WEEK 3 : COGNIZANT DN 4.0 FSE HANDS-ON EXERCISES

(MANDATORY PLUS ADDITIONAL)

* **Spring Core and Maven Exercises:**

**Exercise 1: Configuring a Basic Spring Application**

**Scenario:**

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

**Solution:**

**BookService.java**  
package com.library.service;  
  
import com.library.repository.BookRepository;  
import com.library.utility.Logger;  
  
public class BookService {  
  
 private BookRepository bookRepository;  
 private Logger logger;  
  
 // Constructor Injection  
 public BookService(Logger logger) {  
 this.logger = logger;  
 }  
  
 // Setter Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook() {  
 logger.log("Adding a book...");  
 bookRepository.saveBook();  
 }  
}

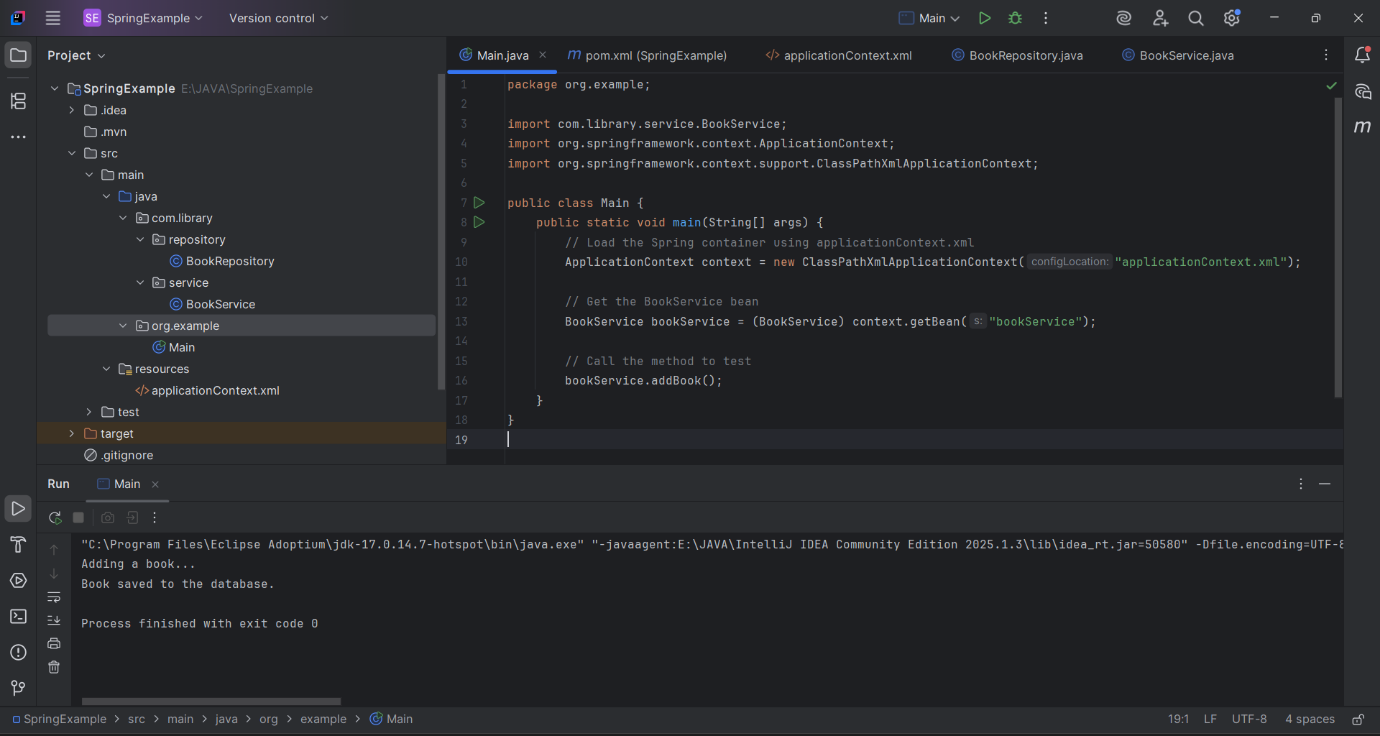
**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook() {  
 System.*out*.println("Book saved to the database.");  
 }  
}

**Main.java**

package org.example;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main {  
 public static void main(String[] args) {  
 // Load the Spring container using applicationContext.xml  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get the BookService bean  
 BookService bookService = (BookService) context.getBean("bookService");  
  
 // Call the method to test  
 bookService.addBook();  
 }  
}

**Output:**

****

**Exercise 2: Implementing Dependency Injection**

**Scenario:**

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

**Solution:**

**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook() {  
 System.*out*.println("Book saved to the database.");  
 }  
}

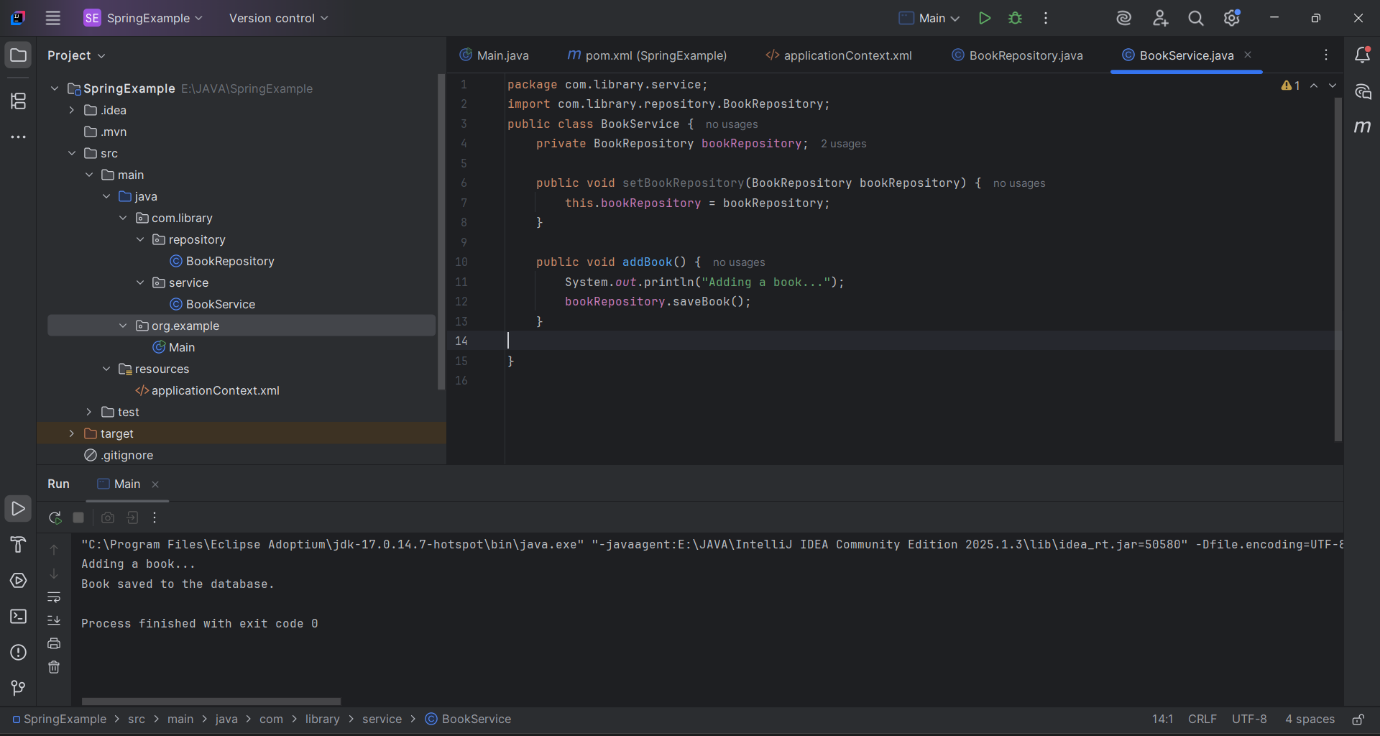
**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
import com.library.utility.Logger;  
  
public class BookService {  
  
 private BookRepository bookRepository;  
 private Logger logger;  
  
 // Constructor Injection  
 public BookService(Logger logger) {  
 this.logger = logger;  
 }  
  
 // Setter Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook() {  
 logger.log("Adding a book...");  
 bookRepository.saveBook();  
 }  
}

**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">  
  
 <!-- Logger Bean -->  
 <bean id="logger" class="com.library.utility.Logger"/>  
  
 <!-- BookRepository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository"/>  
  
 <!-- BookService Bean -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <!-- Constructor Injection -->  
 <constructor-arg ref="logger"/>  
  
 <!-- Setter Injection -->  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

**Output:**

****

**Exercise 4: Creating and Configuring a Maven Project**

**Scenario:**

You need to set up a new Maven project for the library management application and add Spring dependencies.

**Solution:**

**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook() {  
 System.*out*.println("Book saved to the database.");  
 }  
}

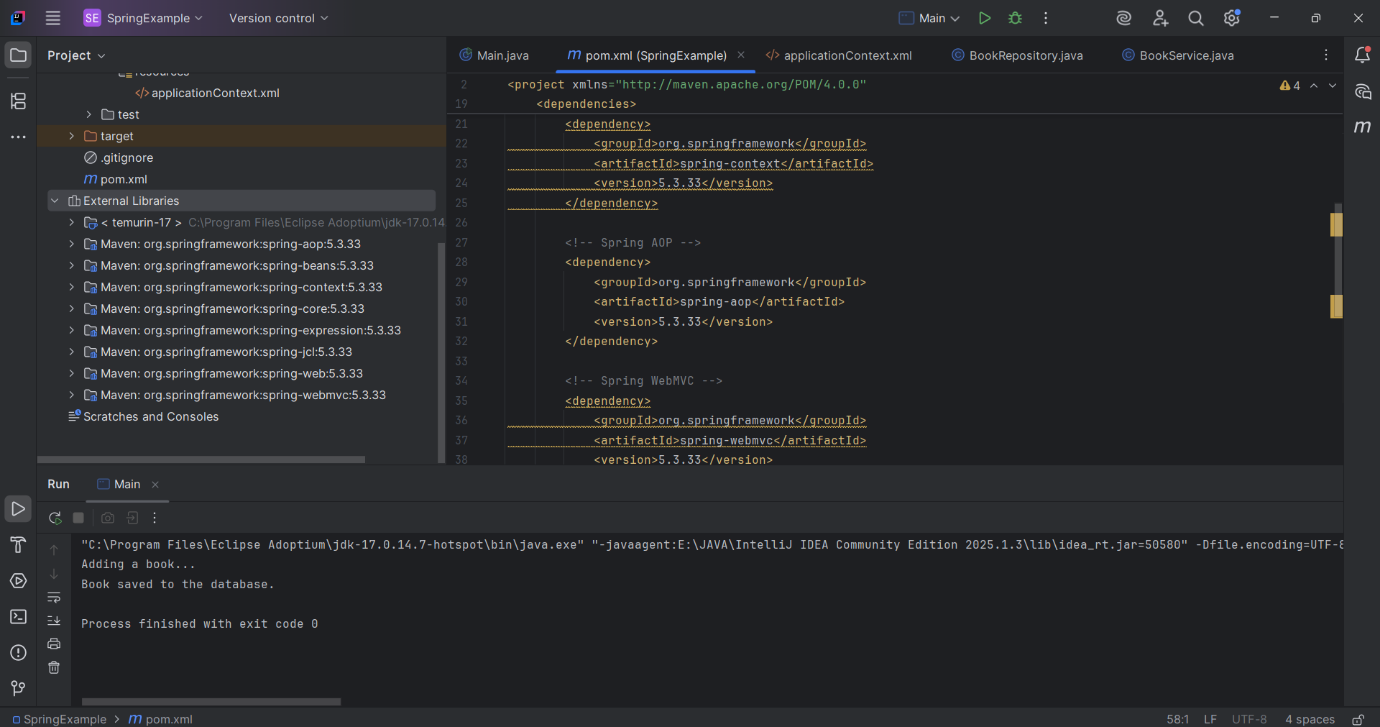
**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
import com.library.utility.Logger;  
  
public class BookService {  
  
 private BookRepository bookRepository;  
 private Logger logger;  
  
 // Constructor Injection  
 public BookService(Logger logger) {  
 this.logger = logger;  
 }  
  
 // Setter Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook() {  
 logger.log("Adding a book...");  
 bookRepository.saveBook();  
 }  
}

**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  
 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>SpringExample</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>17</maven.compiler.source>  
 <maven.compiler.target>17</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Core / Context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
  
 <!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
  
 <!-- Spring WebMVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Maven Compiler Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 <configuration>  
 <source>17</source>  
 <target>17</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Output:**

****

**Exercise 5: Configuring the Spring IoC Container**

**Scenario:**

The library management application requires a central configuration for beans and dependencies.

