**Digital Nurture 4.0 – Week 3**

**2.Spring Data JPA with Hibernate**

**HANDS ON EXERCISE**

**Hands On 1:Spring Data JPA - Quick Example:**

1. Create a Spring Boot Project

* Use [Spring Initializr](https://start.spring.io)
  + Dependencies:
    - Spring Web
    - Spring Data JPA
    - H2 Database (in-memory for quick test)

**2. Define the Entity**

package com.example.demo.entity;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

@Entity

public class Product {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private Double price;

public Product() {}

public Product(String name, Double price) {

this.name = name;

this.price = price;

}

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 Double getPrice() { return price; }

public void setPrice(Double price) { this.price = price; }

}

3. Create the Repository

package com.example.demo.repository;

import com.example.demo.entity.Product;

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

public interface ProductRepository extends JpaRepository<Product, Long> {

}

4. Test it with a CommandLineRunner

package com.example.demo;

import com.example.demo.entity.Product;

import com.example.demo.repository.ProductRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.annotation.Bean;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

@Bean

CommandLineRunner runner(ProductRepository repo) {

return args -> {

repo.save(new Product("Laptop", 80000.0));

repo.save(new Product("Phone", 20000.0));

repo.findAll().forEach(product ->

System.out.println(product.getId() + " " + product.getName() + " " + product.getPrice())

);

};

}

}

5. Configure application.properties

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driver-class-name=org.h2.Driver

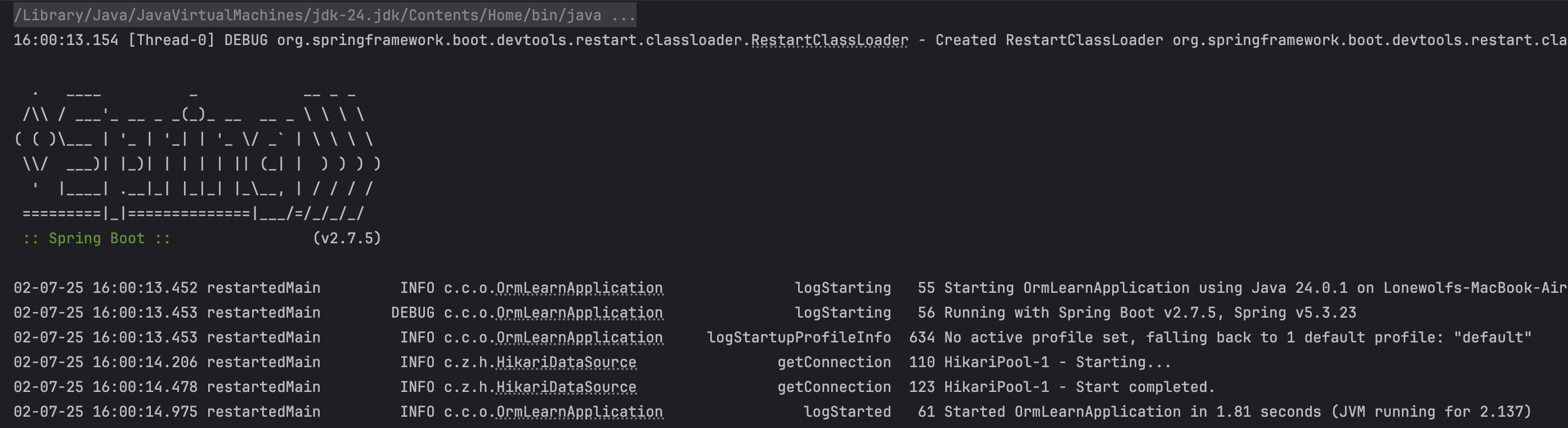
spring.datasource.username=root

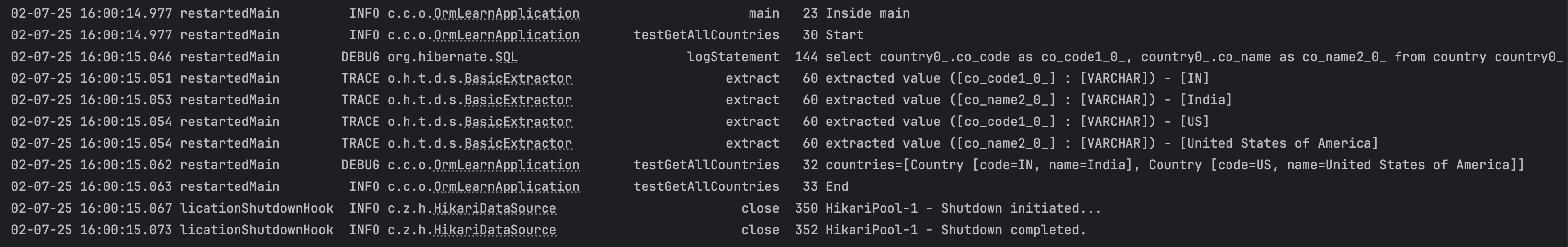
spring.datasource.password=Saran81

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

OUTPUT:





**Hands on 4 : Difference between JPA, Hibernate and Spring Data JPA:**

**Hibernate implentation:**

*pom.xml:*

<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>com.example</groupId>

<artifactId>hibernate-example</artifactId>

<version>1.0</version>

<dependencies>

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.4.4.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>8.3.0</version>

</dependency>

<dependency>

<groupId>jakarta.persistence</groupId>

<artifactId>jakarta.persistence-api</artifactId>

<version>3.1.0</version>

</dependency>

</dependencies>

</project>

*hibernate.cfg.xml:*

<?xml version='1.0' encoding='utf-8'?>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.org/dtd/hibernate-configuration-3.0.dtd">

<hibernate-configuration>

<session-factory>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/test1</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">Saran81</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.hbm2ddl.auto">update</property>

<property name="hibernate.show\_sql">true</property>

<mapping class="com.example.hibernate.Employee"/>

</session-factory>

</hibernate-configuration>

*Java Classes*

**Employee.java**

package com.example.hibernate;

import jakarta.persistence.\*;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String department;

public Employee() {}

public Employee(String name, String department) {

this.name = name;

this.department = department;

}

public Integer getId() { return id; }

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

public String getName() { return name; }

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

public String getDepartment() { return department; }

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

}

**EmployeeDAO.java**

package com.example.hibernate;

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

public class EmployeeDAO {

private static SessionFactory factory;

static {

try {

factory = new Configuration()

.configure()

