**Digital Nurture 4.0**

Week 3- Spring Core and Maven

# Mandatory HandsOn

# File name: 1. spring-data-jpa-handson

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

**Program:**

* Project Setup in IntelliJ IDEA

Directory:

D:\Placement\CTS\orm-learn\orm-learn

Steps:

* Open IntelliJ IDEA.
* Create a new Maven project with Spring Boot.
* Select dependencies:
* Spring Web
* Spring Data JPA
* MySQL Driver
* pom.xml Configuration

**pom.xml:**

<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>com.cognizant</groupId>  
 <artifactId>orm-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>orm-learn</name>  
 <description>Demo project for Spring Boot</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>24</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>  
 <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>

* MySQL Schema and Table Setup

Create a schema named ormlearn in MySQL Workbench and give the Sql query.

USE ormlearn;

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India'), ('US', 'United States');

* Configure the application.properties.

**application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn  
spring.datasource.username=root  
spring.datasource.password=Data@1234\_  
  
spring.jpa.hibernate.ddl-auto=none  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect  
  
logging.level.org.springframework=info  
logging.level.com.cognizant=debug

* Create a entity class named Country.java

**Country.java:**

package com.cognizant.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 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;  
 }  
}

* Create a interface named CountryRepository.java

**CountryRepository.java:**

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

* Create a class named CountryController.java.

**CountryController.java:**

package com.cognizant.ormlearn.controller;  
  
import org.springframework.web.bind.annotation.\*;  
import org.springframework.beans.factory.annotation.Autowired;  
import java.util.List;  
import com.cognizant.ormlearn.model.Country;  
import com.cognizant.ormlearn.repository.CountryRepository;  
  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 @Autowired  
 private CountryRepository repo;  
  
 @GetMapping  
 public List<Country> getAllCountries() {  
 return repo.findAll();  
 }  
  
 @PostMapping  
 public Country addCountry(@RequestBody Country country) {  
 return repo.save(country);  
 }  
  
 @GetMapping("/{code}")  
 public Country getCountryByCode(@PathVariable String code) {  
 return repo.findById(code).orElse(null);  
 }  
  
 @PutMapping("/{code}")  
 public Country updateCountry(@PathVariable String code, @RequestBody Country country) {  
 country.setCode(code);  
 return repo.save(country);  
 }  
  
 @DeleteMapping("/{code}")  
 public void deleteCountry(@PathVariable String code) {  
 repo.deleteById(code);  
 }  
}

