Problem Statement

**1. Spring Boot: Item Delivery Tracking**

Project Based : BackEnd

Attempted

Requirements

Implement REST APIs to add and fetch the employee records.

The Employee entity should have the following attributes:

* **id** : unique id of the employee (Integer)
* **name** : name of the employee (String)
* **state**: name of the state (String)
* **item** : item an employee delivers (String)

Here is an example of a Employee JSON object:

{

"id": 1,

"name": "John",

"state": "MP",

"item": "Food"

}

You are provided with the implementation of the models required for all the APIs. Your task is to implement a set of REST services that exposes the endpoints, enable the users to add the employee details and list the details of an employee id in the following ways:

| API Route | API Type | Success Response Code | Validation Error Code |

|-------------------------|----------|------------------------|------------------------|

| /employee | POST | 201 | 400 |

| /employee/{employeeId} | GET | 200 | 404 |

Task 1: Service Layer Implementation in ItemService

Implement the logic in the service layer using ItemRepository

**Method Details**:

* **addEmployee()**: Implement the POST method which should save the Employee data.
* **getEmployeeById()**: Implement the GET method which should get the Employee data based on the id.

Task 2: REST API Endpoints in EmployeeController

**POST request to /employees**

Save Employee to Database

**HTTP Status Code**:

* 201 - For a successful response
* 400 - If any of the parameters are null or empty

**GET request to /employees/{employeeId}**

Get the given Employee Id

**HTTP Status Code**:

* 200 - If the requested Id is found
* 404 - For the requested Id is not found

Complete the given project to pass all the test cases when running the provided unit tests.

**Example Requests and Responses:**

POST request to /employees:

Request Body

{

"name": "Deepak Jain",

"state": "Rajasthan",

"item": "Food"

}

The response code is 200 and the response body, when converted to JSON, is as follows:

{

"id" : 6,

"name": "Deepak Jain",

"state": "Rajasthan",

"item": "Food"

}

This adds a new object to the collection with the given properties having id 6.

GET request to /employees/{employeeId}

Request: GET /employee/1 Content-Type: application/json

Response:

{

"id" : 6,

"name": "Deepak Jain",

"state": "Rajasthan",

"item": "Food"

}

Response code: 200

**Note:**

* The endpoints for the DELETE method have been provided in the ItemController.java file

----------------------------------------------------------------------------------------------------------

Controller -ItemController.java

package com.example.item.controller;

import java.util.Optional;

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

import org.springframework.http.HttpStatus;

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.RestController;

import com.example.item.entity.Employee;

import com.example.item.service.ItemService;

@RestController

public class ItemController {

    @Autowired

    private ItemService itemService;

    @PostMapping("/employees")

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

        // Code here ...

        return new ResponseEntity<>(null);

    }

    @GetMapping("/employees/{employeeId}")

    public ResponseEntity<Optional<Employee>> getEmployee(@PathVariable Integer employeeId){

        // Code here ...

        return new ResponseEntity<>(null);

    }

    @DeleteMapping("/employees/{employeeId}")

    public ResponseEntity<String> deleteEmployee(@PathVariable Integer employeeId){

        try {

            this.itemService.deleteEmployeeById(employeeId);

            return new ResponseEntity<>(HttpStatus.OK);

        } catch (Exception e) {

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

        }

    }

}

Entity -Employee.java

package com.example.item.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="employees")

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Integer id;

    @Column(name="name")

    private String name;

    @Column(name="state")

    private String state;

    @Column(name="item")

    private String item;

    public Employee() {

    }

    public Employee(String name, String state, String item) {

        super();

        this.name = name;

        this.state = state;

        this.item = item;

    }

    public Integer getId() {

        return id;

    }

    public void setId(Integer id) {

        this.id = id;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getState() {

        return state;

    }

    public void setState(String state) {

        this.state = state;

    }

    public String getItem() {

        return item;

    }

    public void setItem(String item) {

        this.item = item;