.addAnnotatedClass(Employee.class)

.buildSessionFactory();

} catch (Throwable ex) {

System.err.println("SessionFactory creation failed." + ex);

throw new ExceptionInInitializerError(ex);

}

}

public Integer addEmployee(Employee employee) {

Transaction tx = null;

Integer employeeID = null;

try (Session session = factory.openSession()) {

tx = session.beginTransaction();

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

tx.commit();

} catch (Exception e) {

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

e.printStackTrace();

}

return employeeID;

}

public void close() {

factory.close();

}

}

**App.java:**

package com.example.hibernate;

public class App {

public static void main(String[] args) {

EmployeeDAO dao = new EmployeeDAO();

Employee emp = new Employee("Alice", "Engineering");

Integer id = dao.addEmployee(emp);

System.out.println("Employee created with ID: " + id);

dao.close();

}

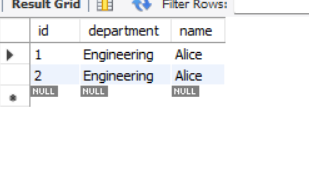
}

**Database**

* Create a **schema named test1**

CREATE DATABASE test1;





**Spring JPA implentation:**

**Step 1:**

Configure spring Initializr by using below table details.

|  |  |
| --- | --- |
| **Field** | **Value** |
| Project | Maven |
| Language | Java |
| Spring Boot | 3.5.3 |
| Group | com.example |
| Artifact | employeeapp |
| Name | employeeapp |
| Package Name | com.example.employeeapp |
| Packaging | Jar |
| Java | 24 |

Dependencies:

1.Spring Data JPA

2.Spring Web

3.MySQL Driver

**Step 2:**

Now open it on eclipse idle.In **com/example/employeeapp** create the classes named **Employee.java,EmployeeRepository.java,EmployeeService.java, and EmployeeController.java.**

**Employee.java:**

package com.example.employeeapp;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String department;

public Employee() {}

public Employee(String name, String department) {

this.name = name;

this.department = department;

}

public Integer getId() { return id; }

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

public String getName() { return name; }

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

public String getDepartment() { return department; }

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

}

**EmployeeRepository.java**:**(Interface)**

package com.example.employeeapp;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java**:

package com.example.employeeapp;

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 Employee addEmployee(Employee employee) {

return employeeRepository.save(employee);

}

}

**EmployeeController.java**:

package com.example.employeeapp;

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

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

@RestController

@RequestMapping("/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping

public Employee createEmployee(@RequestBody Employee employee) {

return employeeService.addEmployee(employee);

}

}

**Step 3:**

In **src/main/resources**,update **application.properties**.

**application.properties**:

spring.datasource.url=jdbc:mysql://localhost:3306/test2

spring.datasource.username=root

spring.datasource.password=root@123

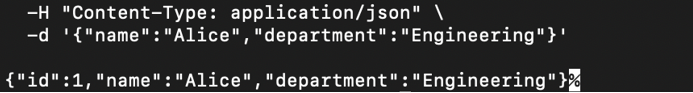
spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

Create scheme called test2 on your MySQL Database.And ensure the username and password

**Step 4:**

Run the **EmployeeappApplication.java**  as a java application.And run the following command on the terminal.



**Additional important hands-on**

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

**(i)Implement services for managing Country,(ii)Find a country based on country code,(iii)Add new Country:**

**STEP 1: Project Setup**

1.Create a Spring Boot project (use [Spring Initializr](https://start.spring.io)):

* Project: Maven
* Language: Java
* Dependencies:
  + Spring Web
  + Spring Data JPA
  + H2 Database

**STEP 2: Define the Country Entity**

package com.example.countryservice.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

@Entity

public class Country {

@Id

private String code;

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;

}

}

**STEP 3: Create the Repository**

package com.example.countryservice.repository;

import com.example.countryservice.model.Country;

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

public interface CountryRepository extends JpaRepository<Country, String> {

}

**STEP 4: Create the Service Layer**

package com.example.countryservice.service;

import com.example.countryservice.model.Country;

import com.example.countryservice.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 List<Country> getAllCountries() {

return countryRepository.findAll();

}

public Optional<Country> getCountryByCode(String code) {

return countryRepository.findById(code);

}

public Country addCountry(Country country) {

return countryRepository.save(country);

}

public void deleteCountry(String code) {

countryRepository.deleteById(code);

}

public Country updateCountry(Country country) {

return countryRepository.save(country);

}

}

**STEP 5: Create the Controller**

package com.example.countryservice.controller;

import com.example.countryservice.model.Country;

import com.example.countryservice.service.CountryService;

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

import org.springframework.http.ResponseEntity;

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

import java.util.List;

@RestController

@RequestMapping("/api/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping

public List<Country> getAllCountries() {

return countryService.getAllCountries();

}

@GetMapping("/{code}")

public ResponseEntity<Country> getCountryByCode(@PathVariable String code) {

return countryService.getCountryByCode(code)

.map(ResponseEntity::ok)

.orElse(ResponseEntity.notFound().build());

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.addCountry(country);

}

@DeleteMapping("/{code}")

public ResponseEntity<Void> deleteCountry(@PathVariable String code) {

countryService.deleteCountry(code);

return ResponseEntity.noContent().build();

}

@PutMapping("/{code}")

public ResponseEntity<Country> updateCountry(

@PathVariable String code,

@RequestBody Country country

) {

if (!countryService.getCountryByCode(code).isPresent()) {

return ResponseEntity.notFound().build();

}

country.setCode(code);

return ResponseEntity.ok(countryService.updateCountry(country));

}

}

**STEP 6: Configure H2 Database**

spring.datasource.url=jdbc:h2:mem:countrydb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.h2.console.enabled=true

**STEP 7:Test the API**

* Start the Spring Boot application.
* Test with Postman or curl:

POST http://localhost:8080/api/countries

Content-Type: application/json

{

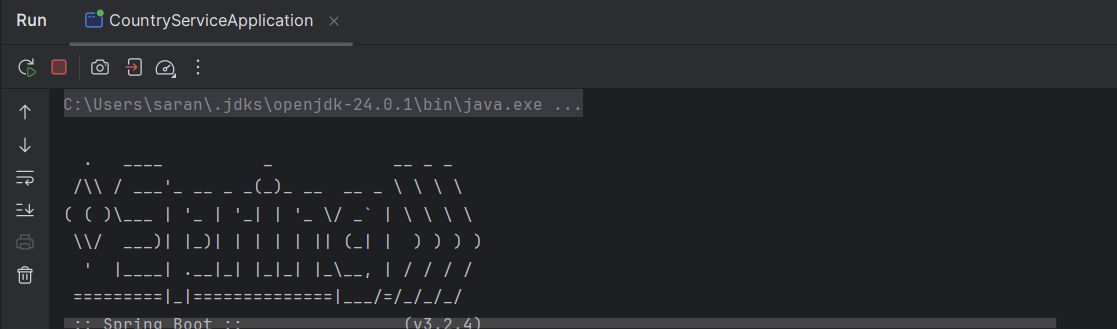
"code": "IN",

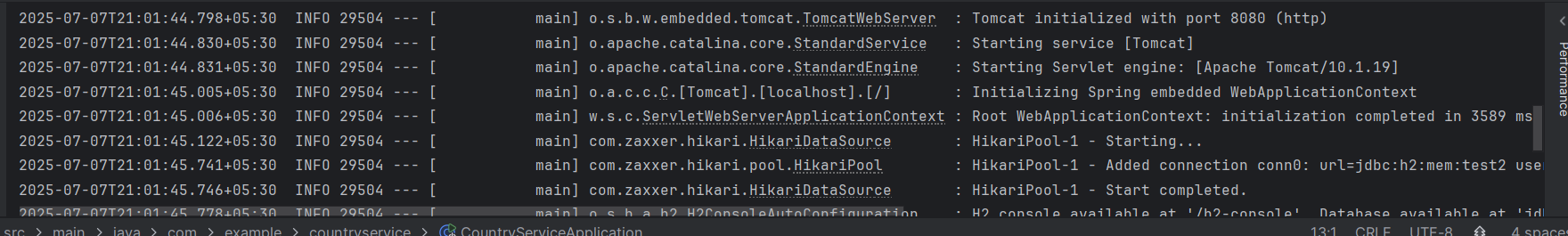
"name": "India"

}

SELECT \* FROM COUNTRY;

OUTPUT:



****

**Filename : 2. spring-data-jpa-handson**

**(i)Demonstrate implementation of Query Methods feature of Spring Data JPA:**

**Step 1:**

Configure spring Initializer by using below table details.

|  |  |
| --- | --- |
| Field | Value |
| Project | Maven |
| Language | Java |
| Spring Boot | 3.5.3 |
| Group | com.cognizant |
| Artifact | Ormlearn |
| Name | Ormlearn |
| Package Name | com.cognizant.ormlearn |
| Packaging | Jar |
| Java | 24 |

Dependencies:

1.Spring Data JPA

2.Spring Web

3.MySQL Driver

**Step 2:**

Update **application.properties** by your username,password and schema name.

**application.properties**:

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn1

spring.datasource.username=root

spring.datasource.password=root@123

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

**Step 3:**

Create the classes named **Country.java** and **Stock.java** in the **src/main/java/com/cognizant/model** repository,Create the interfaces named **CountryRepository.java** and **StockRepository.java src/main/java/com/cognizant/repository,**and finally update the main file named **OrmLearnApplication.java.**

**Country.java**:

package com.cognizant.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 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; }

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**Stock.java**:

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.util.Date;

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

@Column(name = "st\_id")

private int id;

@Column(name = "st\_code")

private String code;

@Temporal(TemporalType.DATE)

@Column(name = "st\_date")

private Date date;

@Column(name = "st\_open")

private BigDecimal open;

@Column(name = "st\_close")

private BigDecimal close;

@Column(name = "st\_volume")

private long volume;

public int getId() { return id; }

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

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public Date getDate() { return date; }

public void setDate(Date date) { this.date = date; }

public BigDecimal getOpen() { return open; }

public void setOpen(BigDecimal open) { this.open = open; }

public BigDecimal getClose() { return close; }

public void setClose(BigDecimal close) { this.close = close; }

public long getVolume() { return volume; }

public void setVolume(long volume) { this.volume = volume; }

@Override

public String toString() {

return "Stock{" +

"id=" + id +

", code='" + code + '\'' +

", date=" + date +

", open=" + open +

", close=" + close +

", volume=" + volume +

'}';

}

}

**CountryRepository.java:**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String text);

List<Country> findByNameContainingOrderByNameAsc(String text);

List<Country> findByNameStartingWith(String letter);

}

**StockRepository.java** :

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Stock;

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

import java.util.Date;

import java.util.List;

public interface StockRepository extends JpaRepository<Stock, Integer> {

List<Stock> findByCodeAndDateBetween(String code, Date start, Date end);

List<Stock> findByCodeAndCloseGreaterThan(String code, double price);

List<Stock> findTop3ByOrderByVolumeDesc();

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**OrmLearnApplication.java**:

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.model.Stock;

import com.cognizant.ormlearn.repository.CountryRepository;

import com.cognizant.ormlearn.repository.StockRepository;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private CountryRepository countryRepository;

@Autowired

private StockRepository stockRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

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

testSearchCountryByContaining();

testSearchCountryByContainingOrdered();

testSearchCountryByStarting();

testFacebookStockSeptember();

testGoogleStockPriceGreaterThan1250();

testTop3HighestVolume();

test3LowestNetflixStocks();

}

private void testSearchCountryByContaining() {

LOGGER.info("Countries containing 'ou':");

List<Country> countries = countryRepository.findByNameContaining("ou");

countries.forEach(c -> LOGGER.info(c.toString()));

}

private void testSearchCountryByContainingOrdered() {

LOGGER.info("Countries containing 'ou' ordered by name asc:");

List<Country> countries = countryRepository.findByNameContainingOrderByNameAsc("ou");

countries.forEach(c -> LOGGER.info(c.toString()));

}

private void testSearchCountryByStarting() {

LOGGER.info("Countries starting with 'Z':");

List<Country> countries = countryRepository.findByNameStartingWith("Z");

countries.forEach(c -> LOGGER.info(c.toString()));

}

private void testFacebookStockSeptember() throws Exception {

LOGGER.info("Facebook stocks in September 2019:");

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

Date start = sdf.parse("2019-09-01");

Date end = sdf.parse("2019-09-30");

List<Stock> stocks = stockRepository.findByCodeAndDateBetween("FB", start, end);

stocks.forEach(s -> LOGGER.info(s.toString()));

}

private void testGoogleStockPriceGreaterThan1250() {

LOGGER.info("Google stocks with price > 1250:");