**Solution:**

**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook() {  
 System.*out*.println("Book saved to the database.");  
 }  
}

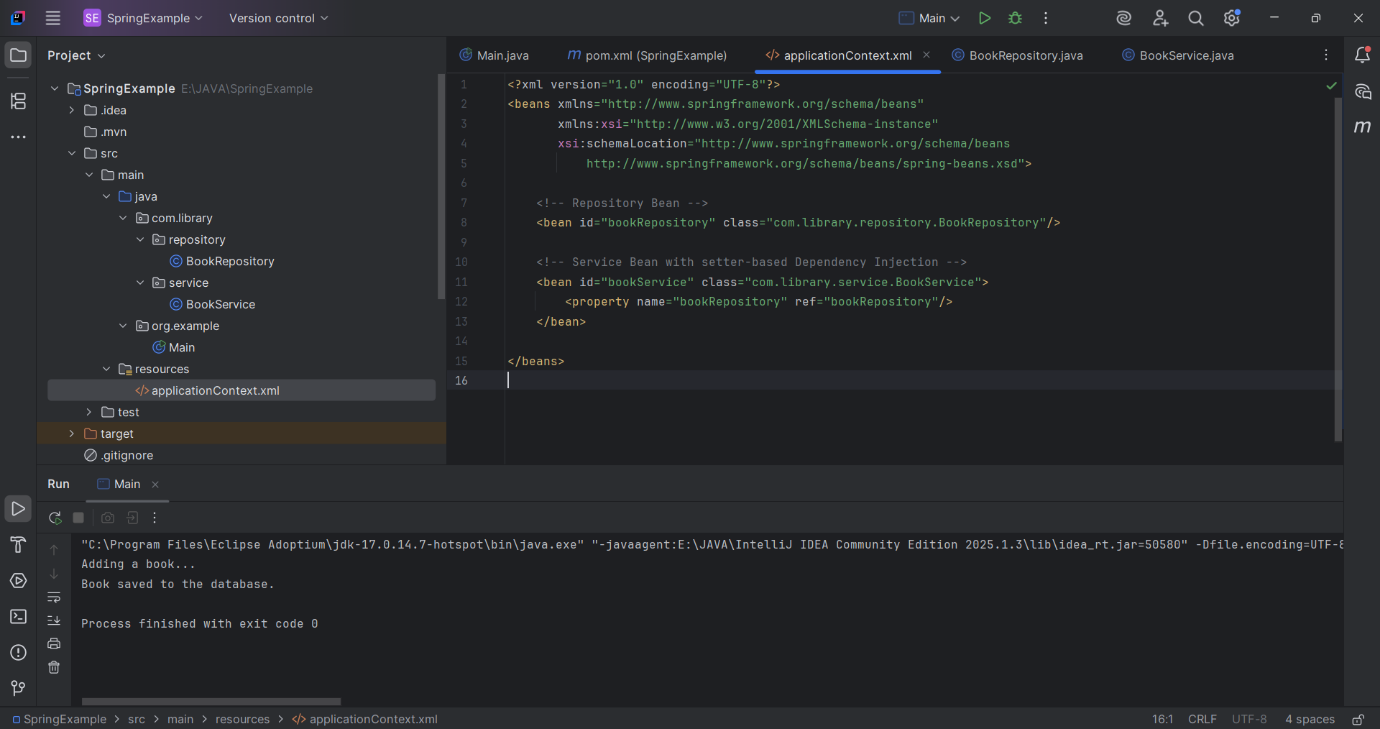
**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
import com.library.utility.Logger;  
  
public class BookService {  
  
 private BookRepository bookRepository;  
 private Logger logger;  
  
 // Constructor Injection  
 public BookService(Logger logger) {  
 this.logger = logger;  
 }  
  
 // Setter Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook() {  
 logger.log("Adding a book...");  
 bookRepository.saveBook();  
 }  
}

**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">  
  
 <!-- Logger Bean -->  
 <bean id="logger" class="com.library.utility.Logger"/>  
  
 <!-- BookRepository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository"/>  
  
 <!-- BookService Bean -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <!-- Constructor Injection -->  
 <constructor-arg ref="logger"/>  
  
 <!-- Setter Injection -->  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

**Output:**

****

**Exercise 7: Implementing Constructor and Setter Injection**

**Scenario:**

The library management application requires both constructor and setter injection for better control over bean initialization.

**Solution:**

**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
import com.library.utility.Logger;  
  
public class BookService {  
  
 private BookRepository bookRepository;  
 private Logger logger;  
  
 // Constructor Injection  
 public BookService(Logger logger) {  
 this.logger = logger;  
 }  
  
 // Setter Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook() {  
 logger.log("Adding a book...");  
 bookRepository.saveBook();  
 }  
}

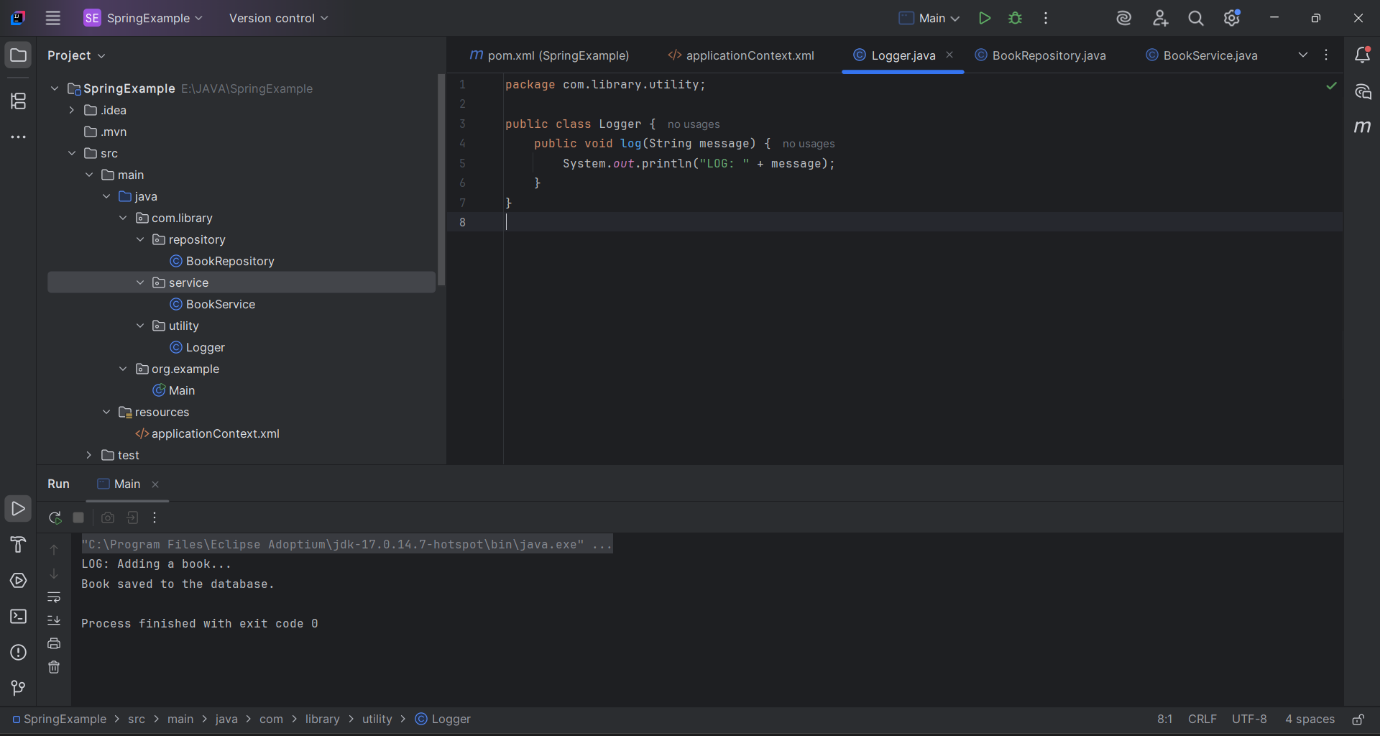
**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook() {  
 System.*out*.println("Book saved to the database.");  
 }  
}

**Logger.java**

package com.library.utility;  
  
public class Logger {  
 public void log(String message) {  
 System.*out*.println("LOG: " + message);  
 }  
}

**Output:**

****

**Exercise 9: Creating a Spring Boot Application**

**Scenario:**

You need to create a Spring Boot application for the library management system to simplify configuration and deployment.

**Solution:**

**LibraryManagementApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication

@ComponentScan(basePackages = {"com.example"})

@EnableJpaRepositories(basePackages = "com.example.repository")

@EntityScan(basePackages = "com.example.model")

public class LibraryManagementApplication {

public static void main(String[] args) {

SpringApplication.run(LibraryManagementApplication.class, args);

    }

}

**Book.java**

package com.example.model;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

private String isbn;

// Constructors

public Book() {}

public Book(String title, String author, String isbn) {

this.title = title;

this.author = author;

this.isbn = isbn;

}

// Getters and Setters

public Long getId() { return id; }

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

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

public String getIsbn() { return isbn; }

public void setIsbn(String isbn) { this.isbn = isbn; }

}

**BookController.java**

import com.example.model.Book;

import com.example.repository.BookRepository;

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

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

import java.util.List;

import java.util.Optional;

@RestController

@RequestMapping("/books")

public class BookController {

@Autowired

private BookRepository bookRepository;

// CREATE

@PostMapping

public Book createBook(@RequestBody Book book) {

return bookRepository.save(book);

}

// READ ALL

@GetMapping

public List<Book> getAllBooks() {

return bookRepository.findAll();

}

// READ BY ID

@GetMapping("/{id}")

public Book getBookById(@PathVariable Long id) {

return bookRepository.findById(id)

.orElseThrow(() -> new RuntimeException("Book not found with id: " + id));

}

// UPDATE

@PutMapping("/{id}")

public Book updateBook(@PathVariable Long id, @RequestBody Book updatedBook) {

return bookRepository.findById(id)

.map(book -> {

book.setTitle(updatedBook.getTitle());

book.setAuthor(updatedBook.getAuthor());

book.setIsbn(updatedBook.getIsbn());

return bookRepository.save(book);

})

.orElseThrow(() -> new RuntimeException("Book not found with id: " + id));

}

// DELETE

@DeleteMapping("/{id}")

public String deleteBook(@PathVariable Long id) {

if (bookRepository.existsById(id)) {

bookRepository.deleteById(id);

return "Book deleted with id: " + id;

} else {

return "Book not found with id: " + id;

       }

    }

}

**BookRepository.java**

package com.example.repository;

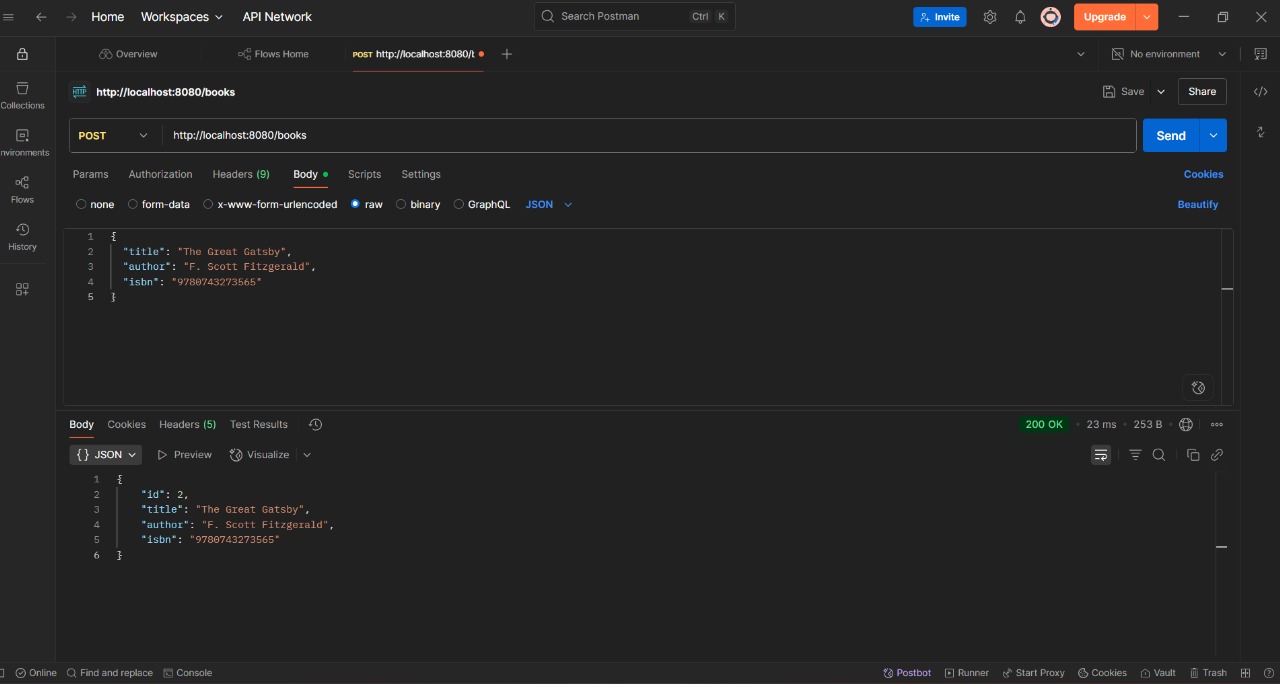
import com.example.model.Book;

