**Digital Nurture 4.0 Deep Skilling - Java FSE**  
**WEEK –3 Additional Hands-on Exercises**  
**Module 6 - Spring Data JPA with Spring Boot, Hibernate**

1. **spring-data-jpa-handson**

**1. Hands on 5:Implement services for managing Country**

**Solution:**

**Code:**

**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.example</groupId>

<artifactId>ormlearn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>ormlearn</name>

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

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.mysql</groupId>

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

<version>8.0.33</version>

<scope>runtime</scope>

</dependency>

<dependency>

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

<artifactId>spring-boot-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

</plugin>

</plugins>

</build>

</project>

**application.properties**

spring.application.name=ormlearn

# Logging

logging.level.org.springframework=info

logging.level.com.example.ormlearn=debug

logging.level.org.hibernate.SQL=debug

logging.level.org.hibernate.type.descriptor.sql=trace

# Database

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=Shri19

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

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

**Country.java**

package com.example.ormlearn.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")

private String code;

@Column(name = "co\_name")

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = 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; }

}

**CountryRepository.java**

package com.example.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.example.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.example.ormlearn.service;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.repository.CountryRepository;

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 Country findCountryByCode(String code) throws Exception {

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

if (result.isEmpty()) throw new Exception("Country Not Found");

return result.get();

}

public void addCountry(Country country) {

countryRepository.save(country);

}

public void updateCountry(Country country) throws Exception {

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

throw new Exception("Country Not Found");

}

countryRepository.save(country);

}

public void deleteCountry(String code) throws Exception {

if (!countryRepository.existsById(code)) {

throw new Exception("Country Not Found");

}

countryRepository.deleteById(code);

}

public List<Country> searchCountriesByPartialName(String partialName) {

return countryRepository.findByNameContaining(partialName);

}

}

**OrmlearnApllication.java**

package com.example.ormlearn;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.service.CountryService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

@Override

public void run(String... args) throws Exception {

System.out.println("Find by code: " +

countryService.findCountryByCode("IN").getName());

Country newCountry = new Country("ZZ", "Zootopia");

countryService.addCountry(newCountry);

newCountry.setName("Zootopia Updated");

countryService.updateCountry(newCountry);

countryService.searchCountriesByPartialName("Uni")

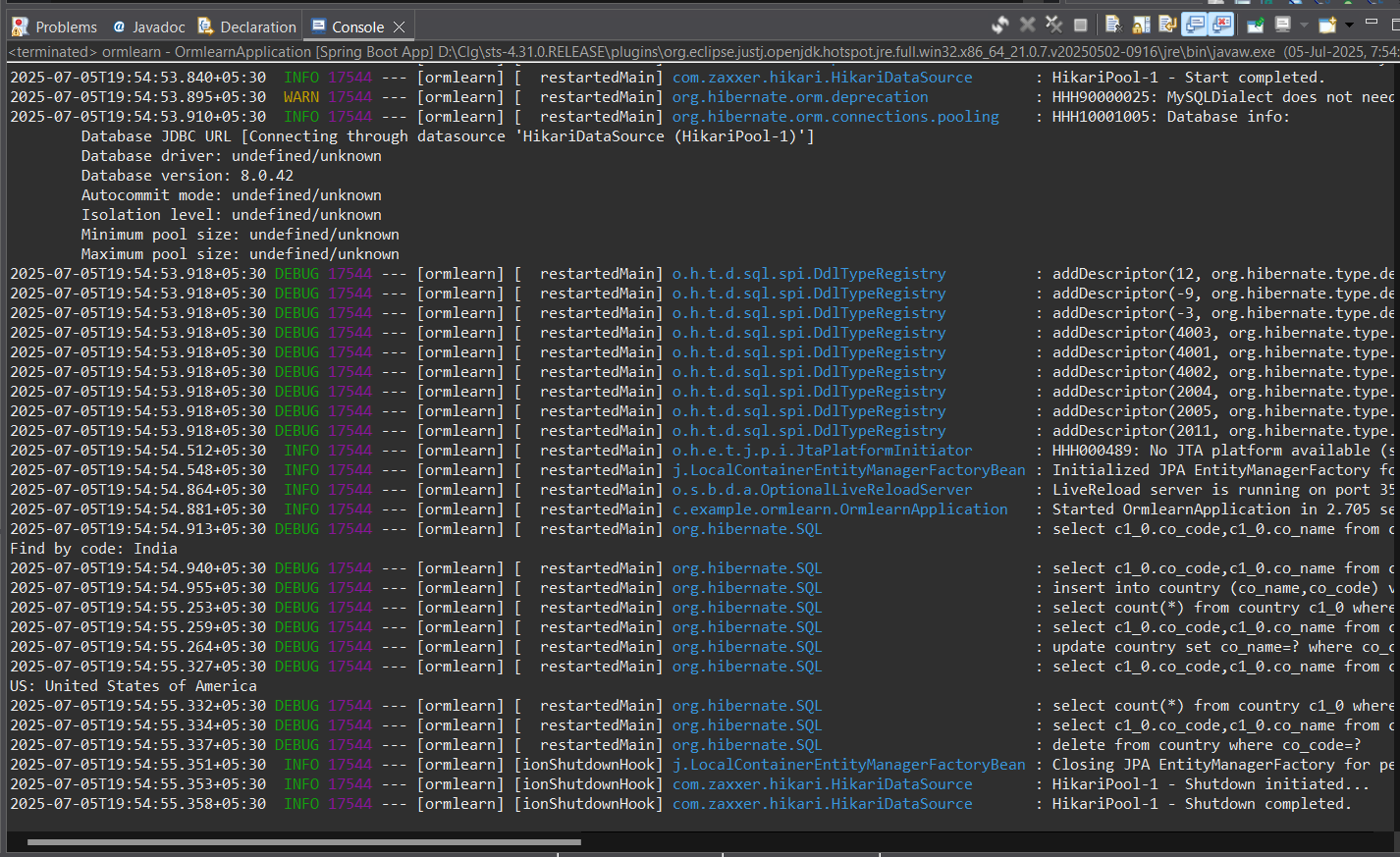
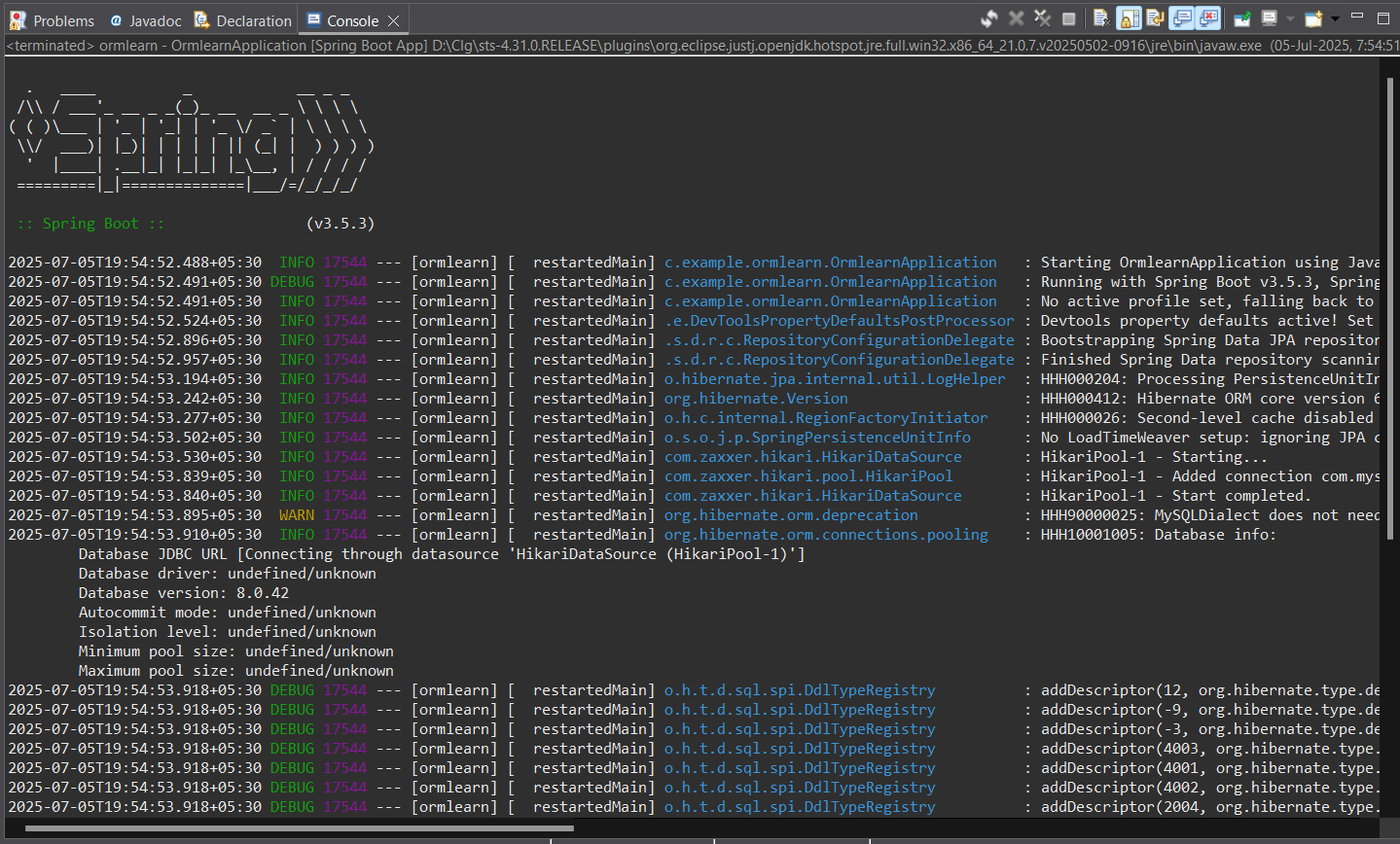
.forEach(c -> System.out.println(c.getCode() + ": " + c.getName()));