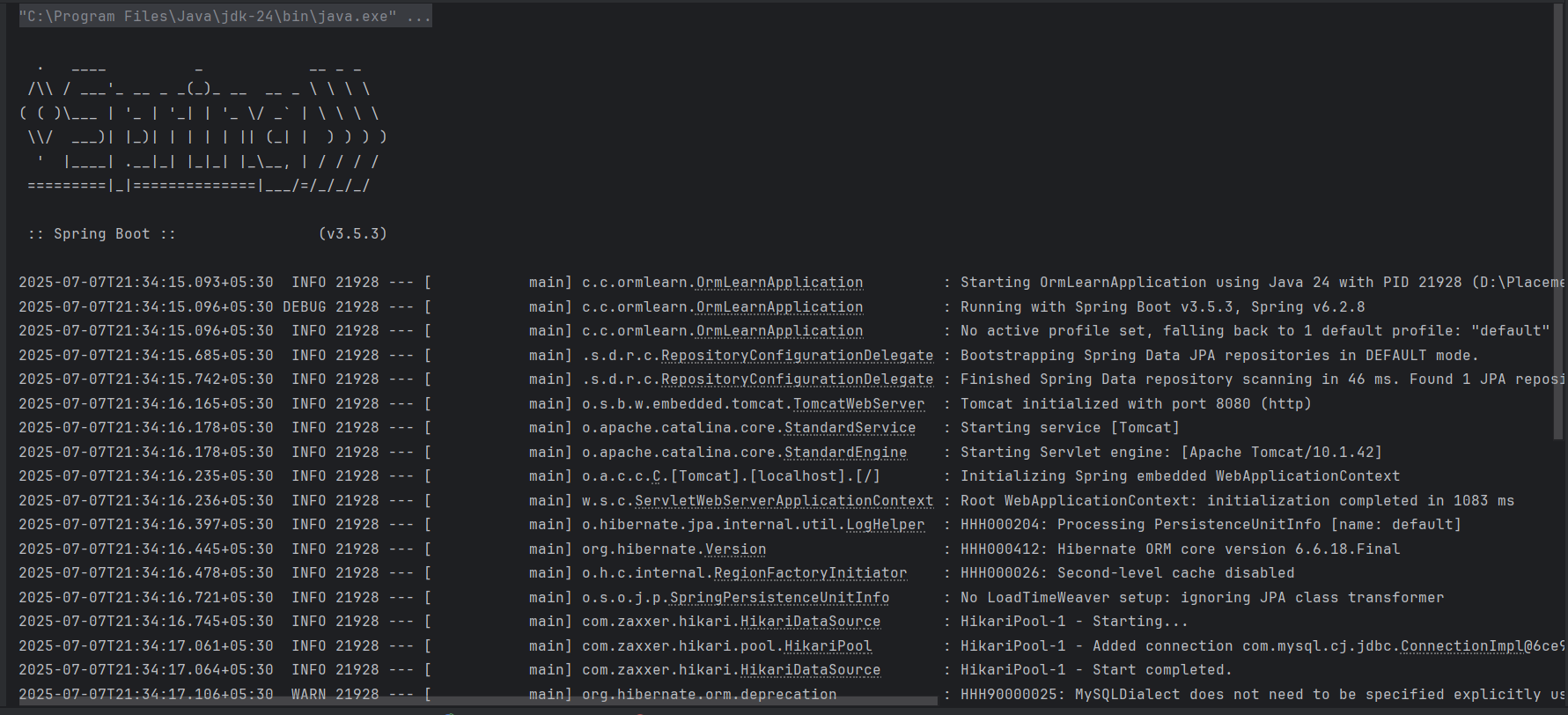
* Create the main class named OrmLearnApplication.java

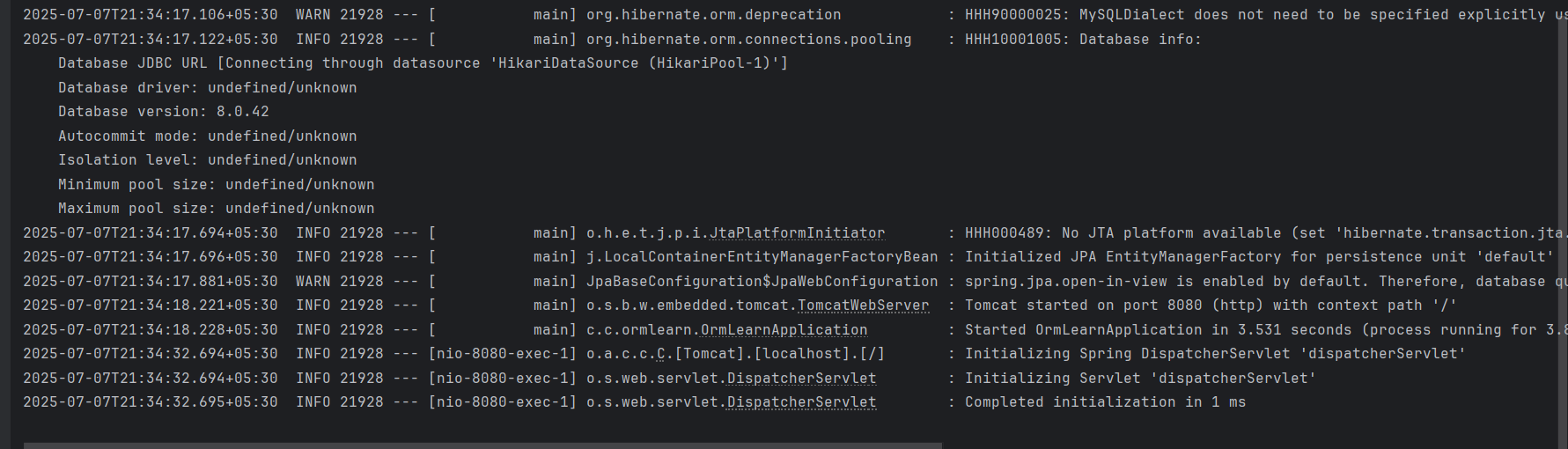
**OrmLearnApplication.java:**

package com.cognizant.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);  
 }  
  
