**EXERCISE 1:SPRING DATA JPA-QUICK EXAMPLE**

**MODEL**

package com.cognizant.ormlearn.model;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@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;

}

@Override

public String toString() {

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

}

}

**REPOSITORY**

package com.cognizant.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**SERVICE**

package com.cognizant.ormlearn.service;

import java.util.List;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**ORM APPLICATION.JAVA**

package com.cognizant.ormlearn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) {

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

LOGGER.info("Inside main");

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

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

LOGGER.info("End");

}

}

**RESOURCES**

# Log configuration

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

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

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# DB Configuration

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

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

spring.datasource.username=root

spring.datasource.password=root

# Hibernate Configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**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.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<parent>

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

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

<version>2.5.0</version>

<relativePath/>

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

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

</dependency>

<dependency>

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

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

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

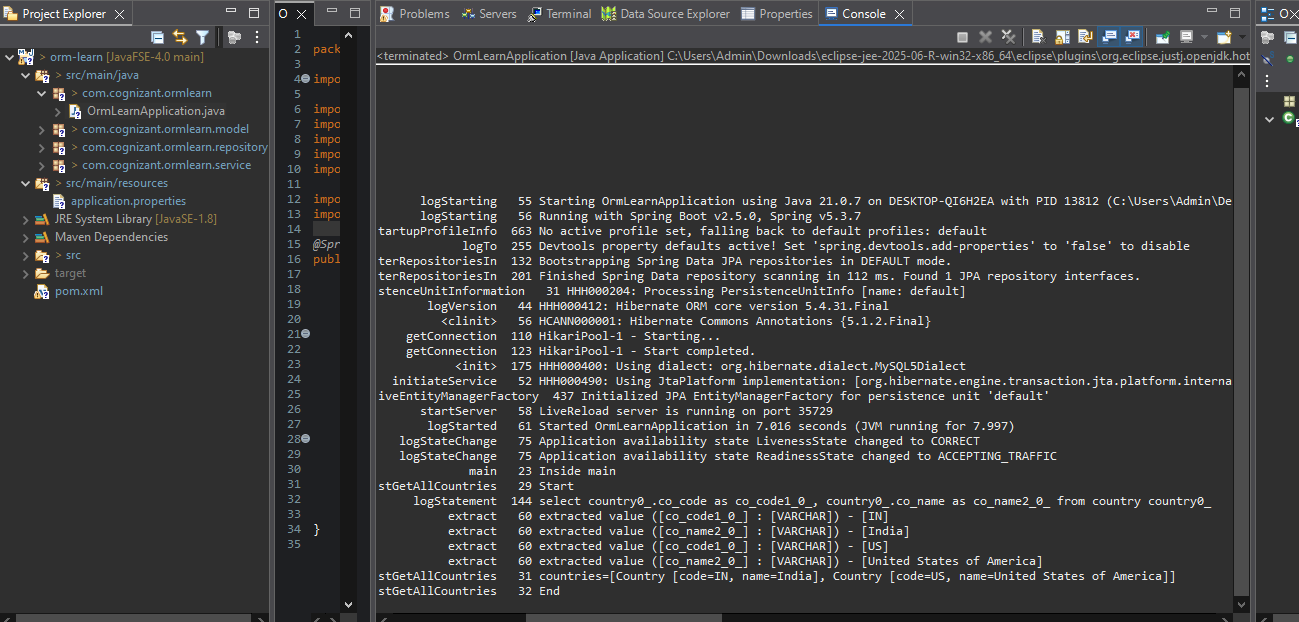
</plugin>

</plugins>

</build>

</project>

**OUTPUT**

****

**EXERCISE 2:DIFFERENCE BETWEEN JPA, HIBERNATE AND SPRING DATA JPA**

**MODEL**

package com.example.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Column;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

@Column(nullable = false)

private String name;

@Column(nullable = false)

private String 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;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", department=" + department + "]";

}

}

**REPOSITORY**

package com.example.repository;

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

import org.springframework.stereotype.Repository;

import com.example.model.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**SERVICE**

package com.example.service;

import javax.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import com.example.model.Employee;

import com.example.repository.EmployeeRepository;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}

**EMPLOYEE DEMO APPLICATION**

package com.example;

import com.example.model.Employee;

import com.example.service.EmployeeService;

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 EmployeeDemoApplication {

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

private static EmployeeService employeeService;

public static void main(String[] args) {

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

employeeService = context.getBean(EmployeeService.class);

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

testAddEmployee(); // <- Add this line

}

private static void testAddEmployee() {

LOGGER.info("Start");

Employee emp = new Employee();

emp.setName("Alice");

emp.setDepartment("IT");

employeeService.addEmployee(emp); // <- Save employee

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

LOGGER.info("End");

}

}

**RESOURCES**

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

spring.datasource.username=root

spring.datasource.password=root

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

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

logging.level.org.hibernate.SQL=debug

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

**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>employee-demo</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<name>employee-demo</name>

