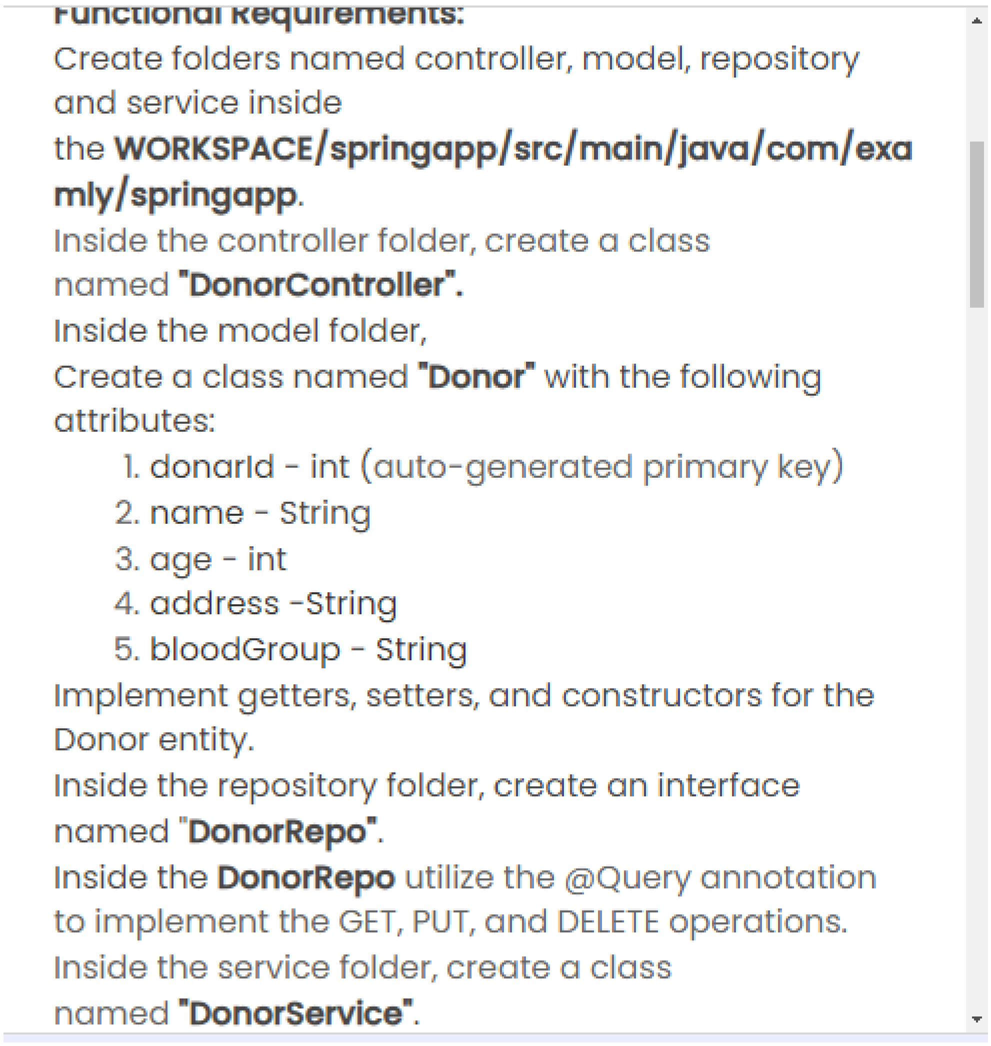
1.Donor



Functional Requirements:

Create folders named controller, model, repository and service inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp**.

Inside the controller folder, create a class named **"DonorController".**

Inside the model folder,

Create a class named **"Donor"** with the following attributes:

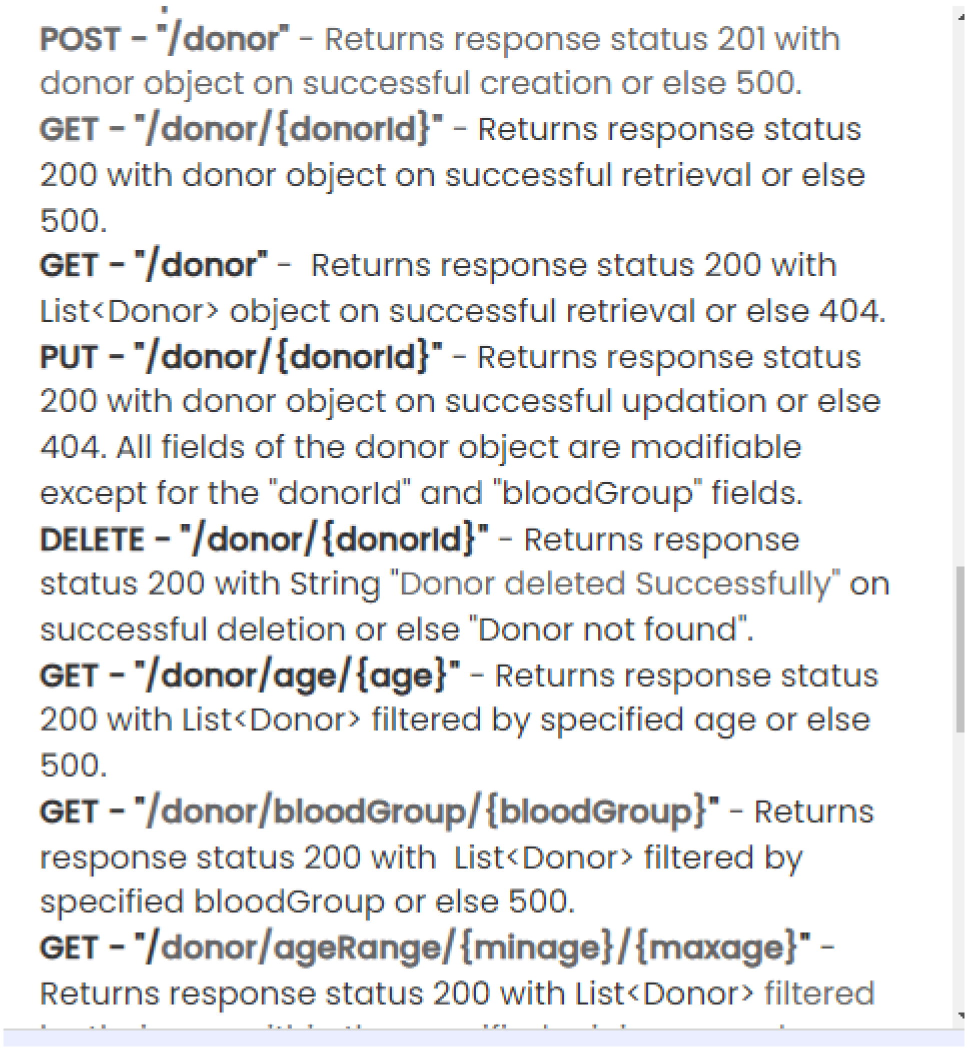
1. donarId - int (auto-generated primary key)
2. name - String
3. age - int
4. address -String
5. bloodGroup - String

Implement getters, setters, and constructors for the Donor entity.

Inside the repository folder, create an interface named "**DonorRepo**".

Inside the **DonorRepo** utilize the @Query annotation to implement the GET, PUT, and DELETE operations.

Inside the service folder, create a class named **"DonorService"**.


POST - "/donor" - Returns response status 201 with donor object on successful creation or else 500.
GET - "/donor/{donorId}" - Returns response status 200 with donor object on successful retrieval or else 500.
GET - "/donor" - Returns response status 200 with List<Donor> object on successful retrieval or else 404.
PUT - "/donor/{donorId}" - Returns response status 200 with donor object on successful updation or else 404. All fields of the donor object are modifiable except for the "donorId" and "bloodGroup" fields.
DELETE - "/donor/{donorId}" - Returns response status 200 with String "Donor deleted Successfully" on successful deletion or else "Donor not found".
GET - "/donor/age/{age}" - Returns response status 200 with List<Donor> filtered by specified age or else 500.
GET - "/donor/bloodGroup/{bloodGroup}" - Returns response status 200 with List<Donor> filtered by specified bloodGroup or else 500.
GET - "/donor/ageRange/{minage}/{maxage}" - Returns response status 200 with List<Donor> filtered

Donor.java
```java
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Donor {
    @Id
    int donorId;
    String name;
    int age;
    String address,bloodGroup;

    public Donor()
    {

    }
    public Donor(int donorId, String name, int age, String address, String
bloodGroup) {
        this.donorId = donorId;
        this.name = name;
        this.age = age;
        this.address = address;
        this.bloodGroup = bloodGroup;
    }

    public int getDonorId() {
        return donorId;
    }

    public void setDonorId(int donorId) {
        this.donorId = donorId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getBloodGroup() {
        return bloodGroup;
    }

    public void setBloodGroup(String bloodGroup) {
        this.bloodGroup = bloodGroup;
    }
```

```
}
```

DonorRepo.java

```
package com.examly.springapp.repository;

import java.util.List;

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

import com.examly.springapp.model.Donor;

public interface DonorRepo extends JpaRepository<Donor,Integer>{
    Donor findByDonorId(int donorId);
    @Query("SELECT d FROM Donor d ORDER BY d.age")
    List<Donor> findByDonorId();

    @Query("SELECT d FROM Donor d ORDER BY d.bloodGroup")
    List<Donor> findByBloodGroup();
    @Query("SELECT d FROM Donor  d where d.age BETWEEN :minAge AND :maxAge")
    List<Donor> findByAgeRange(int minAge, int maxAge);

}
```

DonorService.java

```
package com.examly.springapp.service;


import java.util.List;
import java.util.Optional;

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

import com.examly.springapp.model.Donor;
import com.examly.springapp.repository.DonorRepo;

@Service
public class DonorService {
    @Autowired
    DonorRepo donorRepo;
    public Donor addDonor(Donor donor) {
        return donorRepo.save(donor);
    }
```

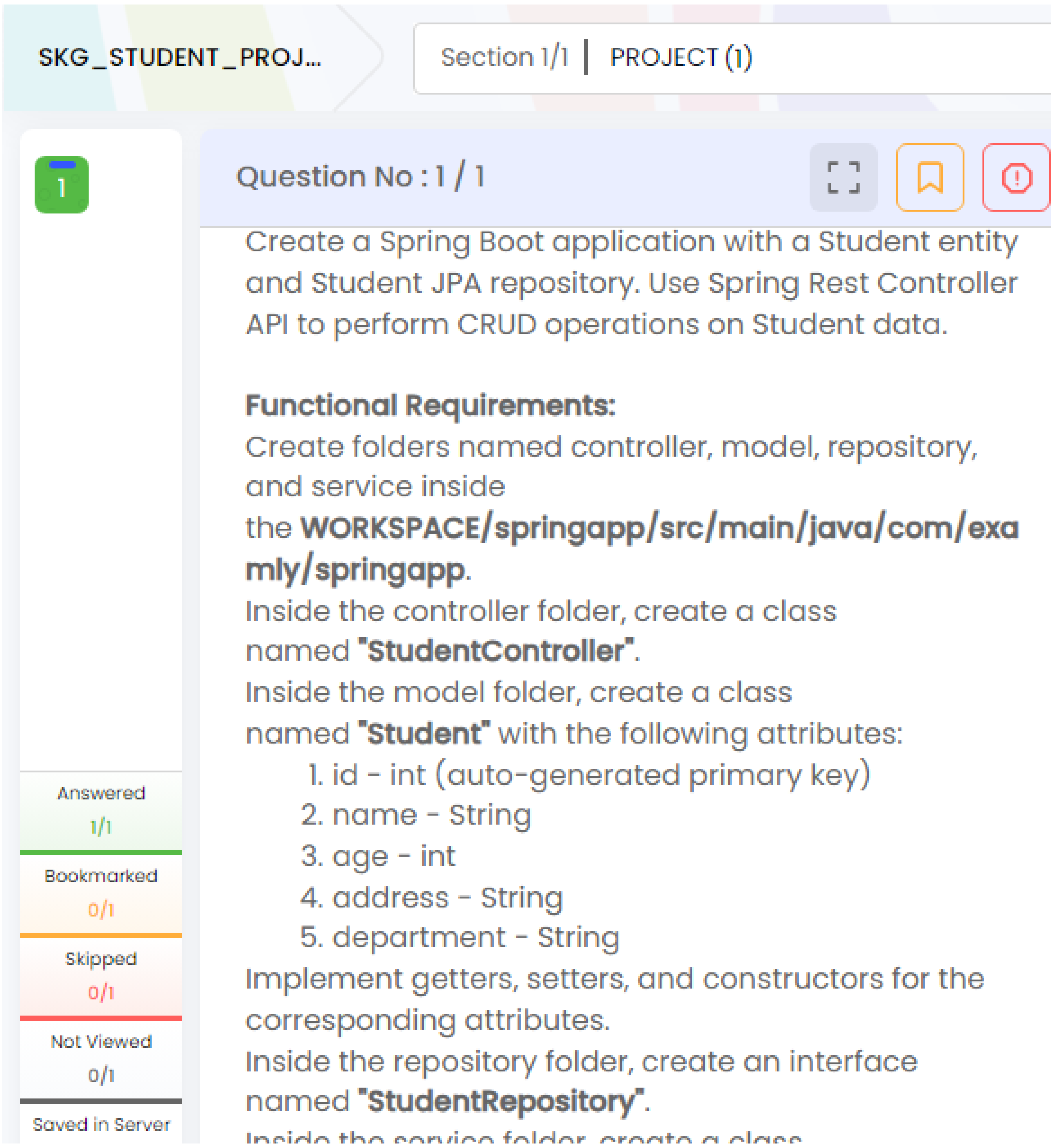
```java
public Donor getsDonor(int donorId) {
    return donorRepo.findByDonorId(donorId);
}

public List<Donor> getDonor() {
    return donorRepo.findAll();
}

public Donor updateDonor(int donorId, Donor donor) {
    Optional<Donor> donorExist = donorRepo.findById(donorId);
    if (donorExist.isPresent()) {
        Donor existingDonor = donorExist.get();
        existingDonor.setName(donor.getName());
        existingDonor.setAge(donor.getAge());
        existingDonor.setAddress(donor.getAddress());
        return donorRepo.save(donor);
    }
    return new Donor();
}

public boolean deleteDonor(int donorId) {
    Optional<Donor> donorExist=donorRepo.findById(donorId);
    if(donorExist.isPresent()){
        donorRepo.deleteById(donorId);
        System.out.println("Donor deleted Succesfully");
        return true;
    }
    else{
        System.out.println("Donor Not Found");
        return false;
    }
}

public List<Donor> sortAge() {
    return donorRepo.findByDonorId();
}

public List<Donor> sortByBloodGroup() {
    return donorRepo.findByBloodGroup();
}

public List<Donor> sortByAgeRange(int minAge, int maxAge) {
    return donorRepo.findByAgeRange(minAge, maxAge);
}


// //post
// public Donor post(Donor d) {
```

```
//     return dr.save(d);
// }

// public List<Donor> get()
// {
//     return dr.findAll();
// }

// public Donor getbyid(int donorId) {
//     return dr.findById(donorId).orElse(null);
// }

// //put
// public boolean update(int donorId,Donor d)
//     {
//         if(this.getbyid(donorId)==null)
//         {
//             return false;
//         }
//         try{
//             dr.save(d);
//         }
//         catch(Exception e)
//         {
//             return false;
//         }
//         return true;
//     }

//     //DELETE
// public boolean delete(int donorId)
//     {
//         if(this.getbyid(donorId) == null)
//         {
//             return false;
//         }
//         dr.deleteById(donorId);
//         return true;
//     }

//     //GET BY AGE
//     public List<Donor> getByAge(int age) {
//         List<Donor> donors = dr.findAll();
//         donors.removeIf(d -> d.getAge() != age);
//         return donors;
//     }

//     public List<Donor> getByBloodGroup(String bloodGroup) {
    //          List<Donor> donors = dr.findAll();
    //          donors.removeIf(d -> !d.getBloodGroup().equals(bloodGroup));
    //          return donors;
    //      }

    //      public List<Donor> getByAgeRange(int minage, int maxage) {
    //          List<Donor> donors = dr.findAll();
    //          donors.removeIf(d -> d.getAge() < minage || d.getAge() >
maxage);
    //          return donors;
    //      }




}
```

DonorController.java

```
package com.examly.springapp.controller;



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

import com.examly.springapp.model.Donor;
import com.examly.springapp.service.DonorService;

@RestController
public class DonorController {
    @Autowired
    DonorService service;

    @PostMapping("/donor")
    public ResponseEntity<?> addDonor(@RequestBody Donor donor) {
        try{
            return new
ResponseEntity<>(service.addDonor(donor),HttpStatus.CREATED);
           }
           catch(Exception e){
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
            }
    }

    @GetMapping("/donor/{donorId}")
    public ResponseEntity<?> getDonorById(@PathVariable int donorId) {
       try{
        return new ResponseEntity<>(service.getsDonor(donorId),HttpStatus.OK);
       }
       catch(Exception e){
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
       }
    }

    @GetMapping("/donor")
    public ResponseEntity<?> getAllDonors() {
        try{
            return new ResponseEntity<>(service.getDonor(),HttpStatus.OK);
           }
           catch(Exception e){
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
           }
    }

    @PutMapping("/donor/{donorId}")
    public ResponseEntity<?> updateDonor(@PathVariable int donorId,
@RequestBody Donor donorDetails) {
        try{
            return new
ResponseEntity<>(service.updateDonor(donorId,donorDetails),HttpStatus.OK);
           }
           catch(Exception e){
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
           }
    }

    @DeleteMapping("/donor/{donorId}")
    public ResponseEntity<?> deleteDonorById(@PathVariable int donorId) {
        boolean deletionStatus=service.deleteDonor(donorId);
        if(deletionStatus){
            return new ResponseEntity<>("Donor deleted
Successfully",HttpStatus.OK);
           }
        else{
            return new ResponseEntity<>("Donor Not
Found",HttpStatus.NOT_FOUND);
           }
    }
    @GetMapping("/donor/age/{age}")
    public ResponseEntity<?> getDonorsByAge()  {
        try{
            return new ResponseEntity<>(service.sortAge(),HttpStatus.OK);
           }
           catch(Exception e){
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
           }
    }

    @GetMapping("/donor/bloodGroup/{bloodGroup}")
    public ResponseEntity<?> getDonorsByBloodGroup() {
        try{
            return new
ResponseEntity<>(service.sortByBloodGroup(),HttpStatus.OK);
           }
           catch(Exception e){
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
           }
    }

    @GetMapping("/donor/ageRange/{minAge}/{maxAge}")
    public ResponseEntity<?> getDonorsByAgeRange(@PathVariable int minAge,
@PathVariable int maxAge) {
        try{
            return new ResponseEntity<>(service.sortByAgeRange(minAge,
maxAge),HttpStatus.OK);
           }
           catch(Exception e){
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }



}
```

2.



SKG_STUDENT_PROJ...