}

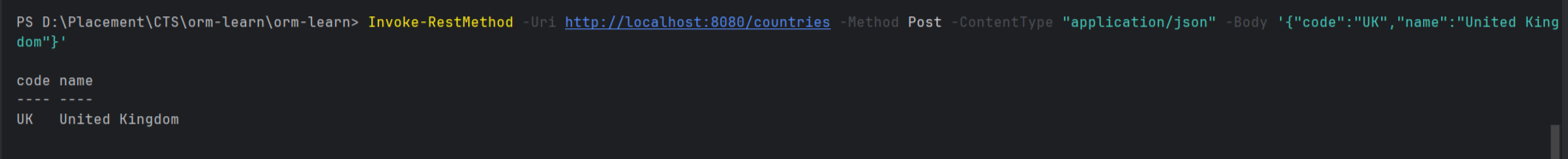
* Build-Rebuild the project.
* Run the OrmLearnApplication.java.

**Output:**

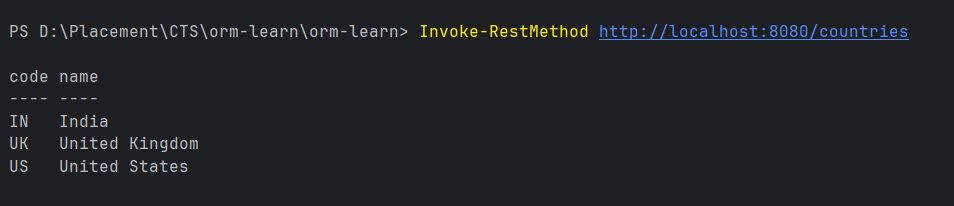
****

****

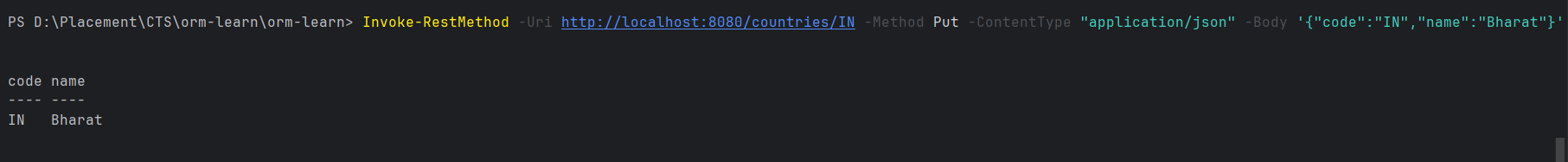
**Create a data:**

****

**Read all data:**

****

**Update a data:**

****

**Delete a data:**

****

**4. Difference between JPA, Hibernate and Spring Data JPA**

**i)Hibernate Implementation:**

**Program:**

* Create a Maven project named hibernate-example.
* Configure the pom.xml with the following dependencies:

**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>  
 <!-- Hibernate Core -->  
 <dependency>  
 <groupId>org.hibernate</groupId>  
 <artifactId>hibernate-core</artifactId>  
 <version>6.4.4.Final</version>  
 </dependency>  
  
 <!-- MySQL Connector -->  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <version>8.3.0</version>  
 </dependency>  
  
 <!-- Jakarta Persistence -->  
 <dependency>  
 <groupId>jakarta.persistence</groupId>  
 <artifactId>jakarta.persistence-api</artifactId>  
 <version>3.1.0</version>  
 </dependency>  
 </dependencies>  
</project>

* Create a file named hibernate.cfg.xml inside src/main/resources

**Hibernate.cfg.xml:**

<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">Data@1234\_</property>  
 <property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>  
 <property name="hibernate.hbm2ddl.auto">update</property>  
 <property name="show\_sql">true</property>  
  
 <mapping class="com.example.hibernate.Employee"/>  
 </session-factory>  
</hibernate-configuration>

* Create the Employee.java under src/main/java/com/example/hibernate package.