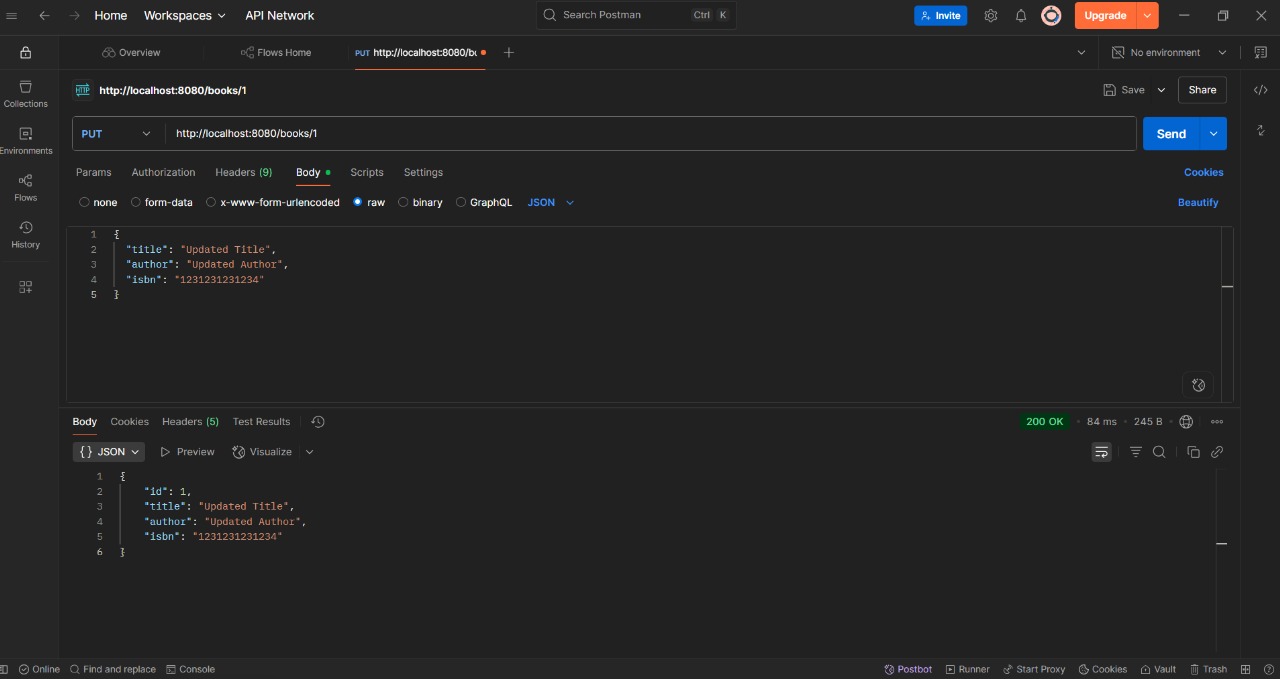
import org.springframework.data.jpa.repository.JpaRepository;

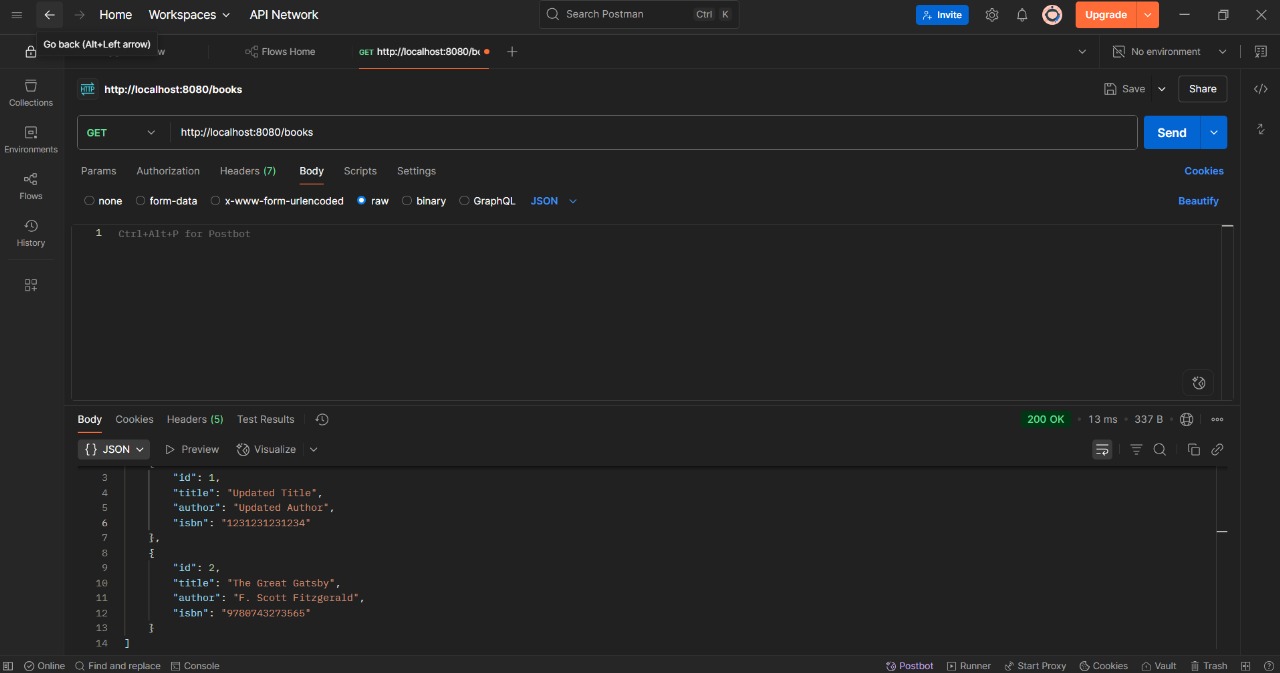
public interface BookRepository extends JpaRepository<Book, Long> {

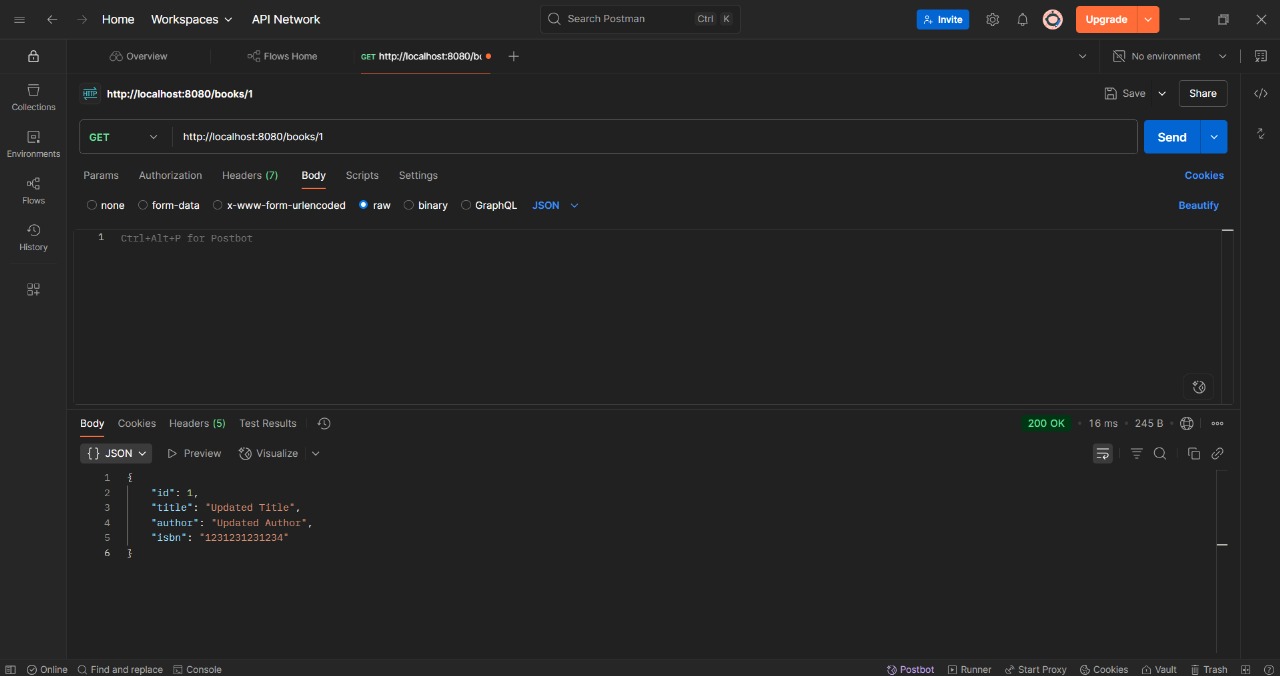
}

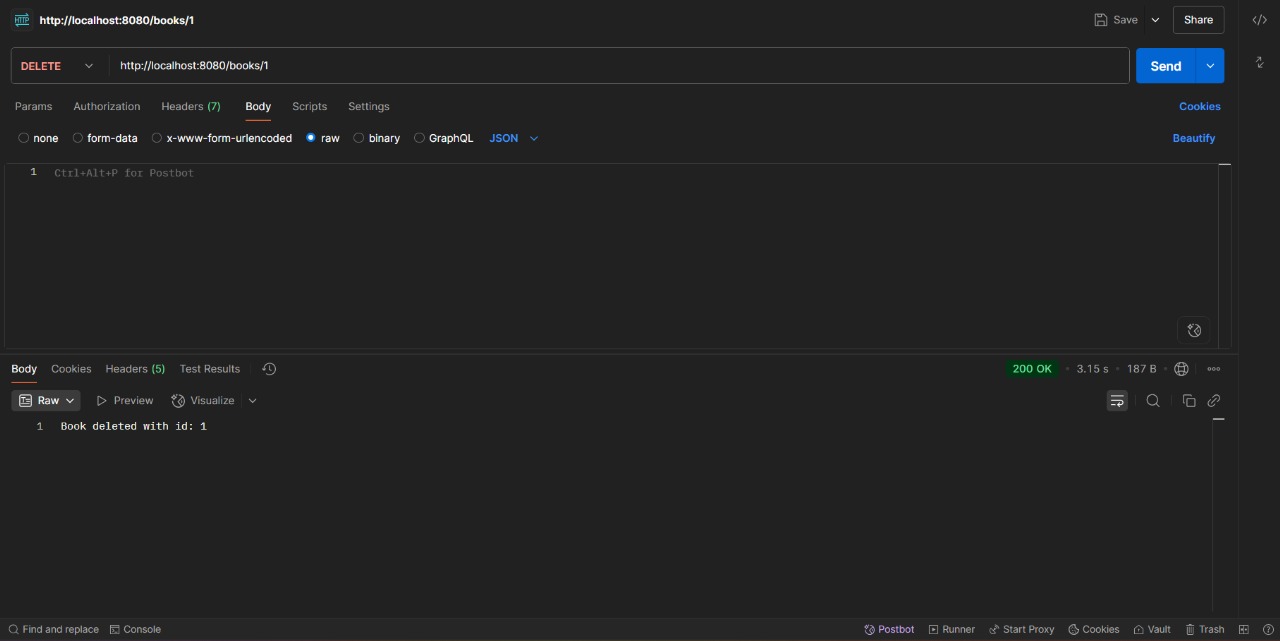
**Output:**

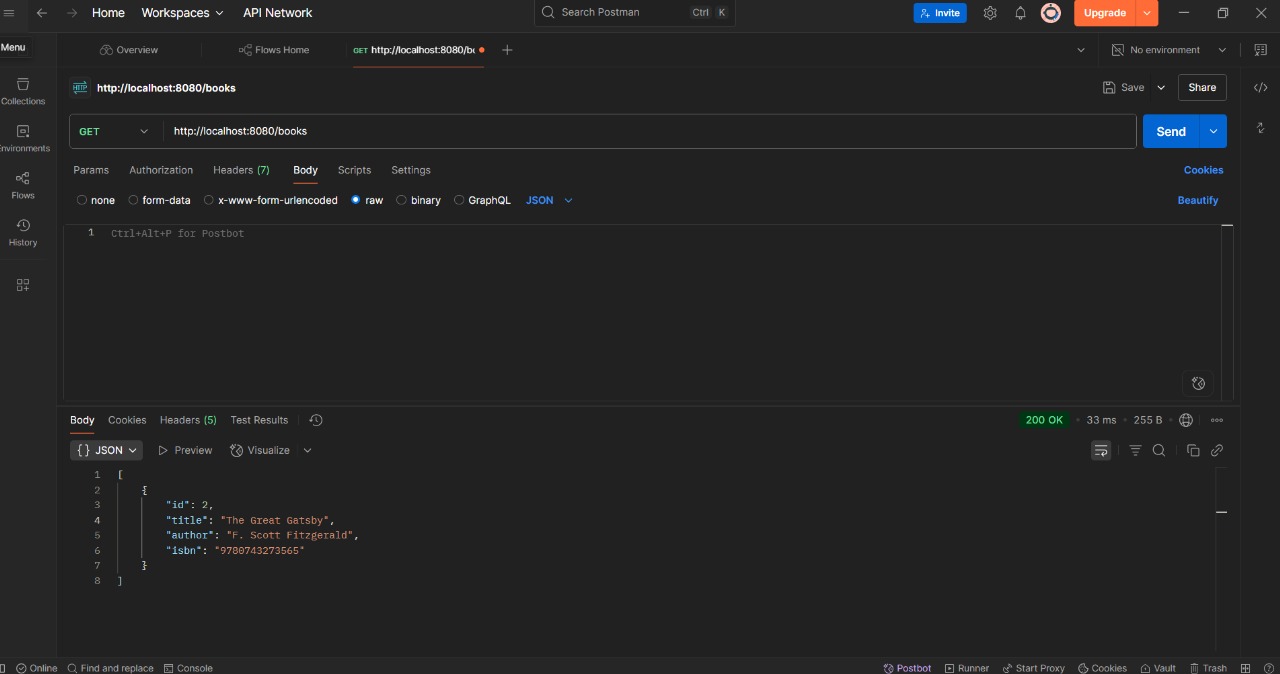












* **Spring Data JPA Hands-on Exercises:**

**Exercise 1: Spring Data JPA - Quick Example**

**Solution:**

**Country.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

*@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 + "]";