<description>Demo project comparing Hibernate and Spring Data JPA</description>

<parent>

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

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

<version>2.5.0</version>

</parent>

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

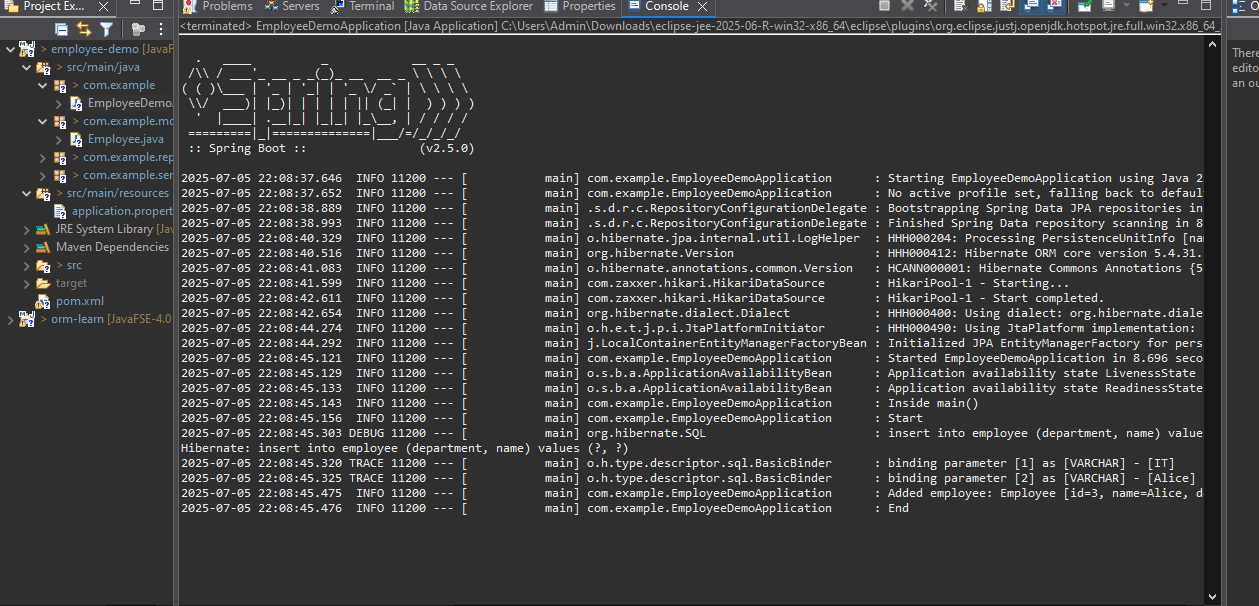
</plugin>

</plugins>

</build>

</project>

**OUTPUT**

****

**EXERCISE 3:Implement services for managing Country, Find a country based on country code, Add a new country**

**CONTROLLER**

package com.cognizant.controller;

import com.cognizant.entity.Country;

import com.cognizant.service.CountryService;

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

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

@Autowired

private CountryService countryService;

@GetMapping("/{code}")

public Country getByCode(@PathVariable String code) {

return countryService.getCountryByCode(code);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.addCountry(country);

}

@PutMapping("/{code}")

public Country updateCountry(@PathVariable String code, @RequestBody Country country) {

return countryService.updateCountry(code, country);

}

@DeleteMapping("/{code}")

public void deleteCountry(@PathVariable String code) {

countryService.deleteCountry(code);

}

@GetMapping("/search")

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

return countryService.searchByPartialName(name);

}

}

**ENTITY**

package com.cognizant.entity;

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; }

}

**REPOSITORY**

package com.cognizant.repository;

import com.cognizant.entity.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContainingIgnoreCase(String name);

}

**SERVICE**

package com.cognizant.service;

import com.cognizant.entity.Country;

import com.cognizant.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository repository;

public Country getCountryByCode(String code) {

return repository.findById(code).orElseThrow(() -> new RuntimeException("Country not found"));

}

public Country addCountry(Country country) {

return repository.save(country);

}

public Country updateCountry(String code, Country updated) {

Country existing = getCountryByCode(code);

existing.setName(updated.getName());

return repository.save(existing);

}

public void deleteCountry(String code) {

repository.deleteById(code);

}

public List<Country> searchByPartialName(String name) {

return repository.findByNameContainingIgnoreCase(name);

}

}

**OrmLearnApplication.java**

package com.cognizant.springlearn;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

@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);

getCountryByCodeTest();

testAddCountry();

}

private static void getCountryByCodeTest() {

LOGGER.info("Start");

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

private static void testAddCountry() {

LOGGER.info("Start");

Country newCountry = new Country();

newCountry.setCode("ZZ");

newCountry.setName("Zootopia");

countryService.addCountry(newCountry);

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

}

**RESOURCES**

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

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.show-sql=true

**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>3.2.4</version>

<relativePath/>

</parent>

<groupId>com.cognizant</groupId>

<artifactId>country-service</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