**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;  
 }  
  
 // Getters and Setters  
 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; }  
}

* Create the EmployeeDAO.java under src/main/java/com/example/hibernate package.

**EmployeeDAO.java:**

package com.example.hibernate;  
  
import org.hibernate.\*;  
import org.hibernate.cfg.Configuration;  
  
public class EmployeeDAO {  
 private static SessionFactory *factory*;  
  
 static {  
 *factory* = new Configuration()  
 .configure()  
 .addAnnotatedClass(Employee.class)  
 .buildSessionFactory();  
 }  
  
 public Integer addEmployee(Employee employee) {  
 Session session = *factory*.openSession();  
 Transaction tx = null;  
 Integer employeeID = null;  
  
 try {  
 tx = session.beginTransaction();  
 employeeID = (Integer) session.save(employee);  
 tx.commit();  
 } catch (Exception e) {  
 if (tx != null) tx.rollback();  
 e.printStackTrace();  
 } finally {  
 session.close();  
 }  
 return employeeID;  
 }  
  
 public void close() {  
 *factory*.close();  
 }  
}

* Create the App.java under src/main/java/com/example/hibernate package.

**App.java:**

package com.example.hibernate;  
  
public class App {  
 public static void main(String[] args) {  
 EmployeeDAO dao = new EmployeeDAO();  
  
 Employee emp1 = new Employee("Viknesh", "HR");  
 Employee emp2 = new Employee("Ram", "Marketing");  
 Employee emp3 = new Employee("Siva", "Testing");  
  
 dao.addEmployee(emp1);  
 dao.addEmployee(emp2);  
 dao.addEmployee(emp3);  
  
 System.*out*.println(" Employees inserted successfully.");  
  
 dao.close();  
 }  
}

* Open MySQL Workbench or IntelliJ Terminal and execute:

CREATE DATABASE test1;

USE test1;

CREATE TABLE employees (

id INT AUTO\_INCREMENT PRIMARY KEY,

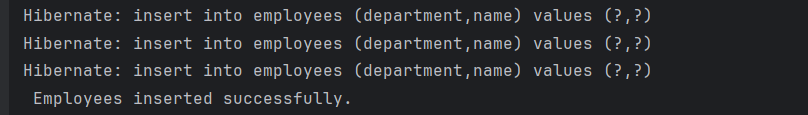
name VARCHAR(255),

department VARCHAR(255)

);

* Run the App.java.

**Output:**

****

**ii)Spring Data JPA Implementation:**

**Program:**

* Update pom.xml.

**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>springdatajpa-example</artifactId>  
 <version>1.0</version>  
 <packaging>jar</packaging>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.5</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <dependencies>  
 <!-- Spring Boot JPA -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <!-- Spring Web -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
  
 <!-- Spring Boot Starter -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
  
 <!-- MySQL Connector -->  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <version>8.3.0</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

* Create the application.properties in src/main/resources.

**application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/test1  
spring.datasource.username=root  
spring.datasource.password=Data@1234\_  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

* Create the Entity class Employee.java in src/main/java/com/example/model.

**Employee.java:**

package com.example.model;  
  
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; }  
}

* Create Repository Interface EmployeeRepository.java in src/main/java/com/example/repository.

**EmployeeRepository.java:**

package com.example.repository;  
  
import com.example.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
}

* Create Service Class EmployeeService.java in java in src/main/java/com/example/service.

**EmployeeService.java:**

package com.example.service;  
  
import com.example.model.Employee;  
import com.example.repository.EmployeeRepository;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
@Service  
public class EmployeeService {  
  
 private final EmployeeRepository repo;  
  
 public EmployeeService(EmployeeRepository repo) {  
 this.repo = repo;  
 }  
  
 @Transactional  
 public void addEmployee(Employee emp) {  
 repo.save(emp);  
 }  
}

* Create Controller EmployeeController.java in java in src/main/java/com/example/controller.