    }

    @Override

    public String toString() {

        return "Employee [id=" + id + ", name=" + name + ", state=" + state + ", item=" + item + "]";

    }

}

Repository -EmployeeRepository.java

package com.example.item.repository;

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

import org.springframework.stereotype.Repository;

import com.example.item.entity.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

Service -ItemService.java

package com.example.item.service;

import java.util.Optional;

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

import org.springframework.http.HttpStatus;

import org.springframework.stereotype.Service;

import org.springframework.web.server.ResponseStatusException;

import com.example.item.entity.Employee;

import com.example.item.repository.EmployeeRepository;

@Service

public class ItemService {

    @Autowired

    private EmployeeRepository employeeRepo;

    public Optional<Employee> getEmployeeById(int employeeId) {

        // Code here ...

        return null;

    }

    public Employee addEmployee(Employee employee) {

        // Code here ...

        return null;

    }

    public void deleteEmployeeById(int employeeId) {

         employeeRepo.deleteById(employeeId);

    }

}

Resource- application.properties

spring.application.name=item

server.port = 8000

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

#local creds

spring.datasource.url=jdbc:mysql://localhost:3306/employeedb?createDatabaseIfNotExist=true

spring.datasource.username=coder

spring.datasource.password=coder

spring.jpa.show-sql=true

logging.level.org.springframework.web: DEBUG

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL57Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.generate-ddl=true

spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

AppTest.java

package com.example.item;

import static org.assertj.core.api.Assertions.assertThat;

import static org.hamcrest.CoreMatchers.is;

import static org.junit.jupiter.api.Assertions.assertEquals;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.delete;

import static org.springframework.test.web.servlet.request.MockMvcRequestBuilders.post;

import static org.springframework.test.web.servlet.result.MockMvcResultHandlers.print;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.jsonPath;

import static org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

import java.nio.charset.Charset;

import java.util.Optional;

import org.junit.jupiter.api.BeforeEach;

import org.junit.jupiter.api.DisplayName;

import org.junit.jupiter.api.Order;

import org.junit.jupiter.api.Test;

import org.mockito.Mockito;

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

import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMvc;

import org.springframework.boot.test.context.SpringBootTest;

import org.springframework.boot.test.mock.mockito.MockBean;

import org.springframework.http.MediaType;

import org.springframework.test.annotation.DirtiesContext;

import org.springframework.test.web.servlet.MockMvc;

import org.springframework.test.web.servlet.MvcResult;

import org.springframework.test.web.servlet.RequestBuilder;

import org.springframework.test.web.servlet.ResultActions;

import org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

import com.example.item.entity.Employee;

import com.example.item.repository.EmployeeRepository;

import com.example.item.service.ItemService;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.fasterxml.jackson.databind.ObjectWriter;

import com.fasterxml.jackson.databind.SerializationFeature;

@SpringBootTest

@DirtiesContext(classMode = DirtiesContext.ClassMode.BEFORE\_EACH\_TEST\_METHOD)

@AutoConfigureMockMvc

class AppTest {

    public static final MediaType APPLICATION\_JSON\_UTF8 = new MediaType(MediaType.APPLICATION\_JSON.getType(), MediaType.APPLICATION\_JSON.getSubtype(), Charset.forName("utf8"));

    @Autowired

    private MockMvc mockMvc;

    @Autowired

    private ItemService itemService;

    @Autowired

    private EmployeeRepository employeeRepository;

    @BeforeEach

    public void setUp() throws Exception {

        Employee employee1 = new Employee();

        employee1.setName("Harsha");

        employee1.setState("UP");

        employee1.setItem("Hoodies");

        employeeRepository.save(employee1);

        Employee employee2 = new Employee();

        employee2.setName("Viraj");

        employee2.setState("Bihar");

        employee2.setItem("EarPods");

        employeeRepository.save(employee2);

    }

    @Test

    @DisplayName("test post Employee entity")

    @Order(1)

    public void testSaveEmployees() throws Exception {

        Employee employee3 = new Employee();

        employee3.setName("Udesh");

        employee3.setState("Gujarat");

        employee3.setItem("Food");

        employeeRepository.save(employee3);

