updatedEmployee != null) {

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

} else {

return new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable Long id) {

employeeService.deleteEmployee(id);

return new ResponseEntity<>(HttpStatus.NO\_CONTENT);

}

@PatchMapping("/{id}/salary")

public ResponseEntity<Void> updateEmployeeSalary(@PathVariable Long id, @RequestParam Double salary) {

employeeService.updateEmployeeSalary(id, salary);

return new ResponseEntity<>(HttpStatus.OK);

}

}

**Employee.java**

package com.rest.curd.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Employee {

@Id

private Long id;

private String name;

private String department;

private Double salary;

// Default constructor (required by JPA)

public Employee() {

}

public Employee(String name, String department, Double salary) {

this.name = name;

this.department = department;

this.salary = salary;

}

// Getters and Setters

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 Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

}

**EmployeeRepository.java**

package com.rest.curd.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.data.jpa.repository.Modifying;

import org.springframework.data.jpa.repository.Query;

import org.springframework.stereotype.Repository;

import org.springframework.transaction.annotation.Transactional;

import com.rest.curd.entity.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

@Transactional

@Modifying

@Query("UPDATE Employee e SET e.salary = :salary WHERE e.id = :id")

void updateSalaryById(Long id, Double salary);

}

**EmployeeService.java**

package com.rest.curd.service;

import java.util.List;

import java.util.Optional;

import com.rest.curd.entity.Employee;

public interface EmployeeService {

List<Employee> getAllEmployees();

Optional<Employee> getEmployeeById(Long id);

Employee createEmployee(Employee employee);

Employee updateEmployee(Long id, Employee employee);

void deleteEmployee(Long id);

void updateEmployeeSalary(Long id, Double salary);

}

**EmployeeServiceImpl.java**

package com.rest.curd.service;

import org.springframework.stereotype.Service;

import com.rest.curd.entity.Employee;

import com.rest.curd.repository.EmployeeRepository;

import java.util.List;

import java.util.Optional;

@Service

public class EmployeeServiceImpl implements EmployeeService {

private final EmployeeRepository employeeRepository;

public EmployeeServiceImpl(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

@Override

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@Override

public Optional<Employee> getEmployeeById(Long id) {

return employeeRepository.findById(id);

}

@Override

public Employee createEmployee(Employee employee) {

return employeeRepository.save(employee);

}

@Override

public Employee updateEmployee(Long id, Employee employee) {

Optional<Employee> existingEmployee = employeeRepository.findById(id);

if (existingEmployee.isPresent()) {

Employee updatedEmployee = existingEmployee.get();

updatedEmployee.setName(employee.getName());

updatedEmployee.setDepartment(employee.getDepartment());

updatedEmployee.setSalary(employee.getSalary());

return employeeRepository.save(updatedEmployee);

}

return null; // Or throw an exception

}

@Override

public void deleteEmployee(Long id) {

employeeRepository.deleteById(id);

}

@Override

public void updateEmployeeSalary(Long id, Double salary) {

employeeRepository.updateSalaryById(id, salary);

}

}

**application.properties**

spring.application.name=RestAPI\_CURD-2

server.port=7007

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

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.h2.console.enabled=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

**Output-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**After updating salary 101-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**QUESTION 3: Create a web MVC Project using Spring Boot which will display “Welcome to Spring boot MVC” and value of request parameter name on HTML page. (CO4&K3)**

**BootController.java**

package com.boot.mvc;

import org.springframework.stereotype.Controller;

import org.springframework.ui.Model;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class BootController {

@GetMapping("/")

public String home(@RequestParam(name = "name", required = false, defaultValue = "Guest") String name, Model model) {

model.addAttribute("message", "Welcome to Spring Boot MVC");

model.addAttribute("name", name);

return "home";

}

}

**Home.html**

<!DOCTYPE html>

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

<head>

<title>Spring Boot MVC</title>

</head>

<body>

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

<p>Your name is: <span th:text="${name}"></span></p>

</body>

</html>

**application.properties**

spring.application.name=springbootmvc

server.port = 2025

**Output-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**

**QUESTION 4: Create a web MVC Project using Spring Boot which has a form name login.jsp, an entity class Users with field id, firstName and lastName. Use Spring Data JPA to save the record in database.(CO4, CO5&K5)**

**UserController.java**

package com.web.mvc;

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestParam;

@Controller

public class UserController {

@Autowired

private UserRepository userRepository;

@GetMapping("/login")

public String showLoginForm() {

return "login";

}

@PostMapping("/register")

public String registerUser(@RequestParam("id") Long id,

@RequestParam("firstName") String firstName,

@RequestParam("lastName") String lastName) {

Users user = new Users(firstName, lastName);

user.setId(id); // Set the ID from the form

userRepository.save(user);

return "success"; // Redirect to a success page

}

}

**Userrepository.java**

package com.web.mvc;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

@Repository

public interface UserRepository extends JpaRepository<Users, Long> {

// Spring Data JPA automatically provides methods for basic CRUD operations

// like save(), findById(), findAll(), deleteById(), etc.

}

**Users.java**

package com.web.mvc;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Users {

@Id

/\* @GeneratedValue(strategy = GenerationType.IDENTITY) \*/

private Long id;

private String firstName;

private String lastName;

public Users() {

}

public Users(String firstName, String lastName) {

this.firstName = firstName;

this.lastName = lastName;

}

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getFirstName() {

return firstName;

}

public void setFirstName(String firstName) {

this.firstName = firstName;

}

public String getLastName() {

return lastName;

}

public void setLastName(String lastName) {

this.lastName = lastName;

}

}

**Application.properties**

spring.application.name=springbootWebMVC

server.port=8585

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

spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect

spring.mvc.view.prefix=/views/

spring.mvc.view.suffix=.jsp

**login.jsp**

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<body>

<h1>User Registration</h1>

<form action="/register" method="post">

<div>

<label for="id">ID:</label>

<input type="text" name="id">

</div>

<br>

<div>

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

<input type="text" name="firstName" required>

</div>

<br>

<div>

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

<input type="text" name="lastName" required>

</div>

<br>

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

</form>

</body>

</html>

**Success.jsp**

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<body>

<h2>Registration Successful!</h2>

<p>The User Details have been saved.</p>

<p><a href="/login">Go back to the login form</a></p>

</body>

</html>

**Output-**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A white rectangular object with black border

AI-generated content may be incorrect.**

**A screenshot of a computer

AI-generated content may be incorrect.**