List<Stock> stocks = stockRepository.findByCodeAndCloseGreaterThan("GOOGL", 1250.0);

stocks.forEach(s -> LOGGER.info(s.toString()));

}

private void testTop3HighestVolume() {

LOGGER.info("Top 3 highest volume stocks:");

List<Stock> stocks = stockRepository.findTop3ByOrderByVolumeDesc();

stocks.forEach(s -> LOGGER.info(s.toString()));

}

private void test3LowestNetflixStocks() {

LOGGER.info("3 lowest Netflix stocks:");

List<Stock> stocks = stockRepository.findTop3ByCodeOrderByCloseAsc("NFLX");

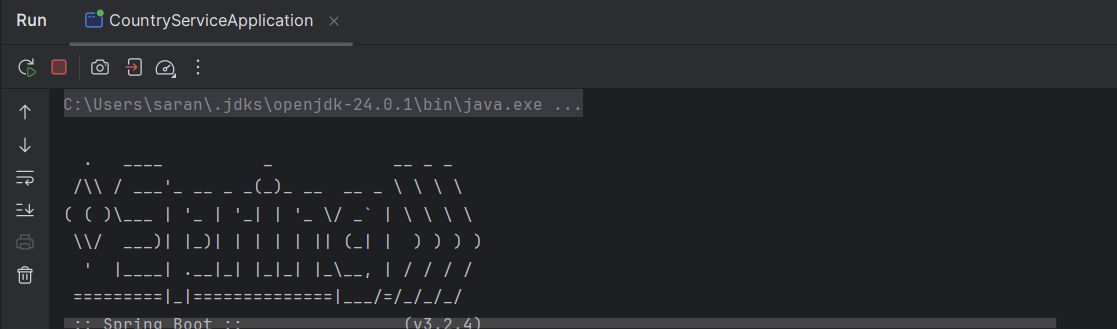
stocks.forEach(s -> LOGGER.info(s.toString()));

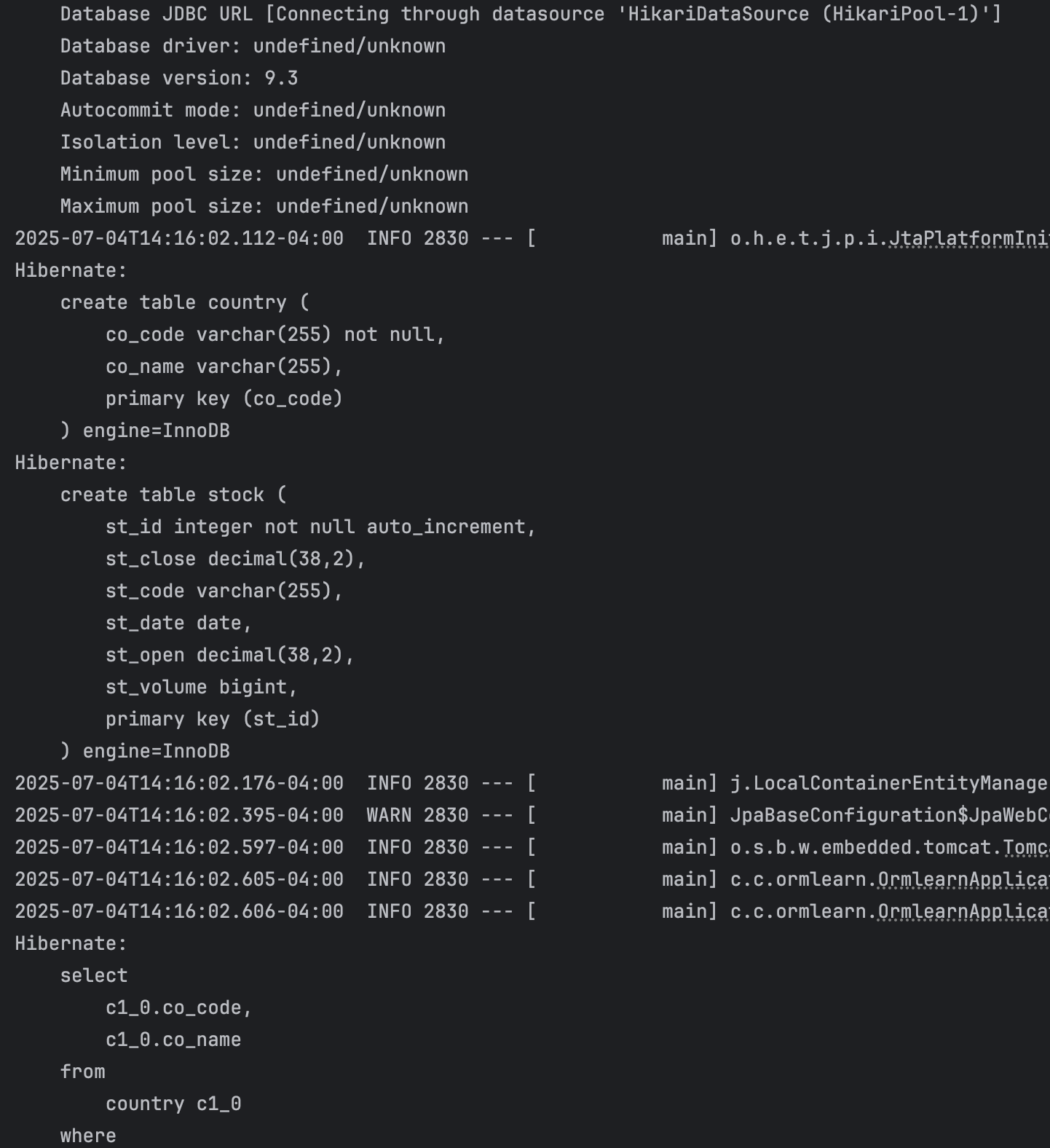
}

}

**Step 4:**

Run **OrmLearnApplication.java** as a java application to see the output on console.