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

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.mysql</groupId>

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

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

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

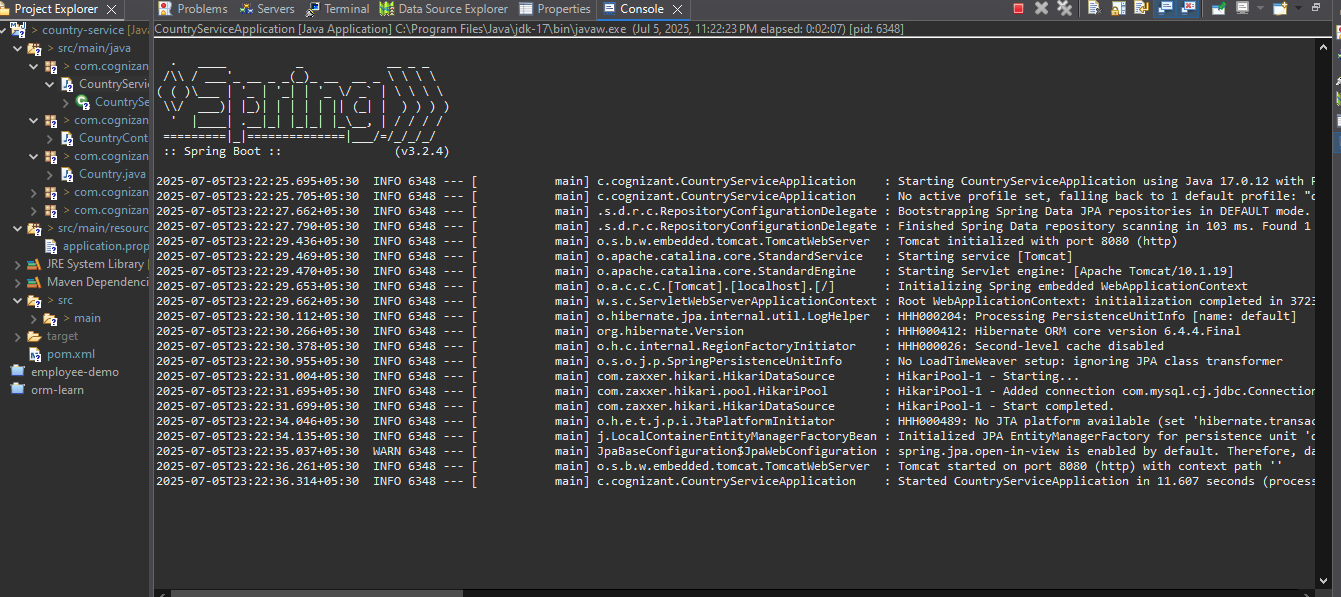
</plugin>

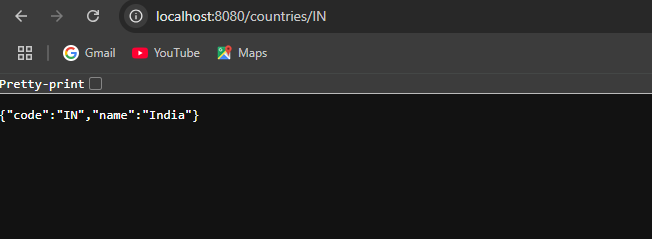
</plugins>

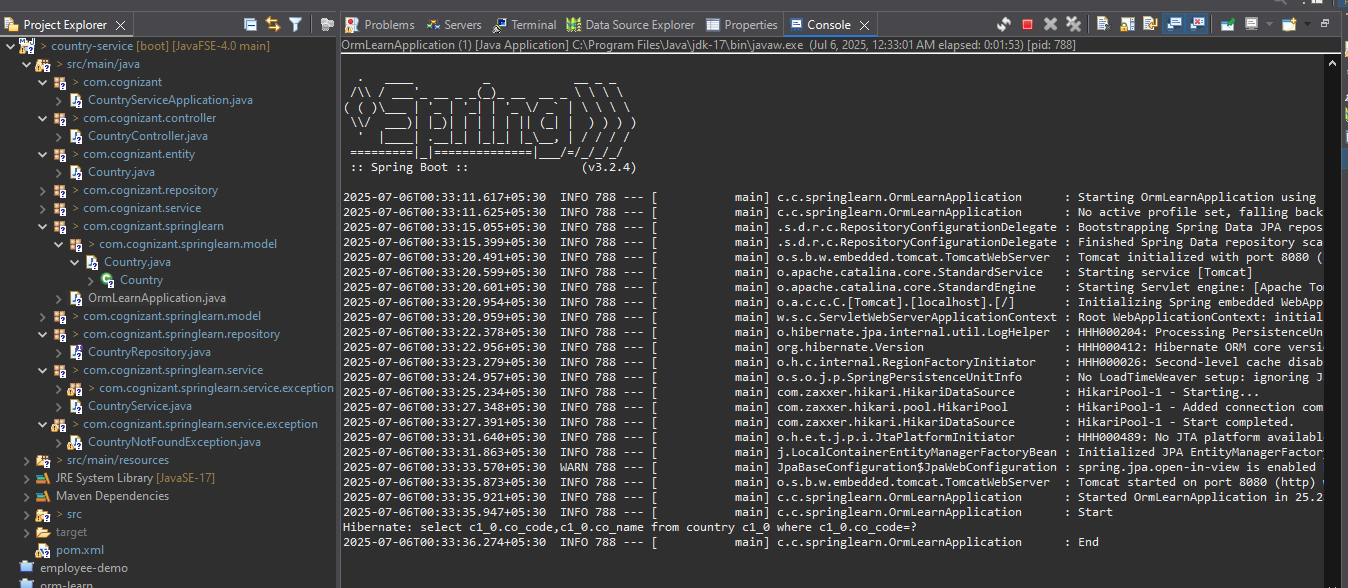
</build>

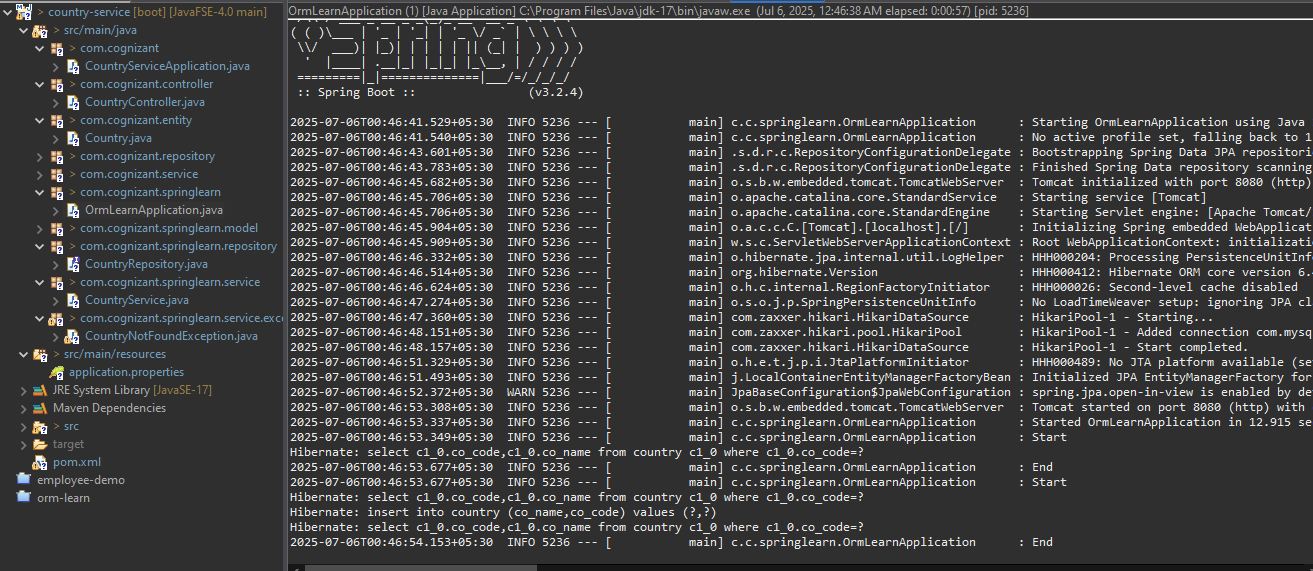
</project>

OUTPUT









**EXERCISE 4:Demonstrate implementation of Query Methods feature of Spring Data JPA**

**MODEL**

package com.example.demo.model;

import javax.persistence.\*;

import java.time.LocalDate;

@Entity

public class Person {

@Id

@GeneratedValue

private Long id;

private String firstName;

private String lastName;

private int age;

private LocalDate createdDate;

public Person() {}

public Person(String firstName, String lastName, int age, LocalDate createdDate) {

this.firstName = firstName;

this.lastName = lastName;

this.age = age;

this.createdDate = createdDate;

}

// Getters and setters omitted for brevity

@Override

public String toString() {

return firstName + " " + lastName + " (" + age + ")";

}

public Long getId() { return id; }

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

public String getFirstName() { return firstName; }

public void setFirstName(String firstName) { this.firstName = firstName; }

public String getLastName() { return lastName; }

public void setLastName(String lastName) { this.lastName = lastName; }

public int getAge() { return age; }

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

public LocalDate getCreatedDate() { return createdDate; }

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

}

**REPOSITORY**

package com.example.demo.repository;

import com.example.demo.model.Person;

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

import java.time.LocalDate;

import java.util.List;

public interface PersonRepository extends JpaRepository<Person, Long> {

List<Person> findByLastNameContaining(String fragment);

List<Person> findByLastNameContainingOrderByAgeDesc(String fragment);

