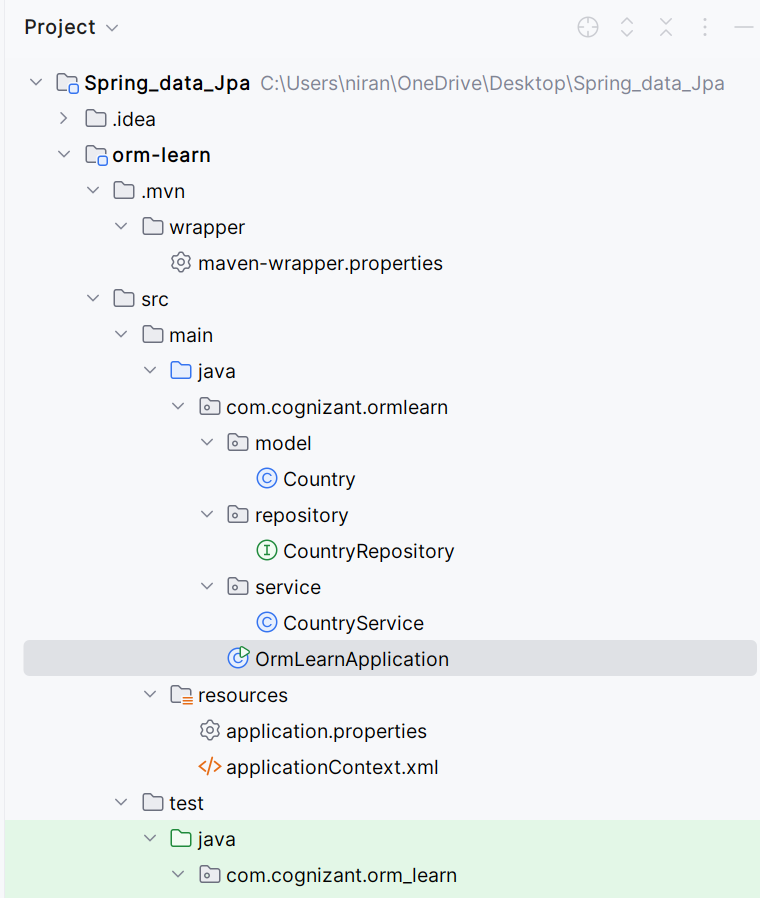
1. **Spring Data JPA - Quick Example**

**Pom.xml**<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <groupId>com.cognizant</groupId>  
 <artifactId>orm-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>orm-learn</name>  
 <description>Project for Spring Data JPA and Hibernate</description>  
  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Boot Starter Data JPA -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
  
 <!-- Spring Boot DevTools -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 <optional>true</optional>  
 </dependency>  
  
 <!-- Spring Boot Test -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**ApplicationContext.xml***<?*xml version="1.0" encoding="UTF-8"*?>*<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="  
 http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 *<!-- Repository Bean -->* <bean id="bookRepository" class="com.library.repository.BookRepository"/>

<bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

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

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect **Country.java**package com.cognizant.ormlearn.model;  
import jakarta.persistence.\*;  
  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 @Column(name = "code")  
 private String code;  
  
 @Column(name = "name")  
 private String name;

public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }

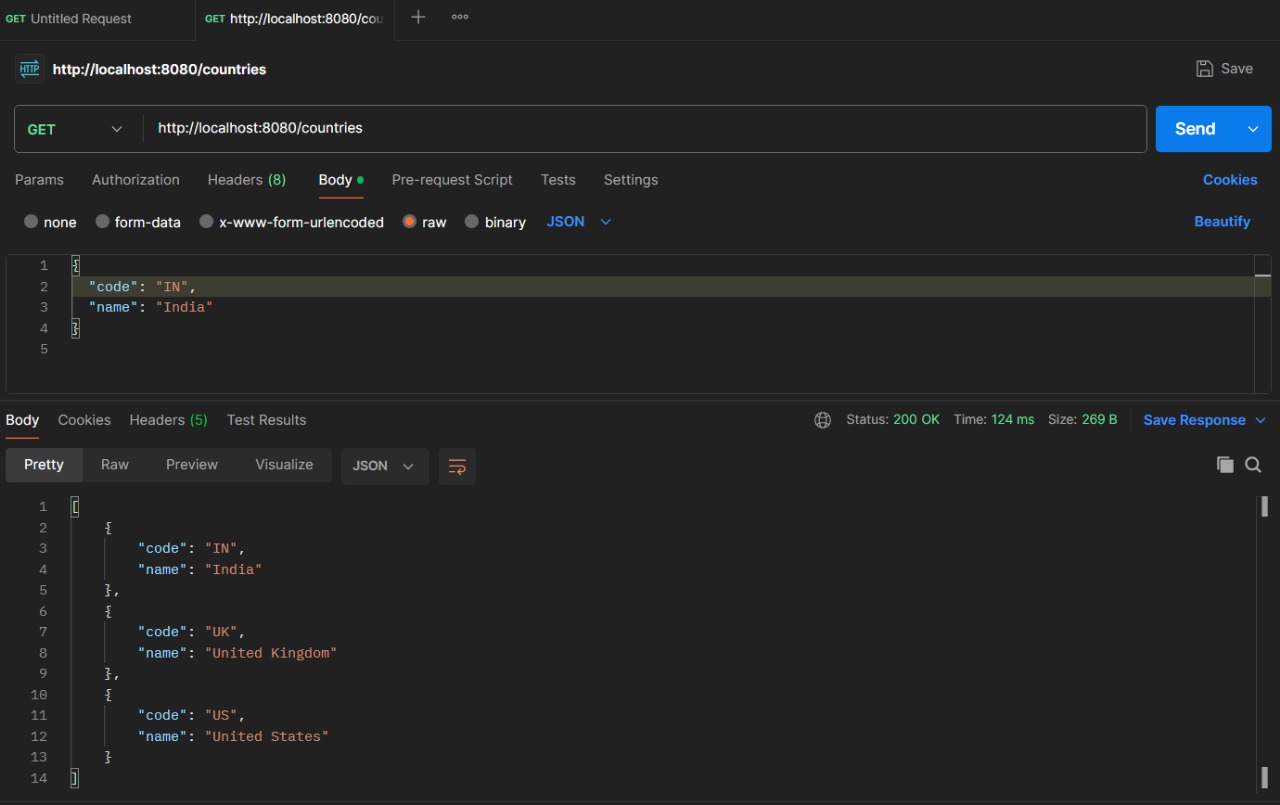
@Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