Section 1/1 | PROJECT (1)

1

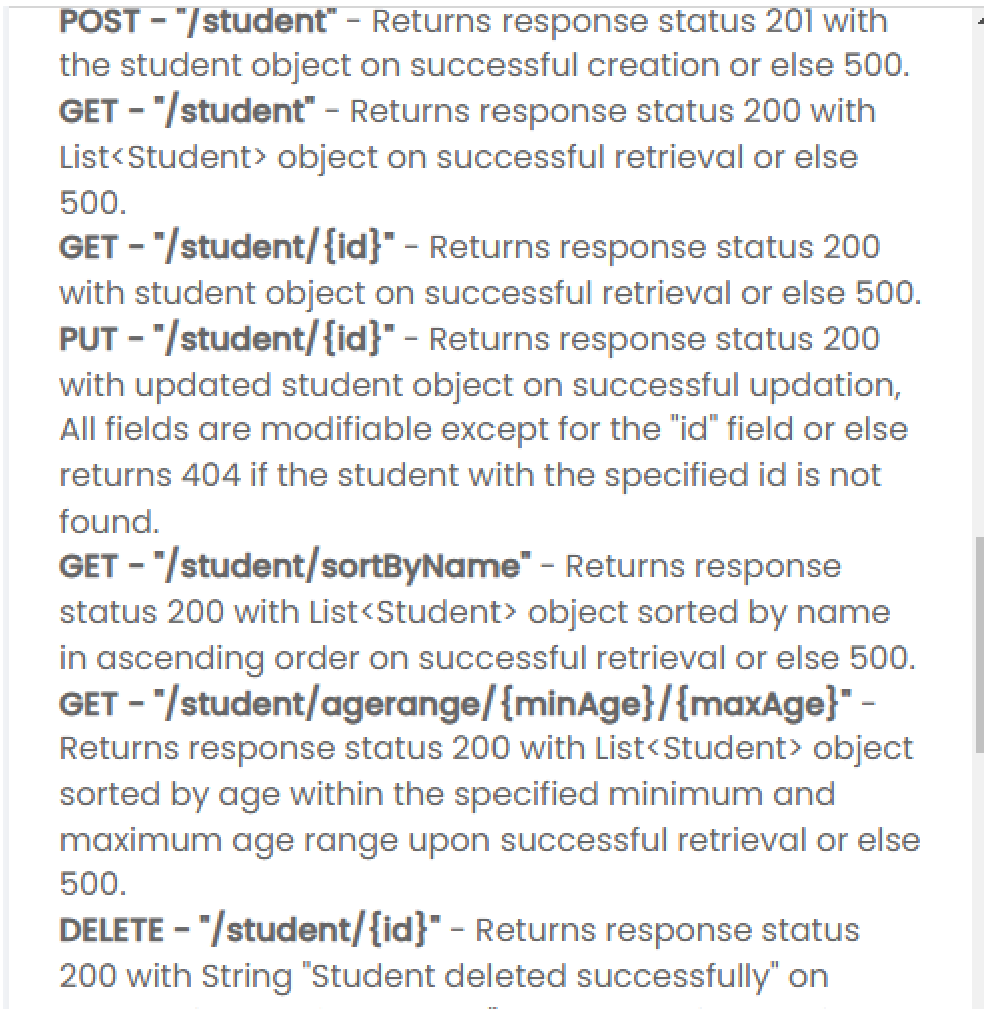
Question No : 1 / 1

Create a Spring Boot application with a Student entity and Student JPA repository. Use Spring Rest Controller API to perform CRUD operations on Student data.

Functional Requirements:
Create folders named controller, model, repository, and service inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp**.
Inside the controller folder, create a class named **"StudentController"**.
Inside the model folder, create a class named **"Student"** with the following attributes:

1. id - int (auto-generated primary key)
2. name - String
3. age - int
4. address - String
5. department - String

Implement getters, setters, and constructors for the corresponding attributes.
Inside the repository folder, create an interface named **"StudentRepository"**.



POST - "/student" - Returns response status 201 with the student object on successful creation or else 500.
GET - "/student" - Returns response status 200 with List<Student> object on successful retrieval or else 500.
GET - "/student/{id}" - Returns response status 200 with student object on successful retrieval or else 500.
PUT - "/student/{id}" - Returns response status 200 with updated student object on successful updation, All fields are modifiable except for the "id" field or else returns 404 if the student with the specified id is not found.
GET - "/student/sortByName" - Returns response status 200 with List<Student> object sorted by name in ascending order on successful retrieval or else 500.
GET - "/student/agerange/{minAge}/{maxAge}" - Returns response status 200 with List<Student> object sorted by age within the specified minimum and maximum age range upon successful retrieval or else 500.
DELETE - "/student/{id}" - Returns response status 200 with String "Student deleted successfully" on

Student.java

```
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Student {
    @Id
    int id;
    String name;
    int age;
    String address,department;

    public Student()
    {

    }
    public Student(int id, String name, int age, String address, String
department) {
        this.id = id;
        this.name = name;
        this.age = age;
        this.address = address;
        this.department = department;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getAddress() {
        return address;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public String getDepartment() {
        return department;
    }

    public void setDepartment(String department) {
        this.department = department;
    }
```

```
}
```

StudentRepository.java

```
package com.examly.springapp.repository;

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

import com.examly.springapp.model.Student;

public interface StudentRepository extends JpaRepository<Student,Integer>{

}
```

StudentService.java

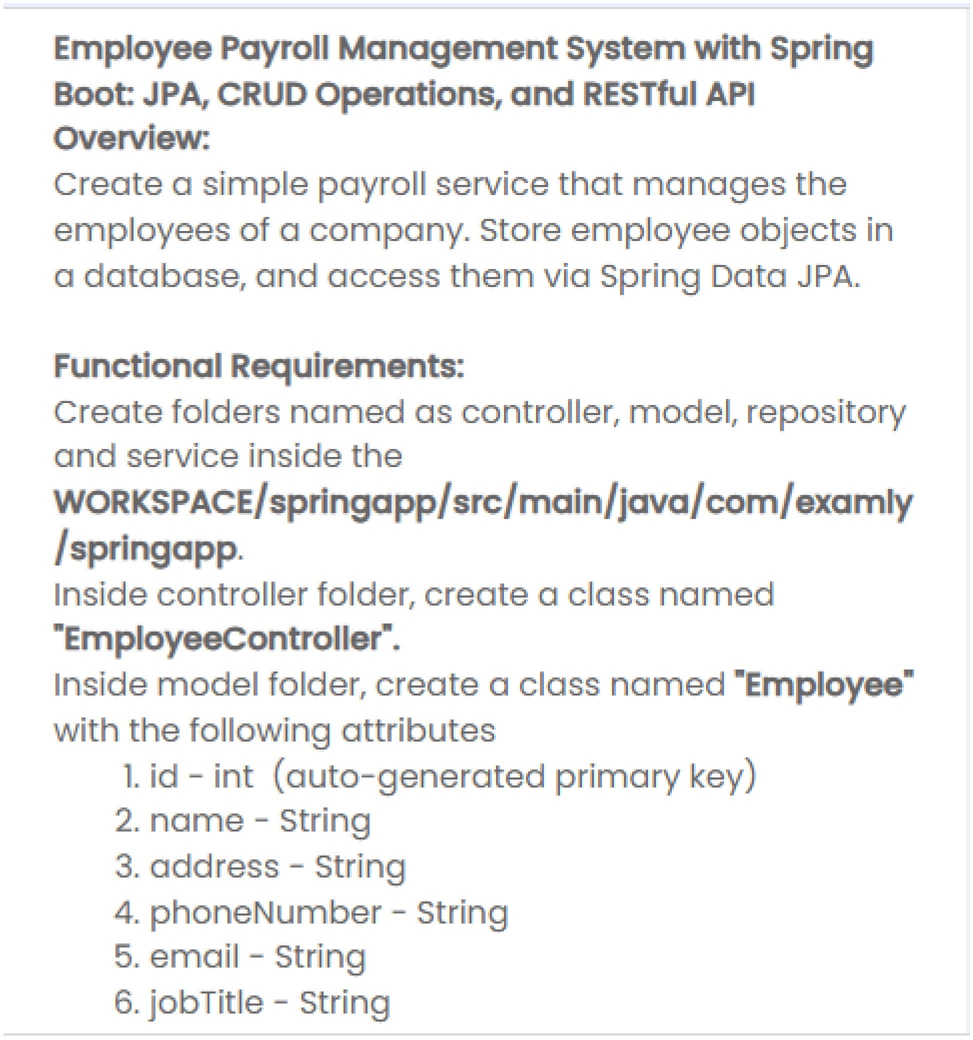
```
package com.examly.springapp.service;

import java.util.Comparator;
import java.util.List;

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

import com.examly.springapp.model.Student;
import com.examly.springapp.repository.StudentRepository;

@Service
public class StudentService {
    @Autowired
    StudentRepository sr;

    //post
    public Student create(Student s)
    {
        return sr.save(s);
    }

    //get
    public List<Student> getall()
    {
        return sr.findAll();
    }

    //getbyid
```

```java
public Student getById(int id)
{
    return sr.findById(id).orElse(null);
}

//putbyid
public boolean update(int id,Student s)
{
    if(this.getById(id)==null)
        {
            return false;
        }
        try{
            sr.save(s);
        }
        catch(Exception e)
        {
            return false;
        }
        return true;

}

//sorting by name
public List<Student> sortbyname()
{
    List<Student> students = sr.findAll();
    students.sort(Comparator.comparing(Student::getName));
    return students;
}

//pagination
public List<Student> getStudentsByAgeRange(int minAge, int maxAge) {
    List<Student> students = sr.findAll();
    students.removeIf(s -> s.getAge() < minAge || s.getAge() > maxAge);
    students.sort(Comparator.comparing(Student::getAge));
    return students;
}

//deletebyid
public boolean delete(int id)
    {
        if(this.getById(id) == null)
        {
            return false;
        }
        sr.deleteById(id);
        return true;
        }


}
```

Studentcontroller.java

```
package com.examly.springapp.controller;


import java.util.List;

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

import com.examly.springapp.model.Student;
import com.examly.springapp.service.StudentService;

@RestController
public class StudentController {
    @Autowired
    StudentService ss;

    //post
    @PostMapping("/student")
    public ResponseEntity<Student> post(@RequestBody Student s)
    {
        Student obj=ss.create(s);
        return new ResponseEntity<>(obj,HttpStatus.CREATED);
    }

    //GET
    @GetMapping("/student")
    public ResponseEntity<List<Student>> get()
    {
        return new ResponseEntity<>(ss.getall(),HttpStatus.OK);
    }

    //GETBYID
    @GetMapping("/student/{id}")
    public ResponseEntity<Student> getbyid(@PathVariable("id") int id)
    {
        return new ResponseEntity<>(ss.getById(id),HttpStatus.OK);
    }

    //PUT
    @PutMapping("/student/{id}")
    public ResponseEntity<Student> putMethod(@PathVariable("id") int
id,@RequestBody Student s)
    {
        if(ss.update(id,s) == true)
        {
            return new ResponseEntity<>(s,HttpStatus.OK);
        }

        return new ResponseEntity<>(null,HttpStatus.NOT_FOUND);
    }

    //delete
    @DeleteMapping("student/{id}")
    public ResponseEntity<?> deleteBookById(@PathVariable int id)
    {
        boolean deleted = ss.delete(id);
        if (deleted) {
            return ResponseEntity.ok("Student deleted successfully");
        } else {
            return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Student
not found with ID: " + id);
        }
    }


    //sorting
    @GetMapping("/student/sortedByName")
    public ResponseEntity<List<Student>> g()
    {
        List<Student> students = ss.sortbyname();
        return ResponseEntity.ok(students);
    }

    //pagination
    @GetMapping("/student/ageRange/{minAge}/{maxAge}")
    public ResponseEntity<List<Student>> getStudentsByAgeRange(@PathVariable
int minAge, @PathVariable int maxAge) {
        List<Student> students = ss.getStudentsByAgeRange(minAge, maxAge);
        return ResponseEntity.ok(students);
    }
}
```

3.WEB_LAB_EMPLOYEE



Employee Payroll Management System with Spring Boot: JPA, CRUD Operations, and RESTful API

Overview:

Create a simple payroll service that manages the employees of a company. Store employee objects in a database, and access them via Spring Data JPA.

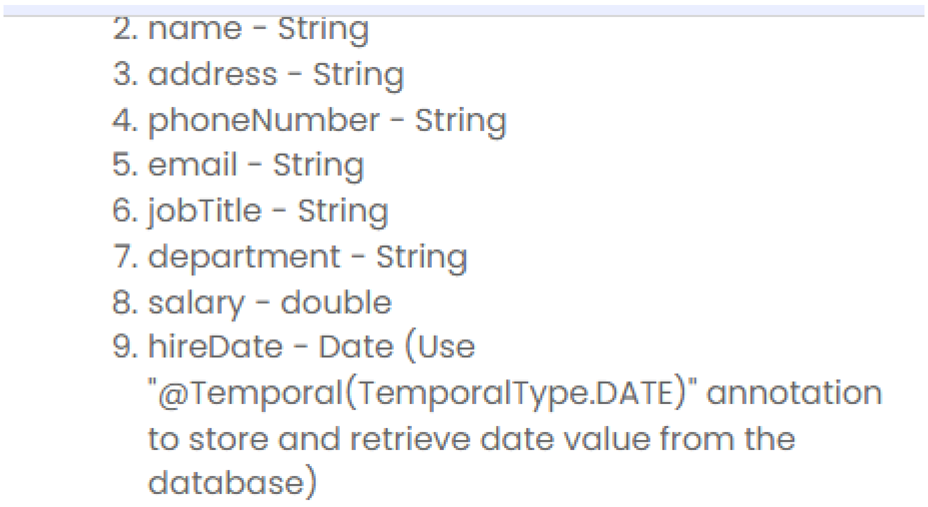
Functional Requirements:

Create folders named as controller, model, repository and service inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp**.

Inside controller folder, create a class named **"EmployeeController".**

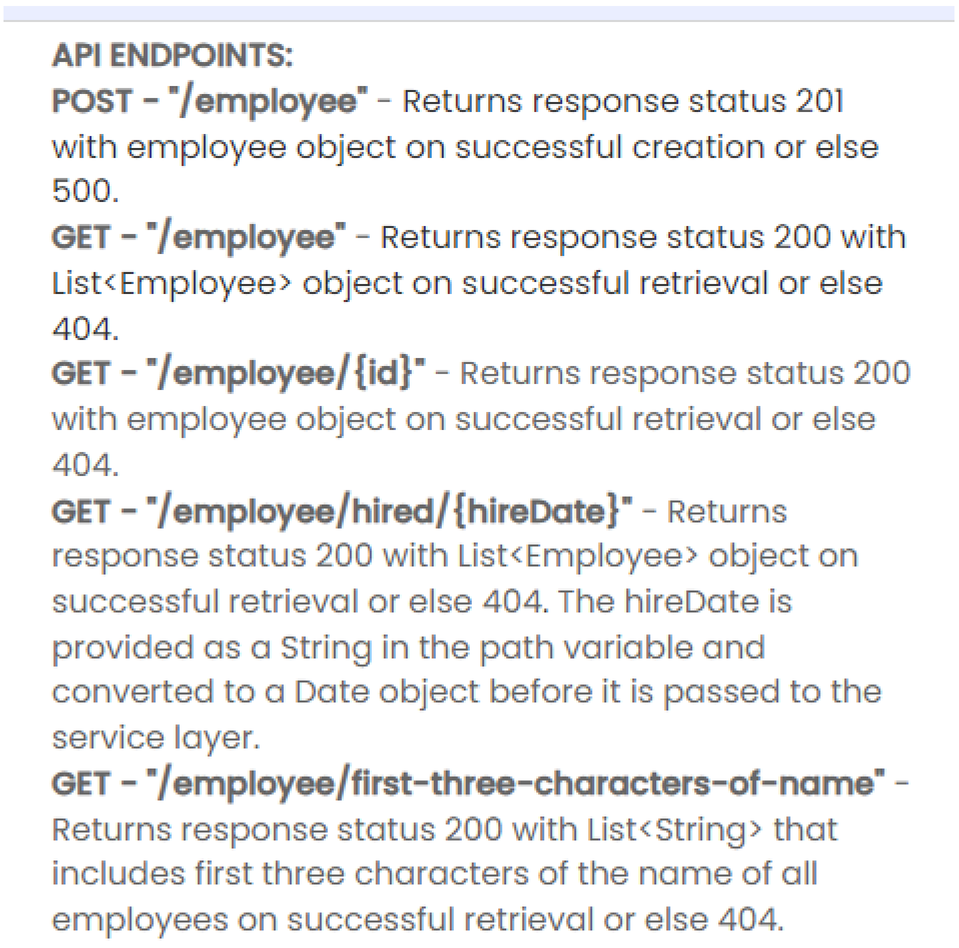
Inside model folder, create a class named **"Employee"** with the following attributes

1. id - int (auto-generated primary key)
2. name - String
3. address - String
4. phoneNumber - String
5. email - String
6. jobTitle - String



2. name - String
3. address - String
4. phoneNumber - String
5. email - String
6. jobTitle - String

7. department - String
8. salary - double
9. hireDate - Date (Use "@Temporal(TemporalType.DATE)" annotation to store and retrieve date value from the database)


API ENDPOINTS:
POST - "/employee" - Returns response status 201 with employee object on successful creation or else 500.
GET - "/employee" - Returns response status 200 with List<Employee> object on successful retrieval or else 404.
GET - "/employee/{id}" - Returns response status 200 with employee object on successful retrieval or else 404.
GET - "/employee/hired/{hireDate}" - Returns response status 200 with List<Employee> object on successful retrieval or else 404. The hireDate is provided as a String in the path variable and converted to a Date object before it is passed to the service layer.
GET - "/employee/first-three-characters-of-name" - Returns response status 200 with List<String> that includes first three characters of the name of all employees on successful retrieval or else 404.