**(ii)Demonstrate implementation of O/R Mapping:**

**STEP 1: Create Spring Boot Project**

Go to [Spring Initializr](https://start.spring.io)

* Project: Maven
* Language: Java
* Dependencies:
  + Spring Web
  + Spring Data JPA
  + Lombok
  + MySQL Driver

**STEP 2: Add application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn1

spring.datasource.username=root

spring.datasource.password=Saran81

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

**STEP 3: Add Dependencies in pom.xml**

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>org.projectlombok</groupId>

<artifactId>lombok</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

</dependency>

**STEP 4: Create model classes**

Department.java

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import lombok.Data;

import java.util.Set;

@Entity

@Table(name = "department")

@Data

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;

}

Skill.java

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import lombok.Data;

import java.util.Set;

@Entity

@Table(name = "skill")

@Data

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", fetch = FetchType.EAGER)

private Set<Employee> employeeList;

}

Employee.javapackage com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import lombok.Data;

import java.util.Date;

import java.util.Set;

@Entity

@Table(name = "employee")

@Data

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")

@Temporal(TemporalType.DATE)

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;

}

**STEP 5: Create repository interfaces**

**DepartmentRepository.java**

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Department;

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

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

EmployeeRepository.java:

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Employee;

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

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

SkillRepository.java

package com.cognizant.ormlearn.repository;

import com.cognizant.ormlearn.model.Skill;

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

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

STEP 6: Create service classes

DepartmentService.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Department;

import com.cognizant.ormlearn.repository.DepartmentRepository;

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

import org.springframework.stereotype.Service;

@Service

public class DepartmentService {

@Autowired

private DepartmentRepository departmentRepository;

public Department get(int id) {

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

}

public void save(Department dept) {

departmentRepository.save(dept);

}

}

EmployeeService.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

public Employee get(int id) {

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

}

public void save(Employee employee) {

employeeRepository.save(employee);

}

}

**SkillService.java**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Skill;

import com.cognizant.ormlearn.repository.SkillRepository;

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

import org.springframework.stereotype.Service;

@Service

public class SkillService {

@Autowired

private SkillRepository skillRepository;

public Skill get(int id) {

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

}

public void save(Skill skill) {

skillRepository.save(skill);

}

}

**STEP 7: Create the SQL schema**

CREATE TABLE department (

dp\_id INT AUTO\_INCREMENT PRIMARY KEY,

dp\_name VARCHAR(255)

);

CREATE TABLE skill (

sk\_id INT AUTO\_INCREMENT PRIMARY KEY,

sk\_name VARCHAR(255)

);

CREATE TABLE employee (

em\_id INT AUTO\_INCREMENT PRIMARY KEY,

em\_name VARCHAR(255),

em\_salary DOUBLE,

em\_permanent BOOLEAN,

em\_date\_of\_birth DATE,

em\_dp\_id INT,

FOREIGN KEY (em\_dp\_id) REFERENCES department(dp\_id)

);

CREATE TABLE employee\_skill (

es\_em\_id INT,

es\_sk\_id INT,

FOREIGN KEY (es\_em\_id) REFERENCES employee(em\_id),

FOREIGN KEY (es\_sk\_id) REFERENCES skill(sk\_id)

);

INSERT INTO department (dp\_name) VALUES ('HR'), ('IT');

INSERT INTO skill (sk\_name) VALUES ('Java'), ('SQL'), ('Spring Boot');

**STEP 8: Create OrmLearnApplication.java**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.\*;

import com.cognizant.ormlearn.service.\*;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.text.SimpleDateFormat;

import java.util.HashSet;

import java.util.Set;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private EmployeeService employeeService;

@Autowired

private DepartmentService departmentService;

@Autowired

private SkillService skillService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

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

LOGGER.info("Start testing");

Employee emp = new Employee();

emp.setName("John Doe");

emp.setSalary(50000);

emp.setPermanent(true);

emp.setDateOfBirth(new SimpleDateFormat("yyyy-MM-dd").parse("1990-01-01"));

emp.setDepartment(departmentService.get(1));

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

skills.add(skillService.get(1));

emp.setSkillList(skills);

employeeService.save(emp);

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

Employee fetched = employeeService.get(emp.getId());

LOGGER.info("Fetched employee: {}", fetched);

LOGGER.info("Department: {}", fetched.getDepartment());

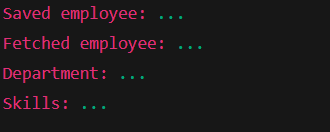
LOGGER.info("Skills: {}", fetched.getSkillList());

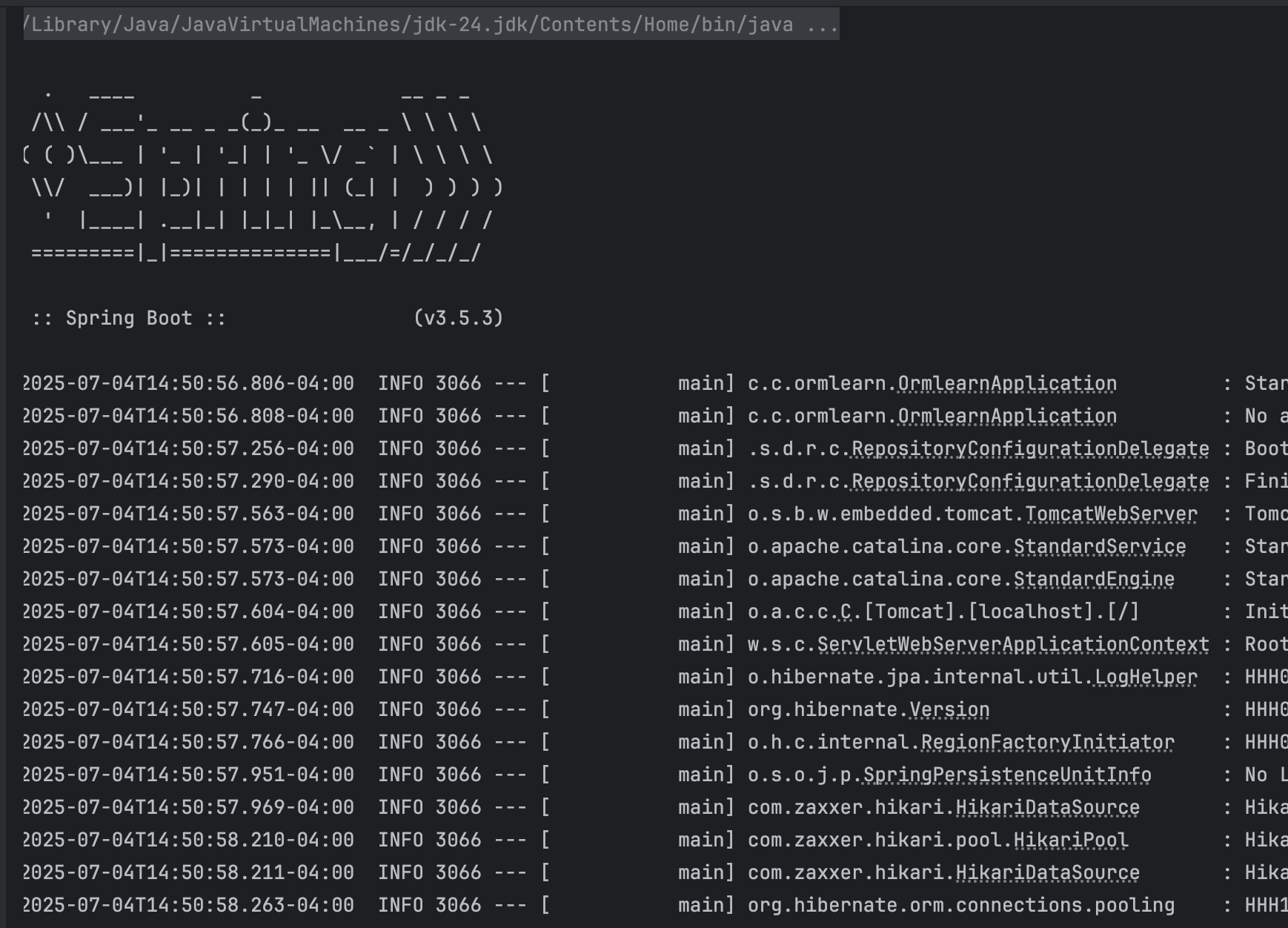
LOGGER.info("End testing");