**CountryRepository.java**package com.cognizant.ormlearn.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import com.cognizant.ormlearn.model.Country;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
}

**CountryService.java**package com.cognizant.ormlearn.service;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.repository.CountryRepository;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

**OrmLearnApplication.java**package com.cognizant.ormlearn;

import java.util.List;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.service.CountryService;  
  
@SpringBootApplication  
public class OrmLearnApplication {  
  
 private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);  
 private static CountryService countryService;  
  
 public static void main(String[] args) {  
 ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);  
 LOGGER.info("Inside main");  
  
 countryService = context.getBean(CountryService.class);  
 testGetAllCountries();  
 }  
  
 private static void testGetAllCountries() {  
 LOGGER.info("Start");  
 List<Country> countries = countryService.getAllCountries();  
 LOGGER.debug("countries={}", countries);  
 LOGGER.info("End");  
 }  
}



**2.** **Difference Between JPA, Hibernate, and Spring Data JPA  
  
1. Java Persistence API (JPA)**

1. A specification (JSR 338) that defines how Java objects interact with relational databases.
2. Provides a set of interfaces and annotations for ORM.
3. Does not provide an implementation — needs a provider like Hibernate, EclipseLink, etc.

**2. Hibernate**

1. A popular **ORM (Object-Relational Mapping)** framework.
2. Provides a **concrete implementation** of the JPA specification.
3. Also includes additional features beyond JPA (e.g., native SQL, custom caching).
4. Developers often write boilerplate code to manage sessions, transactions, etc.

**3. Spring Data JPA**

1. A wrapper/abstraction over JPA providers like Hibernate.
2. Eliminates boilerplate code for DAO layers.
3. Provides built-in repository interfaces (like JpaRepository, CrudRepository).
4. Handles transactions automatically via Spring’s @Transactional.
5. Promotes clean, readable, and maintainable code.

**Hibernate vs Spring Data JPA  
  
Hibernate**public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

}

catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

}

finally {

session.close();

}

return employeeID;

}

**Spring Data JPA**  
  
EmployeeRepository.java  
  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {   
}   
  
EmployeeService.java

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

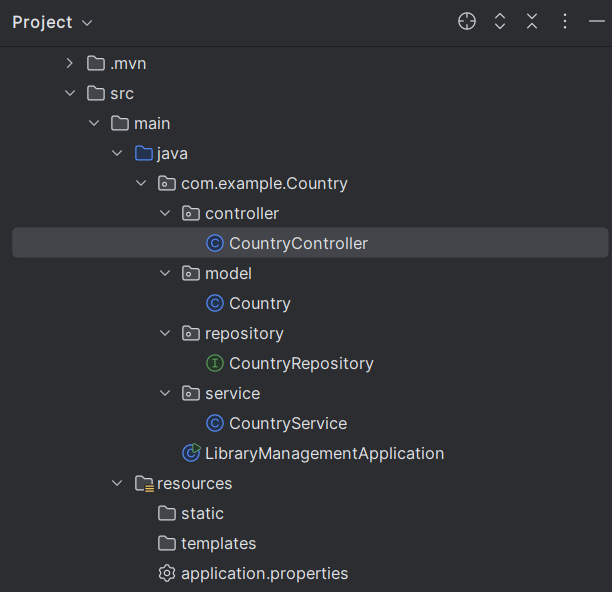
**Benefits:**

1. No need to write boilerplate code.
2. Built-in CRUD methods like save(), findById(), deleteById().
3. Automatically integrates with Spring’s transaction management.

**Additional Important  
 Handson**

**Implement services for managing Country**

**Folder Structure**

****

**Country.java**

package com.example.Country.model;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
import jakarta.persistence.Column;  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 @Column(name = "co\_code", length = 5)  
 private String code;  
  
 @Column(name = "co\_name", length = 100)  
 private String name;  
  
 // Getters and setters  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

**CountryRepository.java**

package com.example.Country.repository;  
  
import com.example.Country.model.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.List;  
  
public interface CountryRepository extends JpaRepository<Country, Long> {  
  
 // FIXED: Now matching the field 'name' from Country entity  
 List<Country> findByNameContainingIgnoreCase(String name);  
}