countryService.deleteCountry("ZZ");

}

}

**Output:**



**Explanation:**

1. Set spring.jpa.hibernate.ddl-auto=validate in application.properties.
2. Populate country table using the given INSERT SQL script.
3. Create Country entity with annotations: @Entity, @Table, @Id, @Column.
4. Create CountryRepository interface extending JpaRepository<Country, String>.
5. Define method in CountryRepository:  
    List<Country> findByNameContainingIgnoreCase(String name);
6. Create CountryService class with @Service annotation.
7. Autowire CountryRepository in CountryService.
8. Implement methods in CountryService:

* getCountry(String code)
* addCountry(Country country)
* updateCountry(Country country)
* deleteCountry(String code)
* findByNameContaining(String partialName)

1. Autowire CountryService in OrmLearnApplication.
2. Create test methods in OrmLearnApplication to call service methods.
3. Run the application and verify logs for output.

**2. Hands on 6:Find a country based on country code**

**Solution:**

**Code:**

**CountryService.java**

package com.example.ormlearn.service;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.repository.CountryRepository;

import com.example.ormlearn.service.exception.CountryNotFoundException;

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

import org.springframework.stereotype.Service;

import javax.transaction.Transactional;

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();

}

}

**CountryNotFoundException.java**

package com.cognizant.ormlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}

**OrmlearnApllication.java**

package com.example.ormlearn;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.service.CountryService;

import com.example.ormlearn.service.exception.CountryNotFoundException;

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) throws CountryNotFoundException {

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

countryService = context.getBean(CountryService.class);

getCountryTest();

}

private static void getCountryTest() throws CountryNotFoundException {

LOGGER.info("Start");