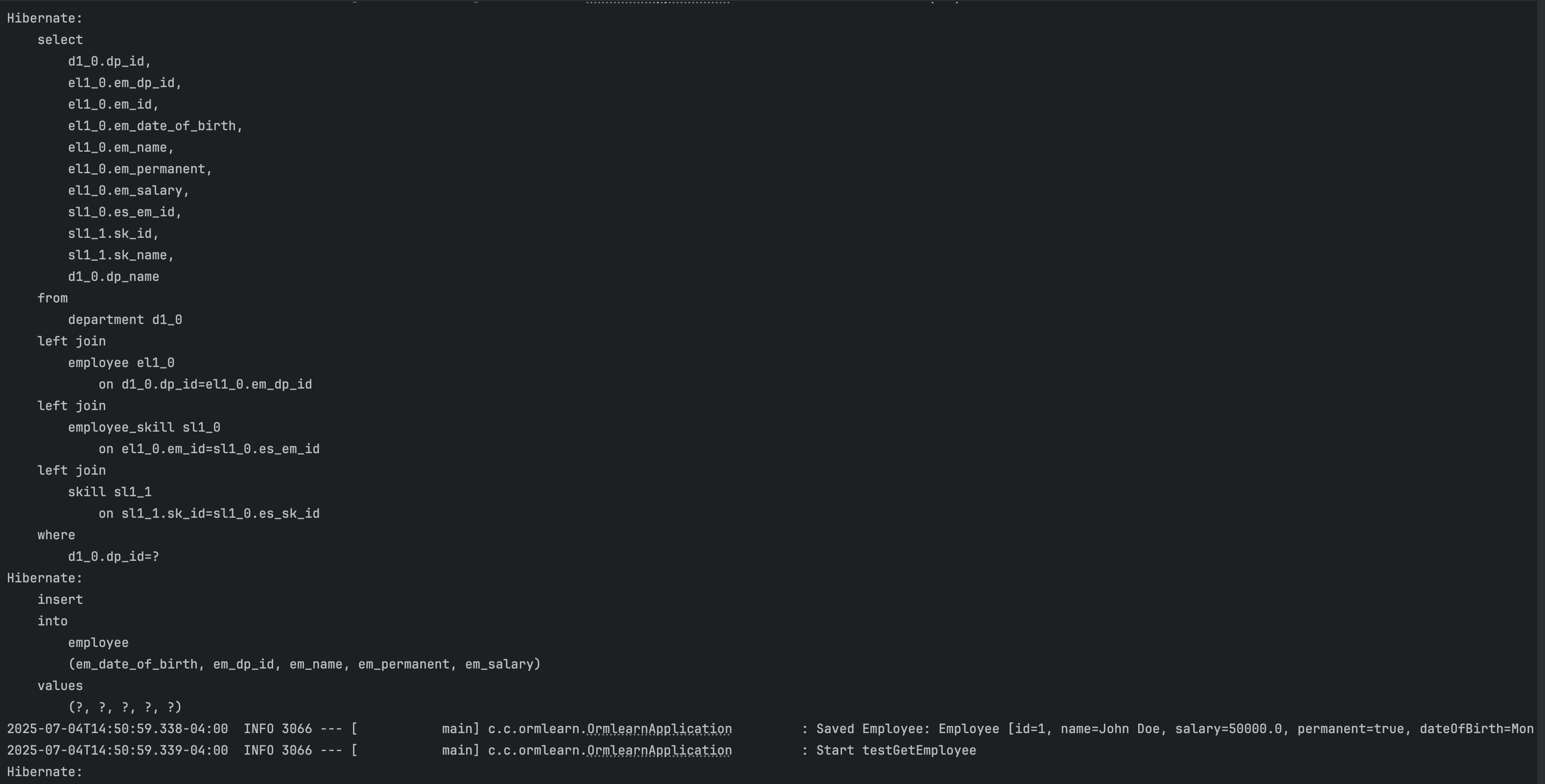
}

}

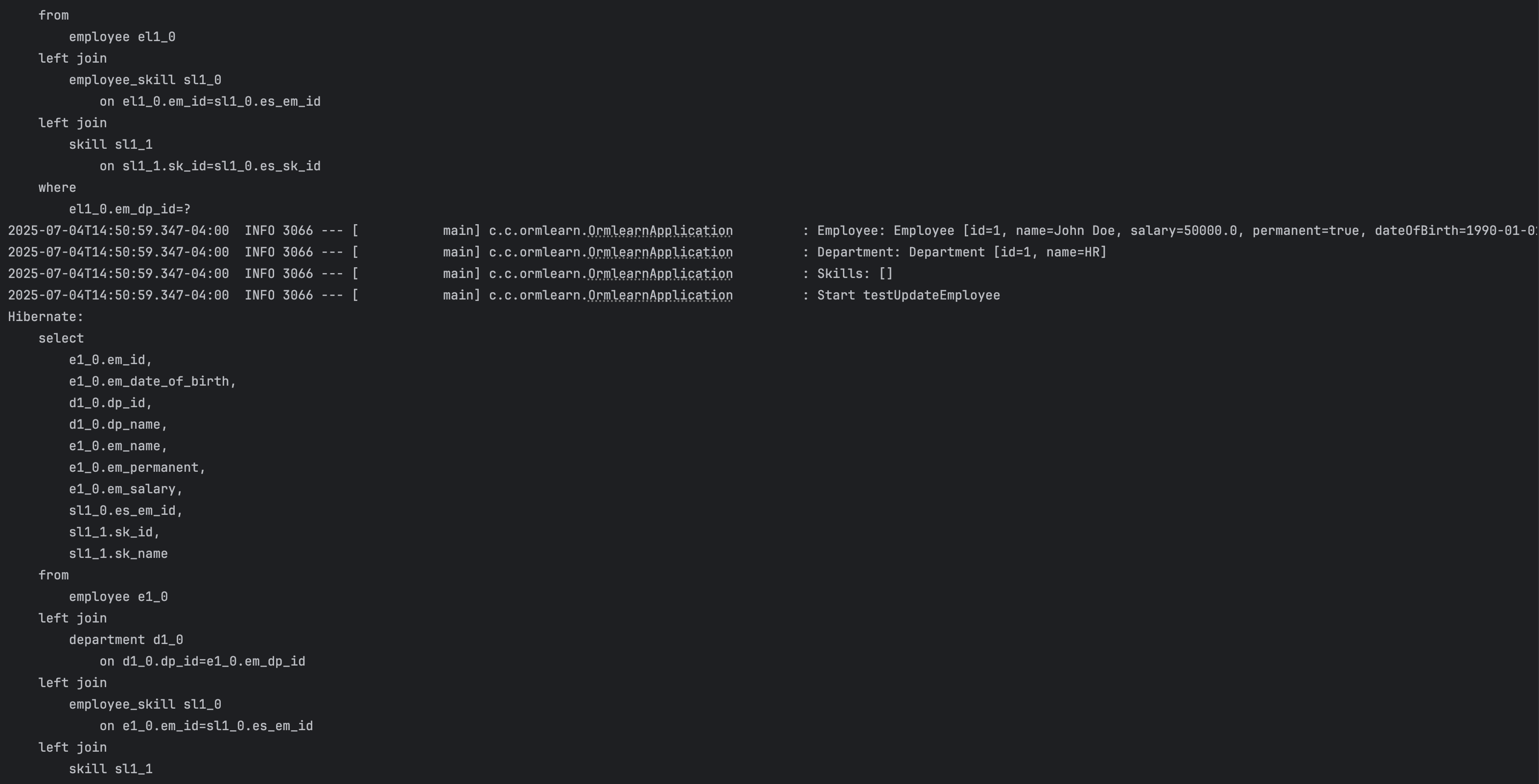
STEP 9: Run the Application

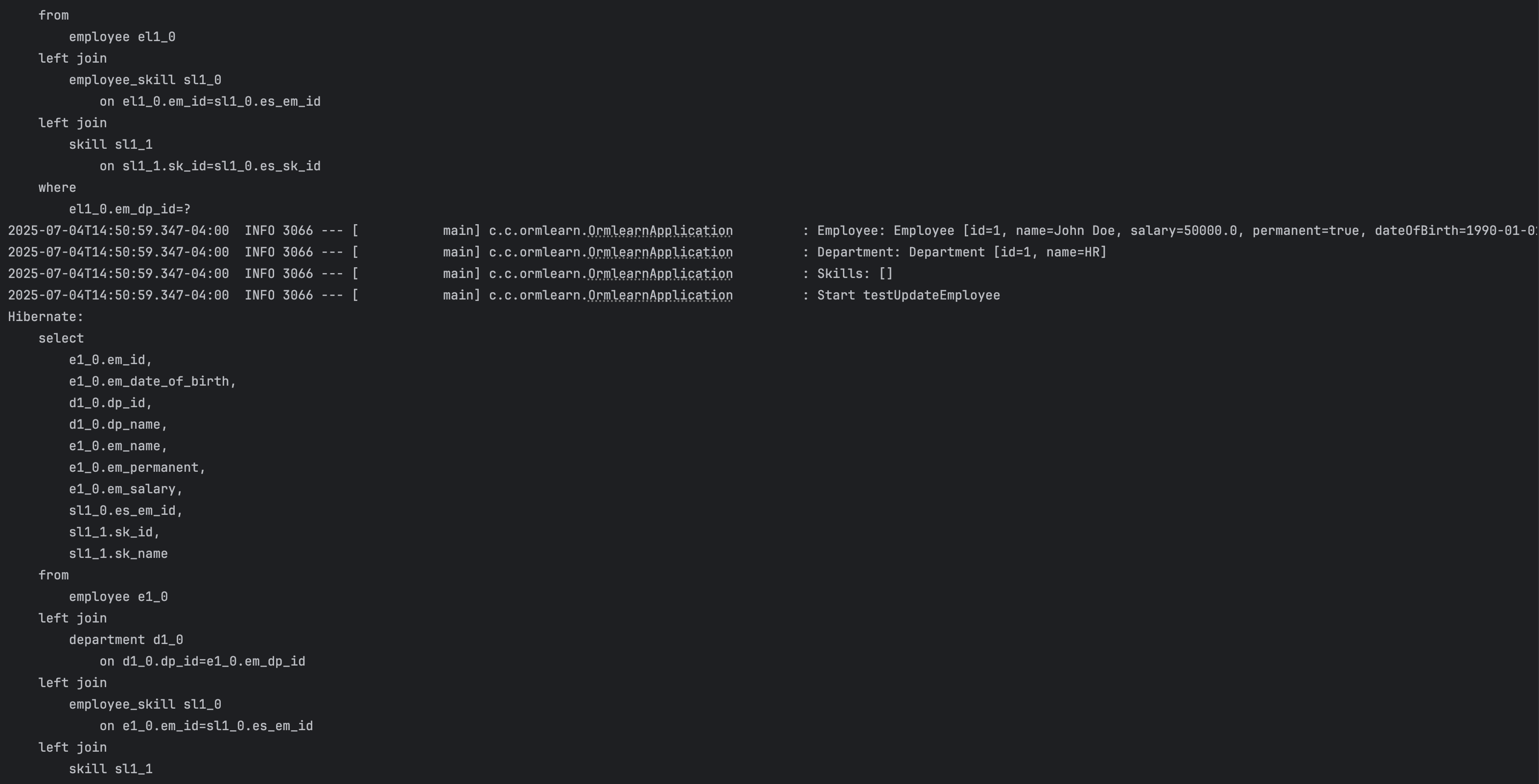














**Filename :** 3. spring-data-jpa-handson

**(i)Demonstrate writing Hibernate Query Language and Native Query**

**STEP 1: Spring Initializer Setup**

| Step | Setting | Value |
| --- | --- | --- |
| 1 | Project | Maven |
| 2 | Language | Java |
| 3 | Spring Boot | 3.5.3 |
| 4 | Group ID / Artifact | com.cognizant / ormlearn |
| 5 | Packaging | Jar |
| 6 | Java Version | 24 |
| 7 | Dependencies | Spring Web, Spring Data JPA, MySQL Driver |