**CountryService.java**

package com.example.Country.service;  
  
import com.example.Country.model.Country;  
import com.example.Country.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository repository;  
  
public List<Country> getAllCountries() {  
 return repository.findAll();  
}

}

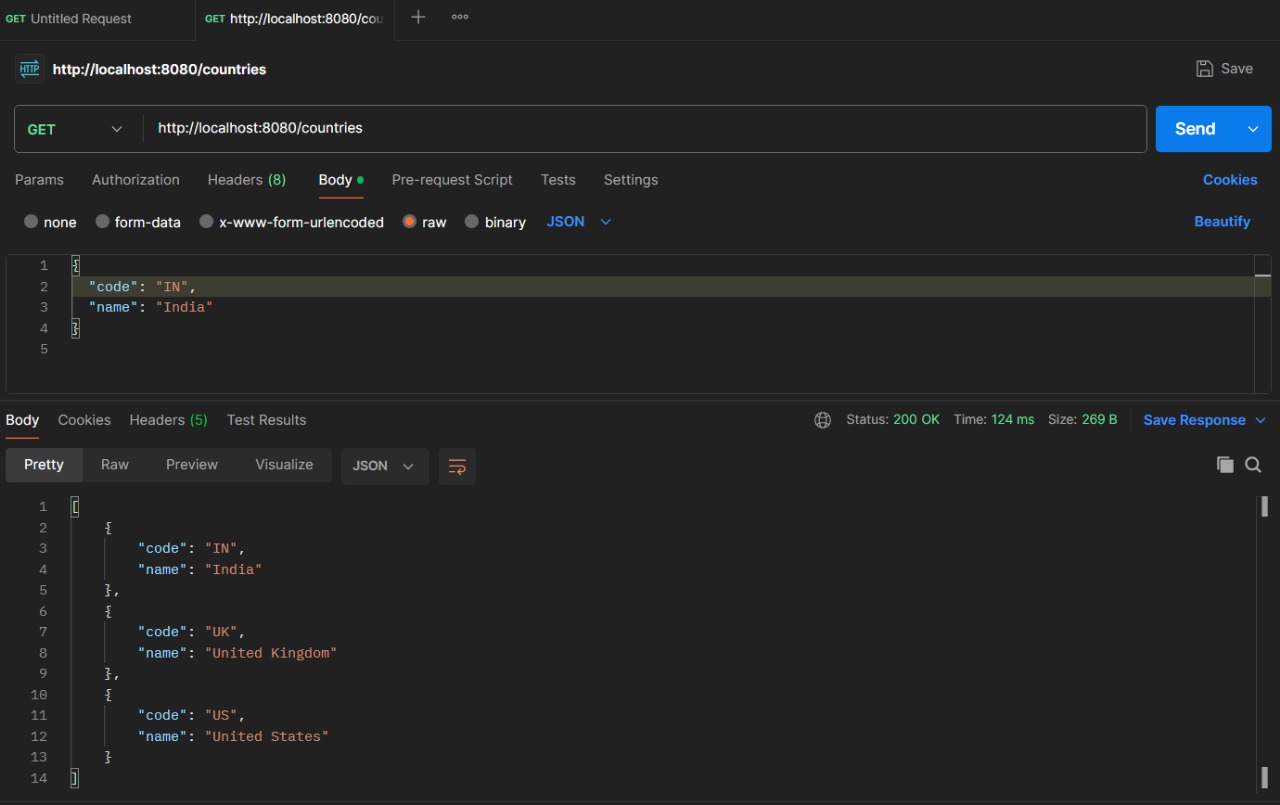
**CountryController.java**

package com.example.Country.controller;  
  
import com.example.Country.model.Country;  
import com.example.Country.service.CountryService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 @Autowired  
 private CountryService service;  
  
@GetMapping  
public List<Country> getAllCountries() {  
 return service.getAllCountries();  
}

**pom.xml**

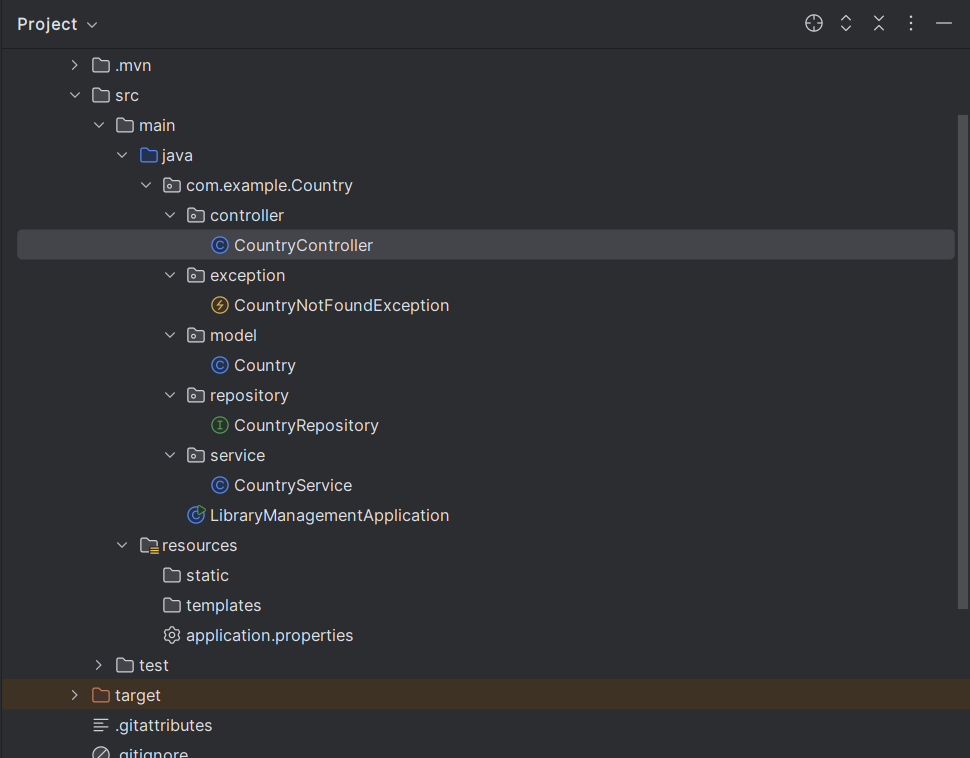
<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>