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

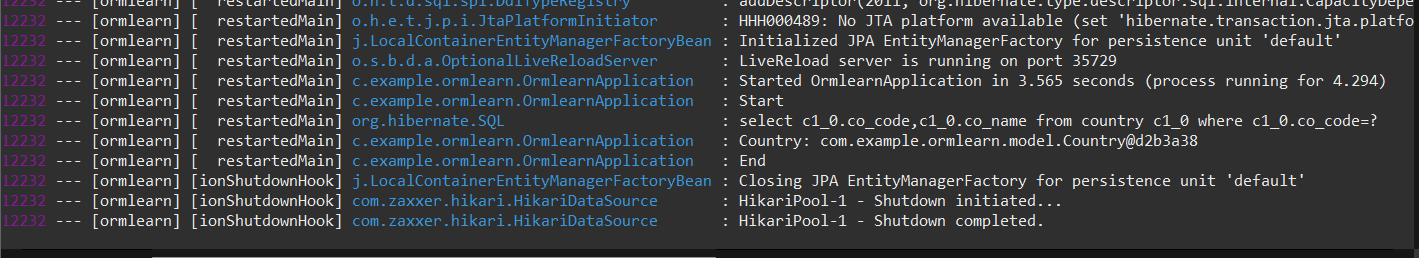
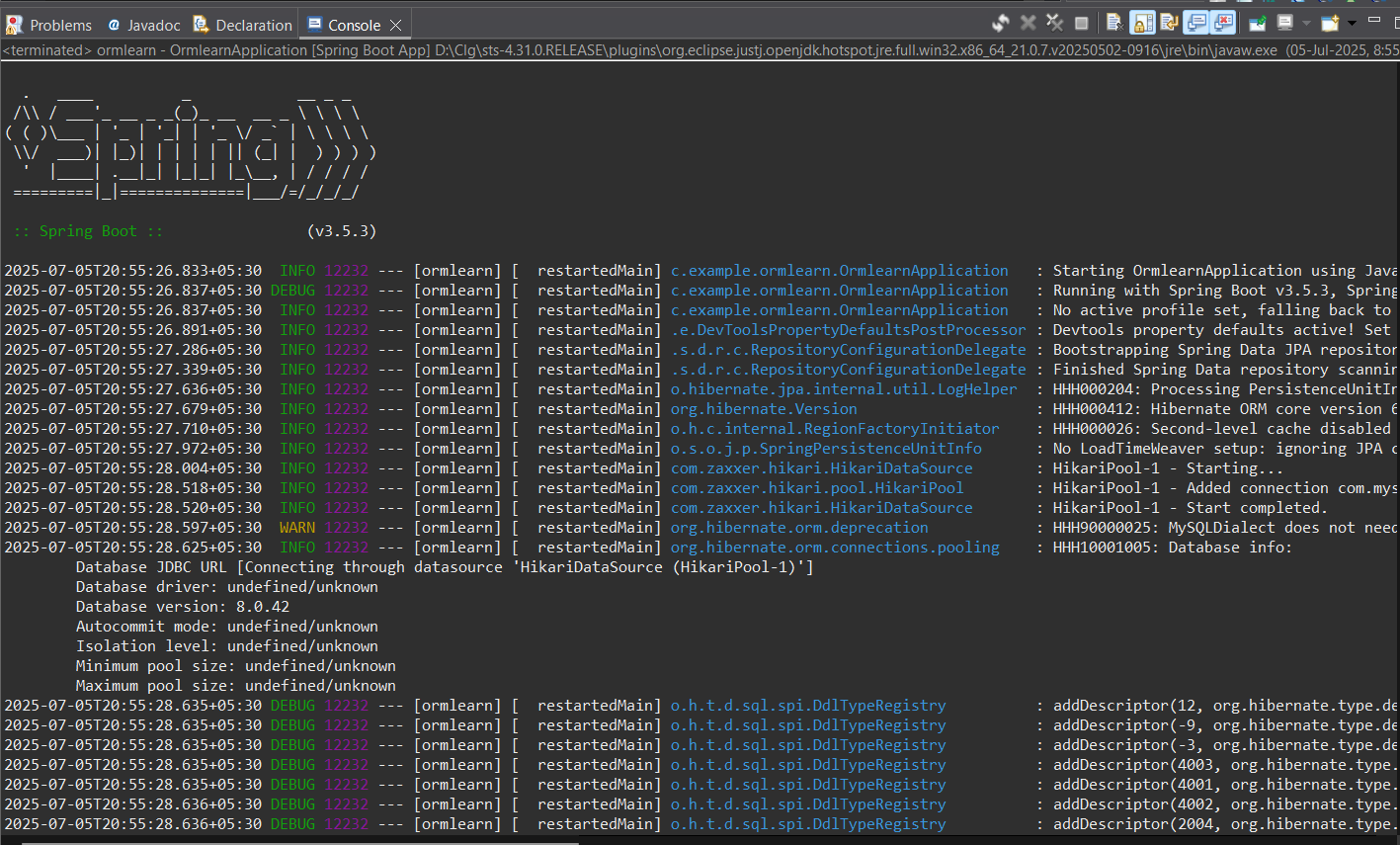
LOGGER.debug("Country: {}", country);

LOGGER.info("End");

}

}

**Output:**



**Explanation:**

1. Create CountryNotFoundException class in com.cognizant.spring-learn.service.exception.
2. In CountryService, create method findCountryByCode() with @Transactional annotation.
3. Inside findCountryByCode(String countryCode):

Use countryRepository.findById(countryCode) to get Optional<Country>.

If not present, throw CountryNotFoundException.

Else, return result.get().

1. In OrmLearnApplication, define method getAllCountriesTest():

Call findCountryByCode("IN").

Log the returned Country object.

1. In main() method:

Call getAllCountriesTest() after setting up application context and service bean.

1. Run and verify output in console logs.

**3. Hands on 7: Add a new country**

**Solution:**

**Code:**

**CountryService.java**

package com.example.ormlearn.service;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.repository.CountryRepository;

import com.example.ormlearn.service.exception.CountryNotFoundException;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String code) throws CountryNotFoundException {

return countryRepository.findById(code)

.orElseThrow(() -> new CountryNotFoundException("Country not found for code: " + code));

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

}

**OrmlearnApllication.java**

package com.example.ormlearn;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.service.CountryService;

import com.example.ormlearn.service.exception.CountryNotFoundException;

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) throws CountryNotFoundException {

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

countryService = context.getBean(CountryService.class);

testAddCountry();

}

private static void getCountryTest() throws CountryNotFoundException {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

private static void testAddCountry() throws CountryNotFoundException {

LOGGER.info("Start");

Country newCountry = new Country();

newCountry.setCode("ZZ");

newCountry.setName("Zootopia");

countryService.addCountry(newCountry);

Country addedCountry = countryService.findCountryByCode("ZZ");