**EmployeeController.java:**

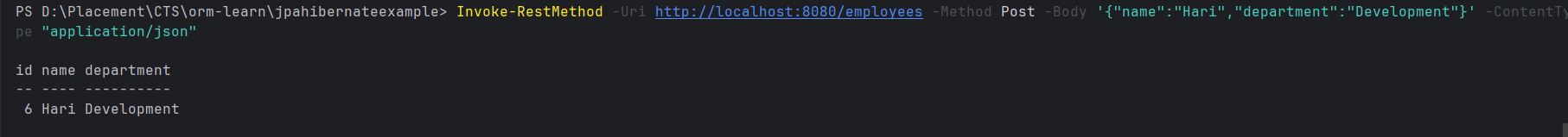
package com.example.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
import com.example.model.Employee;  
import com.example.repository.EmployeeRepository;  
  
@RestController  
@RequestMapping("/employees")  
public class EmployeeController {  
  
 @Autowired  
 private EmployeeRepository repository;  
  
 @PostMapping  
 public Employee createEmployee(@RequestBody Employee employee) {  
 return repository.save(employee);  
 }  
  
 @GetMapping  
 public Iterable<Employee> getAllEmployees() {  
 return repository.findAll();  
 }  
}

* Create Main Class SpringDataJpaApp.java in java in src/main/java/com/example/service.

**SpringDataJpaApp.java:**

package com.example;  
  
import com.example.model.Employee;  
import com.example.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 SpringDataJpaApp implements CommandLineRunner {  
  
 @Autowired  
 private EmployeeService employeeService;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SpringDataJpaApp.class, args);  
 }  
  
 @Override  
 public void run(String... args) {  
 Employee emp = new Employee("Hari", "Engineering");  
 employeeService.addEmployee(emp);  
 System.*out*.println(" Spring Data JPA: Employee inserted");  
 }  
}

* Run the SpringDataJpaApp.java.

**Output:**

# Additional important hands-on

# file name: 1. spring-data-jpa-handson

**HandsOn:**

1. Implement services for managing Country
2. Find a country based on country code
3. Add a new country

**Program:**

* **Database configuration in the application.properties.**

**application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/test1  
spring.datasource.username=root  
spring.datasource.password=Data@1234\_  
  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.show-sql=true

* In MySQL create a database test1.

CREATE TABLE country (

code VARCHAR(5) PRIMARY KEY,

name VARCHAR(50)

);

* Create the entity class named Country.java in src\main\java\com\cts\countryservice\entity\Country.java

**Country.java:**

package com.cts.countryservice.entity;  
  
import jakarta.persistence.Column;  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;  
  
@Entity  
@Table(name = "country")  
public class Country {  
  
 @Id  
 @Column(name = "code")  
 private String code;  
  
 @Column(name = "name")  
 private String name;  
  
 public String getCode() {  
 return code;  
 }  
  
 public void setCode(String code) {  
 this.code = code;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
}

* Create the file named CountryRepository.java in src\main\java\com\cts\countryservice\repository\CountryRepository.java

**CountryRepository.java:**

package com.cts.countryservice.repository;  
  
import com.cts.countryservice.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
import java.util.List;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
  
 List<Country> findByNameContainingIgnoreCase(String name);  
  
}

* Create the file CountryService.java in src\main\java\com\cts\countryservice\service\CountryService.java
* **CountryService.java:**

package com.cts.countryservice.service;  
  
import com.cts.countryservice.entity.Country;  
import com.cts.countryservice.exception.CountryNotFoundException;  
import java.util.List;  
  
public interface CountryService {  
 Country getCountryByCode(String code);  
 Country addCountry(Country country);  
 Country updateCountry(Country country);  
 void deleteCountry(String code);  
 List<Country> searchCountriesByName(String name);  
 Country findCountryByCode(String code) throws CountryNotFoundException;  
}

* Create the file named **CountryServiceImpl.java in** src\main\java\com\cts\countryservice\service\impl\CountryServiceImpl.java
* **CountryServiceImpl.java:**

package com.cts.countryservice.service.impl;  
  
import com.cts.countryservice.entity.Country;  
import com.cts.countryservice.exception.CountryNotFoundException;  
import com.cts.countryservice.repository.CountryRepository;  
import com.cts.countryservice.service.CountryService;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class CountryServiceImpl implements CountryService {  
  
 private final CountryRepository repository;  
  
 public CountryServiceImpl(CountryRepository repository) {  
 this.repository = repository;  
 }  
  