**Testing using postman**



**Find a country based on country code:**

**Folder structure**

****

**Country.java**

package com.example.Country.model;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
import jakarta.persistence.Column;  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 @Column(name = "co\_code", length = 5)  
 private String code;  
  
 @Column(name = "co\_name", length = 100)  
 private String name;  
  
 // Getters and setters  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

**CountryRepository.java**

package com.example.Country.repository;  
  
import com.example.Country.model.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
import java.util.List;  
  
public interface CountryRepository extends JpaRepository<Country, String> {  
 List<Country> findByNameContainingIgnoreCase(String name);  
}

**CountryService.java**

package com.example.Country.service;  
  
import com.example.Country.model.Country;  
import com.example.Country.repository.CountryRepository;  
import com.example.Country.exception.CountryNotFoundException;  
  
import jakarta.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public Country findCountryByCode(String countryCode) throws CountryNotFoundException {  
 Optional<Country> result = countryRepository.findById(countryCode);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country with code " + countryCode + " not found.");  
 }  
 return result.get();  
 }  
  
 public List<Country> searchCountries(String name) {  
 return countryRepository.findByNameContainingIgnoreCase(name);  
 }  
  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

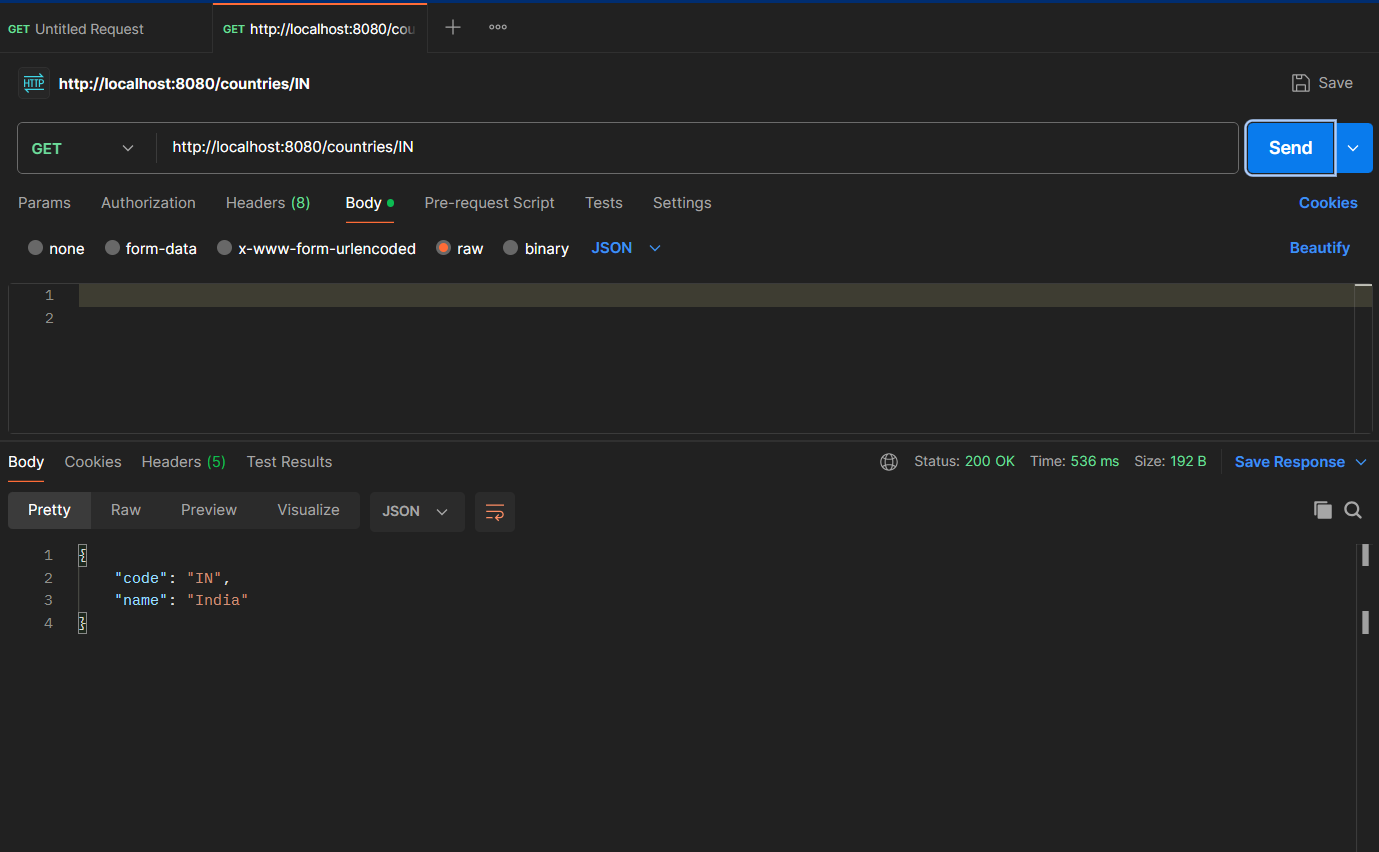
**CountryController.java**

package com.example.Country.controller;  
  
import com.example.Country.model.Country;  
import com.example.Country.service.CountryService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 @Autowired  
 private CountryService service;  
  