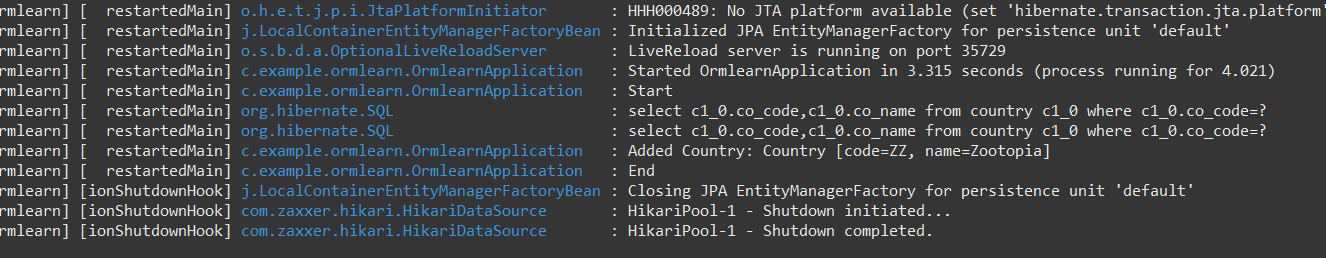
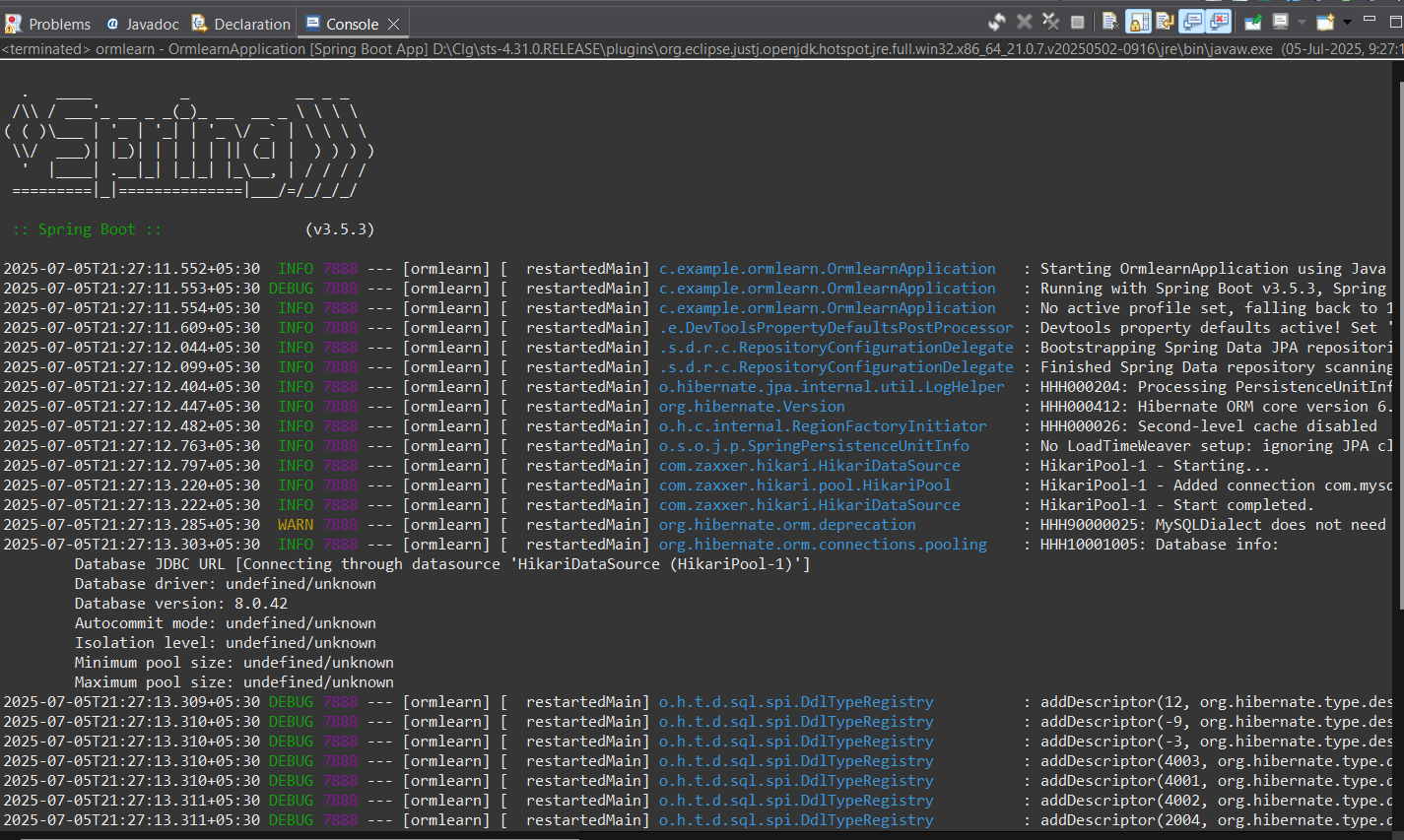
LOGGER.debug("Added Country: {}", addedCountry);

LOGGER.info("End");

}

}

**Output:**



**Explanation:**

1. In CountryService, create method addCountry() with @Transactional annotation.
2. Inside addCountry(Country country), call countryRepository.save(country).
3. In OrmLearnApplication, define testAddCountry() method:

Create a new Country object with code and name.

Call countryService.addCountry(country).

Call countryService.findCountryByCode(code) to verify.

1. In main() method, call testAddCountry() after setting up the context.
2. Run and check if the new country is present in the database.
3. **spring-data-jpa-handson**

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

* + **Query Methods - Search by containing text, sorting, filter with starting text, fetch between dates, greater than or lesser than, top**

**Solution:**

**Code:**

**Country.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.time.LocalDate;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

@Column(name = "created\_date")

private LocalDate createdDate;

public Country() {}

public Country(String code, String name, LocalDate createdDate) {

this.code = code;

this.name = name;

this.createdDate = createdDate;

}

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; }

public LocalDate getCreatedDate() { return createdDate; }

public void setCreatedDate(LocalDate createdDate) { this.createdDate = createdDate; }

@Override

public String toString() {

return "Country{" + "code='" + code + '\'' + ", name='" + name + '\'' + ", createdDate=" + createdDate + '}';

}

}

**CountryRepository.java**

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Country;

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

import java.time.LocalDate;

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

// Find countries by name containing text

List<Country> findByNameContaining(String keyword);

// Find countries by name starting with

List<Country> findByNameStartingWith(String prefix);

// Find countries created after a date

List<Country> findByCreatedDateAfter(LocalDate date);

// Find countries created between two dates

List<Country> findByCreatedDateBetween(LocalDate start, LocalDate end);

// Find top 3 countries by name ascending

List<Country> findTop3ByOrderByNameAsc();

// Find all countries sorted by name descending

List<Country> findAllByOrderByNameDesc();

}

**CountryService.java**

package com.example.ormlearn.service;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.repository.CountryRepository;

import com.example.ormlearn.service.exception.CountryNotFoundException;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.time.LocalDate;

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 Not Found");

}

return result.get();

}

@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}

public List<Country> findByNameContaining(String keyword) {

return countryRepository.findByNameContaining(keyword);

}

public List<Country> findByNameStartingWith(String prefix) {

return countryRepository.findByNameStartingWith(prefix);

}

public List<Country> findByCreatedDateAfter(LocalDate date) {

return countryRepository.findByCreatedDateAfter(date);

}

public List<Country> findByCreatedDateBetween(LocalDate start, LocalDate end) {

return countryRepository.findByCreatedDateBetween(start, end);

}

public List<Country> findTop3ByOrderByNameAsc() {

return countryRepository.findTop3ByOrderByNameAsc();

}

public List<Country> findAllByOrderByNameDesc() {

return countryRepository.findAllByOrderByNameDesc();

}

}

**OrmlearnApllication.java**

package com.example.ormlearn;

import com.example.ormlearn.model.Country;

import com.example.ormlearn.service.CountryService;

import com.example.ormlearn.service.exception.CountryNotFoundException;

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.time.LocalDate;

import java.util.List;

@SpringBootApplication

public class OrmlearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) throws CountryNotFoundException {

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

countryService = context.getBean(CountryService.class);

// getCountryTest();

// testAddCountry();

testQueryMethods();

}

private static void getCountryTest() throws CountryNotFoundException {

LOGGER.info("Start getCountryTest");

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

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

LOGGER.info("End getCountryTest");

}

private static void testAddCountry() throws CountryNotFoundException {

LOGGER.info("Start testAddCountry");

Country newCountry = new Country();

newCountry.setCode("ZZ");

newCountry.setName("Zootopia");

newCountry.setCreatedDate(LocalDate.now());

countryService.addCountry(newCountry);

Country addedCountry = countryService.findCountryByCode("ZZ");

LOGGER.debug("Added Country: {}", addedCountry);