Employee.java

```
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.Id;
import java.util.Date;
@Entity
public class Employee {
    @Id
    int id;
    String name,address,phoneNumber,email,jobTitle,department;
    double salary;
    Date hireDate;
    public Employee() {
    }
    public Employee(int id, String name, String address, String phoneNumber,
String email, String jobTitle,
            String department, double salary, Date hireDate) {
        this.id = id;
        this.name = name;
        this.address = address;
        this.phoneNumber = phoneNumber;
        this.email = email;
        this.jobTitle = jobTitle;
        this.department = department;
        this.salary = salary;
        this.hireDate = hireDate;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getAddress() {
        return address;
    }
    public void setAddress(String address) {
        this.address = address;
    }
    public String getPhoneNumber() {
        return phoneNumber;
    }
    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getJobTitle() {
        return jobTitle;
    }
    public void setJobTitle(String jobTitle) {
        this.jobTitle = jobTitle;
    }
    public String getDepartment() {
        return department;
    }
    public void setDepartment(String department) {
```

```java
        this.department = department;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }
    public Date getHireDate() {
        return hireDate;
    }
    public void setHireDate(Date hireDate) {
        this.hireDate = hireDate;
    }


}
```

EmployeeRepo.java

```java
package com.examly.springapp.repository;

import java.util.Date;
import java.util.List;

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

import com.examly.springapp.model.Employee;

public interface EmployeeRepo extends JpaRepository<Employee,Integer>{

    List<Employee> findByHireDate(Date hireDates);
    @Query("SELECT SUBSTRING(e.name, 1, 3) FROM Employee e")
    List<String> findFirstThreeCharactersOfAllNames();

}
```

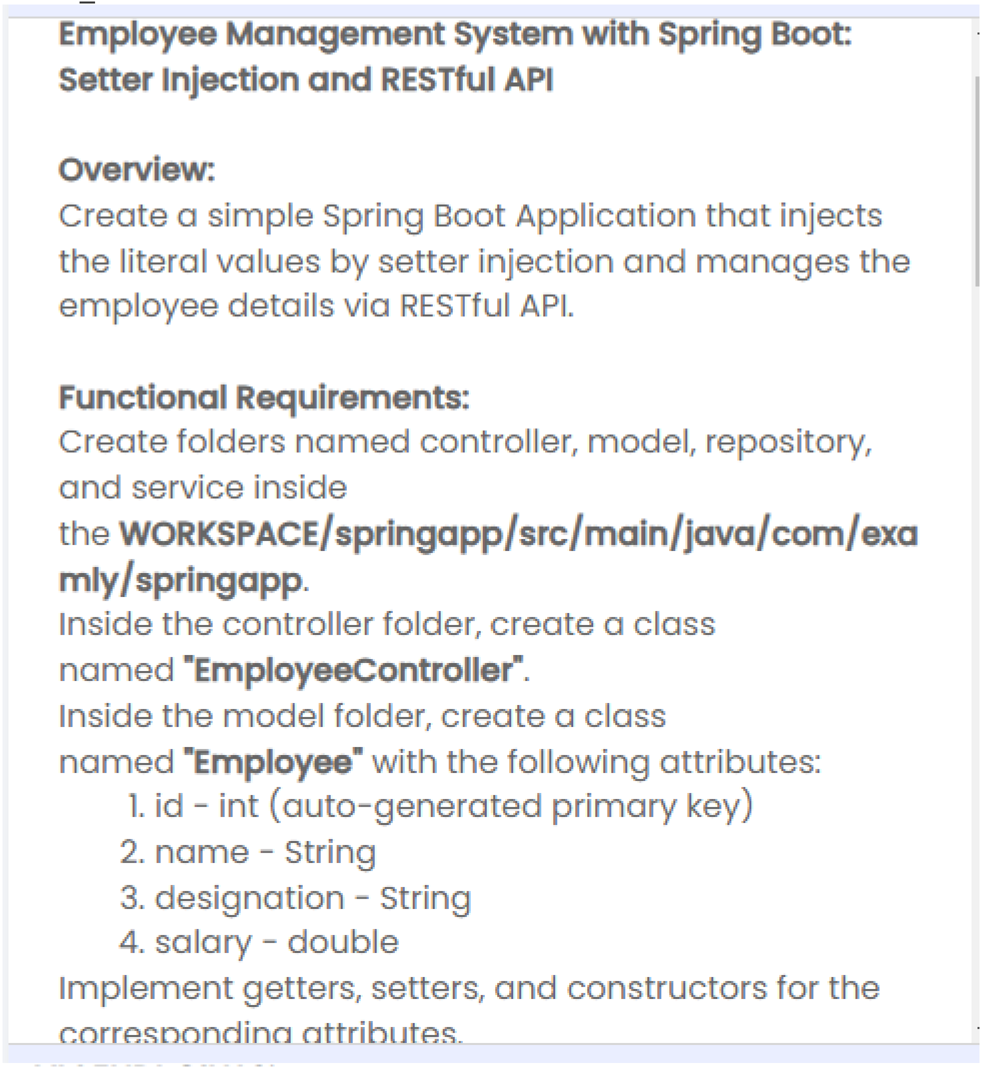
EmployeeService.java

```java
package com.examly.springapp.service;

import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.List;
import java.util.Optional;

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

import com.examly.springapp.model.Employee;
import com.examly.springapp.repository.EmployeeRepo;

@Service
public class EmployeeService {
    @Autowired
    EmployeeRepo er;

    //post
    public Employee postd(Employee emp) {
        return er.save(emp);
    }

    //getall
    public List<Employee> getd() {
        return er.findAll();
    }

    //getbyid
    public Optional<Employee> gettd(int id) {
        return er.findById(id);
    }


    public List<Employee> getttd(String hireDate) throws ParseException {
        Date hireDates = new SimpleDateFormat("yyyy-MM-dd").parse(hireDate);
        return er.findByHireDate(hireDates);
    }

    public List<String> gettttd() {
        return er.findFirstThreeCharactersOfAllNames();
    }
}
```

EmployeeController.java

```java
package com.examly.springapp.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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.examly.springapp.model.Employee;
import com.examly.springapp.service.EmployeeService;

@RestController
public class EmployeeController {
    @Autowired
    EmployeeService es;
     @PostMapping("/employee")
    public ResponseEntity<?> postd(@RequestBody Employee emp)
    {
        try
        {
            return new ResponseEntity<>(es.postd(emp),HttpStatus.CREATED);
        }catch(Exception e)
        {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }

    @GetMapping("/employee")
    public ResponseEntity<?> getd()
    {
        try
        {
            return new ResponseEntity<>(es.getd(),HttpStatus.OK);
        }catch(Exception e)
        {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    @GetMapping("/employee/{id}")
    public ResponseEntity<?> gettd(@PathVariable int id)
    {
        try
        {
            return new ResponseEntity<>(es.gettd(id),HttpStatus.OK);
        }catch(Exception e)
        {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
    @GetMapping("/employee/hired/{hireDate}")
    public ResponseEntity<?> getttd(@PathVariable String hireDate)
    {
        try
        {
            return new ResponseEntity<>(es.gettd(hireDate),HttpStatus.OK);
        }catch(Exception e)
        {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    @GetMapping("/employee/first-three-characters-of-name")
    public ResponseEntity<?> getttd()
    {
        try
        {
            return new ResponseEntity<>(es.getttd(),HttpStatus.OK);
        }catch(Exception e)
        {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

}
```

4.LAB_EMPLOYEE



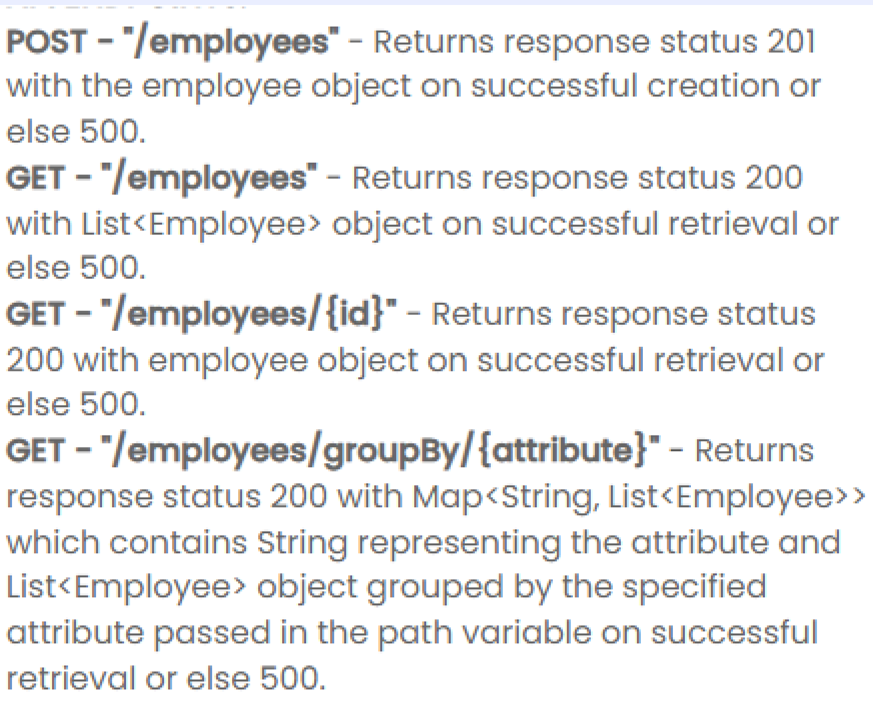
Employee Management System with Spring Boot: Setter Injection and RESTful API

Overview:
Create a simple Spring Boot Application that injects the literal values by setter injection and manages the employee details via RESTful API.

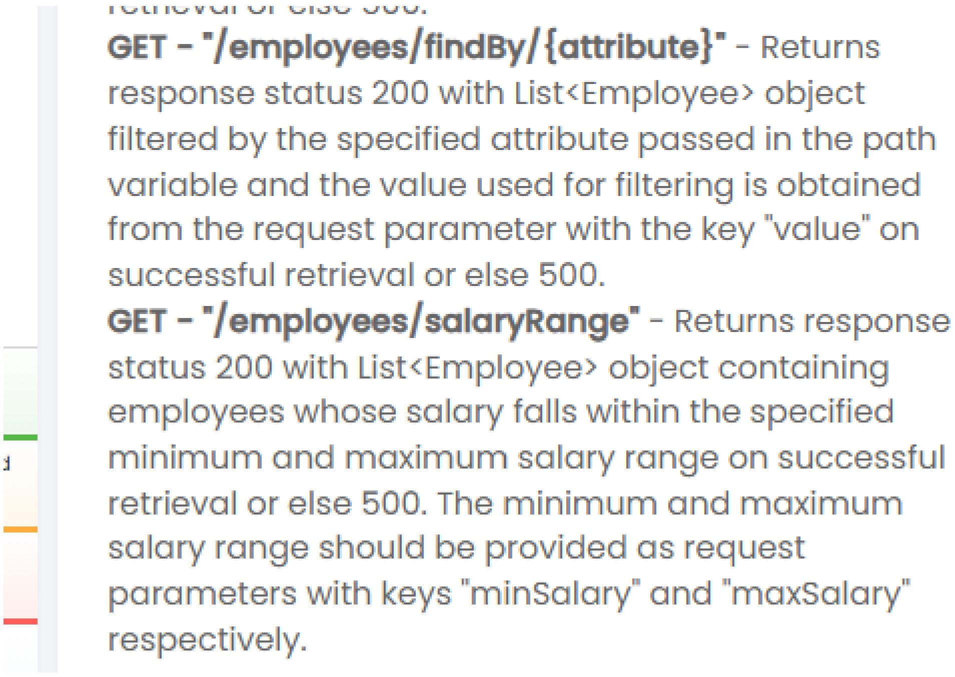
Functional Requirements:
Create folders named controller, model, repository, and service inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp**.
Inside the controller folder, create a class named **"EmployeeController"**.
Inside the model folder, create a class named **"Employee"** with the following attributes:

1. id - int (auto-generated primary key)
2. name - String
3. designation - String
4. salary - double

Implement getters, setters, and constructors for the corresponding attributes.



POST - "/employees" - Returns response status 201 with the employee object on successful creation or else 500.
GET - "/employees" - Returns response status 200 with List<Employee> object on successful retrieval or else 500.
GET - "/employees/{id}" - Returns response status 200 with employee object on successful retrieval or else 500.
GET - "/employees/groupBy/{attribute}" - Returns response status 200 with Map<String, List<Employee>> which contains String representing the attribute and List<Employee> object grouped by the specified attribute passed in the path variable on successful retrieval or else 500.



retrieval or else 500.

GET - "/employees/findBy/{attribute}" - Returns response status 200 with List<Employee> object filtered by the specified attribute passed in the path variable and the value used for filtering is obtained from the request parameter with the key "value" on successful retrieval or else 500.

GET - "/employees/salaryRange" - Returns response status 200 with List<Employee> object containing employees whose salary falls within the specified minimum and maximum salary range on successful retrieval or else 500. The minimum and maximum salary range should be provided as request parameters with keys "minSalary" and "maxSalary" respectively.

```
Employee.java
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Employee {
    @Id
    int id;
    String name,designation;
    double salary;
    public Employee() {
    }
    public Employee(int id, String name, String designation, double salary) {
        this.id = id;
        this.name = name;
        this.designation = designation;
        this.salary = salary;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getDesignation() {
        return designation;
    }
    public void setDesignation(String designation) {
        this.designation = designation;
    }
    public double getSalary() {
        return salary;
    }
    public void setSalary(double salary) {
        this.salary = salary;
    }


}
```

```
EmployeeRepo.java
package com.examly.springapp.repository;

import java.util.List;

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

import com.examly.springapp.model.Employee;

public interface EmployeeRepo extends JpaRepository<Employee,Integer>{
    List<Employee> findByName(String value);

    List<Employee> findByDesignation(String value);

    List<Employee> findBySalaryBetween(double minSalary, double maxSalary);
}
```

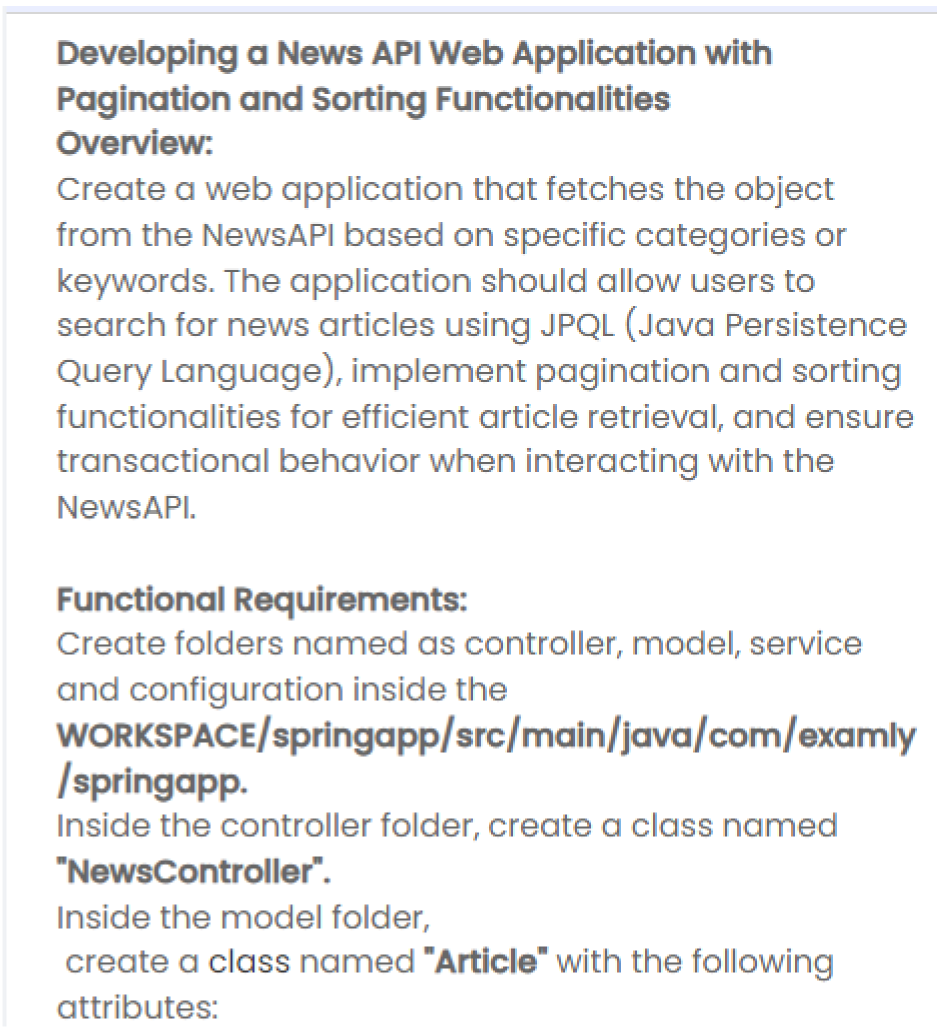
```
EmployeeService.java
package com.examly.springapp.service;

import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

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

import com.examly.springapp.model.Employee;
```

```java
import com.examly.springapp.repository.EmployeeRepo;

@Service
public class EmployeeService {
    @Autowired
    EmployeeRepo er;

    //post
    public Employee create(Employee e)
    {
        return er.save(e);
    }

    //get
    public List<Employee> getall()
    {
        return er.findAll();
    }

    //getbyid
    public Employee getById(int id)
    {
        return er.findById(id).orElse(null);
    }

    //group by
 public Map<String, List<Employee>> groupByAttribute(String attribute) {
        List<Employee> employees = er.findAll();
        return employees.stream().collect(Collectors.groupingBy(
                e -> {
                    switch (attribute) {
                        case "name":
                            return e.getName();
                        case "designation":
                            return e.getDesignation();
                        default:
                            return "Invalid Attribute";
                    }
                }
        ));
    }
//findby
    public List<Employee> findByAttribute(String attribute, String value) {
        if ("name".equals(attribute)) {
            return er.findByName(value);
        } else if ("designation".equals(attribute)) {
            return er.findByDesignation(value);
        } else {
            return null; // Handle invalid attribute
        }
    }
//range
    public List<Employee> getEmployeesInSalaryRange(double minSalary, double
maxSalary) {
        return er.findBySalaryBetween(minSalary, maxSalary);
    }
}
```

EmployeeController.java

```
package com.examly.springapp.controller;

import java.util.List;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.examly.springapp.model.Employee;
import com.examly.springapp.service.EmployeeService;

@RestController
public class EmployeeController {
    @Autowired
    EmployeeService es;

    //post
    @PostMapping("/employees")
    public ResponseEntity<Employee> post(@RequestBody Employee e)
    {
        Employee obj=es.create(e);
        return new ResponseEntity<>(obj,HttpStatus.CREATED);
    }

    //get
    @GetMapping("/employees")
    public ResponseEntity<List<Employee>> get()
    {
        return new ResponseEntity<>(es.getall(),HttpStatus.OK);
    }
    //getbyid
    @GetMapping("/employees/{id}")
    public ResponseEntity<Employee> getbyid(@PathVariable("id") int id)
    {
        return new ResponseEntity<>(es.getById(id),HttpStatus.OK);
    }

     @GetMapping("/employees/groupBy/{attribute}")
    public Map<String, List<Employee>> groupByAttribute(@PathVariable String
attribute) {
        return es.groupByAttribute(attribute);
    }

    @GetMapping("/employees/findBy/{attribute}")
    public List<Employee> findByAttribute(@PathVariable String attribute,
@RequestParam String value) {
        return es.findByAttribute(attribute, value);
    }

    @GetMapping("/employees/salaryRange")
    public List<Employee> getEmployeesInSalaryRange(@RequestParam double
minSalary, @RequestParam double maxSalary) {
        return es.getEmployeesInSalaryRange(minSalary, maxSalary);
    }


}
```

5.WEB_LAB_NEWSAPI



Developing a News API Web Application with Pagination and Sorting Functionalities

Overview:

Create a web application that fetches the object from the NewsAPI based on specific categories or keywords. The application should allow users to search for news articles using JPQL (Java Persistence Query Language), implement pagination and sorting functionalities for efficient article retrieval, and ensure transactional behavior when interacting with the NewsAPI.