 @GetMapping  
 public List<Country> getAllCountries() {  
 return service.getAllCountries();  
 }  
  
 @GetMapping("/search")  
 public List<Country> searchByName(@RequestParam String name) {  
 return service.searchCountries(name);  
 }  
  
 @GetMapping("/{code}")  
 public Country getCountryByCode(@PathVariable String code) {  
 return service.findCountryByCode(code);  
 }  
}

**CountryNotfoundException.java**

package com.example.Country.exception;  
  
public class CountryNotFoundException extends RuntimeException {  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

**OUTPUT**

****

**ADD A NEW COUNTRY**

**CountryService.java**

package com.example.Country.service;  
  
import com.example.Country.exception.CountryNotFoundException;  
import com.example.Country.model.Country;  
import com.example.Country.repository.CountryRepository;  
import jakarta.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public Country findCountryByCode(String countryCode) throws CountryNotFoundException {  
 Optional<Country> result = countryRepository.findById(countryCode);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country with code " + countryCode + " not found.");  
 }  
 return result.get();  
 }  
  
 public List<Country> searchCountries(String name) {  
 return countryRepository.findByNameContainingIgnoreCase(name);  
 }  
  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
  
 @Transactional  
 public void addCountry(Country country) {  
 countryRepository.save(country);  
 }  
}