List<Person> findByFirstNameStartsWith(String prefix);

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

List<Person> findByAgeGreaterThan(int age);

List<Person> findByAgeLessThanEqual(int age);

List<Person> findTop3ByOrderByAgeDesc();

}

**RUNNER**

package com.example.demo.runner;

import com.example.demo.model.Person;

import com.example.demo.repository.PersonRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import java.time.LocalDate;

@Component

public class DataRunner implements CommandLineRunner {

private final PersonRepository repo;

public DataRunner(PersonRepository repo) {

this.repo = repo;

}

@Override

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

repo.save(new Person("John", "Doe", 30, LocalDate.of(2023,1,1)));

repo.save(new Person("Jane", "Doe", 25, LocalDate.of(2024,6,15)));

repo.save(new Person("Alice", "Smith", 35, LocalDate.of(2025,1,10)));

repo.save(new Person("Bob", "Johnson", 40, LocalDate.of(2025,3,20)));

repo.save(new Person("Charlie", "Doe", 20, LocalDate.of(2025,5,5)));

System.out.println("Containing 'Doe': " + repo.findByLastNameContaining("Doe"));

System.out.println("Sorted containing 'Doe': " + repo.findByLastNameContainingOrderByAgeDesc("Doe"));

System.out.println("StartsWith 'J': " + repo.findByFirstNameStartsWith("J"));

System.out.println("Created between 2024-01 and 2025-06: " +

repo.findByCreatedDateBetween(LocalDate.of(2024,1,1), LocalDate.of(2025,6,30)));

System.out.println("Age > 30: " + repo.findByAgeGreaterThan(30));

System.out.println("Age <= 30: " + repo.findByAgeLessThanEqual(30));

System.out.println("Top 3 by age desc: " + repo.findTop3ByOrderByAgeDesc());

}

}

**DemoApplication.java**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

**RESOURCES**

spring.datasource.url=jdbc:h2:mem:testdb;DB\_CLOSE\_DELAY=-1

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

**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>spring-data-jpa-query-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<java.version>11</java.version>

<spring.boot.version>2.2.0.RELEASE</spring.boot.version>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-boot-dependencies</artifactId>

<version>${spring.boot.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

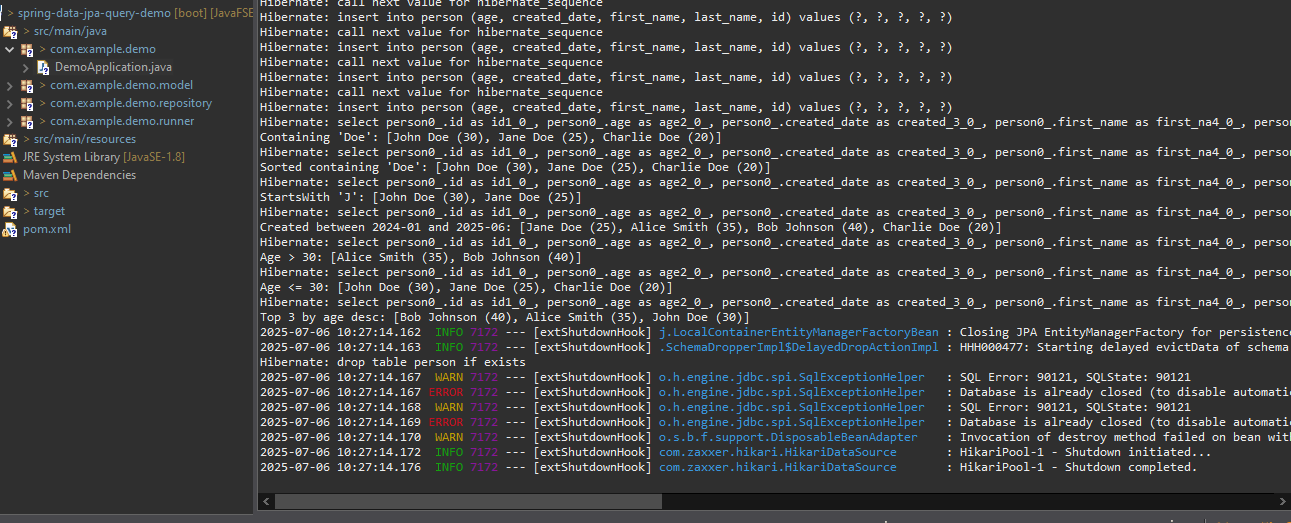
<scope>runtime</scope>

</dependency>

</dependencies>

</project>

**OUTPUT**

****

**EXERCISE 5:Demonstrate implementation of OR Mapping**

**MODEL**

**Department.java**

package com.example.demo.model;

import javax.persistence.\*;

import java.util.List;

@Entity

public class Department {

@Id

@GeneratedValue

private Long id;

private String name;

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

private List<Employee> employees;

// Getters and setters

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 List<Employee> getEmployees() { return employees; }