LOGGER.info("End testAddCountry");

}

private static void testQueryMethods() {

LOGGER.info("Start Query Methods");

List<Country> countriesWithAn = countryService.findByNameContaining("an");

LOGGER.debug("Countries containing 'an': {}", countriesWithAn);

List<Country> countriesStartingWithU = countryService.findByNameStartingWith("U");

LOGGER.debug("Countries starting with 'U': {}", countriesStartingWithU);

List<Country> createdAfter = countryService.findByCreatedDateAfter(LocalDate.of(2024, 1, 1));

LOGGER.debug("Countries created after 2024-01-01: {}", createdAfter);

List<Country> createdBetween = countryService.findByCreatedDateBetween(

LocalDate.of(2023, 1, 1), LocalDate.of(2024, 12, 31));

LOGGER.debug("Countries created between 2023 and 2024: {}", createdBetween);

List<Country> top3Countries = countryService.findTop3ByOrderByNameAsc();

LOGGER.debug("Top 3 countries by name ASC: {}", top3Countries);

List<Country> countriesDesc = countryService.findAllByOrderByNameDesc();

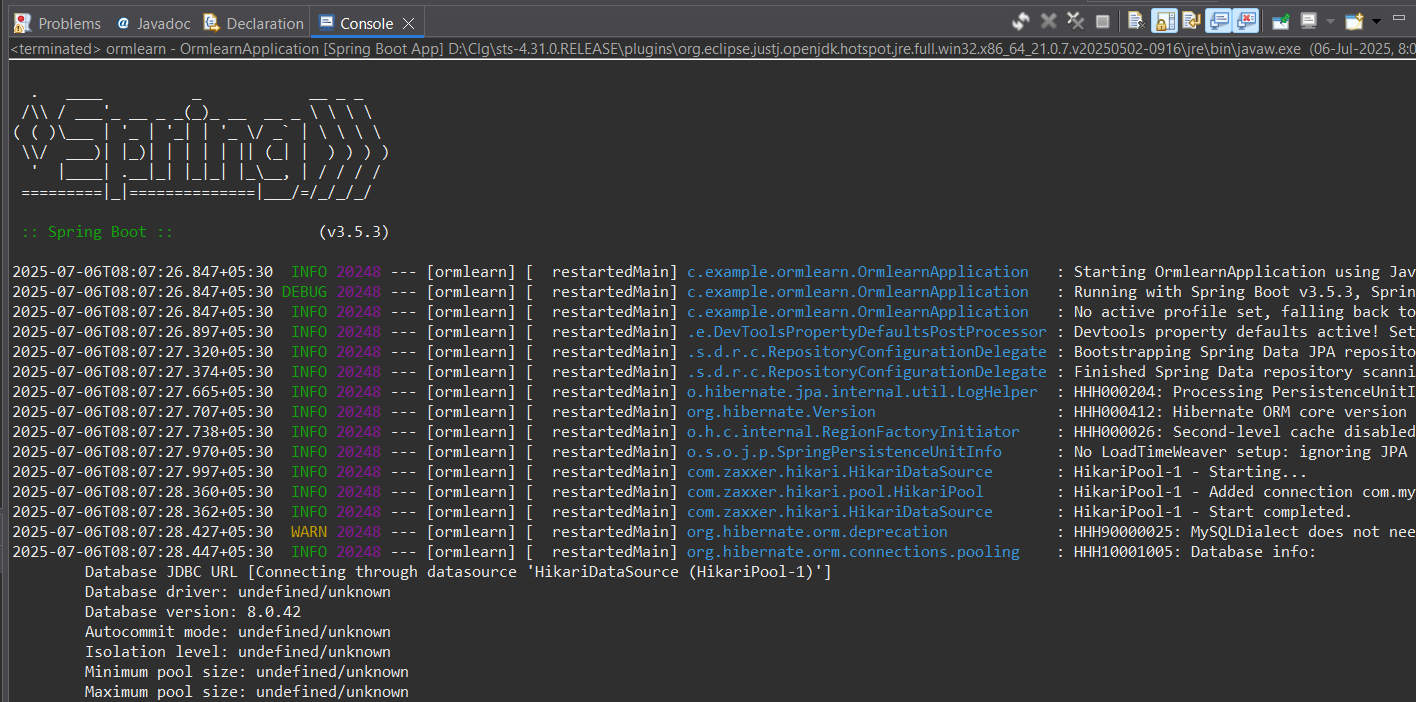
LOGGER.debug("All countries by name DESC: {}", countriesDesc);

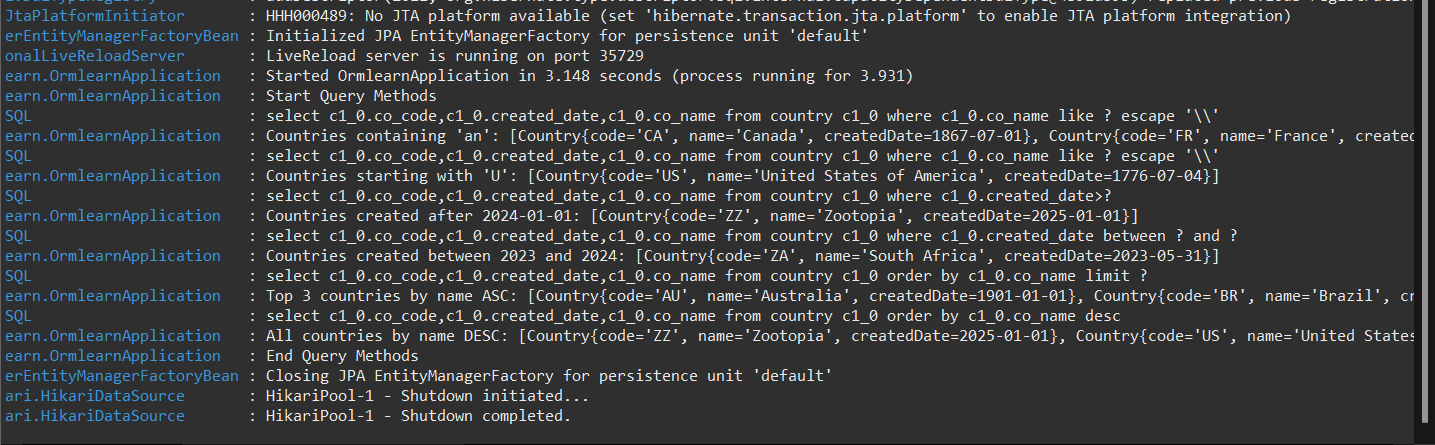
LOGGER.info("End Query Methods");

}

}

**Output:**





**Explanation:**

1. Define Query Methods in Repository Interface:

List<Country> findByNameContaining(String keyword);

List<Country> findByNameStartingWith(String prefix);

List<Country> findTop3ByOrderByNameAsc();

(For date/number fields, if present:)

List<Entity> findByDateBetween(Date start, Date end);

List<Entity> findByValueGreaterThan(int value);

1. Create corresponding test methods in OrmLearnApplication:

Call the query methods from CountryRepository.