**STEP 2: Database Schema**

CREATE DATABASE ormlearn3;

USE ormlearn3;

DROP TABLE IF EXISTS employee\_skill;

DROP TABLE IF EXISTS employee;

DROP TABLE IF EXISTS skill;

DROP TABLE IF EXISTS department;

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE skill (

id INT PRIMARY KEY,

name VARCHAR(50)

);

CREATE TABLE employee (

em\_id INT PRIMARY KEY,

em\_name VARCHAR(50),

em\_salary DOUBLE,

em\_permanent BOOLEAN,

em\_date\_of\_birth VARCHAR(20),

em\_dp\_id INT,

FOREIGN KEY (em\_dp\_id) REFERENCES department(id)

);

CREATE TABLE employee\_skill (

es\_em\_id INT,

es\_sk\_id INT,

PRIMARY KEY (es\_em\_id, es\_sk\_id),

FOREIGN KEY (es\_em\_id) REFERENCES employee(em\_id),

FOREIGN KEY (es\_sk\_id) REFERENCES skill(id)

);

INSERT INTO department VALUES (1, 'HR'), (2, 'IT');

INSERT INTO skill VALUES (1, 'Java'), (2, 'Spring'), (3, 'Communication');

INSERT INTO employee VALUES

(1, 'John Doe', 50000, true, '1990-01-01', 2),

(2, 'Emma Smith', 60000, true, '1992-05-05', 1),

(3, 'Alex Brown', 45000, false, '1988-07-12', 2);

INSERT INTO employee\_skill VALUES (1, 1), (1, 2), (2, 3);

**STEP 3: application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn3

spring.datasource.username=root

spring.datasource.password=Saran81

spring.jpa.hibernate.ddl-auto=none

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

**STEP 4: pom.xml**

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

**STEP 5: Model Classes**

Department.java

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "department")

public class Department {

@Id

private int id;

private String name;

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; }

@Override

public String toString() {

return "Department{id=" + id + ", name='" + name + "'}";

}

}

Skill.java

package com.cognizant.ormlearn.model;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "skill")

public class Skill {

@Id

private int id;

private String name;

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; }

@Override

public String toString() {

return "Skill{id=" + id + ", name='" + name + "'}";

}

}

Employee.java

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

import java.util.List;

import java.util.stream.Collectors;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@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 String dateOfBirth;

@ManyToOne

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

private Department department;

@ManyToMany

@JoinTable(

name = "employee\_skill",

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

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

)

private List<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 String getDateOfBirth() { return dateOfBirth; }

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

public Department getDepartment() { return department; }

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

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

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

@Override

public String toString() {

String dept = department != null ? department.getName() : "null";

String skills = skillList != null ? skillList.stream().map(Skill::getName).collect(Collectors.joining(", ")) : "null";

return "Employee{id=" + id + ", name='" + name + "', salary=" + salary +

", permanent=" + permanent + ", dateOfBirth='" + dateOfBirth +

"', department=" + dept + ", skills=" + skills + "}";

}

}

**STEP 6: Repository**

EmployeeRepository.java

package com.cognizant.ormlearn.repository;

import com.cognizant.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, Integer> {

@Query("SELECT e FROM Employee e WHERE e.permanent = true")

List<Employee> getAllPermanentEmployees();

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

List<Employee> getAllPermanentEmployeesWithFetch();

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

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

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

List<Employee> getAllEmployeesNative();

}

**STEP 7: Service**

EmployeeService.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import java.util.List;

public interface EmployeeService {

List<Employee> getAllPermanentEmployees();

List<Employee> getAllPermanentEmployeesWithFetch();

double getAverageSalary(int deptId);

List<Employee> getAllEmployeesNative();

}

EmployeeServiceImpl.java

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeServiceImpl implements EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Override

public List<Employee> getAllPermanentEmployees() {

return employeeRepository.getAllPermanentEmployees();

}

@Override

public List<Employee> getAllPermanentEmployeesWithFetch() {

return employeeRepository.getAllPermanentEmployeesWithFetch();

}

@Override

public double getAverageSalary(int deptId) {

return employeeRepository.getAverageSalary(deptId);

}

@Override

public List<Employee> getAllEmployeesNative() {

return employeeRepository.getAllEmployeesNative();

}

}

**STEP 8: Main Application**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.service.EmployeeService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

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

@Autowired

private EmployeeService employeeService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) {

LOGGER.info("START");

testPermanentEmployeesWithFetch();

testAverageSalary();

testNativeQuery();

LOGGER.info("END");

}

private void testPermanentEmployeesWithFetch() {

LOGGER.info("START - testPermanentEmployeesWithFetch");

List<Employee> list = employeeService.getAllPermanentEmployeesWithFetch();

list.forEach(e -> LOGGER.info("Employee: {}", e));

LOGGER.info("END - testPermanentEmployeesWithFetch");

}

private void testAverageSalary() {

LOGGER.info("START - testAverageSalary");

double avg = employeeService.getAverageSalary(2);

LOGGER.info("Average salary in department 2: {}", avg);

LOGGER.info("END - testAverageSalary");

}

private void testNativeQuery() {

LOGGER.info("START - testNativeQuery");

List<Employee> list = employeeService.getAllEmployeesNative();

list.forEach(e -> LOGGER.info("Employee Native: id={}, name={}, salary={}, permanent={}",

e.getId(), e.getName(), e.getSalary(), e.isPermanent()));

LOGGER.info("END - testNativeQuery");

}

}

STEP 9: Run OUTPUT