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

}

**Employee.java**

package com.example.demo.model;

import javax.persistence.\*;

import java.util.Set;

@Entity

public class Employee {

@Id

@GeneratedValue

private Long id;

private String name;

@ManyToOne(fetch = FetchType.EAGER)

@JoinColumn(name = "department\_id")

private Department department;

@ManyToMany

@JoinTable(

name = "employee\_project",

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

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

)

private Set<Project> projects;

// Getters and setters

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 Department getDepartment() { return department; }

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

public Set<Project> getProjects() { return projects; }

public void setProjects(Set<Project> projects) { this.projects = projects; }

}

**Project.java**

package com.example.demo.model;

import javax.persistence.\*;

import java.util.Set;

@Entity

public class Project {

@Id

@GeneratedValue

private Long id;

private String title;

@ManyToMany(mappedBy = "projects")

private Set<Employee> employees;

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

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

}

**REPOSITORY**

**DepartmentRepository.java**

package com.example.demo.repository;

import com.example.demo.model.Department;

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

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

**EmployeeRepository.java**

package com.example.demo.repository;

import com.example.demo.model.Employee;

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

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

**ProjectRepository.java**

package com.example.demo.repository;

import com.example.demo.model.Project;

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

public interface ProjectRepository extends JpaRepository<Project, Long> {}

**RUNNER**

package com.example.demo.runner;

import com.example.demo.model.\*;

import com.example.demo.repository.\*;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import java.util.Arrays;

import java.util.HashSet;

@Component

public class DataLoader implements CommandLineRunner {

private final DepartmentRepository deptRepo;

private final EmployeeRepository empRepo;

private final ProjectRepository projRepo;

public DataLoader(DepartmentRepository deptRepo, EmployeeRepository empRepo, ProjectRepository projRepo) {

this.deptRepo = deptRepo;

this.empRepo = empRepo;

this.projRepo = projRepo;

}

@Override

public void run(String... args) {

Department d1 = new Department();

d1.setName("IT");

deptRepo.save(d1);

Project p1 = new Project();

p1.setTitle("Alpha");

Project p2 = new Project();

p2.setTitle("Beta");

projRepo.saveAll(Arrays.asList(p1, p2));

Employee e1 = new Employee();

e1.setName("Alice");

e1.setDepartment(d1);

e1.setProjects(new HashSet<>(Arrays.asList(p1, p2)));

Employee e2 = new Employee();

e2.setName("Bob");

e2.setDepartment(d1);

e2.setProjects(new HashSet<>(Arrays.asList(p2)));

empRepo.saveAll(Arrays.asList(e1, e2));

}

}

**DEMO APPLICATION.JAVA**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

**RESOURCES**

spring.datasource.url=jdbc:h2:mem:testdb;DB\_CLOSE\_DELAY=-1

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

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

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

**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>spring-data-jpa-orm-mapping-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<java.version>11</java.version>

<spring.boot.version>2.2.0.RELEASE</spring.boot.version>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-boot-dependencies</artifactId>

<version>${spring.boot.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

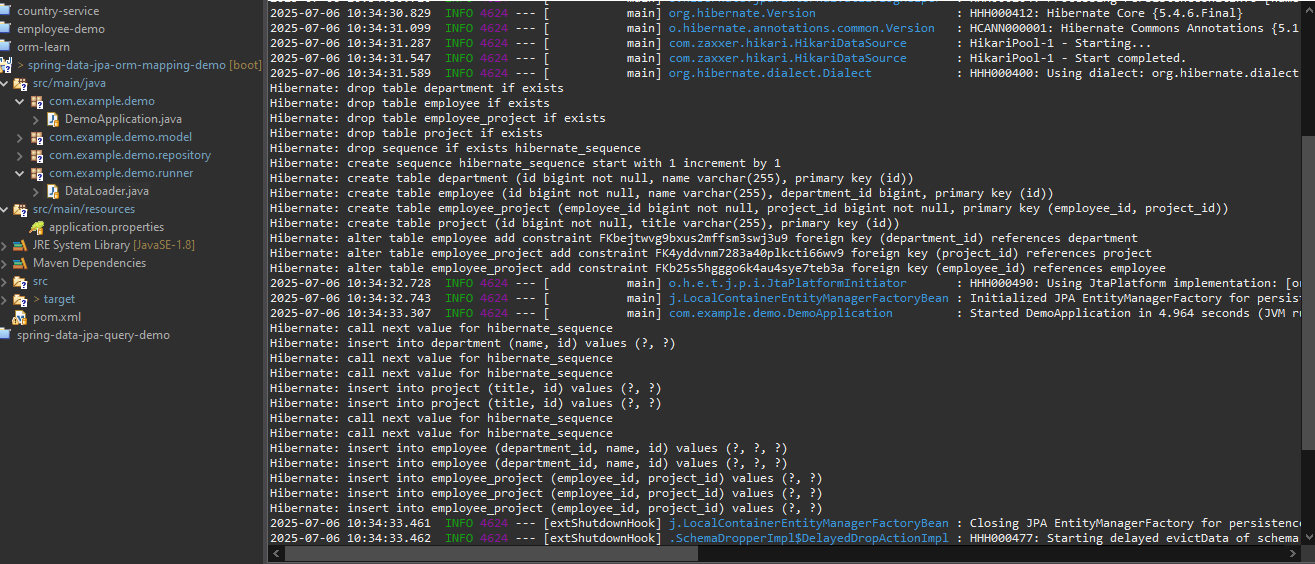
<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