Functional Requirements:

Create folders named as controller, model, service and configuration inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp.**

Inside the controller folder, create a class named **"NewsController".**

Inside the model folder,
 create a class named **"Article"** with the following attributes:

Inside the controller folder, create a class named **"NewsController".**
Inside the model folder,
create a class named **"Article"** with the following attributes:

1. author - String
2. title - String
3. description - String
4. url - String
5. urlToImage - String
6. publishedAt - Date
7. content - String
8. source - Source

create a class named **"Source"** with the following attributes:

1. id - String
2. name - String

create a class named **"ApiReponse"** with the following attributes:

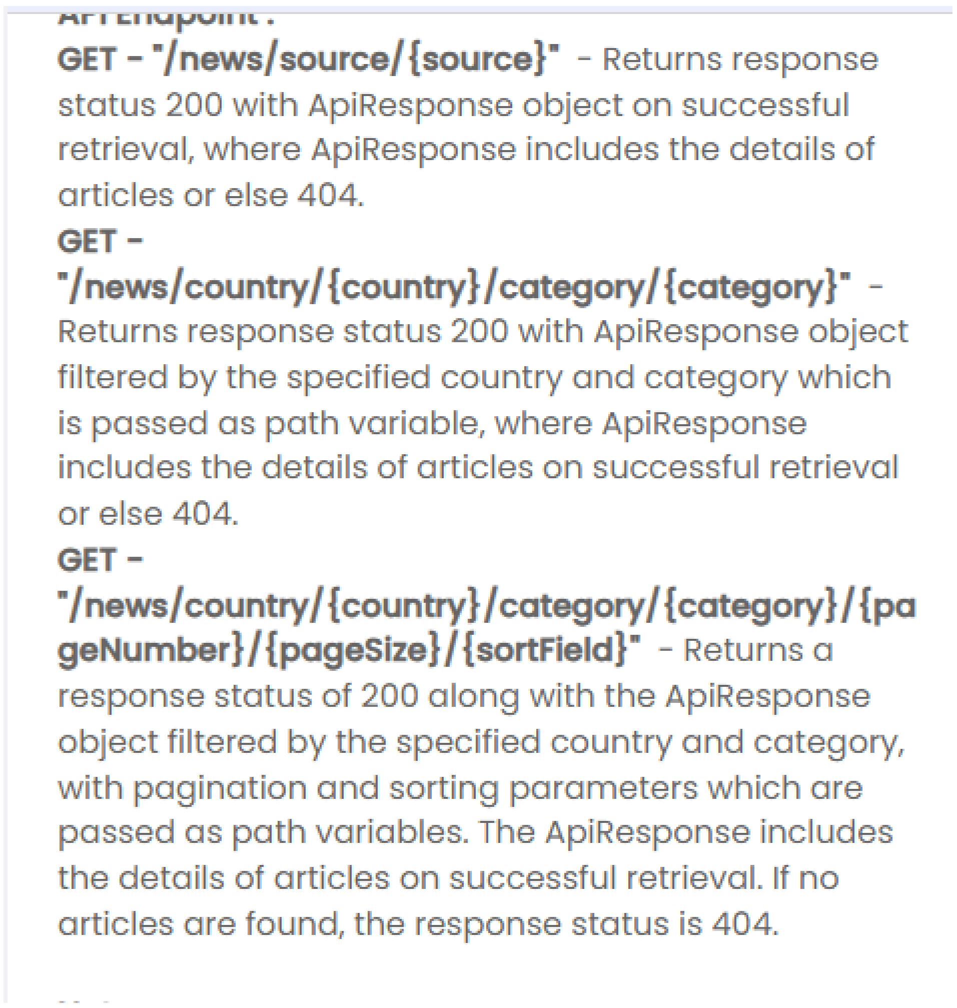
1. articles - Article[ ]

Implement getters, setters and constructors for the
corresponding attributes of Article, Source and

corresponding attributes of Article, Source and ApiResponse classes.
Inside service folder, create a class named "**NewsService**".
Inside configuration folder, create a class named **"AppConfig".**
Annotate the **AppConfig** class with @Configuration annotation and define a **RestTemplate** bean.
Utilize the NewsAPI to fetch news articles.

- Access the API at **https://newsapi.org/**
- The apiKey is already specified in the application.properties file.
- In the NewsService class, use the @Value annotation to retrieve the value of apiKey from the application.properties file.

Example API Endpoint: Use the following API endpoint to retrieve news articles based on specified country:
GET – "https://newsapi.org/v2/top-headlines?country={country}&category={category}&apiKey={API KEY}" - Returns the latest news articles for the specified country and category.



API Endpoint :

GET - "/news/source/{source}" - Returns response status 200 with ApiResponse object on successful retrieval, where ApiResponse includes the details of articles or else 404.

GET - "/news/country/{country}/category/{category}" - Returns response status 200 with ApiResponse object filtered by the specified country and category which is passed as path variable, where ApiResponse includes the details of articles on successful retrieval or else 404.

GET - "/news/country/{country}/category/{category}/{pageNumber}/{pageSize}/{sortField}" - Returns a response status of 200 along with the ApiResponse object filtered by the specified country and category, with pagination and sorting parameters which are passed as path variables. The ApiResponse includes the details of articles on successful retrieval. If no articles are found, the response status is 404.

Article.java

```
package com.examly.springapp.model;

import java.util.Date;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Article {

    @Id
    private String author;
    private String title;
    private String description;
    private String url;
    private String urlToImage;
    private Date publishedAt;
    private String content;
    private Source source;
    public Article() {
    }
    public Article(String author, String title, String description, String
url, String urlToImage, Date publishedAt,
            String content, Source source) {
        this.author = author;
        this.title = title;
        this.description = description;
        this.url = url;
        this.urlToImage = urlToImage;
        this.publishedAt = publishedAt;
        this.content = content;
        this.source = source;
    }
    public String getAuthor() {
        return author;
    }
    public void setAuthor(String author) {
        this.author = author;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getDescription() {
        return description;
    }
    public void setDescription(String description) {
        this.description = description;
    }
    public String getUrl() {
        return url;
    }
    public void setUrl(String url) {
        this.url = url;
    }
    public String getUrlToImage() {
        return urlToImage;
    }
    public void setUrlToImage(String urlToImage) {
        this.urlToImage = urlToImage;
    }
    public Date getPublishedAt() {
        return publishedAt;
    }
    public void setPublishedAt(Date publishedAt) {
        this.publishedAt = publishedAt;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Source getSource() {
        return source;
    }
    public void setSource(Source source) {
        this.source = source;
    }




}
```

Source.java

```
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class Source {
    @Id
    String id;
    String name;
    public Source() {
    }
    public Source(String id, String name) {
        this.id = id;
        this.name = name;
    }
    public String getId() {
        return id;
    }
    public void setId(String id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
```

```java
        this.name = name;
    }



}
```

ApiResponse.java

```java
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.Id;

@Entity
public class ApiResponse {
    @Id
    private Article[] articles;

    public ApiResponse() {
    }

    public ApiResponse(Article[] articles) {
        this.articles = articles;
    }

    public Article[] getArticles() {
        return articles;
    }

    public void setArticles(Article[] articles) {
        this.articles = articles;
    }
}
```

Service:

Newsservice

```java
package com.examly.springapp.service;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Service;
import org.springframework.web.client.RestTemplate;

import com.examly.springapp.model.ApiResponse;
```

```
@Service
public class NewsService {
    @Value("${newsapi.apikey}")
    private String apiKey;

    private final RestTemplate restTemplate;

    public NewsService(RestTemplate restTemplate) {
        this.restTemplate = restTemplate;
    }

    public ApiResponse getNewsBySource(String source) {
        String url = String.format("https://newsapi.org/v2/top-
headlines?sources=%s&apiKey=%s", source, apiKey);
        ResponseEntity<ApiResponse> response = restTemplate.getForEntity(url,
ApiResponse.class);
        return response.getBody();
    }

    public ApiResponse getNewsByCountryAndCategory(String country, String
category) {
        String url = String.format("https://newsapi.org/v2/top-
headlines?country=%s&category=%s&apiKey=%s", country, category, apiKey);
        ResponseEntity<ApiResponse> response = restTemplate.getForEntity(url,
ApiResponse.class);
        return response.getBody();
    }

    public ApiResponse getNewsByCountryCategoryWithPaginationAndSort(String
country, String category, int pageNumber, int pageSize, String sortField) {
        // This example URL does not directly support pagination and sorting
as these features are not standard for the NewsAPI.
        // Instead, this method demonstrates a placeholder for how you might
structure the method.
        // You'll need to handle pagination and sorting manually within your
application or use a different API if required.
        String url = String.format("https://newsapi.org/v2/top-
headlines?country=%s&category=%s&apiKey=%s", country, category, apiKey);
        ResponseEntity<ApiResponse> response = restTemplate.getForEntity(url,
ApiResponse.class);
        return response.getBody(); // Remember to adjust this method based on
your actual pagination and sorting logic.
    }
}
```

Controller:

NewsController.java

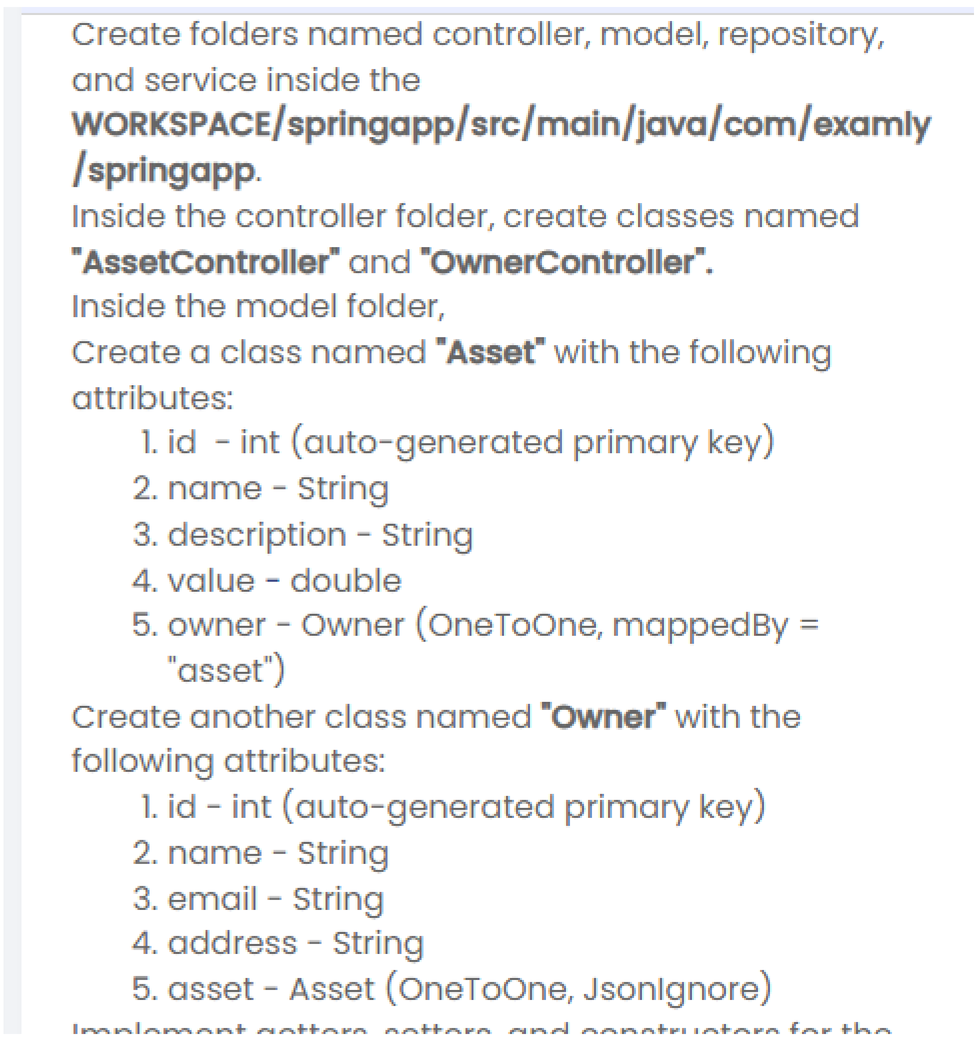
```java
package com.examly.springapp.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.examly.springapp.model.ApiResponse;
import com.examly.springapp.service.NewsService;

@RestController
@RequestMapping("/news")
public class NewsController {

    @Autowired
    NewsService newsService;

    @GetMapping("/source/{source}")
    public ResponseEntity<ApiResponse> getNewsBySource(@PathVariable String
source) {
        ApiResponse apiResponse = newsService.getNewsBySource(source);
        if (apiResponse == null || apiResponse.getArticles() == null ||
apiResponse.getArticles().length == 0) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(apiResponse);
    }

    @GetMapping("/country/{country}/category/{category}")
    public ResponseEntity<ApiResponse>
getNewsByCountryAndCategory(@PathVariable String country, @PathVariable String
category) {
        ApiResponse apiResponse =
newsService.getNewsByCountryAndCategory(country, category);
        if (apiResponse == null || apiResponse.getArticles() == null ||
apiResponse.getArticles().length == 0) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(apiResponse);
    }

    @GetMapping("/country/{country}/category/{category}/{pageNumber}/{pageSize
}/{sortField}")
    public ResponseEntity<ApiResponse>
getNewsByCountryCategoryWithPaginationAndSort(@PathVariable String country,
@PathVariable String category, @PathVariable int pageNumber, @PathVariable int
pageSize, @PathVariable String sortField) {
        ApiResponse apiResponse =
newsService.getNewsByCountryCategoryWithPaginationAndSort(country, category,
pageNumber, pageSize, sortField);
        if (apiResponse == null || apiResponse.getArticles() == null ||
apiResponse.getArticles().length == 0) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(apiResponse);
    }
}
```

Configuration

AppConfig.java

```
package com.examly.springapp.configuration;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.client.RestTemplate;

@Configuration
public class AppConfig {
    @Bean
    public RestTemplate restTemplate() {
        return new RestTemplate();
    }
}
```

AssetTracking:



Create folders named controller, model, repository, and service inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp**.

Inside the controller folder, create classes named **"AssetController"** and **"OwnerController".**

Inside the model folder,

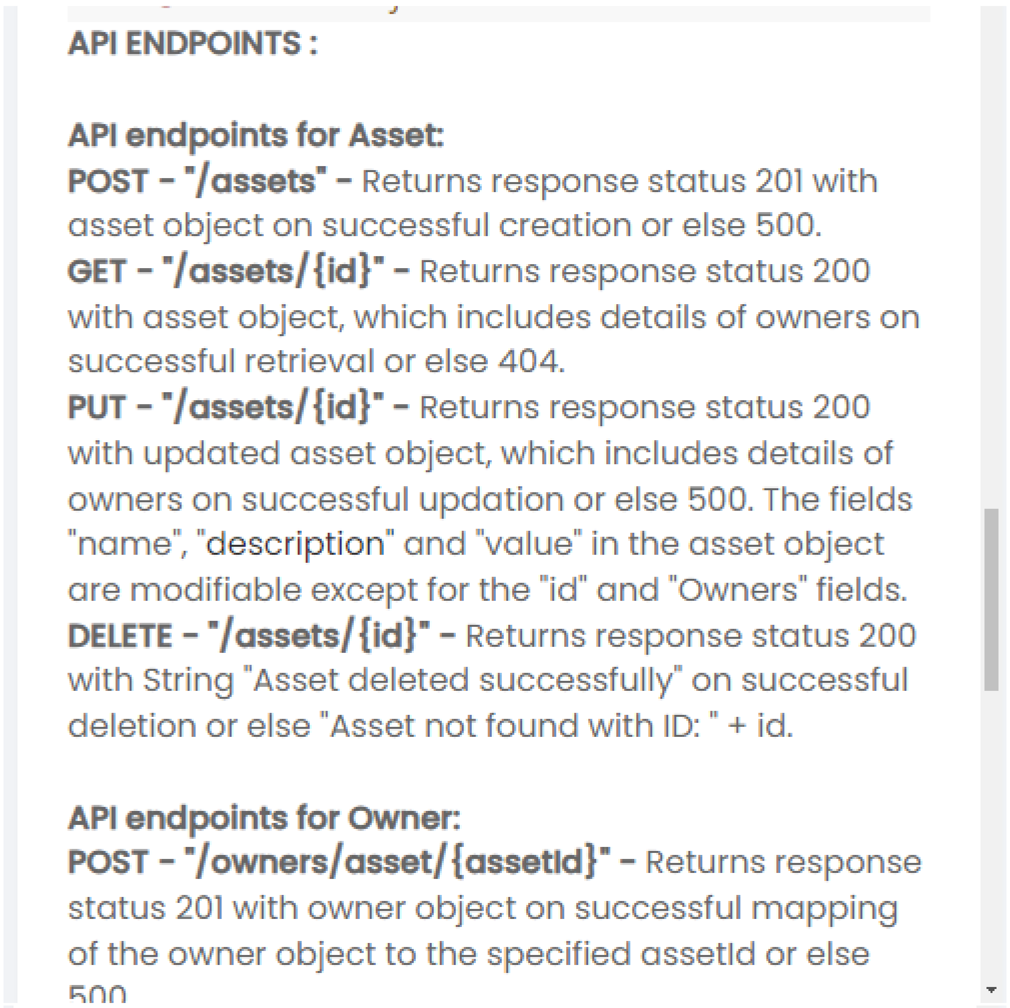
Create a class named **"Asset"** with the following attributes:

1. id - int (auto-generated primary key)
2. name - String
3. description - String
4. value - double
5. owner - Owner (OneToOne, mappedBy = "asset")

Create another class named **"Owner"** with the following attributes:

1. id - int (auto-generated primary key)
2. name - String
3. email - String
4. address - String
5. asset - Asset (OneToOne, JsonIgnore)

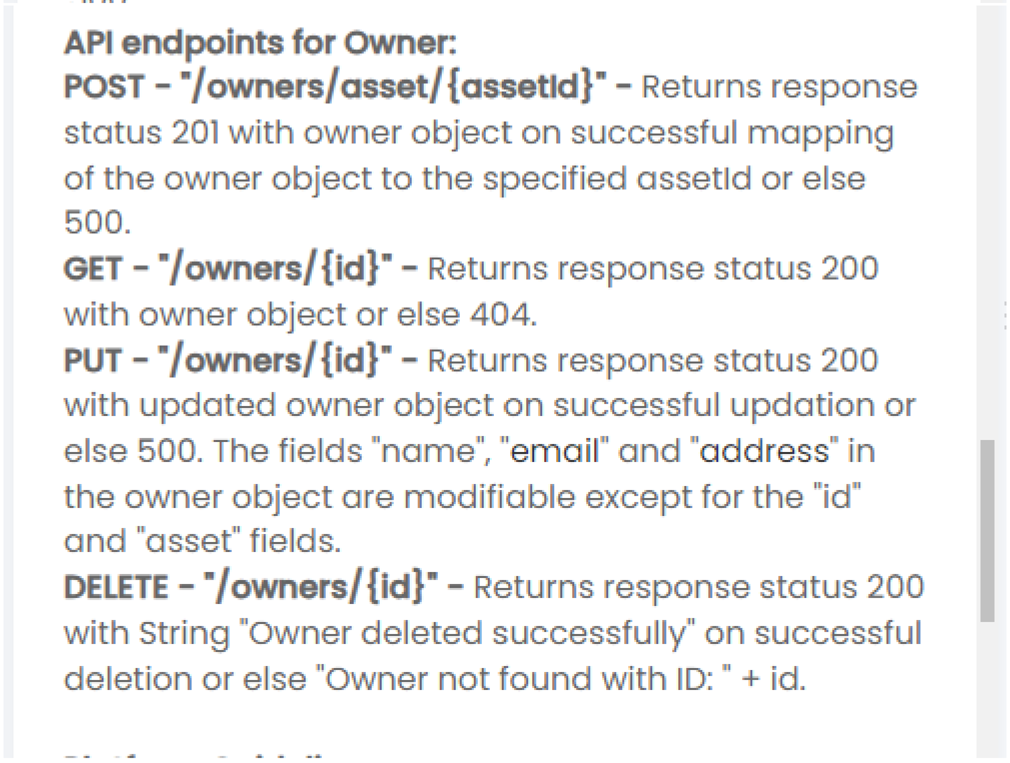
Implement getters, setters, and constructors for the



API ENDPOINTS :

API endpoints for Asset:
POST - "/assets" - Returns response status 201 with asset object on successful creation or else 500.
GET - "/assets/{id}" - Returns response status 200 with asset object, which includes details of owners on successful retrieval or else 404.
PUT - "/assets/{id}" - Returns response status 200 with updated asset object, which includes details of owners on successful updation or else 500. The fields "name", "description" and "value" in the asset object are modifiable except for the "id" and "Owners" fields.
DELETE - "/assets/{id}" - Returns response status 200 with String "Asset deleted successfully" on successful deletion or else "Asset not found with ID: " + id.