Log and verify the output.

1. Run the main application and observe results in the log.

**2. Demonstrate implementation of O/R Mapping**

* + **@ManyToOne, @JoinColumn, @OneToMany, FetchType.EAGER,**

**FetchType.LAZY, @ManyToMany, @JoinTable, mappedBy**

**Solution:**

**Code:**

**Country.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_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;

}

}

**Employee.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

@Table(name = "employee")

public class Employee {

@Id

private int id;

@Column(name = "emp\_name")

private String name;

@ManyToOne

@JoinColumn(name = "co\_code")

private Country country;

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(

name = "employee\_skill",

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

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

)

private List<Skill> skills;

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 Country getCountry() {

return country;

}

public void setCountry(Country country) {

this.country = country;

}

public List<Skill> getSkills() {

return skills;

}

public void setSkills(List<Skill> skills) {

this.skills = skills;

}

}

**CountryRepository.java**

import com.example.ormlearn.model.Country;

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

import org.springframework.stereotype.Repository;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**EmployeeRepository.java**

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**

package com.example.ormlearn.service;

import com.example.ormlearn.model.Employee;

import com.example.ormlearn.model.Skill;

import com.example.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void fetchEmployeeWithSkills(int id) {

Employee employee = employeeRepository.findById(id).orElse(null);

if (employee != null) {

System.out.println("Employee: " + employee.getName());

System.out.println("Country: " + employee.getCountry().getName());

System.out.println("Skills:");

for (Skill skill : employee.getSkills()) {

System.out.println(" - " + skill.getName());

}

} else {

System.out.println("Employee not found.");

}

}

}

**OrmlearnApllication.java**

package com.example.ormlearn;

import com.example.ormlearn.service.EmployeeService;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmlearnApplication implements CommandLineRunner {

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(OrmlearnApplication.class, args);

}

@Override

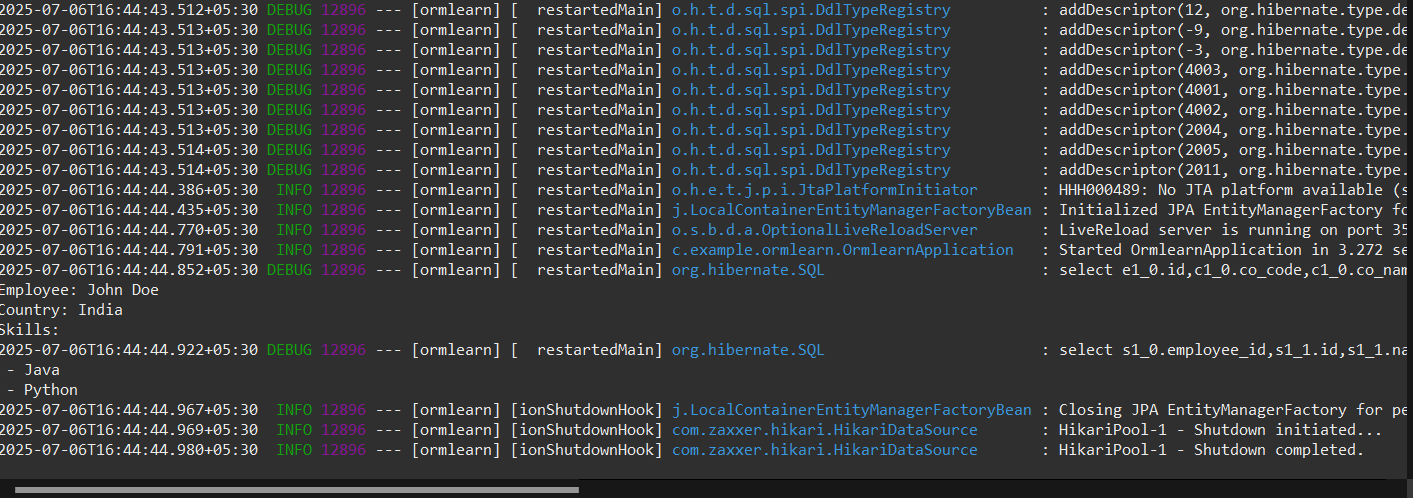
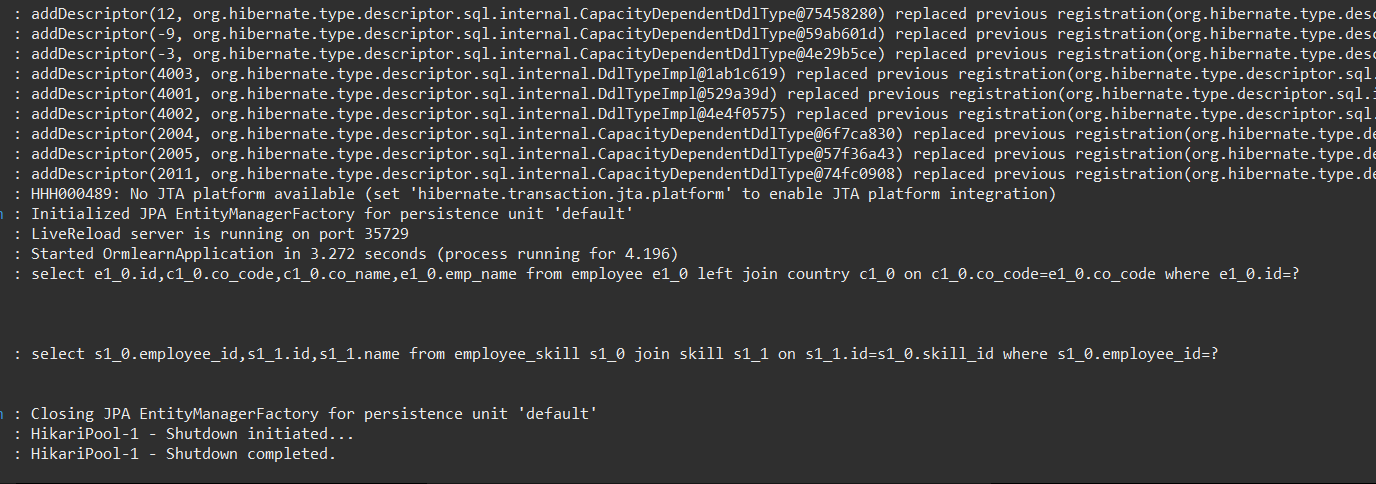
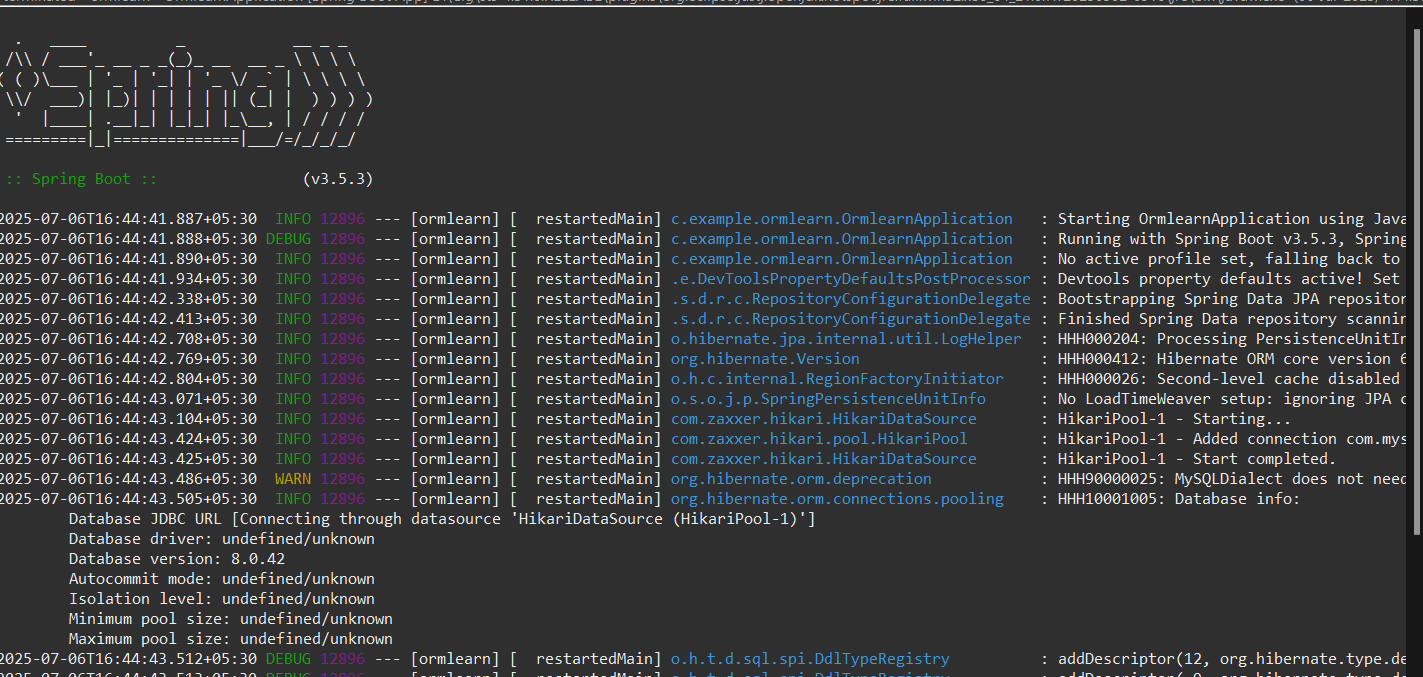
public void run(String... args) throws Exception {