}

}

**CountryService.java**

package com.cognizant.orm\_learn.service;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

import jakarta.transaction.Transactional;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

*@Transactional*

public void addCountry(Country country) {

// Only add if country doesn't already exist

if (!countryRepository.existsById(country.getCode())) {

countryRepository.save(country);

}

}

*@Transactional*

public Country findCountryByCode(String code) {

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

return result.orElse(null);

}

*@Transactional*

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

}

**CountryRepository.java**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.orm\_learn.model.Country;

*@Repository*

public interface CountryRepository extends JpaRepository<Country, String> {

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

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;

*@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);

*countryService* = context.getBean(CountryService.class);

***LOGGER***.info("Inside main");

// 1. Fetch all countries

*testGetAllCountries*();

// 2. Add a new country

*testAddCountry*();

// 3. Find a country by code

*testFindCountryByCode*();

// 4. Delete a country

*testDeleteCountry*();

// 5. Final country list

*testGetAllCountries*();

}

private static void testGetAllCountries() {

***LOGGER***.info("Start");

List<Country> countries = *countryService*.getAllCountries();

***LOGGER***.debug("countries={}", countries);

***LOGGER***.info("End");

}

private static void testAddCountry() {

***LOGGER***.info("Start - Add Country");

Country country = new Country();

country.setCode("JP");

country.setName("Japan");

*countryService*.addCountry(country);

***LOGGER***.info("Added country: {}", country);

***LOGGER***.info("End");

}

private static void testFindCountryByCode() {

***LOGGER***.info("Start - Find Country");

Country country = *countryService*.findCountryByCode("JP");

***LOGGER***.info("Country found: {}", country);

***LOGGER***.info("End");

}

private static void testDeleteCountry() {

***LOGGER***.info("Start - Delete Country");

*countryService*.deleteCountry("JP");

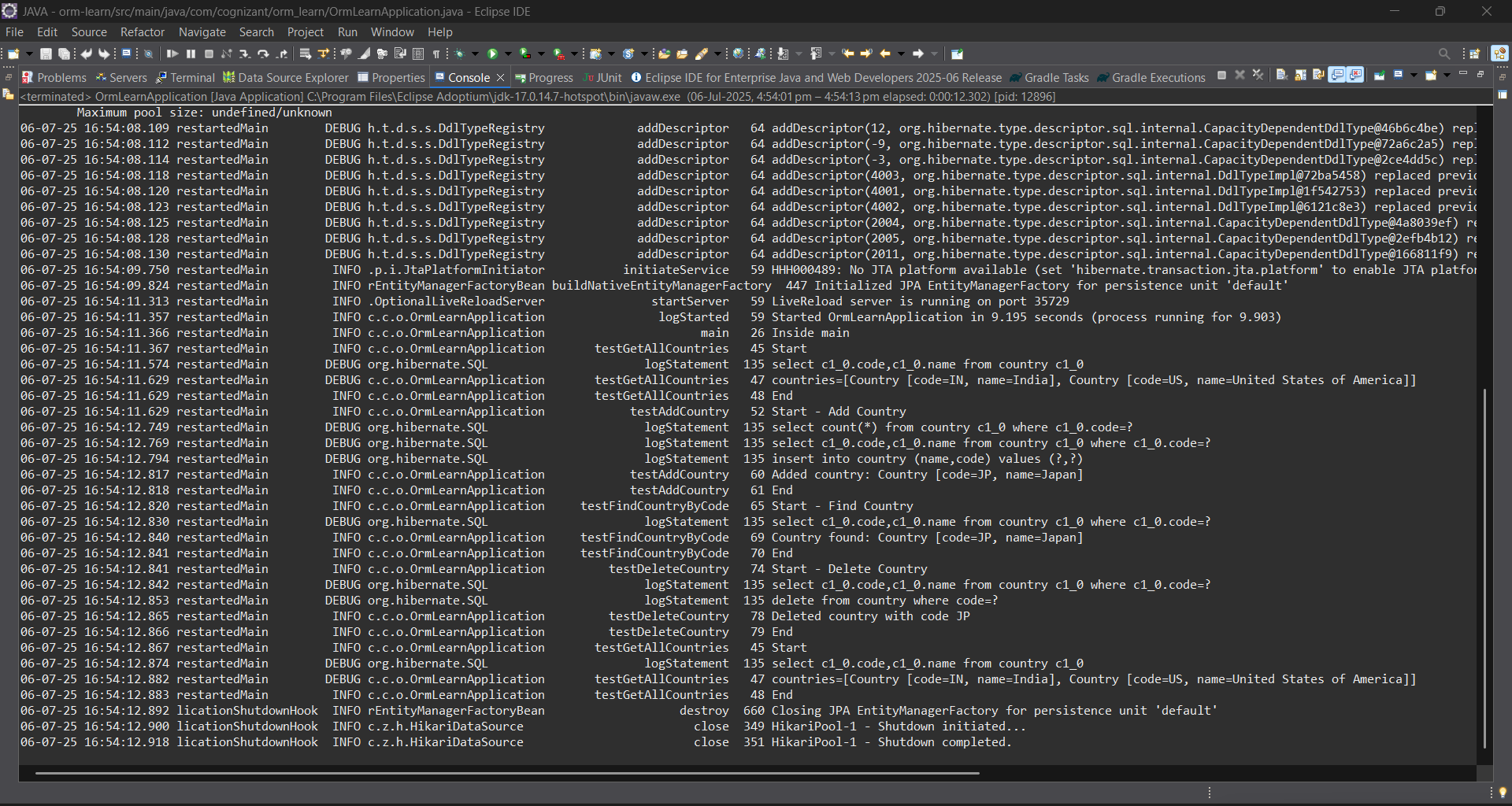
***LOGGER***.info("Deleted country with code JP");

***LOGGER***.info("End");

}

}

**Output:**

****

**Exercise 4: Difference between JPA, Hibernate and Spring Data JPA**

Java Persistence API (JPA)

* JSR 338 Specification for persisting, reading and managing data from Java objects
* Does not contain concrete implementation of the specification
* Hibernate is one of the implementation of JPA

Hibernate

* ORM Tool that implements JPA

Spring Data JPA

* Does not have JPA implementation, but reduces boiler plate code
* This is another level of abstraction over JPA implementation provider like Hibernate
* Manages transactions

**Solution:**

**Employee.java**package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.Set;

*@Entity*

*@Table*(name = "employee")

public class Employee {

*@Id*

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

*@Column*(name = "em\_id")

private int id;

*@Column*(name = "em\_name")

private String name;

*@Column*(name = "em\_salary")

private double salary;

*@Column*(name = "em\_permanent")

private boolean permanent;

*@Column*(name = "em\_date\_of\_birth")

private Date dateOfBirth;

*@ManyToOne*

*@JoinColumn*(name = "em\_dp\_id")

private Department department;

*@ManyToMany*(fetch = *FetchType*.***EAGER***)

*@JoinTable*(

name = "employee\_skill",

joinColumns = *@JoinColumn*(name = "es\_em\_id"),

inverseJoinColumns = *@JoinColumn*(name = "es\_sk\_id")

)

private Set<Skill> skillList;

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 double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

public Department getDepartment() { return department; }

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

public Set<Skill> getSkillList() { return skillList; }

public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

*@Override*

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +

", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth + "]";

}

}

**Department.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

*@Entity*

*@Table*(name = "department")

public class Department {

*@Id*

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

*@Column*(name = "dp\_id")

private int id;

*@Column*(name = "dp\_name")

private String name;

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

private Set<Employee> employeeList;

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 Set<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(Set<Employee> employeeList) {

this.employeeList = employeeList;

}

*@Override*

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

**Skill.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

*@Entity*

*@Table*(name = "skill")

public class Skill {

*@Id*

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

*@Column*(name = "sk\_id")

private int id;

*@Column*(name = "sk\_name")

private String name;

*@ManyToMany*(mappedBy = "skillList")

private Set<Employee> employeeList;

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 Set<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(Set<Employee> employeeList) {

this.employeeList = employeeList;

}

*@Override*

public String toString() {

return "Skill [id=" + id + ", name=" + name + "]";

}

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

**DepartmentRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {}

**SkillRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {}

**EmployeeService.java**package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Employee;

import com.cognizant.orm\_learn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@Service*

public class EmployeeService {

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

*@Autowired*

private EmployeeRepository employeeRepository;

*@Transactional*

public Employee get(int id) {

***LOGGER***.info("Start");

return employeeRepository.findById(id).get();

}

*@Transactional*

public void save(Employee employee) {

***LOGGER***.info("Start");

employeeRepository.save(employee);

***LOGGER***.info("End");

}

}

**DepartmentService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.repository.DepartmentRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class DepartmentService {

*@Autowired*

private DepartmentRepository departmentRepository;

*@Transactional*

public Department get(int id) {

return departmentRepository.findById(id).get();

}

*@Transactional*

public void save(Department department) {

departmentRepository.save(department);

}

}

**SkillService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Skill;

import com.cognizant.orm\_learn.repository.SkillRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class SkillService {

*@Autowired*

private SkillRepository skillRepository;

*@Transactional*

public Skill get(int id) {

return skillRepository.findById(id).get();

}

*@Transactional*

public void save(Skill skill) {

skillRepository.save(skill);

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.model.Employee;

import com.cognizant.orm\_learn.model.Skill;

import com.cognizant.orm\_learn.service.DepartmentService;

import com.cognizant.orm\_learn.service.EmployeeService;

import com.cognizant.orm\_learn.service.SkillService;

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 java.util.Date;

import java.util.HashSet;

import java.util.Set;

*@SpringBootApplication*

public class OrmLearnApplication {

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

private static EmployeeService *employeeService*;

private static DepartmentService *departmentService*;

private static SkillService *skillService*;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);

*employeeService* = context.getBean(EmployeeService.class);

*departmentService* = context.getBean(DepartmentService.class);

*skillService* = context.getBean(SkillService.class);

***LOGGER***.info("Inside main");

// Uncomment the methods below one by one to test

*testAddEmployee*();

//testGetEmployee();

*testUpdateEmployee*();

*testGetDepartment*();

*testAddSkillToEmployee*();

}

private static void testAddEmployee() {

***LOGGER***.info("Start - testAddEmployee");

Employee employee = new Employee();

employee.setName("Jane Doe");

employee.setSalary(60000.0);

employee.setPermanent(true);

employee.setDateOfBirth(new Date());

// Set Department

Department department = *departmentService*.get(1); // ensure this exists in DB

employee.setDepartment(department);

// Set Skills

Set<Skill> skillList = new HashSet<>();

skillList.add(*skillService*.get(1)); // Java

skillList.add(*skillService*.get(2)); // Spring Boot

employee.setSkillList(skillList);

// Save Employee

*employeeService*.save(employee);

***LOGGER***.debug("Employee added: {}", employee);

***LOGGER***.info("End");

}

private static void testGetEmployee() {

***LOGGER***.info("Start - testGetEmployee");

Employee employee = *employeeService*.get(1); // use valid employee id

***LOGGER***.debug("Employee: {}", employee);

***LOGGER***.debug("Department: {}", employee.getDepartment());

***LOGGER***.debug("Skills: {}", employee.getSkillList());

***LOGGER***.info("End");

}

private static void testUpdateEmployee() {

***LOGGER***.info("Start - testUpdateEmployee");

Employee employee = *employeeService*.get(1); // use valid employee id

Department newDept = *departmentService*.get(2); // different department ID

employee.setDepartment(newDept);

*employeeService*.save(employee);

***LOGGER***.debug("Updated Employee: {}", employee);

***LOGGER***.info("End");

}

private static void testGetDepartment() {

***LOGGER***.info("Start - testGetDepartment");

Department department = *departmentService*.get(1); // department with multiple employees