API endpoints for Owner:
POST - "/owners/asset/{assetId}" - Returns response status 201 with owner object on successful mapping of the owner object to the specified assetId or else 500



API endpoints for Owner:
POST - "/owners/asset/{assetId}" - Returns response status 201 with owner object on successful mapping of the owner object to the specified assetId or else 500.
GET - "/owners/{id}" - Returns response status 200 with owner object or else 404.
PUT - "/owners/{id}" - Returns response status 200 with updated owner object on successful updation or else 500. The fields "name", "email" and "address" in the owner object are modifiable except for the "id" and "asset" fields.
DELETE - "/owners/{id}" - Returns response status 200 with String "Owner deleted successfully" on successful deletion or else "Owner not found with ID: " + id.

Asset.java

```java
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.OneToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;

@Entity
public class Asset {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String description;
    private double value;

    @OneToOne(mappedBy = "asset")
    @JsonIgnore
    private Owner owner;

    public Asset() {}

    public Asset(String name, String description, double value) {
        this.name = name;
        this.description = description;
        this.value = value;
    }

    // Getters and setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    public double getValue() {
        return value;
    }

    public void setValue(double value) {
        this.value = value;
    }

    public Owner getOwner() {
        return owner;
    }

    public void setOwner(Owner owner) {
        this.owner = owner;
    }
}
```

Owner.java

```
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.OneToOne;

@Entity
public class Owner {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String email;
    private String address;

    @OneToOne
```

```java
@JoinColumn(name = "asset_id")
private Asset asset;

public Owner() {}

public Owner(String name, String email, String address) {
    this.name = name;
    this.email = email;
    this.address = address;
}

// Getters and setters
public int getId() {
    return id;
}

public void setId(int id) {
    this.id = id;
}

public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getEmail() {
    return email;
}

public void setEmail(String email) {
    this.email = email;
}

public String getAddress() {
    return address;
}

public void setAddress(String address) {
    this.address = address;
}

public Asset getAsset() {
    return asset;
}
```

```
    public void setAsset(Asset asset) {
        this.asset = asset;
    }
}
```

AssetRepo.java

```
package com.examly.springapp.repository;

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

import com.examly.springapp.model.Asset;

public interface AssetRepository extends JpaRepository<Asset,Integer>{

}
```

OwnerRepo.java

```
package com.examly.springapp.repository;

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

import com.examly.springapp.model.Owner;

public interface OwnerRepository extends JpaRepository<Owner,Integer>{

}
```

AssetService.java

```
package com.examly.springapp.service;

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

import com.examly.springapp.model.Asset;
import com.examly.springapp.repository.AssetRepository;

@Service
public class AssetService {
    private final AssetRepository assetRepository;

    @Autowired
    public AssetService(AssetRepository assetRepository) {
```

```java
        this.assetRepository = assetRepository;
    }

    public Asset saveAsset(Asset asset) {
        return assetRepository.save(asset);
    }

    public Asset getAssetById(int id) {
        return assetRepository.findById(id).orElse(null);
    }

    public void deleteAssetById(int id) {
        assetRepository.deleteById(id);
    }
}
```

OwnerService.java

```java
package com.examly.springapp.service;

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

import com.examly.springapp.model.Owner;
import com.examly.springapp.repository.OwnerRepository;

@Service
public class OwnerService {
    private final OwnerRepository ownerRepository;

    @Autowired
    public OwnerService(OwnerRepository ownerRepository) {
        this.ownerRepository = ownerRepository;
    }

    public Owner saveOwner(Owner owner) {
        return ownerRepository.save(owner);
    }

    public Owner getOwnerById(int id) {
        return ownerRepository.findById(id).orElse(null);
    }

    public void deleteOwnerById(int id) {
        ownerRepository.deleteById(id);
    }
}
```

AssetController.java

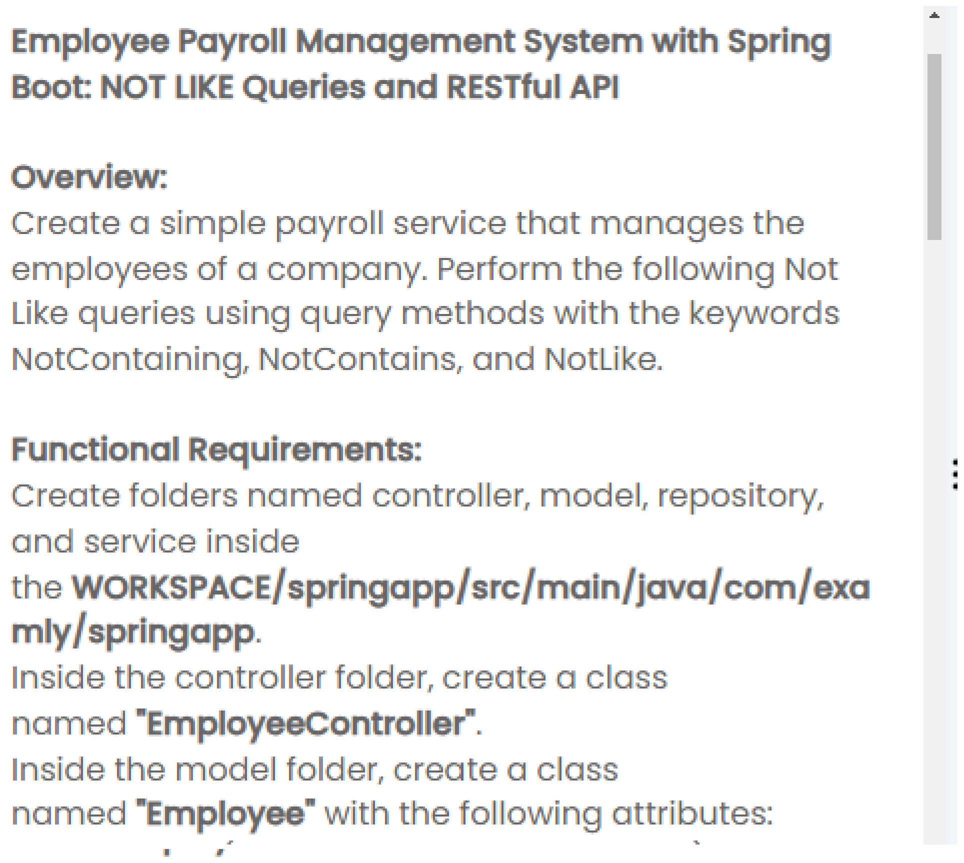
```java
package com.examly.springapp.controller;

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.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.examly.springapp.model.Asset;
import com.examly.springapp.service.AssetService;

@RestController
@RequestMapping("/assets")
public class AssetController {
    private final AssetService assetService;

    @Autowired
    public AssetController(AssetService assetService) {
        this.assetService = assetService;
    }

    @PostMapping
    public ResponseEntity<Asset> createAsset(@RequestBody Asset asset) {
        Asset createdAsset = assetService.saveAsset(asset);
        return new ResponseEntity<>(createdAsset, HttpStatus.CREATED);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Asset> getAssetById(@PathVariable int id) {
        Asset asset = assetService.getAssetById(id);
        if (asset == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<>(asset, HttpStatus.OK);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Asset> updateAsset(@PathVariable int id,
@RequestBody Asset asset) {
        Asset existingAsset = assetService.getAssetById(id);
        if (existingAsset == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }

        // Update fields
        existingAsset.setName(asset.getName());
        existingAsset.setDescription(asset.getDescription());
        existingAsset.setValue(asset.getValue());

        Asset updatedAsset = assetService.saveAsset(existingAsset);
        return new ResponseEntity<>(updatedAsset, HttpStatus.OK);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<String> deleteAsset(@PathVariable int id) {
        Asset asset = assetService.getAssetById(id);
        if (asset == null) {
            return new ResponseEntity<>("Asset not found with ID: " + id,
HttpStatus.NOT_FOUND);
        }
        assetService.deleteAssetById(id);
        return new ResponseEntity<>("Asset deleted successfully",
HttpStatus.OK);
    }
}
```

OwnerController.java

```
package com.examly.springapp.controller;

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.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.examly.springapp.model.Owner;
import com.examly.springapp.service.OwnerService;
```

```
@RestController
@RequestMapping("/owners")
public class OwnerController {
    private final OwnerService ownerService;

    @Autowired
    public OwnerController(OwnerService ownerService) {
        this.ownerService = ownerService;
    }

    @PostMapping("/asset/{assetid}")
    public ResponseEntity<Owner> mapOwnerToAsset(@PathVariable("assetid") int
assetId, @RequestBody Owner owner) {
        // You can implement the mapping logic here
        // Assuming the mapping logic is implemented elsewhere
        Owner mappedOwner = ownerService.saveOwner(owner);
        return new ResponseEntity<>(mappedOwner, HttpStatus.CREATED);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Owner> getOwnerById(@PathVariable int id) {
        Owner owner = ownerService.getOwnerById(id);
        if (owner == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<>(owner, HttpStatus.OK);
    }

    @PutMapping("/{id}")
    public ResponseEntity<Owner> updateOwner(@PathVariable int id,
@RequestBody Owner owner) {
        Owner existingOwner = ownerService.getOwnerById(id);
        if (existingOwner == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }

        // Update fields
        existingOwner.setName(owner.getName());
        existingOwner.setEmail(owner.getEmail());
        existingOwner.setAddress(owner.getAddress());

        Owner updatedOwner = ownerService.saveOwner(existingOwner);
        return new ResponseEntity<>(updatedOwner, HttpStatus.OK);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<String> deleteOwner(@PathVariable int id) {
        Owner owner = ownerService.getOwnerById(id);
        if (owner == null) {
            return new ResponseEntity<>("Owner not found with ID: " + id,
HttpStatus.NOT_FOUND);
        }
        ownerService.deleteOwnerById(id);
        return new ResponseEntity<>("Owner deleted successfully",
HttpStatus.OK);
    }
}
```

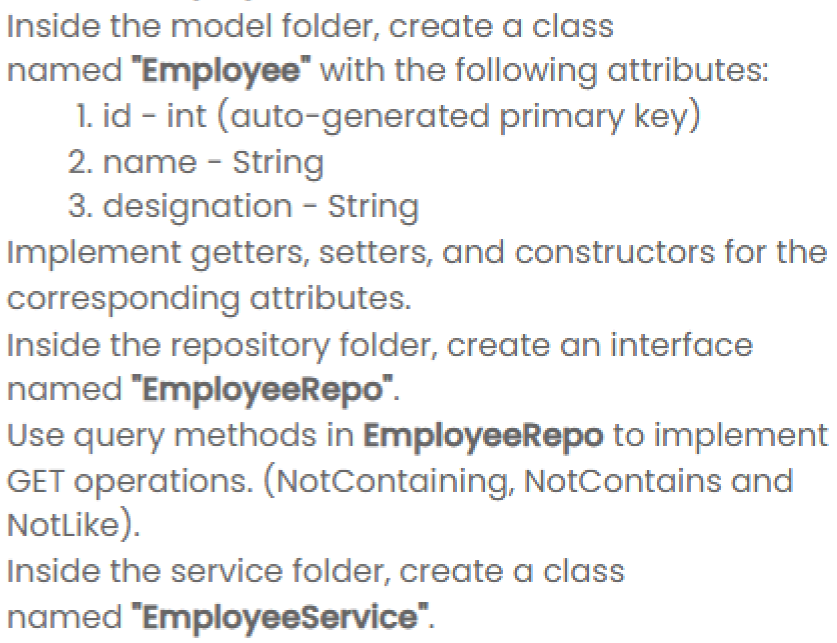
LAB_NOT_CONTAINS



Employee Payroll Management System with Spring Boot: NOT LIKE Queries and RESTful API

Overview:
Create a simple payroll service that manages the employees of a company. Perform the following Not Like queries using query methods with the keywords NotContaining, NotContains, and NotLike.

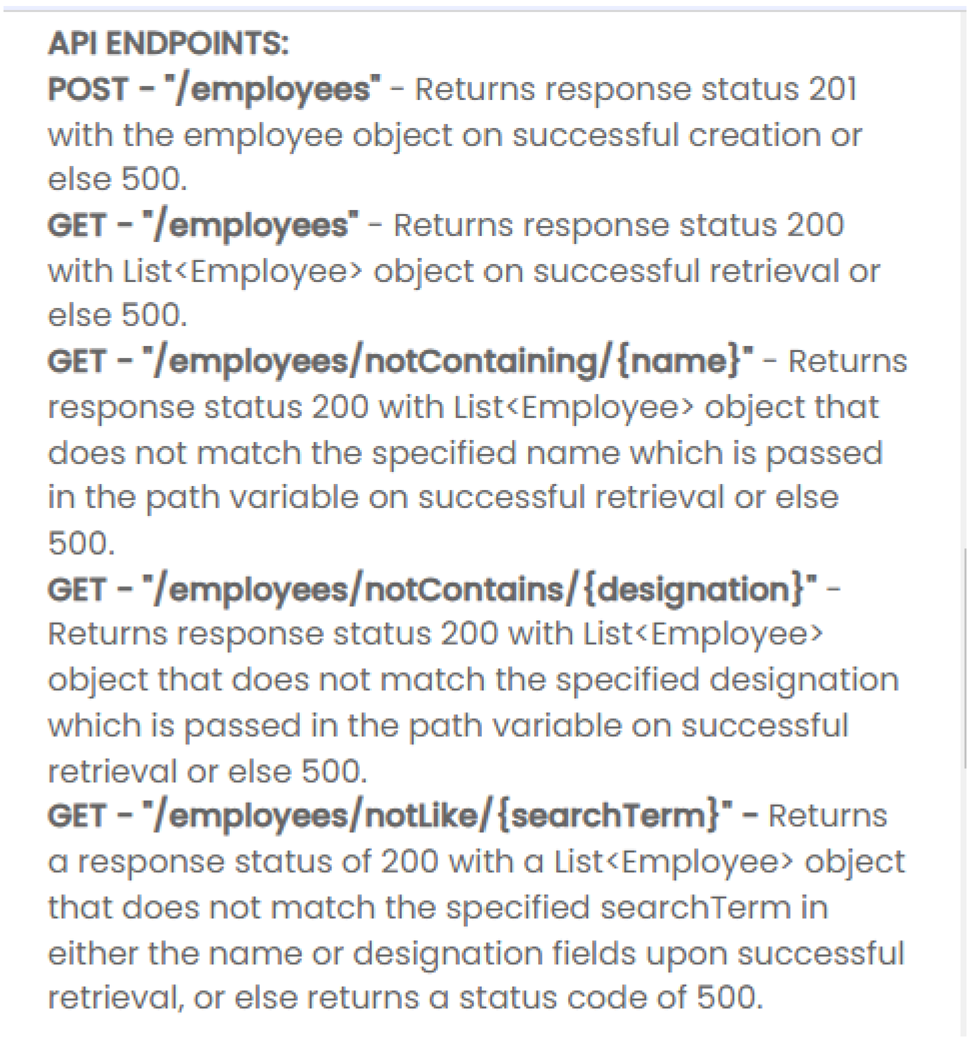
Functional Requirements:
Create folders named controller, model, repository, and service inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp**.
Inside the controller folder, create a class named **"EmployeeController"**.
Inside the model folder, create a class named **"Employee"** with the following attributes:



Inside the model folder, create a class named **"Employee"** with the following attributes:

1. id - int (auto-generated primary key)
2. name - String
3. designation - String

Implement getters, setters, and constructors for the corresponding attributes.
Inside the repository folder, create an interface named **"EmployeeRepo"**.
Use query methods in **EmployeeRepo** to implement GET operations. (NotContaining, NotContains and NotLike).
Inside the service folder, create a class named **"EmployeeService"**.


API ENDPOINTS:
POST - "/employees" - Returns response status 201 with the employee object on successful creation or else 500.
GET - "/employees" - Returns response status 200 with List<Employee> object on successful retrieval or else 500.
GET - "/employees/notContaining/{name}" - Returns response status 200 with List<Employee> object that does not match the specified name which is passed in the path variable on successful retrieval or else 500.
GET - "/employees/notContains/{designation}" - Returns response status 200 with List<Employee> object that does not match the specified designation which is passed in the path variable on successful retrieval or else 500.
GET - "/employees/notLike/{searchTerm}" - Returns a response status of 200 with a List<Employee> object that does not match the specified searchTerm in either the name or designation fields upon successful retrieval, or else returns a status code of 500.