employeeService.fetchEmployeeWithSkills(1); // Replace 1 with a valid employee ID in your DB

}

}

**Output:**



**Explanation:**

1. Create Entity Classes

Annotate classes with @Entity and @Table.

1. Implement Relationships

Use @ManyToOne and @JoinColumn for many-to-one mapping.

Use @OneToMany(mappedBy = "...", fetch = FetchType.LAZY/EAGER) for one-to-many.

Use @ManyToMany and @JoinTable for many-to-many.

1. Create Repositories

Extend JpaRepository for each entity.

1. Autowire in Service Layer

Autowire repositories into a service class.

1. Create Test Methods

Fetch entity with related data and log the output.

1. Run Application
2. **spring-data-jpa-handson**

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

* + **HQL stands for Hibernate Query Language, JPQL stands for Java Persistence Query Language, Compare HQL and JPQL, @Query annotation, HQL fetch keyword, aggregate functions in HQL, Native Query, nativeQuery attribute**

**Solution:**

HQL (Hibernate Query Language) is an object-oriented query language similar to SQL, but it operates on Java objects instead of database tables.

JPQL (Java Persistence Query Language) is part of JPA and is also object-oriented. It works similarly to HQL and is considered vendor-agnostic (works across different JPA providers).

**Code:**

**Employee.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.util.HashSet;

import java.util.Set;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(name = "name", nullable = false)

private String name;

@Column(name = "department")

private String department;

@Column(name = "salary")

private Double salary;

@ManyToMany(fetch = FetchType.LAZY)

@JoinTable(

name = "employee\_skill",

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

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

)

private Set<Skill> skills = new HashSet<>();

public Long getId() {

return id;

}

public void setId(Long id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

public Double getSalary() {

return salary;

}

public void setSalary(Double salary) {

this.salary = salary;

}

public Set<Skill> getSkills() {

return skills;

}

public void setSkills(Set<Skill> skills) {

this.skills = skills;

}

}

**Skill.java**

package com.example.ormlearn.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToMany(mappedBy = "skills")

private Set<Employee> employees;

public Long getId() { return id; }

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

public String getName() { return name; }

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

public Set<Employee> getEmployees() { return employees; }

public void setEmployees(Set<Employee> employees) { this.employees = employees; }

}

**EmployeeRepository.java**

package com.example.ormlearn.repository;

import com.example.ormlearn.model.Employee;

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

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

import org.springframework.data.repository.query.Param;

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// JPQL Query

@Query("SELECT e FROM Employee e WHERE e.department = :dept")

List<Employee> findByDepartment(@Param("dept") String department);

// Fetch Join Query

@Query("SELECT e FROM Employee e JOIN FETCH e.skills WHERE e.id = :id")

Employee findEmployeeWithSkills(@Param("id") Long id);

// Aggregate function

@Query("SELECT AVG(e.salary) FROM Employee e")

Double findAverageSalary();

// Native SQL Query

@Query(value = "SELECT \* FROM employee WHERE salary > :minSalary", nativeQuery = true)

List<Employee> findEmployeesWithHighSalary(@Param("minSalary") double salary);

}

**TestEmployeeService.java**

package com.example.ormlearn.service;

import com.example.ormlearn.model.Employee;

import com.example.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import jakarta.annotation.PostConstruct;

import java.util.List;

@Service

public class TestEmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@PostConstruct

public void testQueries() {

System.out.println("Testing JPQL Query by Department:");

List<Employee> itEmployees = employeeRepository.findByDepartment("IT");

itEmployees.forEach(e -> System.out.println(e.getName()));

System.out.println("\nTesting Fetch Join (Employee with Skills):");

Employee empWithSkills = employeeRepository.findEmployeeWithSkills(1L);

System.out.println("Employee: " + empWithSkills.getName());

empWithSkills.getSkills().forEach(skill -> System.out.println(skill.getName()));

System.out.println("\nTesting Aggregate (Average Salary):");

Double avgSalary = employeeRepository.findAverageSalary();