***LOGGER***.debug("Department: {}", department);

***LOGGER***.debug("Employees: {}", department.getEmployeeList());

***LOGGER***.info("End");

}

private static void testAddSkillToEmployee() {

***LOGGER***.info("Start - testAddSkillToEmployee");

Employee employee = *employeeService*.get(1); // valid employee id

Skill skill = *skillService*.get(2); // valid skill id not already assigned

Set<Skill> skillList = employee.getSkillList();

if (skillList == null) {

skillList = new HashSet<>();

}

skillList.add(skill);

employee.setSkillList(skillList);

*employeeService*.save(employee);

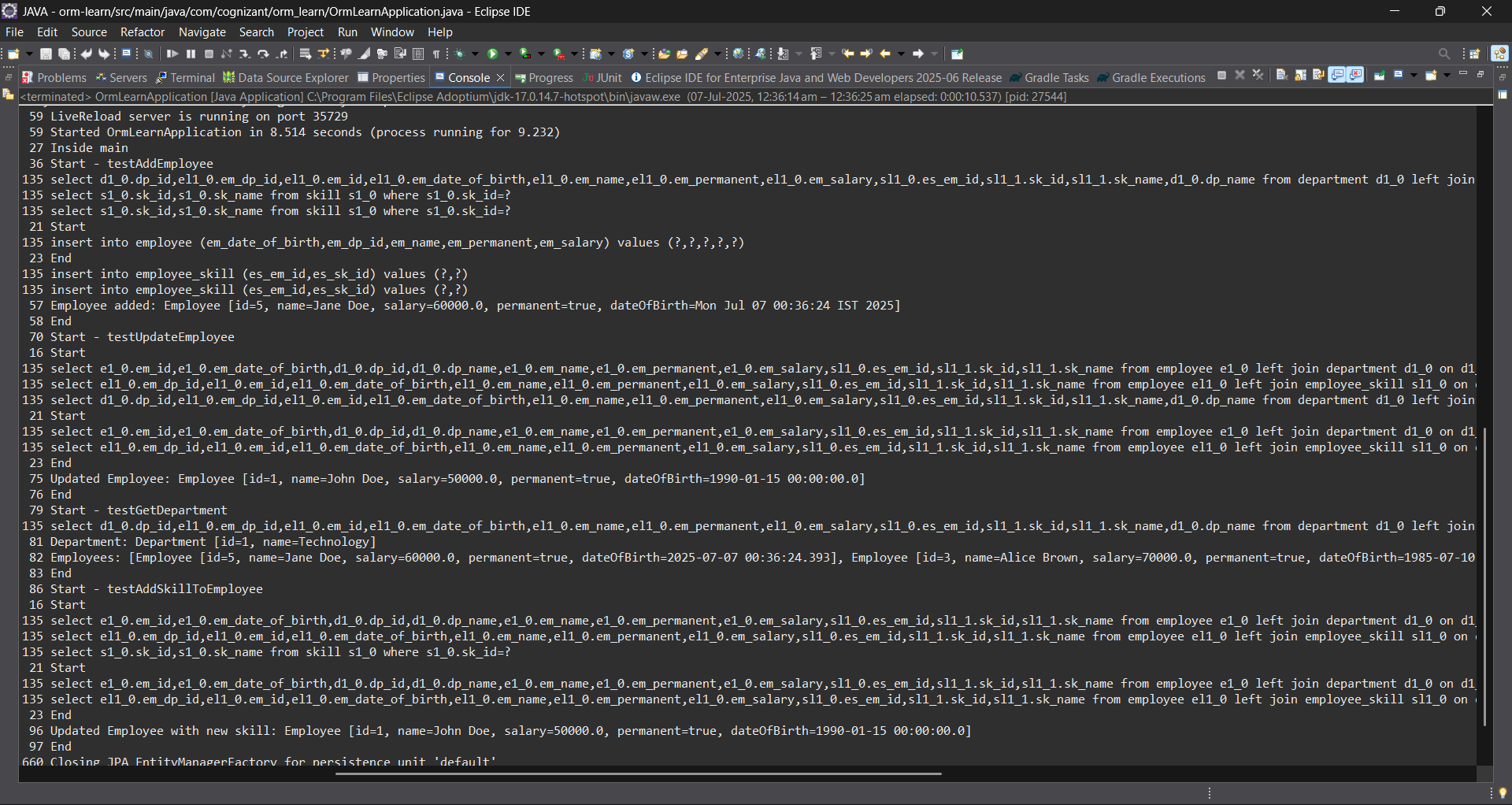
***LOGGER***.debug("Updated Employee with new skill: {}", employee);

***LOGGER***.info("End");

}

}

**Output:**



**Exercise 5: Implement services for managing Country**

**An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.**

* **Find a country based on country code**
* **Add new country**
* **Update country**
* **Delete country**
* **Find list of countries matching a partial country name**

**Solution:**

**Country.java:**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

*@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 + "]";

}

}

**CountryService.java**package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Country;

import java.util.List;

public interface CountryService {

Country findCountryByCode(String code);

void addCountry(Country country);

void updateCountry(Country country);

void deleteCountry(String code);

List<Country> findCountriesByPartialName(String name);

List<Country> getAllCountries(); // ✅ This fixes getAllCountries()

}

**CountryServiceImpl.java**package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.List;

*@Service*

*@Transactional*

public class CountryServiceImpl implements CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Override*

public Country findCountryByCode(String code) {

return countryRepository.findById(code).orElse(null);

}

*@Override*

public void addCountry(Country country) {

countryRepository.save(country);

}

*@Override*

public void updateCountry(Country country) {

Country existing = countryRepository.findById(country.getCode()).orElse(null);

if (existing != null) {

existing.setName(country.getName());

countryRepository.save(existing);

}

}

*@Override*

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

*@Override*

public List<Country> findCountriesByPartialName(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

*@Override*

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

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;

*@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);

*countryService* = context.getBean(CountryService.class);

***LOGGER***.info("Inside main");

// 1. Fetch all countries

*testGetAllCountries*();

// 2. Add a new country

*testAddCountry*();

// 3. Find a country by code

*testFindCountryByCode*();

// 4. Delete a country

*testDeleteCountry*();

// 5. Final country list

*testGetAllCountries*();

// 6. Find countries by partial name

*testFindCountriesByPartialName*();

}

private static void testGetAllCountries() {

***LOGGER***.info("Start - Get All Countries");

List<Country> countries = *countryService*.getAllCountries();

***LOGGER***.debug("countries={}", countries);

***LOGGER***.info("End");

}

private static void testAddCountry() {

***LOGGER***.info("Start - Add Country");

Country country = new Country();

country.setCode("JP");

country.setName("Japan");

*countryService*.addCountry(country);

***LOGGER***.info("Added country: {}", country);

***LOGGER***.info("End");

}

private static void testFindCountryByCode() {

***LOGGER***.info("Start - Find Country by Code");

Country country = *countryService*.findCountryByCode("JP");

***LOGGER***.info("Country found: {}", country);

***LOGGER***.info("End");

}

private static void testDeleteCountry() {

***LOGGER***.info("Start - Delete Country");

*countryService*.deleteCountry("JP");

***LOGGER***.info("Deleted country with code JP");

***LOGGER***.info("End");

}

private static void testFindCountriesByPartialName() {

***LOGGER***.info("Start - Find Countries by Partial Name");

String partialName = "Uni"; // Example input

List<Country> countries = *countryService*.findCountriesByPartialName(partialName);

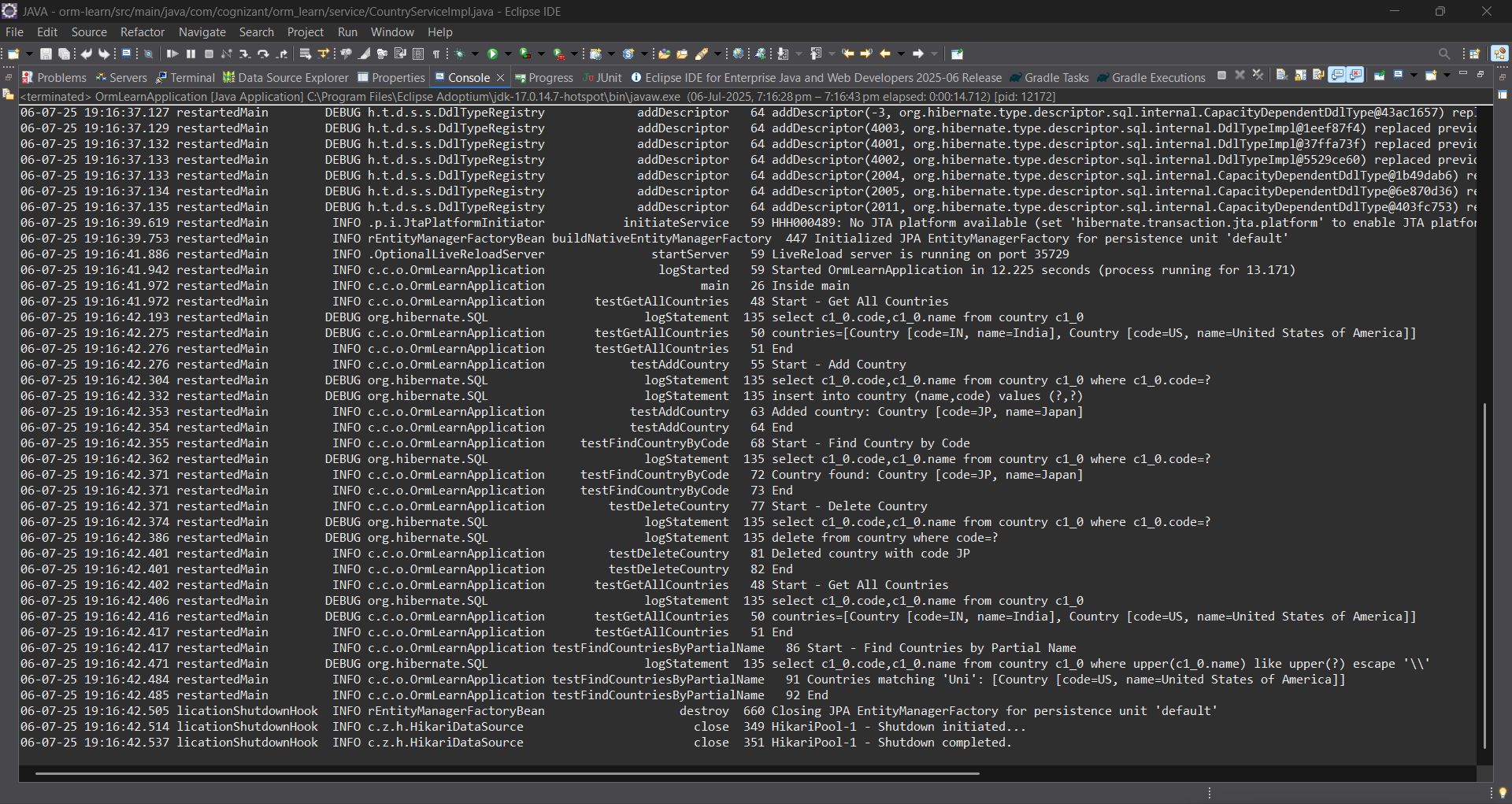
***LOGGER***.info("Countries matching '{}': {}", partialName, countries);

***LOGGER***.info("End");

}

}

**Output:**

****

**Exercise 6: Find a country based on country code**

**Solution:**

**CountryService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;

import java.util.List;

import jakarta.transaction.Transactional;

public interface CountryService {

@Transactional

Country findCountryByCode(String code) throws CountryNotFoundException;

void addCountry(Country country);

void updateCountry(Country country);

void deleteCountry(String code);

List<Country> findCountriesByPartialName(String name);

List<Country> getAllCountries();

}

**CountryServiceImpl.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

import com.cognizant.orm\_learn.service.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

@Transactional

public class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository countryRepository;

@Override

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

@Override

public void addCountry(Country country) {

countryRepository.save(country);

}

@Override

public void updateCountry(Country country) {

Country existing = countryRepository.findById(country.getCode()).orElse(null);

if (existing != null) {

existing.setName(country.getName());

countryRepository.save(existing);

}

}

@Override

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

@Override

public List<Country> findCountriesByPartialName(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

@Override

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;

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;

@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);

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

// 1. Fetch all countries

testGetAllCountries();

// 2. Add a new country

testAddCountry();

// 3. Find a country by code (with exception handling)

testFindCountryByCode();

// 4. Delete a country

testDeleteCountry();

// 5. Final country list

testGetAllCountries();

// 6. Find countries by partial name

testFindCountriesByPartialName();

}

private static void testGetAllCountries() {

LOGGER.info("Start - Get All Countries");

List<Country> countries = countryService.getAllCountries();

LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

private static void testAddCountry() {

LOGGER.info("Start - Add Country");

Country country = new Country();

country.setCode("JP");

country.setName("Japan");

countryService.addCountry(country);

LOGGER.info("Added country: {}", country);

LOGGER.info("End");

}

private static void testFindCountryByCode() {

LOGGER.info("Start - Find Country by Code");

try {

Country country = countryService.findCountryByCode("IN"); // Change "IN" to "JP" or other

LOGGER.info("Country found: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End");