</project>

**OUTPUT**

**EXERCISE 6: Demonstrate writing Hibernate Query Language and Native Query**

**MODEL**

package com.example.demo.model;

import javax.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue

private Long id;

private String name;

private String department;

private double salary;

// Getters and setters

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; }

@Override

public String toString() {

return name + " - " + department + " - " + salary;

}

}

**REPOSITORY**

package com.example.demo.repository;

import com.example.demo.model.Employee;

import org.springframework.data.jpa.repository.\*;

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

import org.springframework.stereotype.Repository;

import java.util.List;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

// JPQL (HQL) - simple select

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

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

// HQL with fetch keyword (not required here but applicable with relationships)

// HQL aggregate function

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

Double findAverageSalaryByDepartment(@Param("dept") String dept);

// Native Query

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

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

}

**RUNNER**

package com.example.demo.runner;

import com.example.demo.model.Employee;

import com.example.demo.repository.EmployeeRepository;

import org.springframework.boot.CommandLineRunner;

import org.springframework.stereotype.Component;

import java.util.List;

@Component

public class DataLoader implements CommandLineRunner {

private final EmployeeRepository employeeRepository;

public DataLoader(EmployeeRepository employeeRepository) {

this.employeeRepository = employeeRepository;

}

@Override

public void run(String... args) {

Employee emp1 = new Employee();

emp1.setName("Alice");

emp1.setDepartment("IT");

emp1.setSalary(70000);

Employee emp2 = new Employee();

emp2.setName("Bob");

emp2.setDepartment("IT");

emp2.setSalary(80000);

Employee emp3 = new Employee();

emp3.setName("Charlie");

emp3.setDepartment("HR");

emp3.setSalary(50000);

Employee emp4 = new Employee();

emp4.setName("Diana");

emp4.setDepartment("Finance");

emp4.setSalary(60000);

employeeRepository.save(emp1);

employeeRepository.save(emp2);

employeeRepository.save(emp3);

employeeRepository.save(emp4);

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

System.out.println("IT Department Employees: " + itEmployees);

Double avgSalary = employeeRepository.findAverageSalaryByDepartment("IT");

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

List<Employee> highEarners = employeeRepository.findHighEarners(60000);

System.out.println("High Earners: " + highEarners);

}

}

**DEMO APPLICATION.JAVA**

package com.example.demo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

**RESOURCES**

spring.datasource.url=jdbc:h2:mem:testdb;DB\_CLOSE\_DELAY=-1;DB\_CLOSE\_ON\_EXIT=FALSE

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=create-drop

spring.jpa.show-sql=true

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

logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE

**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>spring-data-jpa-hql-nativequery-demo</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<java.version>11</java.version>