System.out.println("Average Salary: " + avgSalary);

System.out.println("\nTesting Native SQL Query:");

List<Employee> highEarners = employeeRepository.findEmployeesWithHighSalary(50000.0);

highEarners.forEach(e -> System.out.println(e.getName() + " - " + e.getSalary()));

}

}

**OrmlearnApllication.java**

package com.example.ormlearn;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class OrmlearnApplication {

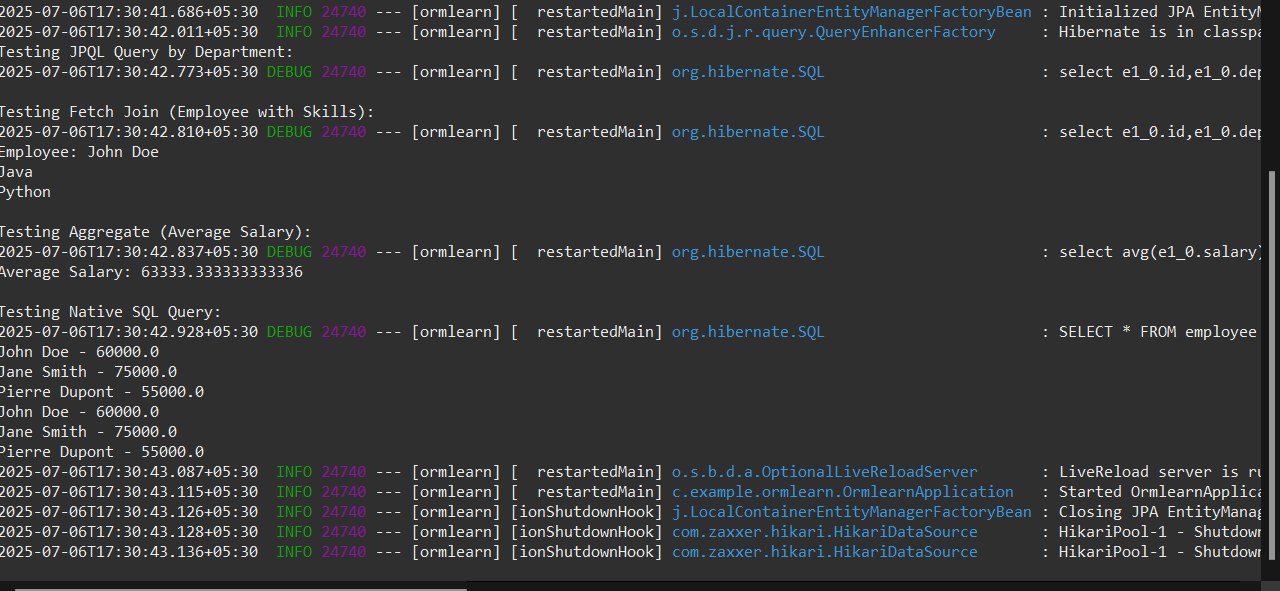
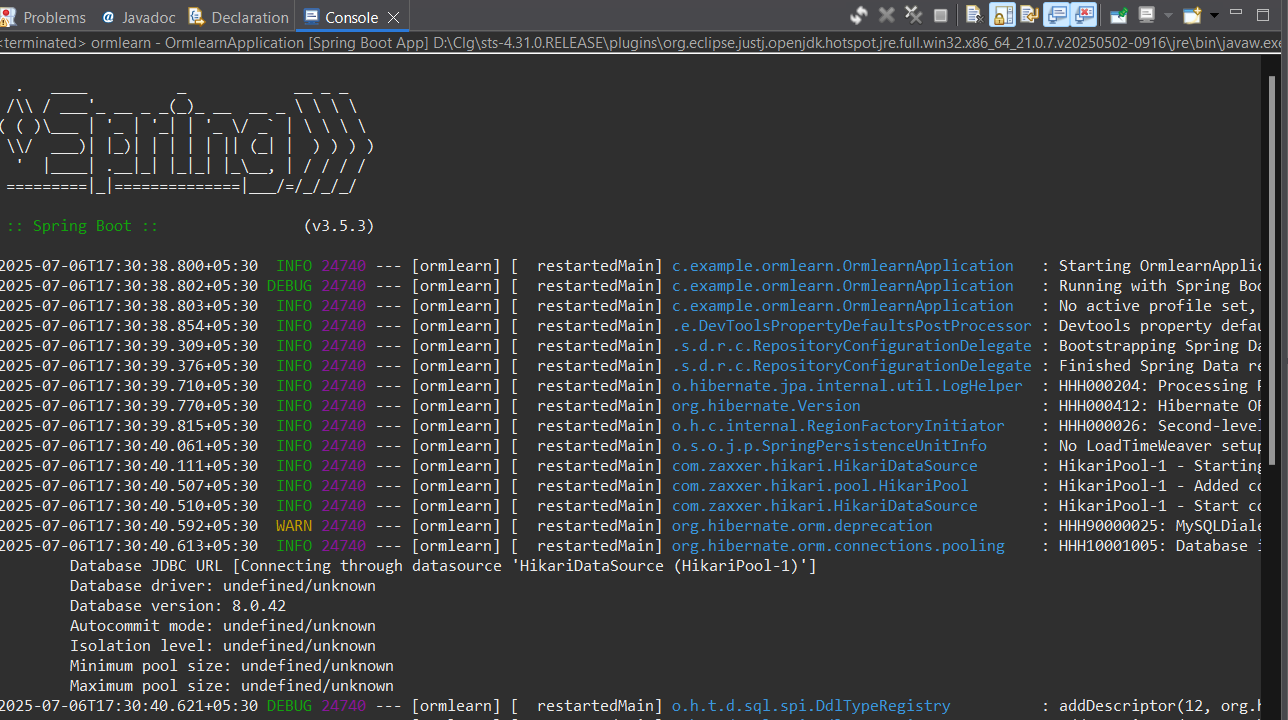
public static void main(String[] args) {

SpringApplication.run(OrmlearnApplication.class, args);

}

}

**Output:**



**Explanation:**

Steps to Demonstrate HQL & JPQL

1. Create HQL/JPQL method in Repository:

Use @Query annotation for custom queries.

1. JPQL - Find by Department:

Define: @Query("SELECT e FROM Employee e WHERE e.department = :dept")

Use parameter binding with @Param.

1. HQL Fetch Join (Many-to-Many Skills):

Define: @Query("SELECT e FROM Employee e JOIN FETCH e.skills WHERE e.id = :id")

1. HQL Aggregate Function (Average Salary):

Define: @Query("SELECT AVG(e.salary) FROM Employee e")

Steps to Demonstrate Native SQL Query

1. Create Native Query Method:

Define: @Query(value = "SELECT \* FROM employee WHERE salary > ?1", nativeQuery = true)

1. Run and Test All Methods in Main Class:

Call each repository method.

Print/log the results to verify output.