}

private static void testDeleteCountry() {

LOGGER.info("Start - Delete Country");

countryService.deleteCountry("JP");

LOGGER.info("Deleted country with code JP");

LOGGER.info("End");

}

private static void testFindCountriesByPartialName() {

LOGGER.info("Start - Find Countries by Partial Name");

String partialName = "Uni"; // Example input

List<Country> countries = countryService.findCountriesByPartialName(partialName);

LOGGER.info("Countries matching '{}': {}", partialName, countries);

LOGGER.info("End");

}

}

**CountryNotFoundException.java**

package com.cognizant.orm\_learn.service.exception;

public class CountryNotFoundException extends Exception {

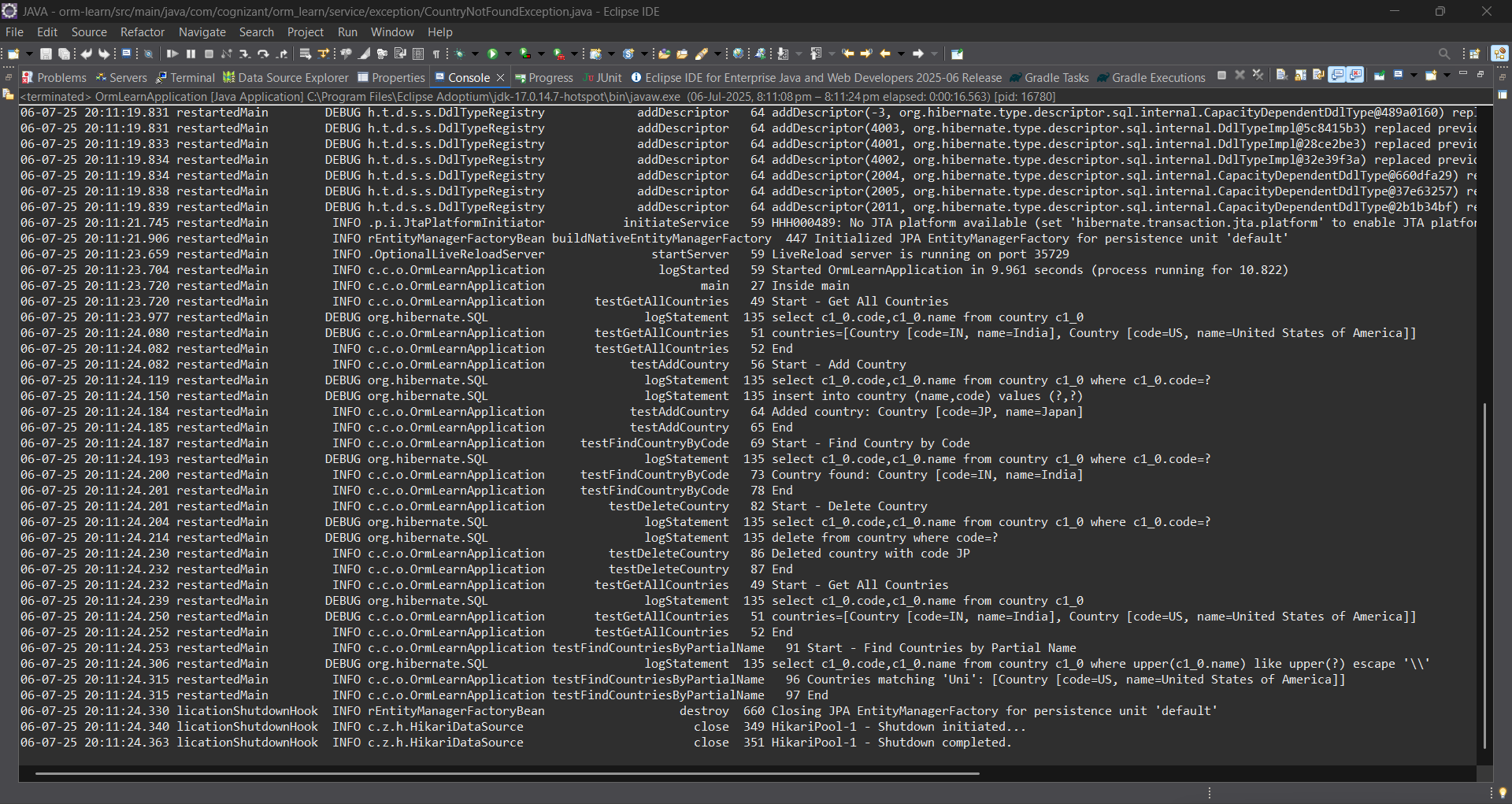
public CountryNotFoundException(String message) {

super(message);

}

}

**Output:**

****

**Exercise 7: Add a new country**

**Solution:**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;

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;

*@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);

*countryService* = context.getBean(CountryService.class);

***LOGGER***.info("Inside main");

// 1. Fetch all countries

*testGetAllCountries*();

// 2. Add a new country

*testAddCountry*();

// 3. Find a country by code (with exception handling)

*testFindCountryByCode*();

// 4. Delete a country

*testDeleteCountry*();

// 5. Final country list

*testGetAllCountries*();

// 6. Find countries by partial name

*testFindCountriesByPartialName*();

}

private static void testGetAllCountries() {

***LOGGER***.info("Start - Get All Countries");

List<Country> countries = *countryService*.getAllCountries();

***LOGGER***.debug("countries={}", countries);

***LOGGER***.info("End");

}

private static void testFindCountryByCode() {

***LOGGER***.info("Start - Find Country by Code");

try {

Country country = *countryService*.findCountryByCode("IN"); // Change "IN" to "JP" or other

***LOGGER***.info("Country found: {}", country);

} catch (CountryNotFoundException e) {

***LOGGER***.error("Exception: {}", e.getMessage());

}

***LOGGER***.info("End");

}

private static void testDeleteCountry() {

***LOGGER***.info("Start - Delete Country");

*countryService*.deleteCountry("JP");

***LOGGER***.info("Deleted country with code JP");

***LOGGER***.info("End");

}

private static void testFindCountriesByPartialName() {

***LOGGER***.info("Start - Find Countries by Partial Name");

String partialName = "Uni"; // Example input

List<Country> countries = *countryService*.findCountriesByPartialName(partialName);

***LOGGER***.info("Countries matching '{}': {}", partialName, countries);

***LOGGER***.info("End");

}

private static void testAddCountry() {

***LOGGER***.info("Start - Add Country");

Country country = new Country();

country.setCode("FR"); // Use a new unique code

country.setName("France");

*countryService*.addCountry(country);

***LOGGER***.info("Added country: {}", country);

try {

Country retrieved = *countryService*.findCountryByCode("FR");

***LOGGER***.info("Country fetched from DB: {}", retrieved);

} catch (Exception e) {

***LOGGER***.error("Country not found after adding: {}", e.getMessage());

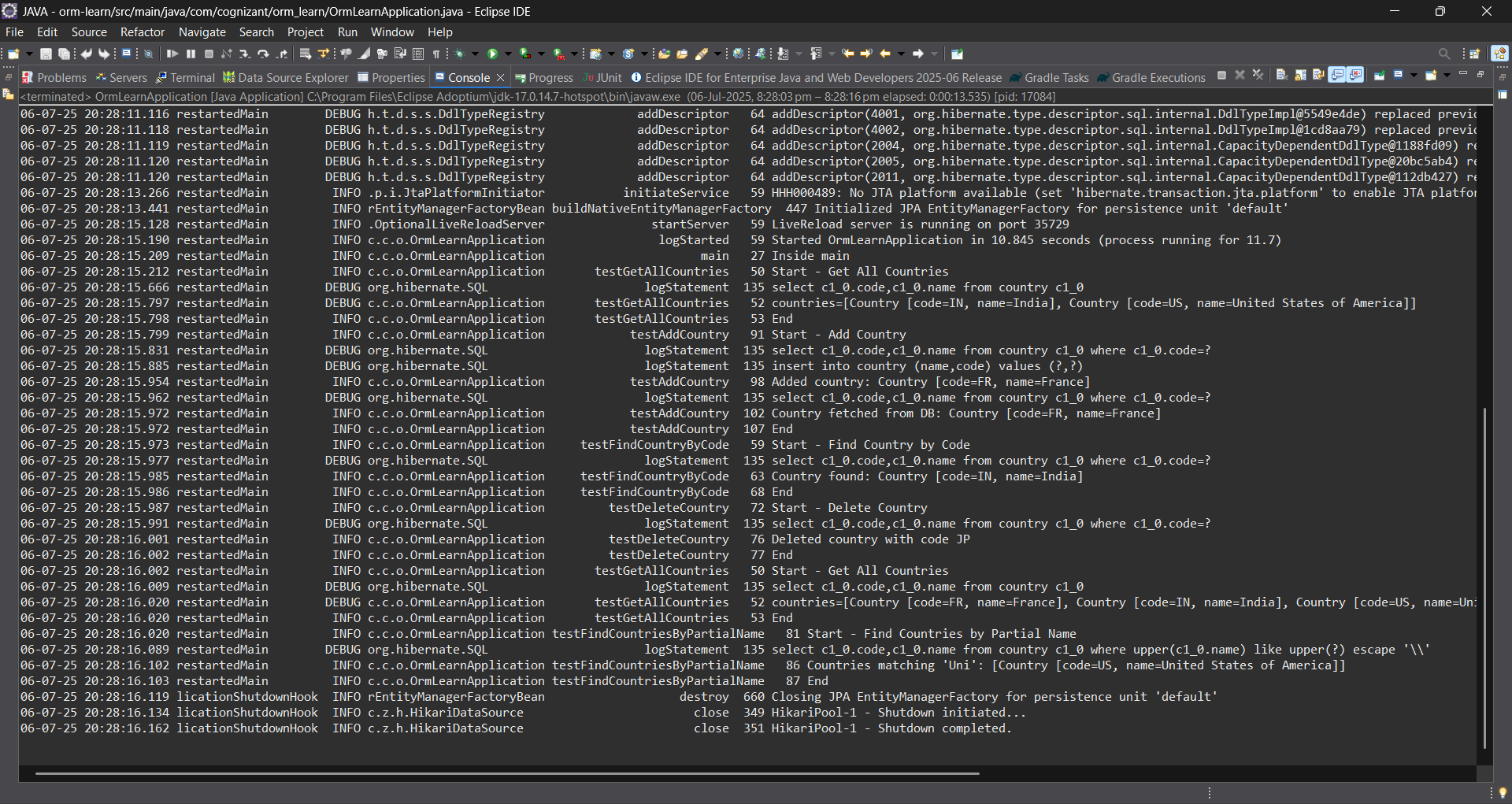
}

***LOGGER***.info("End");

}

}

**Output:**



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

**Exercise 8: Update Country**

* **Add a new method updateCountry() in CountryService and CountryServiceImpl.**
* **Use @Transactional and save() to update the country name.**
* **Fetch the country using findCountryByCode() and update its name.**

**Exercise 9: Delete Country**

* **Use deleteById() in service layer to remove a country.**

**Solution for both:**

**CountryService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;

import java.util.List;

import jakarta.transaction.Transactional;

public interface CountryService {

@Transactional

Country findCountryByCode(String code) throws CountryNotFoundException;

@Transactional

void addCountry(Country country);

@Transactional

void updateCountry(String code, String newName) throws CountryNotFoundException;

@Transactional

void deleteCountry(String code);

List<Country> findCountriesByPartialName(String name);

List<Country> getAllCountries();

}

**CountryServiceImpl.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.repository.CountryRepository;

import com.cognizant.orm\_learn.service.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

@Transactional

public class CountryServiceImpl implements CountryService {

@Autowired

private CountryRepository countryRepository;

@Override

public Country findCountryByCode(String code) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

@Override

public void addCountry(Country country) {

countryRepository.save(country);

}

@Override

public void updateCountry(String code, String newName) throws CountryNotFoundException {

Country existing = findCountryByCode(code); // reuse with exception handling

existing.setName(newName);

countryRepository.save(existing);

}

@Override

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

@Override

public List<Country> findCountriesByPartialName(String name) {

return countryRepository.findByNameContainingIgnoreCase(name);

}

@Override

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Country;

import com.cognizant.orm\_learn.service.CountryService;

import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;

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;

@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);

countryService = context.getBean(CountryService.class);

LOGGER.info("Inside main");

testAddCountry();

testUpdateCountry();

testFindCountryByCode();

testDeleteCountry();

testGetAllCountries();

testFindCountriesByPartialName();

}

private static void testAddCountry() {

LOGGER.info("Start - Add Country");

Country country = new Country();

country.setCode("DE");

country.setName("Germany");

countryService.addCountry(country);

LOGGER.info("Added country: {}", country);

LOGGER.info("End");

}

private static void testUpdateCountry() {

LOGGER.info("Start - Update Country");

try {

countryService.updateCountry("DE", "Deutschland");

LOGGER.info("Updated country name for DE to Deutschland");

} catch (CountryNotFoundException e) {

LOGGER.error("Error updating country: {}", e.getMessage());

}

LOGGER.info("End");

}

private static void testFindCountryByCode() {

LOGGER.info("Start - Find Country by Code");

try {

Country country = countryService.findCountryByCode("DE");