Employee.java

```java
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String designation;

    public Employee() {
    }

    public Employee(int id, String name, String designation) {
        this.id = id;
        this.name = name;
        this.designation = designation;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDesignation() {
        return designation;
    }

    public void setDesignation(String designation) {
        this.designation = designation;
    }
}
```

EmployeeRepo.java

```java
package com.examly.springapp.repository;

import com.examly.springapp.model.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.stereotype.Repository;

import java.util.List;

@Repository
public interface EmployeeRepo extends JpaRepository<Employee, Integer> {
    List<Employee> findByNameNotContaining(String name);
    List<Employee> findByDesignationNotContaining(String designation);
    // List<Employee> findByNameNotLikeOrDesignationNotLike(String name,
String designation);
```

```
    //List<Employee> findByNameNotContainingOrDesignationNotContaining(String
name, String designation);
    @Query("SELECT e FROM Employee e WHERE e.name NOT LIKE %?1%")
    List<Employee> findByNameNotLike(String name);
}
```

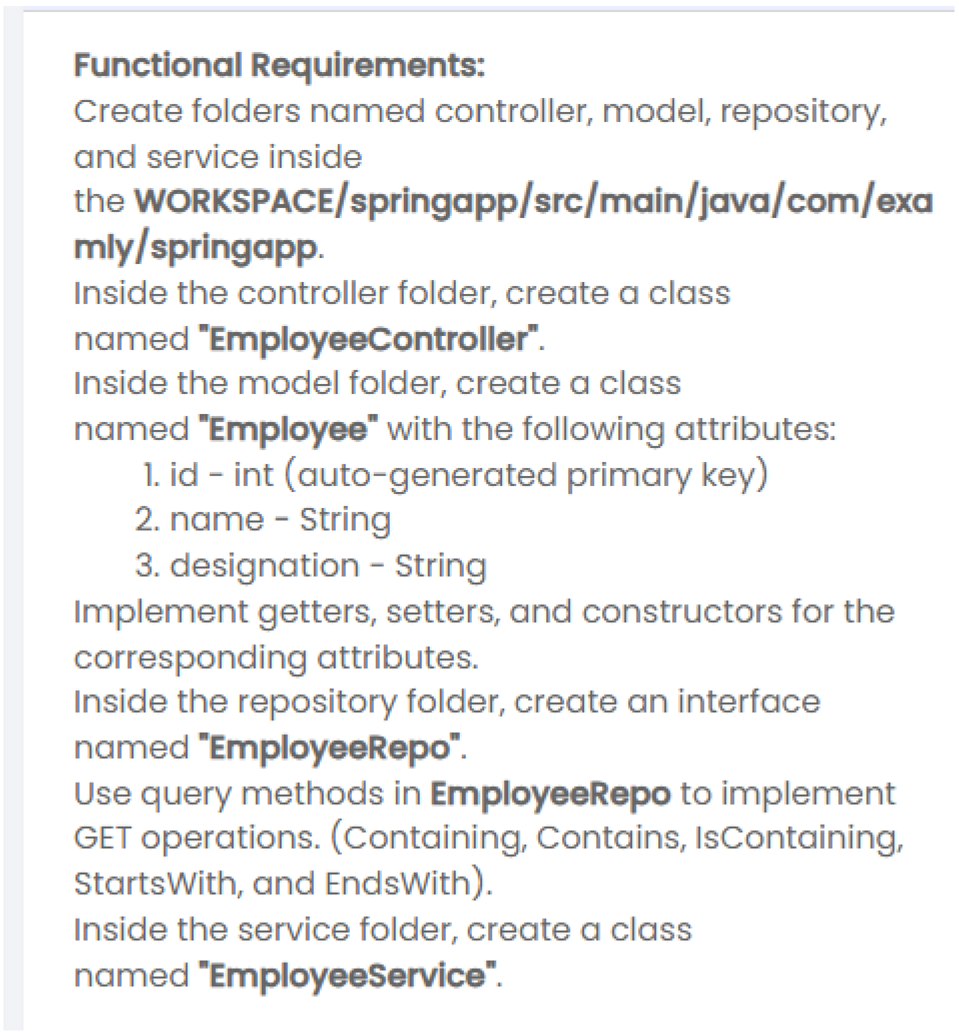
EmployeeService.java

```
package com.examly.springapp.service;

import java.util.List;

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

import com.examly.springapp.model.Employee;
import com.examly.springapp.repository.EmployeeRepo;

@Service
public class EmployeeService {
    private final EmployeeRepo employeeRepo;

    @Autowired
    public EmployeeService(EmployeeRepo employeeRepo) {
        this.employeeRepo = employeeRepo;
    }

    public List<Employee> getAllEmployees() {
        return employeeRepo.findAll();
    }

    public List<Employee> getEmployeesNotContainingName(String name) {
        return employeeRepo.findByNameNotContaining(name);
    }


    public List<Employee> getEmployeesNotContainingDesignation(String
designation) {
        return employeeRepo.findByDesignationNotContaining(designation);
    }

    public List<Employee> getEmployeesByNameNotLike(String name) {
        return employeeRepo.findByNameNotLike(name);
    }
    public Employee create(Employee emp) {
        return employeeRepo.save(emp);
    }
}
```

EmployeeController.java

```
package com.examly.springapp.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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;

import com.examly.springapp.model.Employee;
import com.examly.springapp.service.EmployeeService;

@RestController
@RequestMapping("/employees")
public class EmployeeController {
    @Autowired
    private  EmployeeService employeeService;

    @PostMapping
    public ResponseEntity<?>create(@RequestBody Employee emp)
    {
        try{
            return new
ResponseEntity<>(employeeService.create(emp),HttpStatus.CREATED);
        }catch(Exception e)
        {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
    }

    @GetMapping
    public ResponseEntity<List<Employee>> getAllEmployees() {
        List<Employee> employees = employeeService.getAllEmployees();
        return new ResponseEntity<>(employees, HttpStatus.OK);
    }

    @GetMapping("/notContaining/{name}")
    public ResponseEntity<List<Employee>>
getEmployeesNotContainingName(@PathVariable String name) {
        List<Employee> employees =
employeeService.getEmployeesNotContainingName(name);
        return new ResponseEntity<>(employees, HttpStatus.OK);
    }

    @GetMapping("/notContains/{designation}")
    public ResponseEntity<List<Employee>>
getEmployeesNotContainingDesignation(@PathVariable String designation) {
        List<Employee> employees =
employeeService.getEmployeesNotContainingDesignation(designation);
        return new ResponseEntity<>(employees, HttpStatus.OK);
    }

    @GetMapping("/notLike/{name}")
    public ResponseEntity<List<Employee>>
getEmployeesByNameNotLike(@PathVariable String name) {
        List<Employee> employees =
employeeService.getEmployeesByNameNotLike(name);
        return ResponseEntity.ok(employees);
    }
}
```

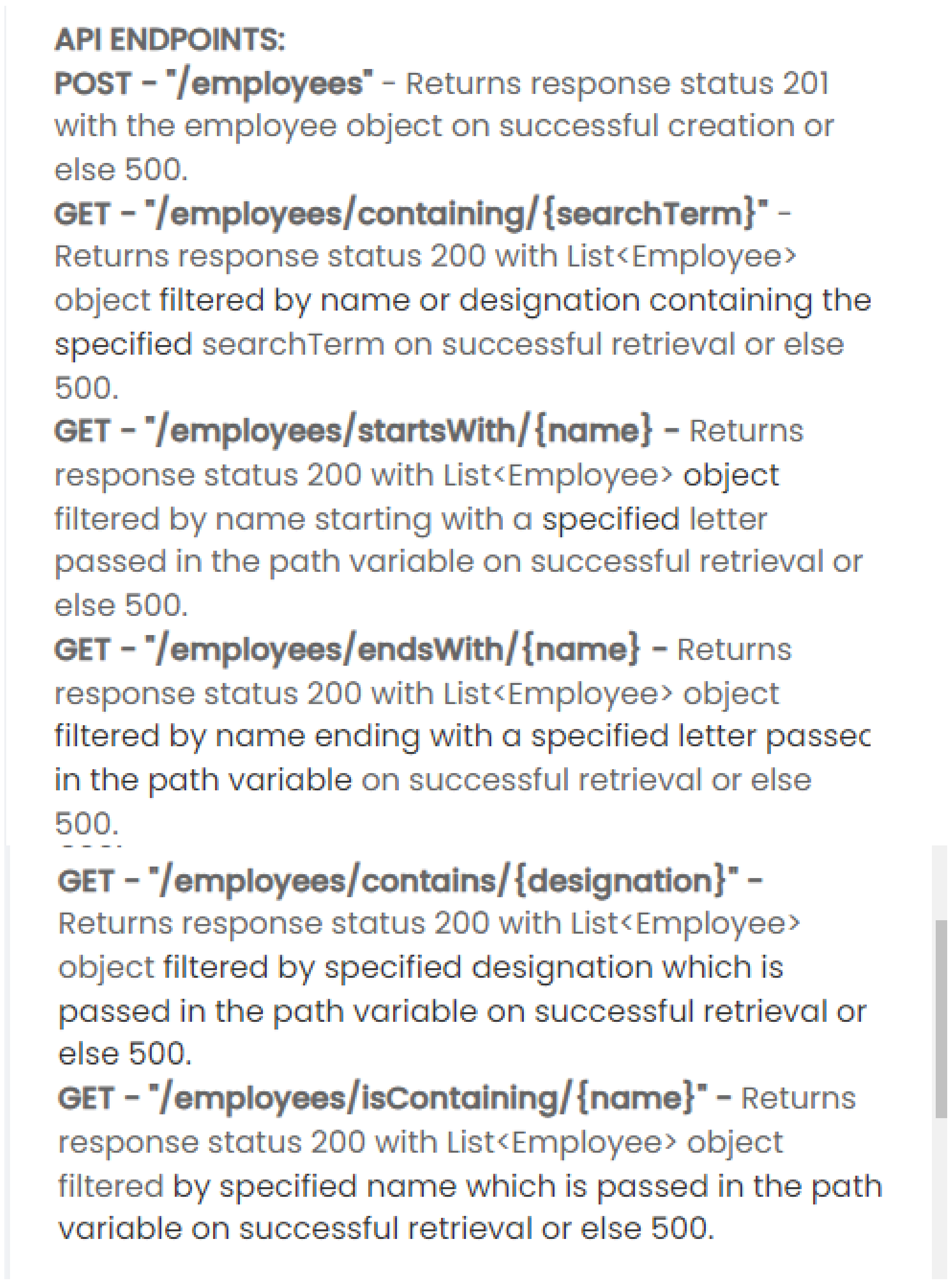
NOT_CONTAONING:



Functional Requirements:
Create folders named controller, model, repository, and service inside the **WORKSPACE/springapp/src/main/java/com/examly/springapp**.
Inside the controller folder, create a class named **"EmployeeController"**.
Inside the model folder, create a class named **"Employee"** with the following attributes:

1. id - int (auto-generated primary key)
2. name - String
3. designation - String

Implement getters, setters, and constructors for the corresponding attributes.
Inside the repository folder, create an interface named **"EmployeeRepo"**.
Use query methods in **EmployeeRepo** to implement GET operations. (Containing, Contains, IsContaining, StartsWith, and EndsWith).
Inside the service folder, create a class named **"EmployeeService"**.


API ENDPOINTS:
POST - "/employees" - Returns response status 201 with the employee object on successful creation or else 500.
GET - "/employees/containing/{searchTerm}" - Returns response status 200 with List<Employee> object filtered by name or designation containing the specified searchTerm on successful retrieval or else 500.
GET - "/employees/startsWith/{name} - Returns response status 200 with List<Employee> object filtered by name starting with a specified letter passed in the path variable on successful retrieval or else 500.
GET - "/employees/endsWith/{name} - Returns response status 200 with List<Employee> object filtered by name ending with a specified letter passec in the path variable on successful retrieval or else 500.
GET - "/employees/contains/{designation}" - Returns response status 200 with List<Employee> object filtered by specified designation which is passed in the path variable on successful retrieval or else 500.
GET - "/employees/isContaining/{name}" - Returns response status 200 with List<Employee> object filtered by specified name which is passed in the path variable on successful retrieval or else 500.

Employee.java
```java
package com.examly.springapp.model;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;

@Entity
public class Employee {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String name;
    private String designation;

    public Employee() {
    }

    public Employee(String name, String designation) {
        this.name = name;
        this.designation = designation;
    }

    // Getters and setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getDesignation() {
        return designation;
    }

    public void setDesignation(String designation) {
        this.designation = designation;
    }


}
```

EmployeeRepo.java

```
package com.examly.springapp.repository;

import java.util.List;

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

import com.examly.springapp.model.Employee;
```

```java
public interface EmployeeRepo extends JpaRepository<Employee,Integer>{
    List<Employee>
findByNameContainingIgnoreCaseOrDesignationContainingIgnoreCase(String name,
String designation);

    List<Employee> findByNameStartingWithIgnoreCase(String name);

    List<Employee> findByNameEndingWithIgnoreCase(String name);

    List<Employee> findByDesignationContainingIgnoreCase(String designation);

    List<Employee> findByNameIsContainingIgnoreCase(String name);

}
```

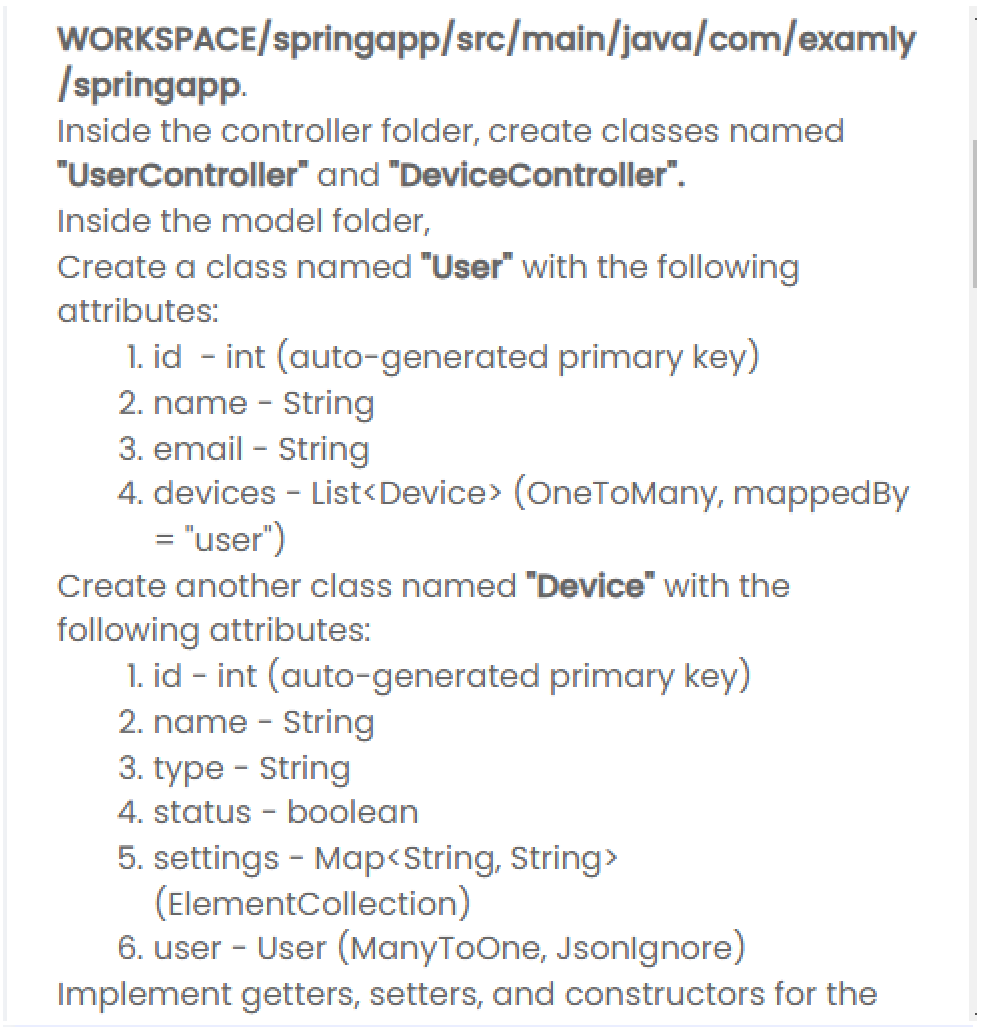
EmployeeService.java

```java
package com.examly.springapp.service;

import java.util.List;

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

import com.examly.springapp.model.Employee;
import com.examly.springapp.repository.EmployeeRepo;

@Service
public class EmployeeService {
    @Autowired
    private EmployeeRepo employeeRepo;

    public Employee saveEmployee(Employee employee) {
        return employeeRepo.save(employee);
    }

    public List<Employee> getEmployeesContaining(String searchTerm) {
        return
employeeRepo.findByNameContainingIgnoreCaseOrDesignationContainingIgnoreCase(s
earchTerm, searchTerm);
    }

    public List<Employee> getEmployeesStartingWith(String name) {
        return employeeRepo.findByNameStartingWithIgnoreCase(name);
    }

    public List<Employee> getEmployeesEndingWith(String name) {
        return employeeRepo.findByNameEndingWithIgnoreCase(name);
    }

    public List<Employee> getEmployeesContains(String designation) {
        return
employeeRepo.findByDesignationContainingIgnoreCase(designation);
    }

    public List<Employee> getEmployeesIsContaining(String name) {
        return employeeRepo.findByNameIsContainingIgnoreCase(name);
    }
}
```

EmployeeController.java

```
package com.examly.springapp.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
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;

import com.examly.springapp.model.Employee;
import com.examly.springapp.service.EmployeeService;

@RestController
@RequestMapping("/employees")
public class EmployeeController {
    @Autowired
    private EmployeeService employeeService;

    @PostMapping
    public ResponseEntity<Employee> addEmployee(@RequestBody Employee
employee) {
        return new ResponseEntity<>(employeeService.saveEmployee(employee),
HttpStatus.CREATED);
    }

    @GetMapping("/containing/{searchTerm}")
    public ResponseEntity<List<Employee>> getEmployeesContaining(@PathVariable
String searchTerm) {
        return new
ResponseEntity<>(employeeService.getEmployeesContaining(searchTerm),
HttpStatus.OK);
    }

    @GetMapping("/startsWith/{name}")
    public ResponseEntity<List<Employee>>
getEmployeesStartingWith(@PathVariable String name) {
        return new
ResponseEntity<>(employeeService.getEmployeesStartingWith(name),
HttpStatus.OK);
    }

    @GetMapping("/endsWith/{name}")
    public ResponseEntity<List<Employee>> getEmployeesEndingWith(@PathVariable
String name) {
        return new
ResponseEntity<>(employeeService.getEmployeesEndingWith(name), HttpStatus.OK);
    }

    @GetMapping("/contains/{designation}")
    public ResponseEntity<List<Employee>> getEmployeesContains(@PathVariable
String designation) {
        return new
ResponseEntity<>(employeeService.getEmployeesContains(designation),
HttpStatus.OK);
    }

    @GetMapping("/isContaining/{name}")
    public ResponseEntity<List<Employee>>
getEmployeesIsContaining(@PathVariable String name) {
        return new
ResponseEntity<>(employeeService.getEmployeesIsContaining(name),
HttpStatus.OK);
    }
}
```