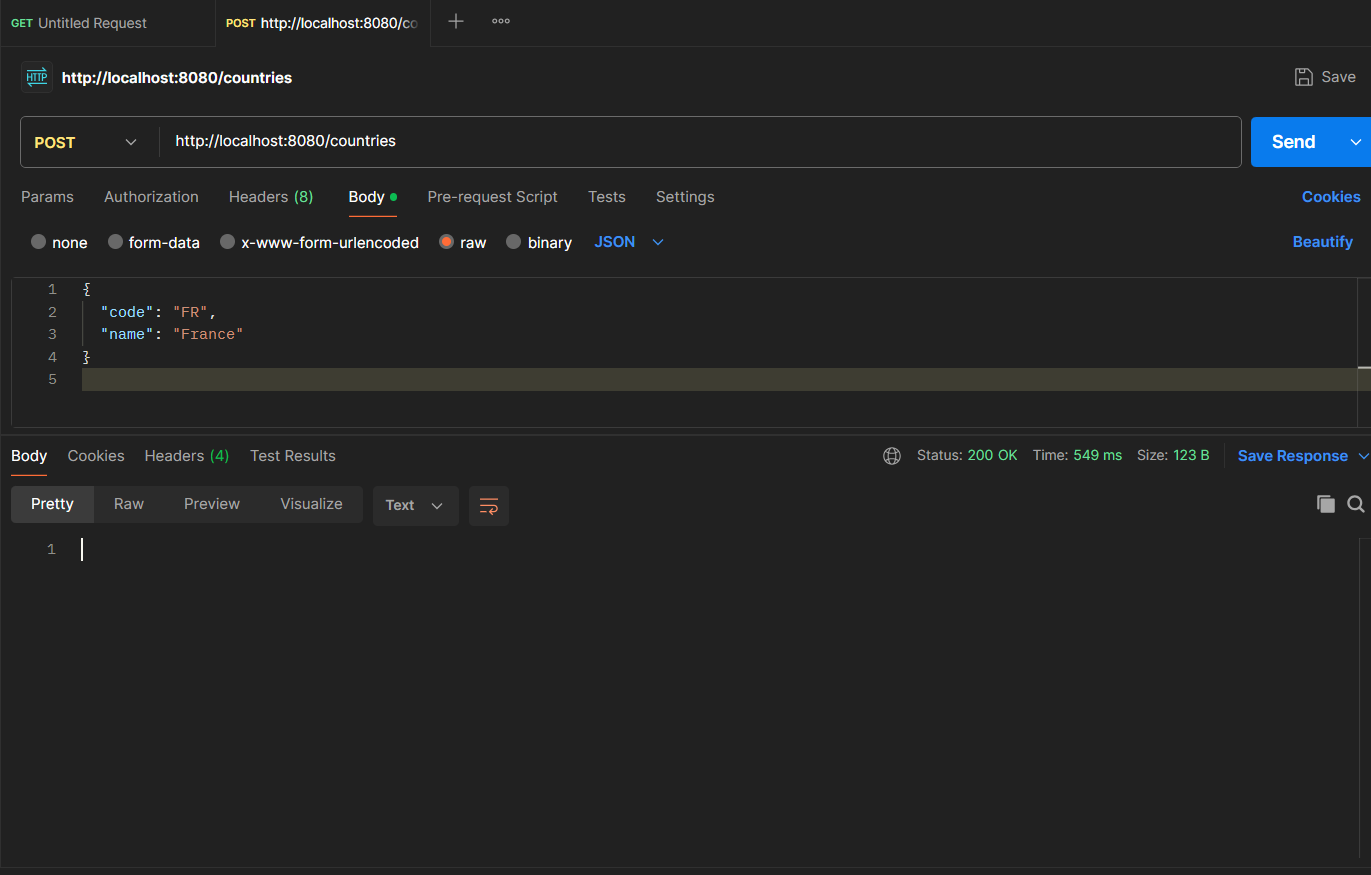
**LibraryManagementApplication.java**

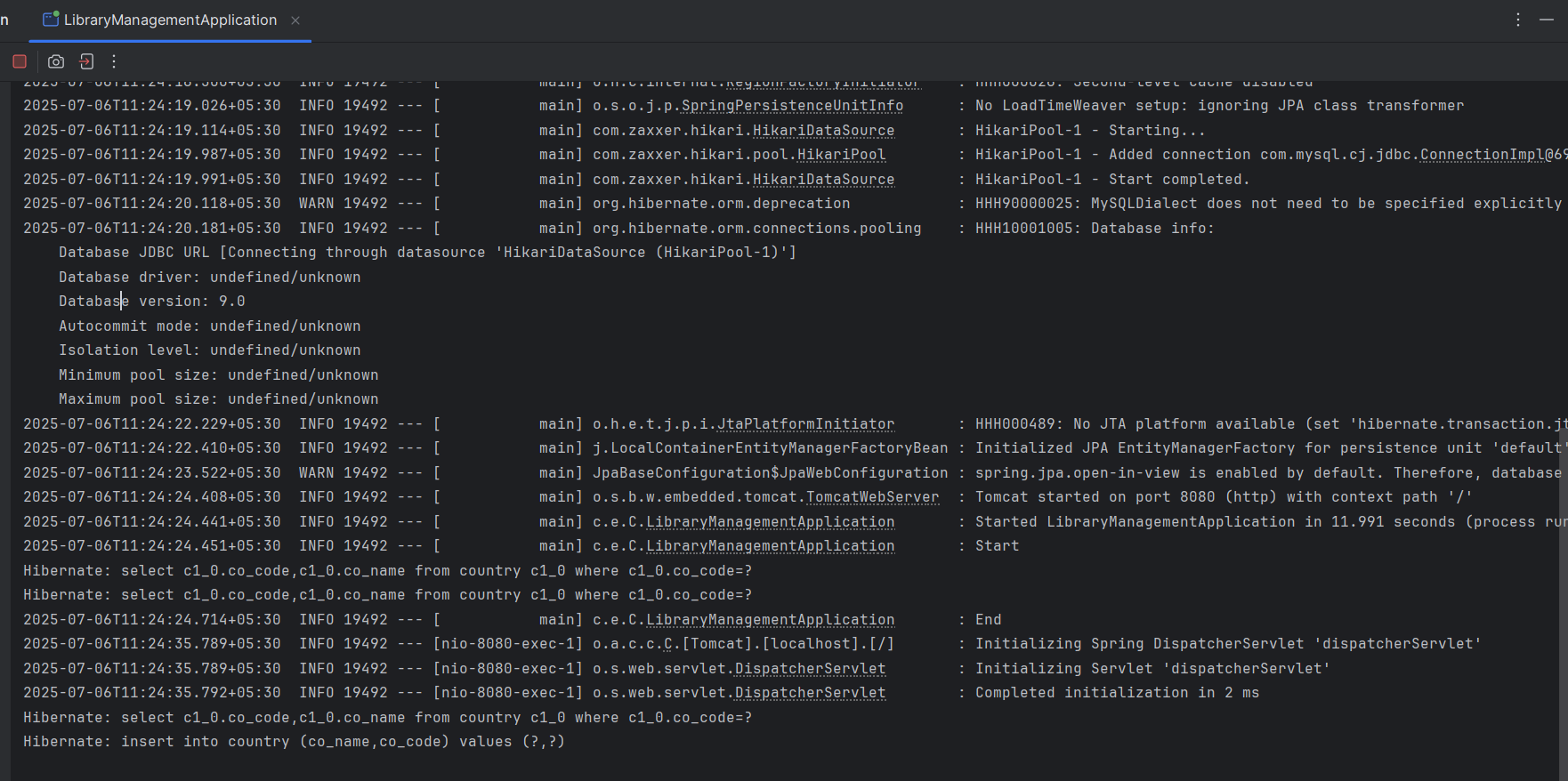
package com.example.Country;  
  
import com.example.Country.exception.CountryNotFoundException;  
import com.example.Country.model.Country;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import com.example.Country.service.CountryService;  
import org.springframework.context.ConfigurableApplicationContext;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
@SpringBootApplication  
public class LibraryManagementApplication {  
  
 private static CountryService *countryService*;  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(LibraryManagementApplication.class);  
  
 public static void main(String[] args) {  
 ConfigurableApplicationContext context = SpringApplication.*run*(LibraryManagementApplication.class, args);  
  
 *countryService* = context.getBean(CountryService.class);  
  
 *testAddCountry*(); // <- CALLING THE TEST METHOD  
 }  
  
 private static void testAddCountry() {  
 *LOGGER*.info("Start");  
  
 Country newCountry = new Country();  
 newCountry.setCode("JP");  
 newCountry.setName("Japan");  
  
 *countryService*.addCountry(newCountry); // Saving new country  
  
 try {  
 Country fetched = *countryService*.findCountryByCode("JP"); // Verify it was added  
 *LOGGER*.debug("Fetched Country: {}", fetched);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Country not found!", e);  
 }  
  
 *LOGGER*.info("End");  
 }  
}

**CountryService.java**

package com.example.Country.controller;  
  
import com.example.Country.model.Country;  
import com.example.Country.service.CountryService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 @Autowired  
 private CountryService service;  
  