LOGGER.info("Country found: {}", country);

} catch (CountryNotFoundException e) {

LOGGER.error("Exception: {}", e.getMessage());

}

LOGGER.info("End");

}

private static void testDeleteCountry() {

LOGGER.info("Start - Delete Country");

countryService.deleteCountry("DE");

LOGGER.info("Deleted country with code DE");

LOGGER.info("End");

}

private static void testGetAllCountries() {

LOGGER.info("Start - Get All Countries");

List<Country> countries = countryService.getAllCountries();

LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

private static void testFindCountriesByPartialName() {

LOGGER.info("Start - Find Countries by Partial Name");

List<Country> countries = countryService.findCountriesByPartialName("Uni");

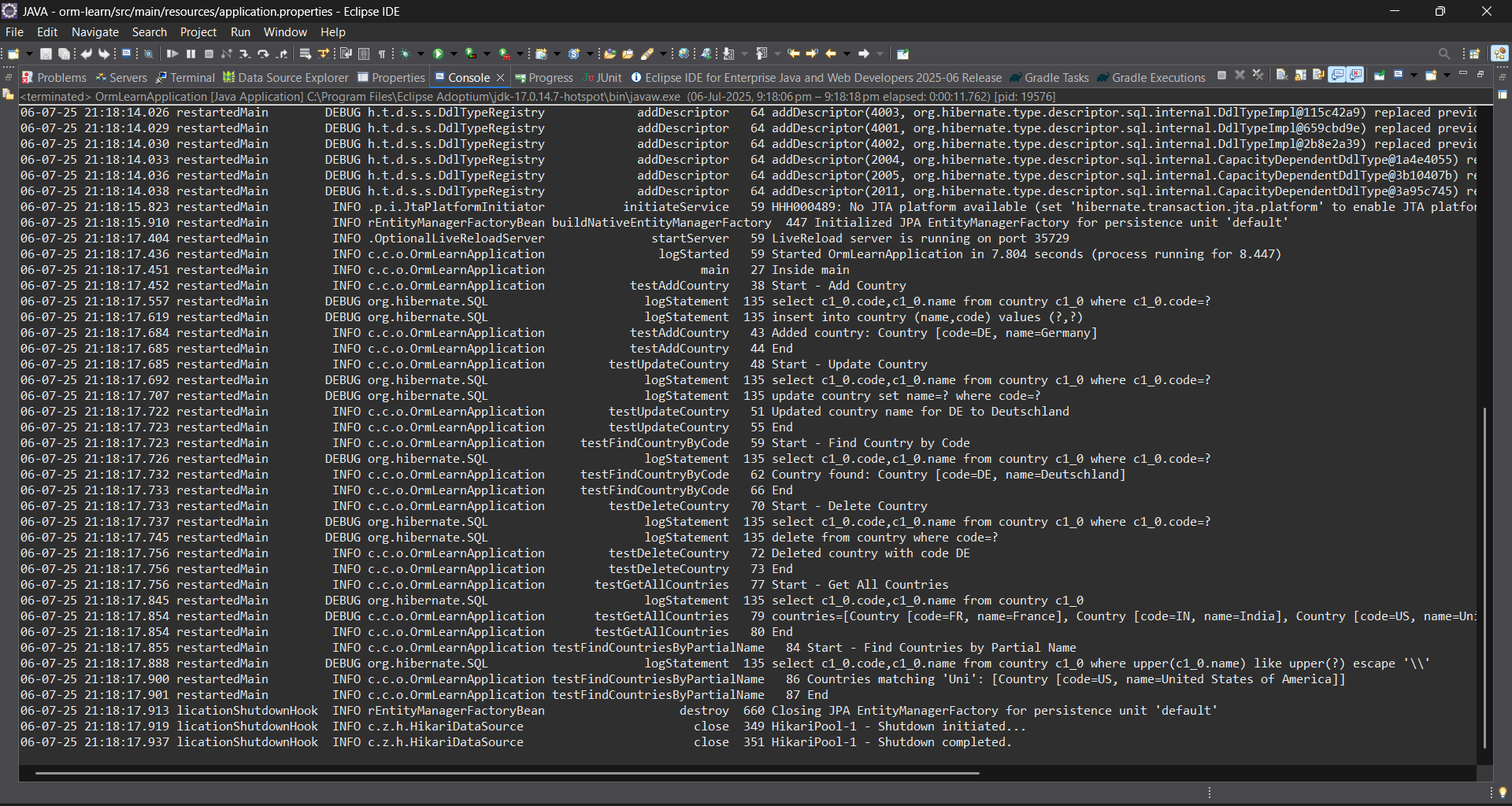
LOGGER.info("Countries matching 'Uni': {}", countries);

LOGGER.info("End");

}

}

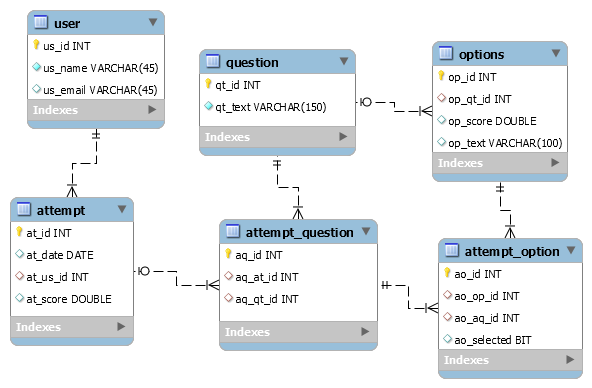
**Output:**

****

**Demonstrate implementation of O/R Mapping**

**Exercise 3: Fetch quiz attempt details using HQL**

* **Username**
* **Attempted Date**
* **All questions as part of the attempt**
* **List of options under each quiz**
* **The option that is correct answer**
* **The score for correct answer**



**Exercise 4: Get average salary using HQL**

**Compute the average salary of a department using HQL.**

**Solution:**

**Employee.java**package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.Set;

*@Entity*

*@Table*(name = "employee")

public class Employee {

*@Id*

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

*@Column*(name = "em\_id")

private int id;

*@Column*(name = "em\_name")

private String name;

*@Column*(name = "em\_salary")

private double salary;

*@Column*(name = "em\_permanent")

private boolean permanent;

*@Column*(name = "em\_date\_of\_birth")

private Date dateOfBirth;

*@ManyToOne*

*@JoinColumn*(name = "em\_dp\_id")

private Department department;

*@ManyToMany*(fetch = *FetchType*.***EAGER***)

*@JoinTable*(

name = "employee\_skill",

joinColumns = *@JoinColumn*(name = "es\_em\_id"),

inverseJoinColumns = *@JoinColumn*(name = "es\_sk\_id")

)

private Set<Skill> skillList;

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 double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

public Department getDepartment() { return department; }

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

public Set<Skill> getSkillList() { return skillList; }

public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

*@Override*

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +

", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth + "]";

}

}

**Department.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

*@Entity*

*@Table*(name = "department")

public class Department {

*@Id*

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

*@Column*(name = "dp\_id")

private int id;

*@Column*(name = "dp\_name")

private String name;

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

private Set<Employee> employeeList;

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 Set<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(Set<Employee> employeeList) {

this.employeeList = employeeList;

}

*@Override*

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

**Skill.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

*@Entity*

*@Table*(name = "skill")

public class Skill {

*@Id*

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

*@Column*(name = "sk\_id")

private int id;

*@Column*(name = "sk\_name")

private String name;

*@ManyToMany*(mappedBy = "skillList")

private Set<Employee> employeeList;

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 Set<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(Set<Employee> employeeList) {

this.employeeList = employeeList;

}

*@Override*

public String toString() {

return "Skill [id=" + id + ", name=" + name + "]";

}

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

**DepartmentRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {}

**SkillRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {}

**EmployeeService.java**package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Employee;

import com.cognizant.orm\_learn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@Service*

public class EmployeeService {

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

*@Autowired*

private EmployeeRepository employeeRepository;

*@Transactional*

public Employee get(int id) {

***LOGGER***.info("Start");

return employeeRepository.findById(id).get();

}

*@Transactional*

public void save(Employee employee) {

***LOGGER***.info("Start");

employeeRepository.save(employee);

***LOGGER***.info("End");

}

}

**DepartmentService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.repository.DepartmentRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class DepartmentService {

*@Autowired*

private DepartmentRepository departmentRepository;

*@Transactional*

public Department get(int id) {

return departmentRepository.findById(id).get();

}

*@Transactional*

public void save(Department department) {

departmentRepository.save(department);

}

}

**SkillService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Skill;

import com.cognizant.orm\_learn.repository.SkillRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class SkillService {

*@Autowired*

private SkillRepository skillRepository;

*@Transactional*

public Skill get(int id) {

return skillRepository.findById(id).get();

}

*@Transactional*

public void save(Skill skill) {

skillRepository.save(skill);

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.model.Employee;

import com.cognizant.orm\_learn.model.Skill;

import com.cognizant.orm\_learn.service.DepartmentService;

import com.cognizant.orm\_learn.service.EmployeeService;

import com.cognizant.orm\_learn.service.SkillService;

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 java.util.Date;

import java.util.HashSet;

import java.util.Set;

*@SpringBootApplication*

public class OrmLearnApplication {

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

private static EmployeeService *employeeService*;

private static DepartmentService *departmentService*;

private static SkillService *skillService*;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);

*employeeService* = context.getBean(EmployeeService.class);

*departmentService* = context.getBean(DepartmentService.class);

*skillService* = context.getBean(SkillService.class);

***LOGGER***.info("Inside main");

// Uncomment the methods below one by one to test

*testAddEmployee*();

//testGetEmployee();

*testUpdateEmployee*();

*testGetDepartment*();

*testAddSkillToEmployee*();

}

private static void testAddEmployee() {

***LOGGER***.info("Start - testAddEmployee");

Employee employee = new Employee();

employee.setName("John Doe");

employee.setSalary(50000.0);

employee.setPermanent(true);

employee.setDateOfBirth(new Date());

Department department = *departmentService*.get(1); // ensure this ID exists

employee.setDepartment(department);

*employeeService*.save(employee);

***LOGGER***.debug("Employee: {}", employee);

***LOGGER***.info("End");

}

private static void testGetEmployee() {

***LOGGER***.info("Start - testGetEmployee");

Employee employee = *employeeService*.get(1); // use valid employee id

***LOGGER***.debug("Employee: {}", employee);

***LOGGER***.debug("Department: {}", employee.getDepartment());

***LOGGER***.debug("Skills: {}", employee.getSkillList());

***LOGGER***.info("End");

}

private static void testUpdateEmployee() {

***LOGGER***.info("Start - testUpdateEmployee");

Employee employee = *employeeService*.get(1); // use valid employee id

Department newDept = *departmentService*.get(2); // different department ID

employee.setDepartment(newDept);

*employeeService*.save(employee);

***LOGGER***.debug("Updated Employee: {}", employee);

***LOGGER***.info("End");

}

private static void testGetDepartment() {

***LOGGER***.info("Start - testGetDepartment");

Department department = *departmentService*.get(1); // department with multiple employees

***LOGGER***.debug("Department: {}", department);

***LOGGER***.debug("Employees: {}", department.getEmployeeList());

***LOGGER***.info("End");

}

private static void testAddSkillToEmployee() {

***LOGGER***.info("Start - testAddSkillToEmployee");

Employee employee = *employeeService*.get(1); // valid employee id

Skill skill = *skillService*.get(2); // valid skill id not already assigned

Set<Skill> skillList = employee.getSkillList();

if (skillList == null) {

skillList = new HashSet<>();

}

skillList.add(skill);

employee.setSkillList(skillList);

*employeeService*.save(employee);