SMARTHOME:



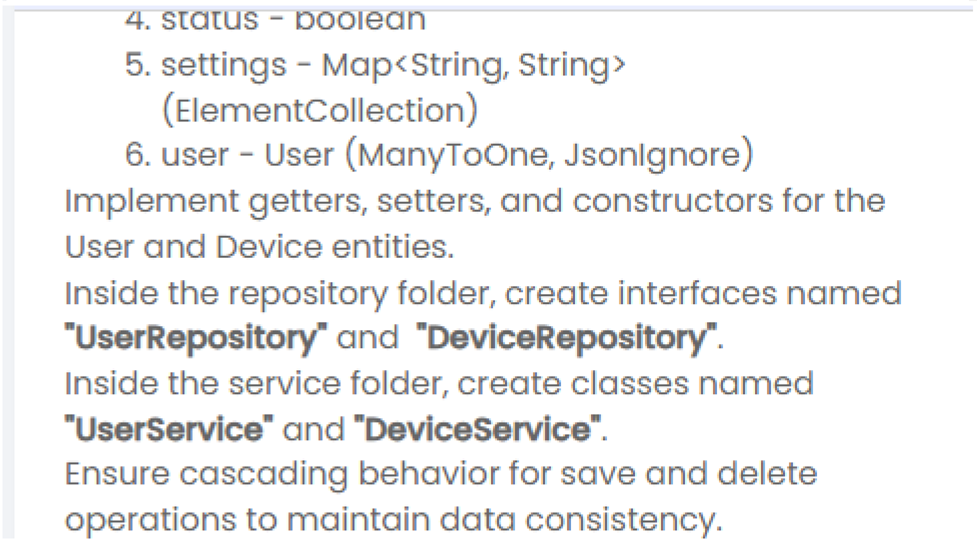
WORKSPACE/springapp/src/main/java/com/examly/springapp.
Inside the controller folder, create classes named **"UserController"** and **"DeviceController".**
Inside the model folder,
Create a class named **"User"** with the following attributes:

1. id - int (auto-generated primary key)
2. name - String
3. email - String
4. devices - List<Device> (OneToMany, mappedBy = "user")

Create another class named **"Device"** with the following attributes:

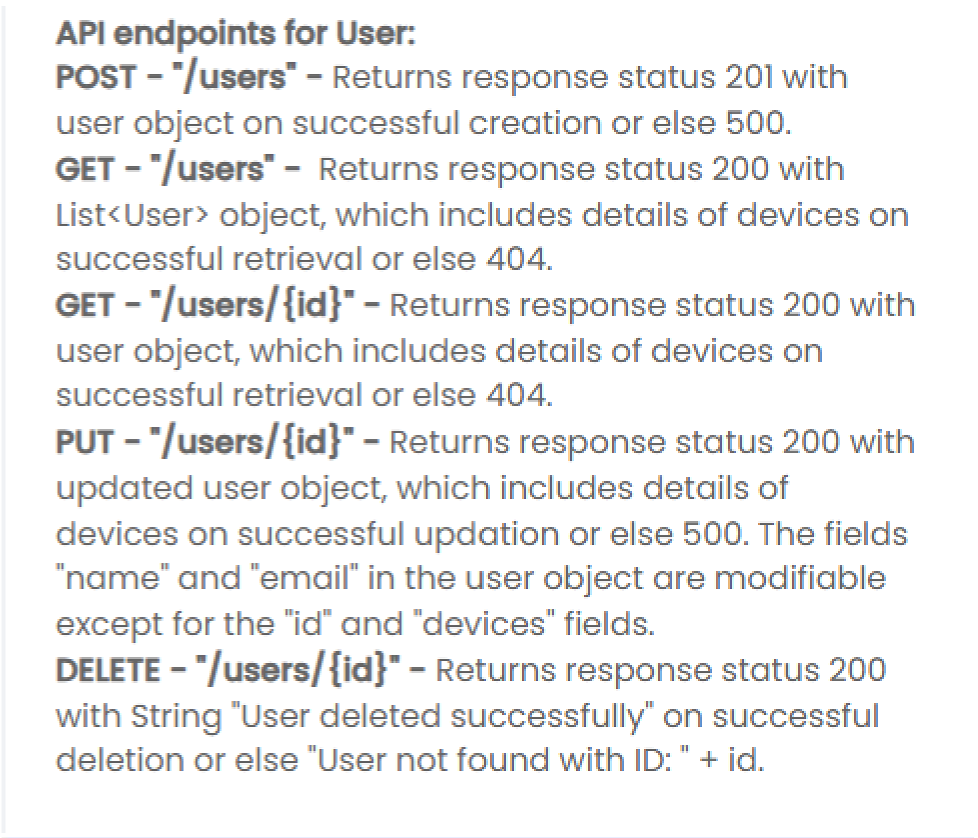
1. id - int (auto-generated primary key)
2. name - String
3. type - String
4. status - boolean
5. settings - Map<String, String> (ElementCollection)
6. user - User (ManyToOne, JsonIgnore)

Implement getters, setters, and constructors for the

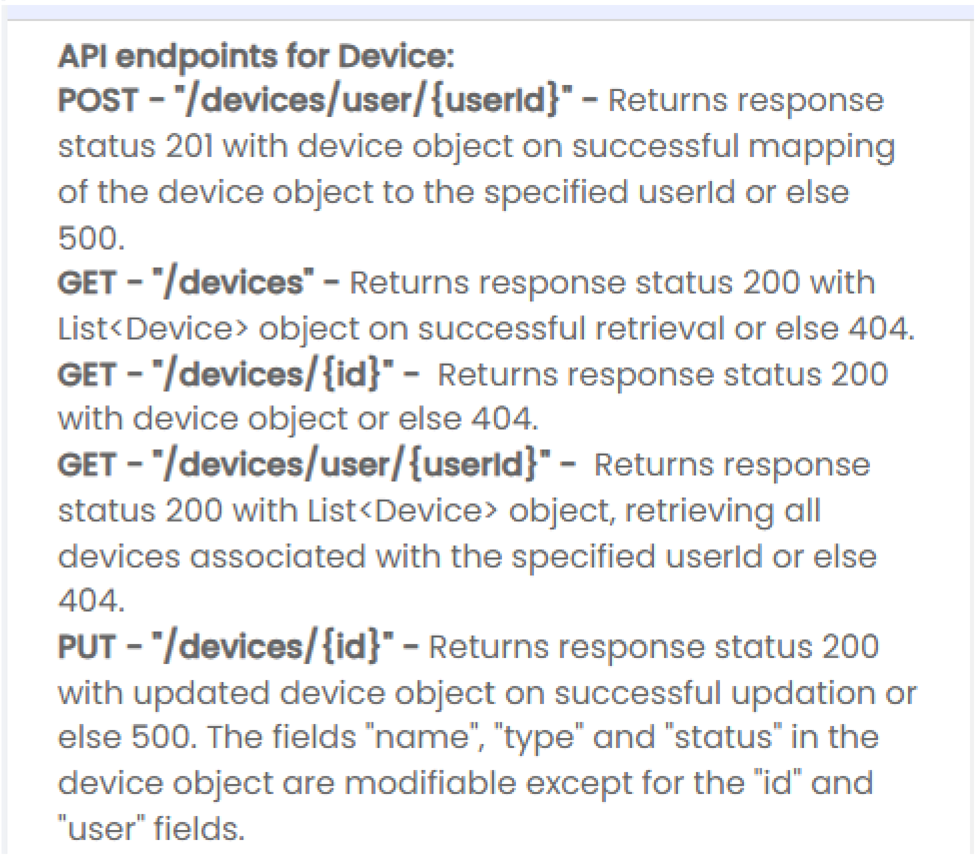


4. status - boolean
5. settings - Map<String, String> (ElementCollection)
6. user - User (ManyToOne, JsonIgnore)

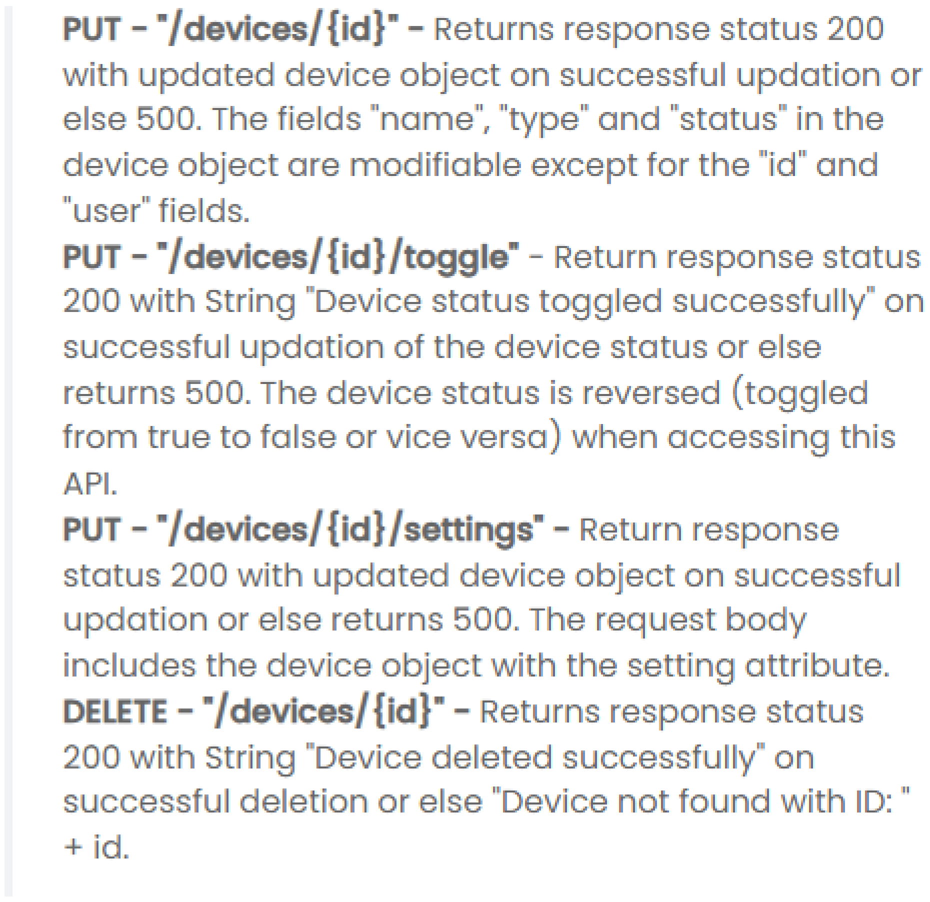
Implement getters, setters, and constructors for the
User and Device entities.
Inside the repository folder, create interfaces named **"UserRepository"** and **"DeviceRepository"**.
Inside the service folder, create classes named **"UserService"** and **"DeviceService"**.
Ensure cascading behavior for save and delete operations to maintain data consistency.



API endpoints for User:
POST - "/users" - Returns response status 201 with user object on successful creation or else 500.
GET - "/users" - Returns response status 200 with List<User> object, which includes details of devices on successful retrieval or else 404.
GET - "/users/{id}" - Returns response status 200 with user object, which includes details of devices on successful retrieval or else 404.
PUT - "/users/{id}" - Returns response status 200 with updated user object, which includes details of devices on successful updation or else 500. The fields "name" and "email" in the user object are modifiable except for the "id" and "devices" fields.
DELETE - "/users/{id}" - Returns response status 200 with String "User deleted successfully" on successful deletion or else "User not found with ID: " + id.



API endpoints for Device:
POST - "/devices/user/{userId}" - Returns response status 201 with device object on successful mapping of the device object to the specified userId or else 500.
GET - "/devices" - Returns response status 200 with List<Device> object on successful retrieval or else 404.
GET - "/devices/{id}" - Returns response status 200 with device object or else 404.
GET - "/devices/user/{userId}" - Returns response status 200 with List<Device> object, retrieving all devices associated with the specified userId or else 404.
PUT - "/devices/{id}" - Returns response status 200 with updated device object on successful updation or else 500. The fields "name", "type" and "status" in the device object are modifiable except for the "id" and "user" fields.



PUT - "/devices/{id}" - Returns response status 200 with updated device object on successful updation or else 500. The fields "name", "type" and "status" in the device object are modifiable except for the "id" and "user" fields.
PUT - "/devices/{id}/toggle" - Return response status 200 with String "Device status toggled successfully" on successful updation of the device status or else returns 500. The device status is reversed (toggled from true to false or vice versa) when accessing this API.
PUT - "/devices/{id}/settings" - Return response status 200 with updated device object on successful updation or else returns 500. The request body includes the device object with the setting attribute.
DELETE - "/devices/{id}" - Returns response status 200 with String "Device deleted successfully" on successful deletion or else "Device not found with ID: " + id.

Device.java

```java
package com.examly.springapp.model;

import javax.persistence.ElementCollection;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.ManyToOne;

import com.fasterxml.jackson.annotation.JsonIgnore;

import java.util.Map;

@Entity
public class Device {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private String type;
    private boolean status;

    @ElementCollection
    private Map<String, String> settings;

    @ManyToOne
    @JsonIgnore
    private User user;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getType() {
        return type;
    }

    public void setType(String type) {
        this.type = type;
    }

    public boolean isStatus() {
        return status;
    }

    public void setStatus(boolean status) {
        this.status = status;
    }

    public Map<String, String> getSettings() {
        return settings;
    }

    public void setSettings(Map<String, String> settings) {
        this.settings = settings;
    }
```

```
    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }



}
```

User.java

```
package com.examly.springapp.model;

import com.fasterxml.jackson.annotation.JsonIgnore;

import javax.persistence.*;
import java.util.List;

@Entity
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    private String email;

    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private List<Device> devices;

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public List<Device> getDevices() {
        return devices;
    }

    public void setDevices(List<Device> devices) {
        this.devices = devices;
    }


}
```

DeviceRepo.java

```
package com.examly.springapp.repository;

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

import com.examly.springapp.model.Device;

public interface DeviceRepository extends JpaRepository<Device,Integer>{

}
```

UserRepo.java

```
package com.examly.springapp.repository;

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

import com.examly.springapp.model.User;

public interface UserRepository extends JpaRepository<User,Integer>{

}
```

DeviceService.java

```java
package com.examly.springapp.service;

import com.examly.springapp.model.Device;
import com.examly.springapp.model.User;
import com.examly.springapp.repository.DeviceRepository;
import com.examly.springapp.repository.UserRepository;

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

import java.util.Collections;
import java.util.List;
import java.util.Optional;

@Service
public class DeviceService {

    @Autowired
    private DeviceRepository deviceRepository;

    @Autowired
    private UserRepository userRepository;

    public Device createDevice(int userId, Device device) {

        User obj = userRepository.findById(userId).orElse(null);

        if(obj!=null)
        {
            device.setUser(obj);
            obj.getDevices().add(device);
            userRepository.save(obj);
            deviceRepository.save(device);
            return device;
        }
        return null;
    }

    public List<Device> getAllDevices() {
        return deviceRepository.findAll();
    }

    public Device getDeviceById(int id) {

        return deviceRepository.findById(id).orElse(null);
    }
    public List<Device> getDevicesByUserId(int userId) {
        // Retrieve devices by user ID
        Optional<User> optionalUser = userRepository.findById(userId);
        if (optionalUser.isPresent()) {
            User user = optionalUser.get();
            return user.getDevices();
        }
        return Collections.emptyList(); // Return empty list if user not found
or has no devices
    }


    public Device updateDevice(int id, Device updatedDevice) {
        Device device = deviceRepository.findById(id).orElse(null);
        if (device != null) {
            device.setName(updatedDevice.getName());
            device.setType(updatedDevice.getType());
            device.setStatus(updatedDevice.isStatus());
            device.setSettings(updatedDevice.getSettings());
            return deviceRepository.save(device);
        }
        return null;
    }

    public boolean toggleDeviceStatus(int id) {
        Device device = deviceRepository.findById(id).orElse(null);
        if (device != null) {
            device.setStatus(!device.isStatus());
            deviceRepository.save(device);
            return true;
        }
        return false;
    }

    public Device updateDeviceSettings(int id, Device updatedDevice) {
        Device device = deviceRepository.findById(id).orElse(null);
        if (device != null) {
            device.setSettings(updatedDevice.getSettings());
            return deviceRepository.save(device);
        }
        return null;
    }

    public boolean deleteDevice(int id) {
        if (deviceRepository.existsById(id)) {
            deviceRepository.deleteById(id);
            return true;
        }
```

```java
        return false;
    }
}
```

UserService.java

```java
package com.examly.springapp.service;

import com.examly.springapp.model.User;
import com.examly.springapp.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserService {

    @Autowired
    private UserRepository userRepository;

    public User createUser(User user) {
        return userRepository.save(user);
    }

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    public User getUserById(int id) {
        return userRepository.findById(id).orElse(null);
    }

    public User updateUser(int id, User updatedUser) {
        User user = userRepository.findById(id).orElse(null);
        if (user != null) {
            user.setName(updatedUser.getName());
            user.setEmail(updatedUser.getEmail());
            return userRepository.save(user);
        }
        return null;
    }

    public boolean deleteUser(int id) {
        if (userRepository.existsById(id)) {
            userRepository.deleteById(id);
            return true;
        }
        return false;
    }
}
```

DeviceController.java

```
package com.examly.springapp.controller;

import com.examly.springapp.model.Device;
import com.examly.springapp.service.DeviceService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class DeviceController {

    @Autowired
    private DeviceService deviceService;

    @PostMapping("/devices/user/{userId}")
    public ResponseEntity<Device> createDevice(@PathVariable int userId,
@RequestBody Device device) {
        Device newDevice = deviceService.createDevice(userId, device);
        return new ResponseEntity<>(newDevice, HttpStatus.CREATED);
    }

    @GetMapping("/devices")
    public ResponseEntity<List<Device>> getAllDevices() {
        List<Device> devices = deviceService.getAllDevices();
        return new ResponseEntity<>(devices, HttpStatus.OK);
    }

    @GetMapping("/devices/{id}")
    public ResponseEntity<Device> getDeviceById(@PathVariable int id) {
        Device device = deviceService.getDeviceById(id);
        if (device == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<>(device, HttpStatus.OK);
    }

    @GetMapping("/devices/user/{userId}")
    public ResponseEntity<List<Device>> getDevicesByUserId(@PathVariable int
userId) {
        List<Device> devices = deviceService.getDevicesByUserId(userId);
        return new ResponseEntity<>(devices, HttpStatus.OK);
    }

    @PutMapping("/devices/{id}")
    public ResponseEntity<Device> updateDevice(@PathVariable int id,
@RequestBody Device updatedDevice) {
        Device device = deviceService.updateDevice(id, updatedDevice);
        if (device == null) {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
        return new ResponseEntity<>(device, HttpStatus.OK);
    }

    @PutMapping("/devices/{id}/toggle")
    public ResponseEntity<String> toggleDeviceStatus(@PathVariable int id) {
        boolean toggled = deviceService.toggleDeviceStatus(id);
        if (toggled) {
            return new ResponseEntity<>("Device status toggled successfully",
HttpStatus.OK);
        }
        return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @PutMapping("/devices/{id}/settings")
    public ResponseEntity<Device> updateDeviceSettings(@PathVariable int id,
@RequestBody Device updatedDevice) {
        Device device = deviceService.updateDeviceSettings(id, updatedDevice);
        if (device == null) {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
        return new ResponseEntity<>(device, HttpStatus.OK);
    }

    @DeleteMapping("/devices/{id}")
    public ResponseEntity<String> deleteDevice(@PathVariable int id) {

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

    }
}
```