 @GetMapping  
 public List<Country> getAllCountries() {  
 return service.getAllCountries();  
 }  
  
 @PostMapping  
 public void addCountry(@RequestBody Country country) {  
 service.addCountry(country);  
 }  
  
 @GetMapping("/code/{code}")  
 public Country getCountryByCode(@PathVariable String code) throws Exception {  
 return service.findCountryByCode(code);  
 }  
  
 @GetMapping("/search")  
 public List<Country> searchByName(@RequestParam String name) {  
 return service.searchCountries(name);  
 }  
}

**OUTPUT**

****



**Demonstrate implementation of Query Methods feature of Spring Data JPA**

**Country.java**

package com.example.Country.model;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 private String code;  
 private String name;  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 @Override  
 public String toString() {  
 return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + '}';  
 }  
}

**CountryRepository.java**

package com.example.Country.repository;

import com.example.Country.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String name);

List<Country> findByNameStartingWith(String prefix);

List<Country> findTop3ByOrderByNameAsc();

}

**CountryService.java**

package com.example.Country.service;

import com.example.Country.exception.CountryNotFoundException;

import com.example.Country.model.Country;

import com.example.Country.repository.CountryRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.List;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

public List<Country> searchCountries(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

public List<Country> getCountriesStartingWith(String prefix) {

return countryRepository.findByNameStartingWith(prefix);

}

public List<Country> getTop3CountriesAlphabetically() {

return countryRepository.findTop3ByOrderByNameAsc();

}

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

Optional<Country> result = countryRepository.findById(countryCode);

if (!result.isPresent()) {

throw new CountryNotFoundException("Country with code " + countryCode + " not found.");

}

return result.get();

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

}

**CountryController.java**

package com.example.Country.controller;

import com.example.Country.model.Country;

import com.example.Country.service.CountryService;

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

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService service;

@GetMapping

public List<Country> getAllCountries() {

return service.getAllCountries();

}

@GetMapping("/search")

public List<Country> searchByName(@RequestParam String name) {

return service.searchCountries(name);

}

@GetMapping("/starts-with")

public List<Country> getByPrefix(@RequestParam String prefix) {

return service.getCountriesStartingWith(prefix);

}

@GetMapping("/top3")

public List<Country> getTop3Countries() {

return service.getTop3CountriesAlphabetically();

}

@PostMapping

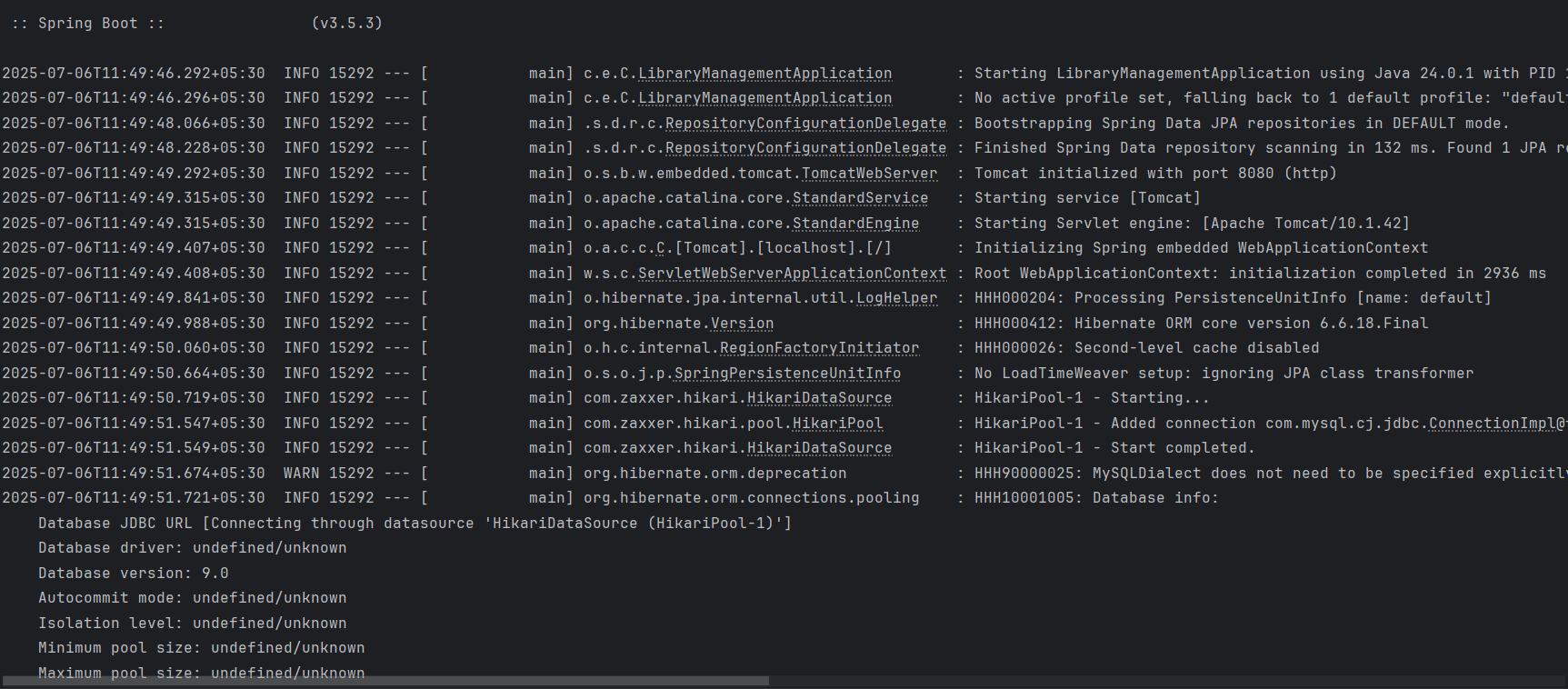
public void addCountry(@RequestBody Country country) {

service.addCountry(country);