        Employee Employee = itemService.addEmployee(employee3);

        assertEquals(employee3.getId(), Employee.getId());

        assertEquals(employee3.getName(), Employee.getName());

        assertEquals(employee3.getState(), Employee.getState());

        assertEquals(employee3.getItem(), Employee.getItem());

    }

    @Test

    @DisplayName("test post Employee entity created response")

    @Order(2)

    public void testSaveEmployeesCreatedResponse() throws Exception {

        Employee employee3 = new Employee();

        employee3.setName("Udesh");

        employee3.setState("Gujarat");

        employee3.setItem("Food");

        employeeRepository.save(employee3);

        ObjectMapper mapper = new ObjectMapper();

        mapper.configure(SerializationFeature.WRAP\_ROOT\_VALUE, false);

        ObjectWriter ow = mapper.writer().withDefaultPrettyPrinter();

        String requestJson=ow.writeValueAsString(employee3);

        mockMvc.perform(post("/employees").contentType(APPLICATION\_JSON\_UTF8).content(requestJson))

                .andDo(print())

                .andExpect(status().isCreated())

                .andExpect(jsonPath("$.id", is(3)))

                .andExpect(jsonPath("$.name", is("Udesh")))

                .andExpect(jsonPath("$.state", is("Gujarat")))

                .andExpect(jsonPath("$.item", is("Food")));

    }

    @Test

    @DisplayName("test post Employee entity bad request for null")

    @Order(3)

    public void testSaveEmployeesBadRequestNull() throws Exception {

        Employee employee3 = new Employee();

        employee3.setName("Udesh");

        employee3.setState(null);

        employee3.setItem("Food");

        employeeRepository.save(employee3);

        ObjectMapper mapper = new ObjectMapper();

        mapper.configure(SerializationFeature.WRAP\_ROOT\_VALUE, false);

        ObjectWriter ow = mapper.writer().withDefaultPrettyPrinter();

        String requestJson=ow.writeValueAsString(employee3);

        mockMvc.perform(post("/employees").contentType(APPLICATION\_JSON\_UTF8).content(requestJson))

                .andDo(print())

                .andExpect(status().isBadRequest());

    }

    @Test

    @DisplayName("test post Employee entity bad request for empty data")

    @Order(4)

    public void testSaveEmployeesBadRequestEmpty() throws Exception {

        Employee employee3 = new Employee();

        employee3.setName("Udesh");

        employee3.setState("Gujarat");

        employee3.setItem("");

        employeeRepository.save(employee3);

        ObjectMapper mapper = new ObjectMapper();

        mapper.configure(SerializationFeature.WRAP\_ROOT\_VALUE, false);

        ObjectWriter ow = mapper.writer().withDefaultPrettyPrinter();

        String requestJson=ow.writeValueAsString(employee3);

        mockMvc.perform(post("/employees").contentType(APPLICATION\_JSON\_UTF8).content(requestJson))

                .andDo(print())

                .andExpect(status().isBadRequest());

    }

    @Test

    @Order(5)

    @DisplayName("check for get item by id")

    public void getItemsByIdOk() throws Exception{

        mockMvc.perform(MockMvcRequestBuilders

                .get("/employees/2")

                .contentType(MediaType.APPLICATION\_JSON))

                .andExpect(status().isOk())

                .andExpect(jsonPath("$.id", is(2)))

                .andExpect(jsonPath("$.name", is("Viraj")))

                .andExpect(jsonPath("$.state", is("Bihar")))

                .andExpect(jsonPath("$.item", is("EarPods")));

    }

    @Test

    @Order(6)

    @DisplayName("check for get item by id")

    public void getItemsByIdNotFound() throws Exception{

        mockMvc.perform(MockMvcRequestBuilders

                .get("/employees/3")

                .contentType(MediaType.APPLICATION\_JSON))

                .andExpect(status().isNotFound());

    }

}