UserController.java

```java
package com.examly.springapp.controller;

import com.examly.springapp.model.User;
import com.examly.springapp.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
public class UserController {

    @Autowired
    private UserService userService;

    @PostMapping("/users")
    public ResponseEntity<User> createUser(@RequestBody User user) {
        User newUser = userService.createUser(user);
        return new ResponseEntity<>(newUser, HttpStatus.CREATED);
    }

    @GetMapping("/users")
    public ResponseEntity<List<User>> getAllUsers() {
        List<User> users = userService.getAllUsers();
        return new ResponseEntity<>(users, HttpStatus.OK);
    }

    @GetMapping("/users/{id}")
    public ResponseEntity<User> getUserById(@PathVariable int id) {
        User user = userService.getUserById(id);
        if (user == null) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
        return new ResponseEntity<>(user, HttpStatus.OK);
    }

    @PutMapping("/users/{id}")
    public ResponseEntity<User> updateUser(@PathVariable int id, @RequestBody 
User updatedUser) {
        User user = userService.updateUser(id, updatedUser);
        if (user == null) {
            return new ResponseEntity<>(HttpStatus.INTERNAL_SERVER_ERROR);
        }
        return new ResponseEntity<>(user, HttpStatus.OK);
    }

    @DeleteMapping("/users/{id}")
    public ResponseEntity<String> deleteUser(@PathVariable int id) {
        boolean deleted = userService.deleteUser(id);
        if (deleted) {
            return new ResponseEntity<>("User deleted successfully",
HttpStatus.OK);
        }
        return new ResponseEntity<>("User not found with ID: " + id,
HttpStatus.NOT_FOUND);
    }
}
```

LIBRARY MANAGEMENT SYSTEM:

Day 11a daily challenge

SKG_LIBRARY_MANAGEMENT:

Library Management:

Member Controller:

```
// MemberController.java

package com.examly.springapp.controller;


import com.examly.springapp.model.Member;

import com.examly.springapp.service.MemberService;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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


import java.util.List;
```

```
@RestController
@RequestMapping("/members")
public class MemberController {
  @Autowired
  private MemberService memberService;

  @PostMapping
  public ResponseEntity<Member> addMember(@RequestBody Member member) {
    Member newMember = memberService.addMember(member);
    return new ResponseEntity<>(newMember, HttpStatus.CREATED);
  }

  @GetMapping("/{id}")
  public ResponseEntity<Member> getMemberById(@PathVariable("id") int id) {
    Member member = memberService.getMemberById(id);
    return new ResponseEntity<>(member, HttpStatus.OK);
  }

  @GetMapping
  public ResponseEntity<List<Member>> getAllMembers() {
    List<Member> members = memberService.getAllMembers();
    return new ResponseEntity<>(members, HttpStatus.OK);
  }

  @PutMapping("/{id}")
  public ResponseEntity<Member> updateMember(@PathVariable("id") int id, @RequestBody
Member member) {
    if (member.getId() != id) {
      return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
    }
    Member updatedMember = memberService.updateMember(member);
```

```
        return new ResponseEntity<>(updatedMember, HttpStatus.OK);
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<String> deleteMember(@PathVariable("id") int id) {
        memberService.deleteMember(id);
        return new ResponseEntity<>("The Member Deleted Successfully", HttpStatus.OK);
    }
}
```

MembershipCard Controller:

```
// MembershipCardController.java
package com.examly.springapp.controller;

import com.examly.springapp.model.MembershipCard;
import com.examly.springapp.service.MembershipService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/membership-cards")
public class MembershipCardController {
    @Autowired
    private MembershipService membershipService;

    @PostMapping("/member/{id}")
```

```
   public ResponseEntity<MembershipCard> addMembership(@PathVariable("id") int memberId,
@RequestBody MembershipCard membershipCard) {
      MembershipCard newMembership = membershipService.addMembership(membershipCard);
      return new ResponseEntity<>(newMembership, HttpStatus.CREATED);
   }

   @GetMapping("/{id}")
   public ResponseEntity<MembershipCard> getMembershipById(@PathVariable("id") int id) {
      MembershipCard membershipCard = membershipService.getMembershipById(id);
      return new ResponseEntity<>(membershipCard, HttpStatus.OK);
   }

   @PutMapping("/{id}")
   public ResponseEntity<MembershipCard> updateMembership(@PathVariable("id") int id,
@RequestBody MembershipCard membershipCard) {
      if (membershipCard.getId() != id) {
         return new ResponseEntity<>(HttpStatus.BAD_REQUEST);
      }
      MembershipCard updatedMembership =
membershipService.updateMembership(membershipCard);
      return new ResponseEntity<>(updatedMembership, HttpStatus.OK);
   }

   @DeleteMapping("/{id}")
   public ResponseEntity<String> deleteMembership(@PathVariable("id") int id) {
      membershipService.deleteMembership(id);
      return new ResponseEntity<>("Membership Card Deleted Successfully", HttpStatus.OK);
   }
}
```

Member model:

```java
// Member.java
package com.examly.springapp.model;

import com.fasterxml.jackson.annotation.JsonIgnore;
import javax.persistence.*;

@Entity
public class Member {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private int id;
   private String name;
   private String email;

   @OneToOne(mappedBy = "member")
   @JsonIgnore
   private MembershipCard membershipCard;

   // Constructors
   public Member() {}

   public Member(String name, String email) {
      this.name = name;
      this.email = email;
   }

   // Getters and setters
   // Id
   public int getId() {
      return id;
```

```
    }

    // Name
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    // Email
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    // MembershipCard
    public MembershipCard getMembershipCard() {
        return membershipCard;
    }

    public void setMembershipCard(MembershipCard membershipCard) {
        this.membershipCard = membershipCard;
    }
}
```

Membershipcard model:

```
// MembershipCard.java
package com.examly.springapp.model;

import javax.persistence.*;
import java.time.LocalDate;

@Entity
public class MembershipCard {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    private String cardNumber;
    private LocalDate expirationDate;

    @OneToOne
    @JoinColumn(name = "member_id")
    private Member member;

    // Constructors
    public MembershipCard() {}

    public MembershipCard(String cardNumber, LocalDate expirationDate) {
        this.cardNumber = cardNumber;
        this.expirationDate = expirationDate;
    }

    // Getters and setters
    // Id
    public int getId() {
        return id;
    }

    // CardNumber
    public String getCardNumber() {
        return cardNumber;
    }

    public void setCardNumber(String cardNumber) {
        this.cardNumber = cardNumber;
    }

    // ExpirationDate
    public LocalDate getExpirationDate() {
        return expirationDate;
    }

    public void setExpirationDate(LocalDate expirationDate) {
        this.expirationDate = expirationDate;
    }

    // Member
    public Member getMember() {
        return member;
    }

    public void setMember(Member member) {
        this.member = member;
    }
}
```

Member Repo:

```
// MemberRepo.java
package com.examly.springapp.repository;

import com.examly.springapp.model.Member;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface MemberRepo extends JpaRepository<Member, Integer> {
}
```

MembershipCard Repo:

```
// MembershipRepo.java
package com.examly.springapp.repository;

import com.examly.springapp.model.MembershipCard;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

@Repository
public interface MembershipRepo extends JpaRepository<MembershipCard, Integer> {
}
```

Member Service:

```
// MemberService.java
package com.examly.springapp.service;

import com.examly.springapp.model.Member;
import com.examly.springapp.repository.MemberRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;

@Service
public class MemberService {
   @Autowired
   private MemberRepo memberRepo;

   public Member addMember(Member member) {
      return memberRepo.save(member);
   }

   public Member getMemberById(int id) {
      return memberRepo.findById(id).orElse(null);
   }

   public List<Member> getAllMembers() {
      return memberRepo.findAll();
   }

   public Member updateMember(Member member) {
      return memberRepo.save(member);
   }
```

```
    public void deleteMember(int id) {
        memberRepo.deleteById(id);
    }
}
```

MembershipCard Service:

```
// MembershipService.java
package com.examly.springapp.service;

import com.examly.springapp.model.MembershipCard;
import com.examly.springapp.repository.MembershipRepo;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.Optional;

@Service
public class MembershipService {
    @Autowired
    private MembershipRepo membershipRepo;

    public MembershipCard addMembership(MembershipCard membershipCard) {
        return membershipRepo.save(membershipCard);
    }

    public MembershipCard getMembershipById(int id) {
        Optional<MembershipCard> optionalMembershipCard = membershipRepo.findById(id);
        return optionalMembershipCard.orElse(null);
    }
    public MembershipCard updateMembership(MembershipCard membershipCard) {
        return membershipRepo.save(membershipCard);
    }

    public void deleteMembership(int id) {
        membershipRepo.deleteById(id);
    }
}
```

FINANCE:

SKG_SPRINGBOOT_FINANCE

Personal Finance Management System

Overview:

Develop a Spring Boot REST API for managing personal finances, allowing users to track their income, expenses, and budget goals. Implement a one-to-many bidirectional relationship between users and transactions, enabling users to view their transaction history and financial summaries.

Functional Requirements:

Create folders named controller, model, repository and service inside the WORKSPACE/springapp/src/main/java/com/examly/springapp.

Inside the controller folder, create classes named "UserController” and "TransactionController".

Inside the model folder,

Create a class named "User" with the following attributes:

id - int (auto-generated primary key)

username - String

email - String

transactions - List<Transaction> (OneToMany, mappedBy = "user")

Create another class named "Transaction" with the following attributes:

id - int (auto-generated primary key)

type - String

amount - double

description - String

user - User (ManyToOne, JsonIgnore)

Implement getters, setters, and constructors for the User and Transaction entities.

Inside the repository folder, create interfaces named "UserRepository" and "TransactionRepository".

Inside the service folder, create classes named "UserService" and "TransactionService".

Ensure cascading behavior for save and delete operations to maintain data consistency.

Refer to the below image for the project structure:

API ENDPOINTS :

API endpoints for User:

POST - "/users" - Returns response status 201 with user object on successful creation or else 500.

GET - "/users/{id}" - Returns response status 200 with user object, which includes details of transactions on successful retrieval or else 404.

PUT - "/users/{id}" - Returns response status 200 with updated user object, which includes details of transactions on successful updation or else 500. The fields "username" and "email" in the user object are modifiable except for the "id" and "transactions" fields.

DELETE - "/users/{id}" - Returns response status 200 with String "User deleted successfully" on successful deletion or else "User not found with ID: " + id.

API endpoints for Transaction:

POST - "/transactions/user/{userId}" - Returns response status 201 with transaction object on successful mapping of the transaction object to the specified userId or else 500.

GET - "/transactions/{id}" - Returns response status 200 with transaction object or else 404.

GET - "/transactions/user/{userId}" - Returns response status 200 with List<Transaction> object, retrieving all transactions associated with the specified userId or else 404.

PUT - "/transactions/{id}" - Returns response status 200 with updated transaction object on successful updation or else 500. The fields "type", "amount" and "description" in the transaction object are modifiable except for the "id" and "user" fields.

DELETE - "/transactions/{id}" - Returns response status 200 with String "Transaction deleted successfully" on successful deletion or else "Transaction not found with ID: " + id.

```
package com.examly.springapp.controller;

import java.util.List;

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

import com.examly.springapp.model.Transaction;
import com.examly.springapp.model.User;
import com.examly.springapp.service.TransactionService;
import com.examly.springapp.service.UserService;

@RestController
public class TransactionController {

    @Autowired
    TransactionService ts;
    @Autowired
    UserService us;

    @PostMapping("/transactions/user/{userId}")
    public ResponseEntity<Transaction> add(@RequestBody Transaction te, @PathVariable int userId)
    {
        te.setUser(us.getById(userId));
        Transaction obj = ts.add(te);
        return new ResponseEntity<>(obj,HttpStatus.CREATED);
    }

    @GetMapping("/transactions/{id}")
    public ResponseEntity<Transaction> getById(@PathVariable int id)
    {
        Transaction obj = ts.getById(id);
        return new ResponseEntity<>(obj,HttpStatus.OK);
    }

    @GetMapping("/transactions/user/{userId}")
    public ResponseEntity<List<Transaction>> getAll(@PathVariable int userId)
    {
        User user = us.getById(userId);
        List<Transaction> obj = ts.getAll(user);
        return new ResponseEntity<>(obj,HttpStatus.OK);
    }

    @PutMapping("/transactions/{id}")
    public ResponseEntity<Transaction> updateById(@RequestBody Transaction ue,@PathVariable int
id)
    {
        Transaction obj = ts.updateById(id, ue);
        return new ResponseEntity<>(obj,HttpStatus.OK);
    }

    @DeleteMapping("/transactions/{id}")
    public ResponseEntity<String> deleteById(@PathVariable int id)
    {
        if(ts.deleteById(id))
            return new ResponseEntity<>("Transaction deleted successfully",HttpStatus.OK);
        else
            return new ResponseEntity<>("Not deleted",HttpStatus.NOT_FOUND);
    }
}

package com.examly.springapp.controller;

import javax.websocket.server.PathParam;

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

import com.examly.springapp.model.User;
import com.examly.springapp.service.UserService;
```

```
@RestController
public class UserController {

    @Autowired
    UserService us;

    @PostMapping("/users")
    public ResponseEntity<User> add(@RequestBody User ue)
    {
        User obj = us.add(ue);
        return new ResponseEntity<>(obj,HttpStatus.CREATED);
    }

    @GetMapping("/users/{id}")
    public ResponseEntity<User> getById(@PathVariable int id)
    {
        User obj = us.getById(id);
        return new ResponseEntity<>(obj,HttpStatus.OK);


    }

    @PutMapping("/users/{id}")
    public ResponseEntity<User> updateById(@RequestBody User ue,@PathVariable int id)
    {
        User obj = us.getById(id);
        obj.setEmail(ue.getEmail());
        obj.setUsername(ue.getUsername());
        us.add(obj);
        return new ResponseEntity<>(obj,HttpStatus.OK);
    }
    @DeleteMapping("/users/{id}")
    public ResponseEntity<String> deleteById(@PathVariable int id)
    {
        if(us.deleteById(id))
            return new ResponseEntity<>("User deleted successfully",HttpStatus.OK);
        else
            return new ResponseEntity<>("Not deleted",HttpStatus.NOT_FOUND);
    }




}



package com.examly.springapp.model;


import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.ManyToOne;


import com.fasterxml.jackson.annotation.JsonBackReference;


@Entity
public class Transaction {
    @Id
    int id;
    String type,description;
    double amount;


    @ManyToOne
    @JsonBackReference
```

```
User user;

public int getId() {
   return id;
}

public void setId(int id) {
   this.id = id;
}

public String getType() {
   return type;
}

public void setType(String type) {
   this.type = type;
}

public String getDescription() {
   return description;
}

public void setDescription(String description) {
   this.description = description;
}

public double getAmount() {
   return amount;
}

public void setAmount(double amount) {
        this.amount = amount;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }



}

package com.examly.springapp.model;

import java.util.List;

import javax.persistence.CascadeType;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.OneToMany;

import com.fasterxml.jackson.annotation.JsonManagedReference;

@Entity
public class User {

    @Id
    int id;
    String username,email;
```

```
@OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
@JsonManagedReference
List<Transaction> transactions;

public int getId() {
   return id;
}

public void setId(int id) {
   this.id = id;
}

public String getUsername() {
   return username;
}

public void setUsername(String username) {
   this.username = username;
}

public String getEmail() {
   return email;
}

public void setEmail(String email) {
   this.email = email;
}

public List<Transaction> getTransactions() {
   return transactions;
```

```
    }

    public void setTransactions(List<Transaction> transactions) {
        this.transactions = transactions;
    }


}


package com.examly.springapp.repository;

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

import com.examly.springapp.model.Transaction;

public interface TransactionRepository extends JpaRepository<Transaction,Integer>{

}

package com.examly.springapp.repository;

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

import com.examly.springapp.model.User;

public interface UserRepository extends JpaRepository<User,Integer> {

}
```

```java
package com.examly.springapp.service;

import java.util.List;

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

import com.examly.springapp.model.Transaction;
import com.examly.springapp.model.User;
import com.examly.springapp.repository.TransactionRepository;

@Service
public class TransactionService {

   @Autowired
   TransactionRepository tr;

   public Transaction add(Transaction te)
   {
      return tr.save(te);
   }

   public List<Transaction> getAll(User user)
   {
      return user.getTransactions();
   }

   public Transaction getById(int id)
   {
      return tr.findById(id).orElse(null);
   }
```

```
    public Transaction updateById(int id, Transaction te)
    {
        return tr.save(te);
    }

    public boolean deleteById(int id)
    {
        if(this.getById(id)==null)
            return false;

        tr.deleteById(id);
        return true;
    }
}


package com.examly.springapp.service;

import java.util.List;

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

import com.examly.springapp.model.Transaction;
import com.examly.springapp.model.User;
import com.examly.springapp.repository.UserRepository;

@Service
public class UserService {
    @Autowired
    UserRepository ur;

    public User add(User ue)
    {
        return ur.save(ue);
    }

    public User getById(int id)
    {
        return ur.findById(id).orElse(null);
    }

    public User updateById(int id, User ue)
    {
        return ur.save(ue);
    }

    public boolean deleteById(int id)
    {
        if(this.getById(id)==null)
            return false;

        ur.deleteById(id);
        return true;
    }
}
```

```
package com.examly.springapp;
```

```
import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.boot.autoconfigure.domain.EntityScan;


@SpringBootApplication

@EntityScan(basePackages = "com.examly.springapp.model")

public class SpringappApplication {


	public static void main(String[] args) {

		SpringApplication.run(SpringappApplication.class, args);

	}


}
```

```
spring.jpa.hibernate.ddl-auto=update

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

spring.datasource.usernaAme=root

spring.datasource.password=examly

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

spring.jpa.show-sql= true

spring.jpa.database-platform=org.hibernate.dialect.MySQL8Dialect


enable.swagger.plugin=true

spring.mvc.pathmatch.matching-strategy=ant-path-matcher
```