<spring.boot.version>2.2.0.RELEASE</spring.boot.version>

</properties>

<dependencyManagement>

<dependencies>

<dependency>

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

<artifactId>spring-boot-dependencies</artifactId>

<version>${spring.boot.version}</version>

<type>pom</type>

<scope>import</scope>

</dependency>

</dependencies>

</dependencyManagement>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

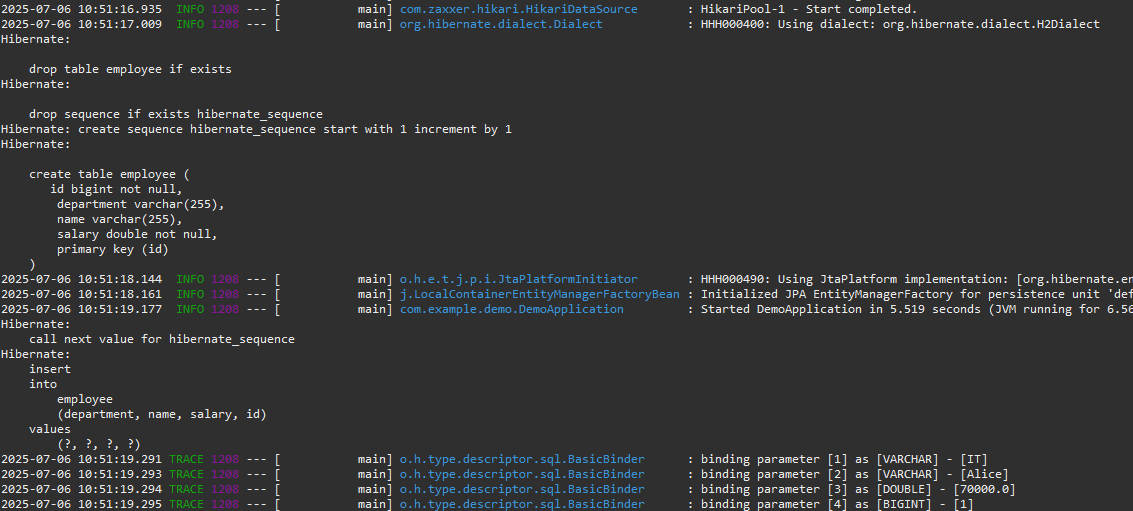
<scope>runtime</scope>

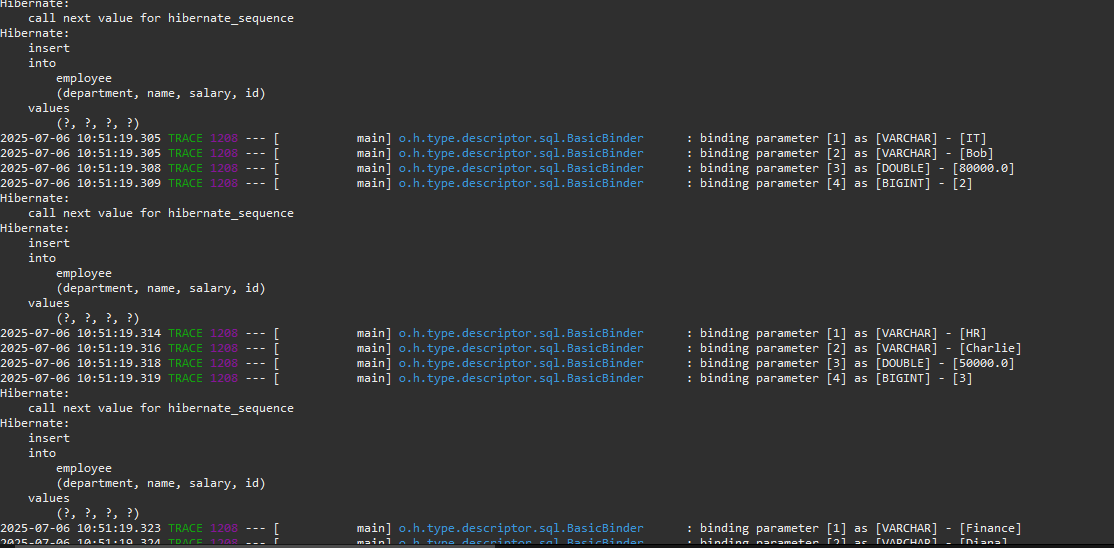
</dependency>

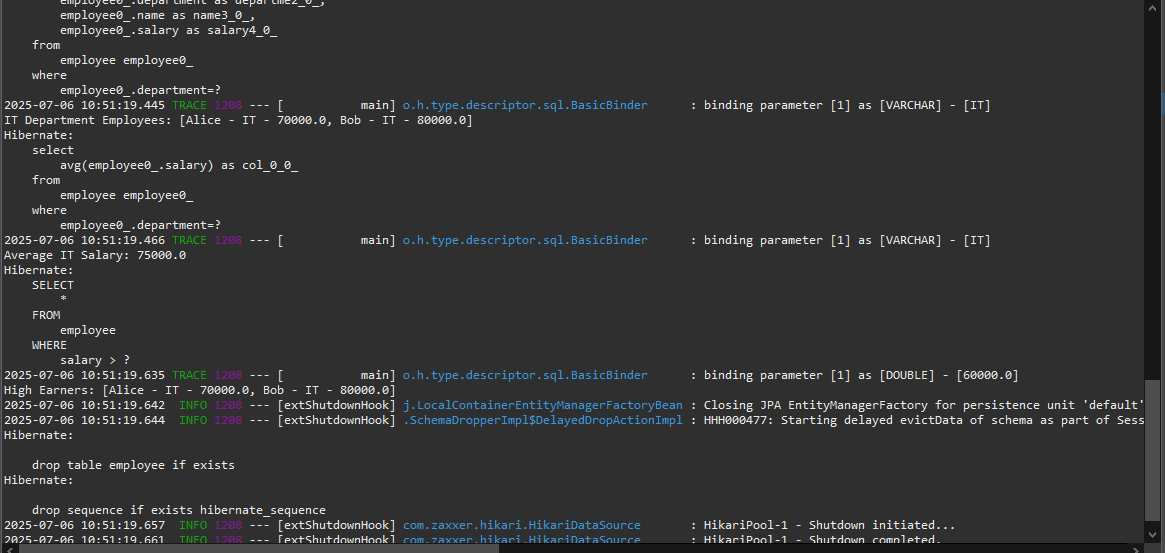
</dependencies>

</project>

**OUTPUT**

****

****

****