}

}

**OUTPUT**

****

**Demonstrate implementation of O/R Mapping**

**Employee.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.List;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private double salary;

@Temporal(TemporalType.DATE)

private Date dateOfBirth;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "department\_id")

private Department department;

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(name = "employee\_skill",

joinColumns = @JoinColumn(name = "employee\_id"),

inverseJoinColumns = @JoinColumn(name = "skill\_id"))

private List<Skill> skills;

// Getters and Setters

}

**Department.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@OneToMany(mappedBy = "department", fetch = FetchType.LAZY)

private List<Employee> employees;

// Getters and Setters

}

**Skill.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToMany(mappedBy = "skills", fetch = FetchType.LAZY)

private List<Employee> employees;

// Getters and Setters

}

**EmployeeRepository.java**

package com.example.orm.repository;

import com.example.orm.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**DepartmentRepository.java**

package com.example.orm.repository;

import com.example.orm.model.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

**SkillRepository.java**

package com.example.orm.repository;

import com.example.orm.model.Skill;

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

public interface SkillRepository extends JpaRepository<Skill, Integer> {

}

**OrmLearnApplication.java**

package com.example.orm;

import com.example.orm.model.Department;

import com.example.orm.model.Employee;

import com.example.orm.model.Skill;

import com.example.orm.repository.DepartmentRepository;

import com.example.orm.repository.EmployeeRepository;

import com.example.orm.repository.SkillRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

import java.util.Arrays;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Bean

public CommandLineRunner demo(EmployeeRepository employeeRepo,

DepartmentRepository departmentRepo,

SkillRepository skillRepo) {

return args -> {

Department dept = new Department();

dept.setName("Engineering");

departmentRepo.save(dept);

Skill skill1 = new Skill();

skill1.setName("Java");

Skill skill2 = new Skill();

skill2.setName("Spring Boot");

skillRepo.saveAll(Arrays.asList(skill1, skill2));

Employee emp = new Employee();

emp.setName("John Doe");

emp.setDepartment(dept);

emp.setSkills(Arrays.asList(skill1, skill2));

employeeRepo.save(emp);

LOGGER.info("Employee Saved: {}", emp.getName());

};

}

}

**DepartmentController.java**

package com.example.orm.controller;

import com.example.orm.model.Department;

import com.example.orm.repository.DepartmentRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/departments")

public class DepartmentController {

@Autowired

private DepartmentRepository departmentRepository;

@GetMapping

public List<Department> getAllDepartments() {

return departmentRepository.findAll();

}

@PostMapping

public Department createDepartment(@RequestBody Department department) {

return departmentRepository.save(department);

}

@GetMapping("/{id}")

public Department getDepartmentById(@PathVariable int id) {

return departmentRepository.findById(id).orElse(null);

}

}

**SkillController.java**

package com.example.orm.controller;

import com.example.orm.model.Skill;

import com.example.orm.repository.SkillRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/skills")

public class SkillController {

@Autowired

private SkillRepository skillRepository;

@GetMapping

public List<Skill> getAllSkills() {

return skillRepository.findAll();

}

@PostMapping

public Skill createSkill(@RequestBody Skill skill) {

return skillRepository.save(skill);

}

@GetMapping("/{id}")

public Skill getSkillById(@PathVariable int id) {

return skillRepository.findById(id).orElse(null);

}

}

**EmployeeController.java**

package com.example.orm.controller;

import com.example.orm.model.Employee;

import com.example.orm.model.Department;

import com.example.orm.model.Skill;

import com.example.orm.repository.EmployeeRepository;

import com.example.orm.repository.DepartmentRepository;

import com.example.orm.repository.SkillRepository;

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

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

import java.util.List;

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeRepository employeeRepository;

@Autowired

private DepartmentRepository departmentRepository;

@Autowired

private SkillRepository skillRepository;

@GetMapping

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

// Ensure department and skills exist before saving

Department dept = departmentRepository.findById(employee.getDepartment().getId()).orElse(null);

List<Skill> skills = skillRepository.findAllById(

employee.getSkills().stream().map(Skill::getId).toList()

);

employee.setDepartment(dept);

employee.setSkills(skills);

return employeeRepository.save(employee);

}

@GetMapping("/{id}")

public Employee getEmployeeById(@PathVariable int id) {

return employeeRepository.findById(id).orElse(null);

}

}

**Get All Permanent Employees (with department & skill list)**

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query("SELECT e FROM Employee e LEFT JOIN FETCH e.department d LEFT JOIN FETCH e.skillList WHERE e.permanent = 1")

List<Employee> getAllPermanentEmployees();

}

**Get Average Salary of a Department**

@Query("SELECT AVG(e.salary) FROM Employee e WHERE e.department.id = :id")

double getAverageSalary(@Param("id") int departmentId);

**Native SQL Query Example**

**Get All Employees**

@Query(value = "SELECT \* FROM employee", nativeQuery = true)

List<Employee> getAllEmployeesNative();