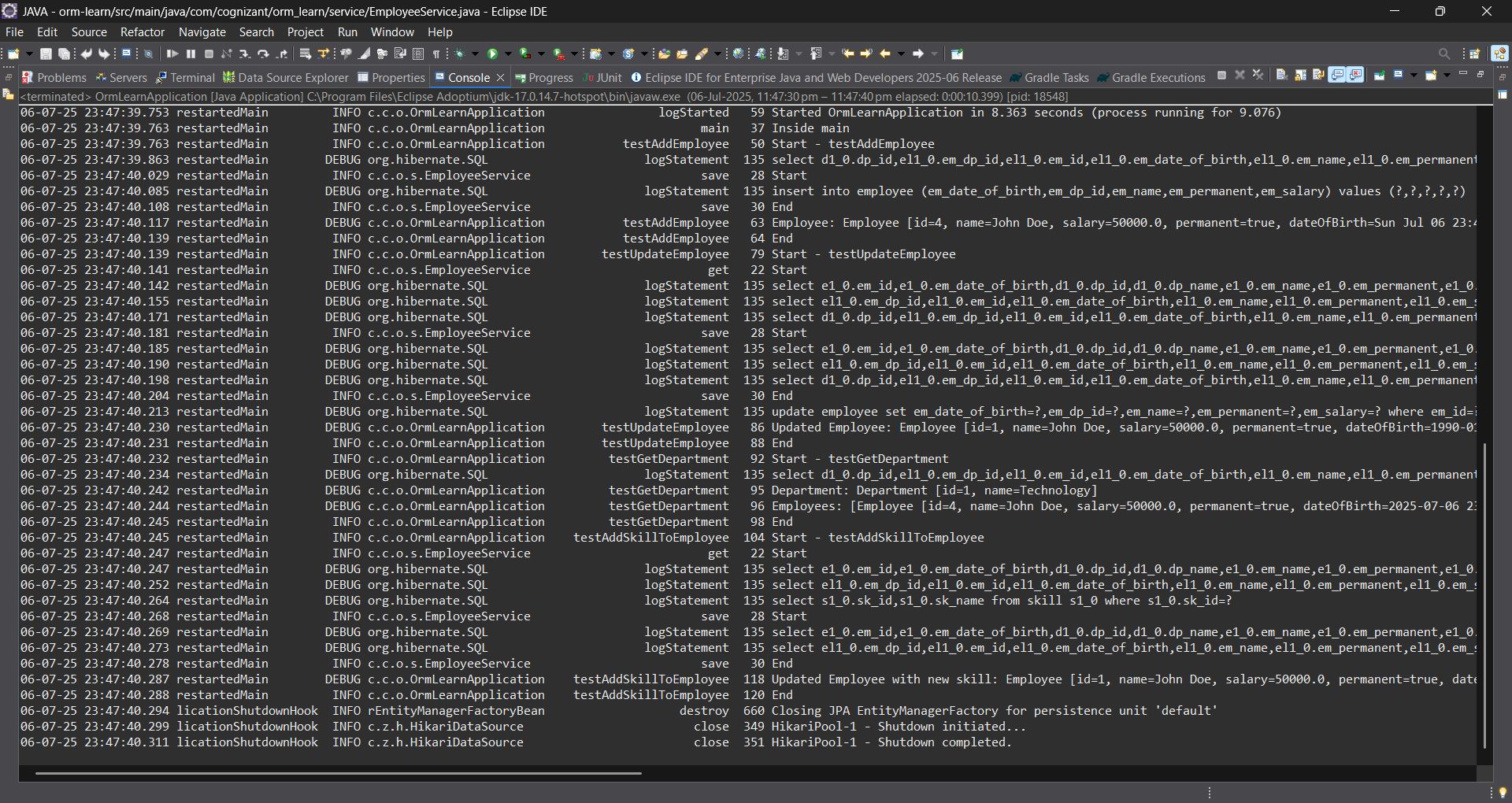
***LOGGER***.debug("Updated Employee with new skill: {}", employee);

***LOGGER***.info("End");

}

}

**Output:**

****

**Demonstrate writing Hibernate Query Language and Native Query**

**Exercise 4: Get average salary using HQL**

**Compute the average salary of a department using HQL.**

**Exercise 5: Get all employees using Native Query**

**Solution for Both:**

**Employee.java**package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.Set;

*@Entity*

*@Table*(name = "employee")

public class Employee {

*@Id*

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

*@Column*(name = "em\_id")

private int id;

*@Column*(name = "em\_name")

private String name;

*@Column*(name = "em\_salary")

private double salary;

*@Column*(name = "em\_permanent")

private boolean permanent;

*@Column*(name = "em\_date\_of\_birth")

private Date dateOfBirth;

*@ManyToOne*

*@JoinColumn*(name = "em\_dp\_id")

private Department department;

*@ManyToMany*(fetch = *FetchType*.***EAGER***)

*@JoinTable*(

name = "employee\_skill",

joinColumns = *@JoinColumn*(name = "es\_em\_id"),

inverseJoinColumns = *@JoinColumn*(name = "es\_sk\_id")

)

private Set<Skill> skillList;

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 double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth = dateOfBirth; }

public Department getDepartment() { return department; }

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

public Set<Skill> getSkillList() { return skillList; }

public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

*@Override*

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +

", permanent=" + permanent + ", dateOfBirth=" + dateOfBirth + "]";

}

}

**Department.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

*@Entity*

*@Table*(name = "department")

public class Department {

*@Id*

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

*@Column*(name = "dp\_id")

private int id;

*@Column*(name = "dp\_name")

private String name;

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

private Set<Employee> employeeList;

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 Set<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(Set<Employee> employeeList) {

this.employeeList = employeeList;

}

*@Override*

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

**Skill.java**

package com.cognizant.orm\_learn.model;

import jakarta.persistence.\*;

import java.util.Set;

*@Entity*

*@Table*(name = "skill")

public class Skill {

*@Id*

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

*@Column*(name = "sk\_id")

private int id;

*@Column*(name = "sk\_name")

private String name;

*@ManyToMany*(mappedBy = "skillList")

private Set<Employee> employeeList;

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 Set<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(Set<Employee> employeeList) {

this.employeeList = employeeList;

}

*@Override*

public String toString() {

return "Skill [id=" + id + ", name=" + name + "]";

}

}

**EmployeeRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

**DepartmentRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {}

**SkillRepository.java**

package com.cognizant.orm\_learn.repository;

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

import com.cognizant.orm\_learn.model.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {}

**EmployeeService.java**package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Employee;

import com.cognizant.orm\_learn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*@Service*

public class EmployeeService {

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

*@Autowired*

private EmployeeRepository employeeRepository;

*@Transactional*

public Employee get(int id) {

***LOGGER***.info("Start");

return employeeRepository.findById(id).get();

}

*@Transactional*

public void save(Employee employee) {

***LOGGER***.info("Start");

employeeRepository.save(employee);

***LOGGER***.info("End");

}

}

**DepartmentService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.repository.DepartmentRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class DepartmentService {

*@Autowired*

private DepartmentRepository departmentRepository;

*@Transactional*

public Department get(int id) {

return departmentRepository.findById(id).get();

}

*@Transactional*

public void save(Department department) {

departmentRepository.save(department);

}

}

**SkillService.java**

package com.cognizant.orm\_learn.service;

import com.cognizant.orm\_learn.model.Skill;

import com.cognizant.orm\_learn.repository.SkillRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

*@Service*

public class SkillService {

*@Autowired*

private SkillRepository skillRepository;

*@Transactional*

public Skill get(int id) {

return skillRepository.findById(id).get();

}

*@Transactional*

public void save(Skill skill) {

skillRepository.save(skill);

}

}

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.model.Department;

import com.cognizant.orm\_learn.model.Employee;

import com.cognizant.orm\_learn.model.Skill;

import com.cognizant.orm\_learn.service.DepartmentService;

import com.cognizant.orm\_learn.service.EmployeeService;

import com.cognizant.orm\_learn.service.SkillService;

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 java.util.Date;

import java.util.HashSet;

import java.util.Set;

*@SpringBootApplication*

public class OrmLearnApplication {

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

private static EmployeeService *employeeService*;

private static DepartmentService *departmentService*;

private static SkillService *skillService*;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);

*employeeService* = context.getBean(EmployeeService.class);

*departmentService* = context.getBean(DepartmentService.class);

*skillService* = context.getBean(SkillService.class);

***LOGGER***.info("Inside main");

// Uncomment the methods below one by one to test

//*testAddEmployee*();

testGetEmployee();

//*testUpdateEmployee*();

//*testGetDepartment*();

//*testAddSkillToEmployee*();

}

private static void testAddEmployee() {

***LOGGER***.info("Start - testAddEmployee");

Employee employee = new Employee();

employee.setName("John Doe");

employee.setSalary(50000.0);

employee.setPermanent(true);

employee.setDateOfBirth(new Date());

Department department = *departmentService*.get(1); // ensure this ID exists

employee.setDepartment(department);

*employeeService*.save(employee);

***LOGGER***.debug("Employee: {}", employee);

***LOGGER***.info("End");

}

private static void testGetEmployee() {

***LOGGER***.info("Start - testGetEmployee");

Employee employee = *employeeService*.get(1); // use valid employee id

***LOGGER***.debug("Employee: {}", employee);

***LOGGER***.debug("Department: {}", employee.getDepartment());

***LOGGER***.debug("Skills: {}", employee.getSkillList());

***LOGGER***.info("End");

}

private static void testUpdateEmployee() {

***LOGGER***.info("Start - testUpdateEmployee");

Employee employee = *employeeService*.get(1); // use valid employee id

Department newDept = *departmentService*.get(2); // different department ID

employee.setDepartment(newDept);

*employeeService*.save(employee);

***LOGGER***.debug("Updated Employee: {}", employee);

***LOGGER***.info("End");

}

private static void testGetDepartment() {

***LOGGER***.info("Start - testGetDepartment");

Department department = *departmentService*.get(1); // department with multiple employees

***LOGGER***.debug("Department: {}", department);

***LOGGER***.debug("Employees: {}", department.getEmployeeList());

***LOGGER***.info("End");

}

private static void testAddSkillToEmployee() {

***LOGGER***.info("Start - testAddSkillToEmployee");

Employee employee = *employeeService*.get(1); // valid employee id

Skill skill = *skillService*.get(2); // valid skill id not already assigned

Set<Skill> skillList = employee.getSkillList();

if (skillList == null) {

skillList = new HashSet<>();

}

skillList.add(skill);

employee.setSkillList(skillList);

*employeeService*.save(employee);

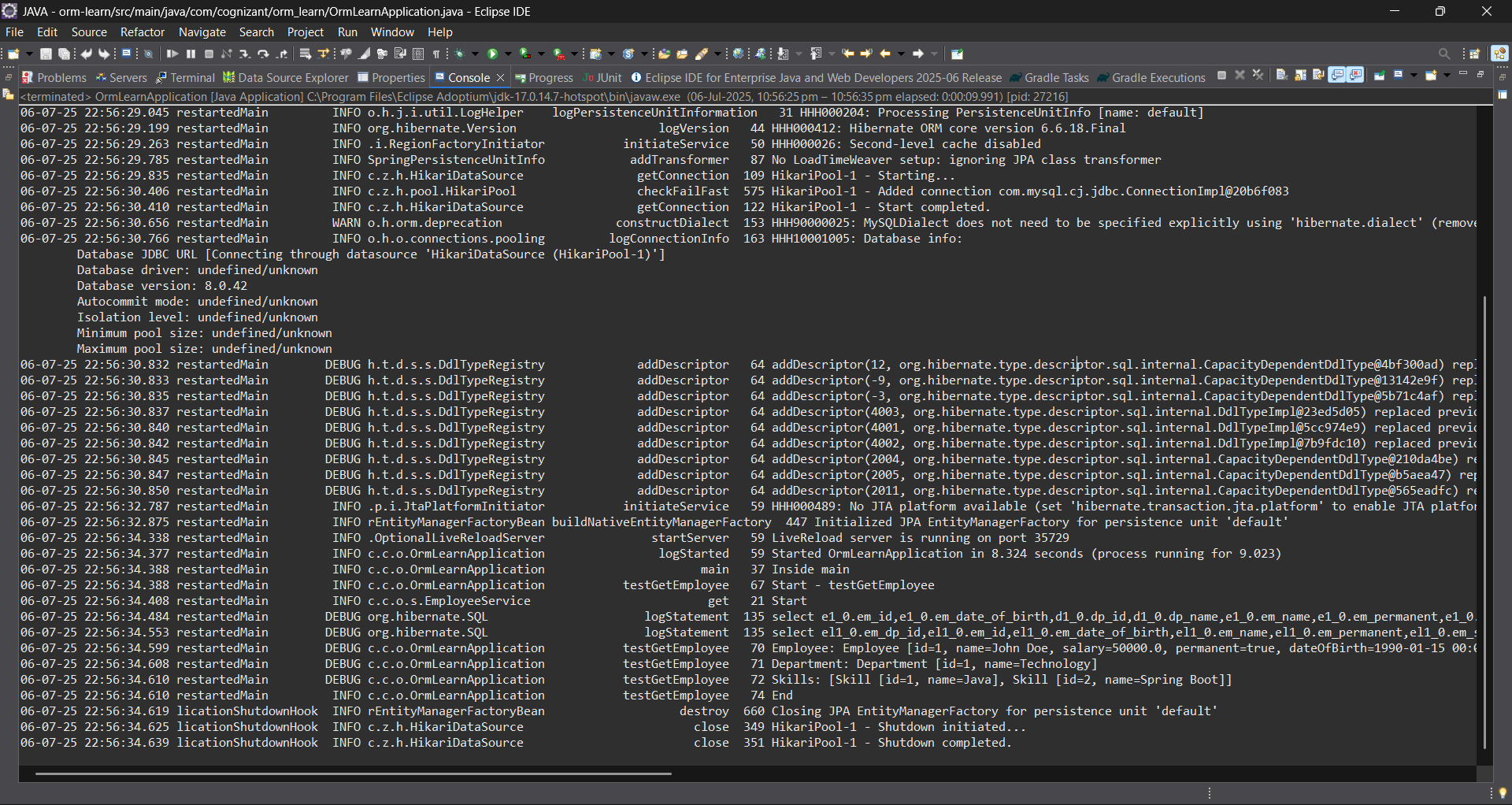
***LOGGER***.debug("Updated Employee with new skill: {}", employee);

***LOGGER***.info("End");

}

}

**Output:**

****