 @Override  
 public Country getCountryByCode(String code) {  
 return repository.findById(code).orElse(null);  
 }  
  
 @Override  
 public Country addCountry(Country country) {  
 return repository.save(country);  
 }  
  
 @Override  
 public Country updateCountry(Country country) {  
 return repository.save(country);  
 }  
  
 @Override  
 public void deleteCountry(String code) {  
 repository.deleteById(code);  
 }  
  
 @Override  
 public List<Country> searchCountriesByName(String name) {  
 return repository.findByNameContainingIgnoreCase(name);  
 }  
  
 @Override  
 @Transactional  
 public Country findCountryByCode(String countryCode) throws CountryNotFoundException {  
 Optional<Country> result = repository.findById(countryCode);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country not found for code: " + countryCode);  
 }  
 return result.get();  
 }  
}

* Create a file named CountryController.java in src\main\java\com\cts\countryservice\controller\CountryController.java

**CountryController.java:**

package com.cts.countryservice.controller;  
import com.cts.countryservice.exception.CountryNotFoundException;  
  
import com.cts.countryservice.entity.Country;  
import com.cts.countryservice.service.CountryService;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/countries")  
public class CountryController {  
  
 private final CountryService service;  
  
 public CountryController(CountryService service) {  
 this.service = service;  
 }  
  
 @GetMapping("/{code}")  
 public Country getByCode(@PathVariable String code) {  
 return service.getCountryByCode(code);  
 }  
  
 @PostMapping  
 public Country add(@RequestBody Country country) {  
 return service.addCountry(country);  
 }  
  
 @PutMapping  
 public Country update(@RequestBody Country country) {  
 return service.updateCountry(country);  
 }  
  
 @DeleteMapping("/{code}")  
 public void delete(@PathVariable String code) {  
 service.deleteCountry(code);  
 }  
  
 @GetMapping("/search")  
 public List<Country> search(@RequestParam String name) {  
 return service.searchCountriesByName(name);  
 }  
 @GetMapping("/find/{code}")  
 public Country findByCode(@PathVariable String code) throws CountryNotFoundException {  
 return service.findCountryByCode(code);  
 }  
  
}

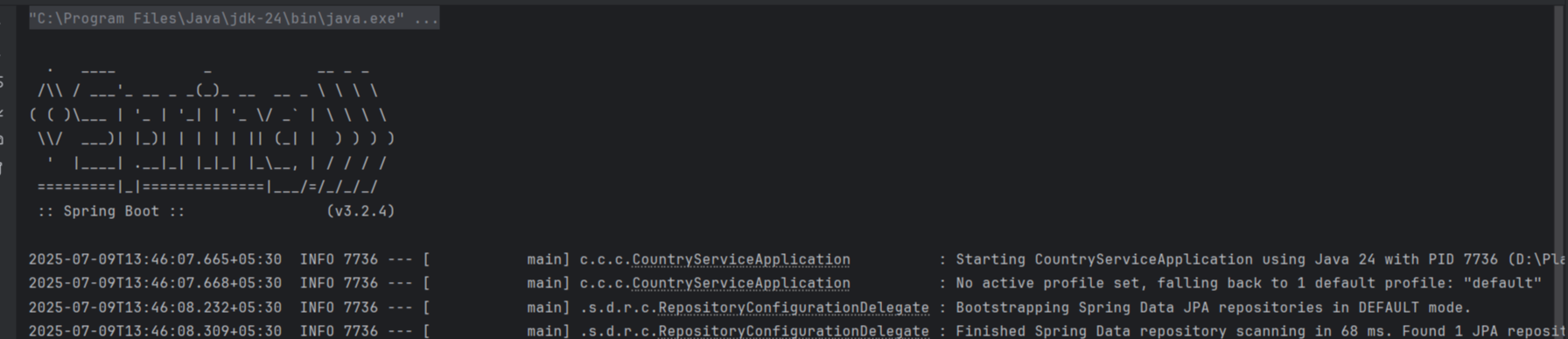
* Create a file named CountryNotFoundException.java in src\main\java\com\cts\countryservice\exception\CountryNotFoundException.java
* **CountryNotFoundException.java:**

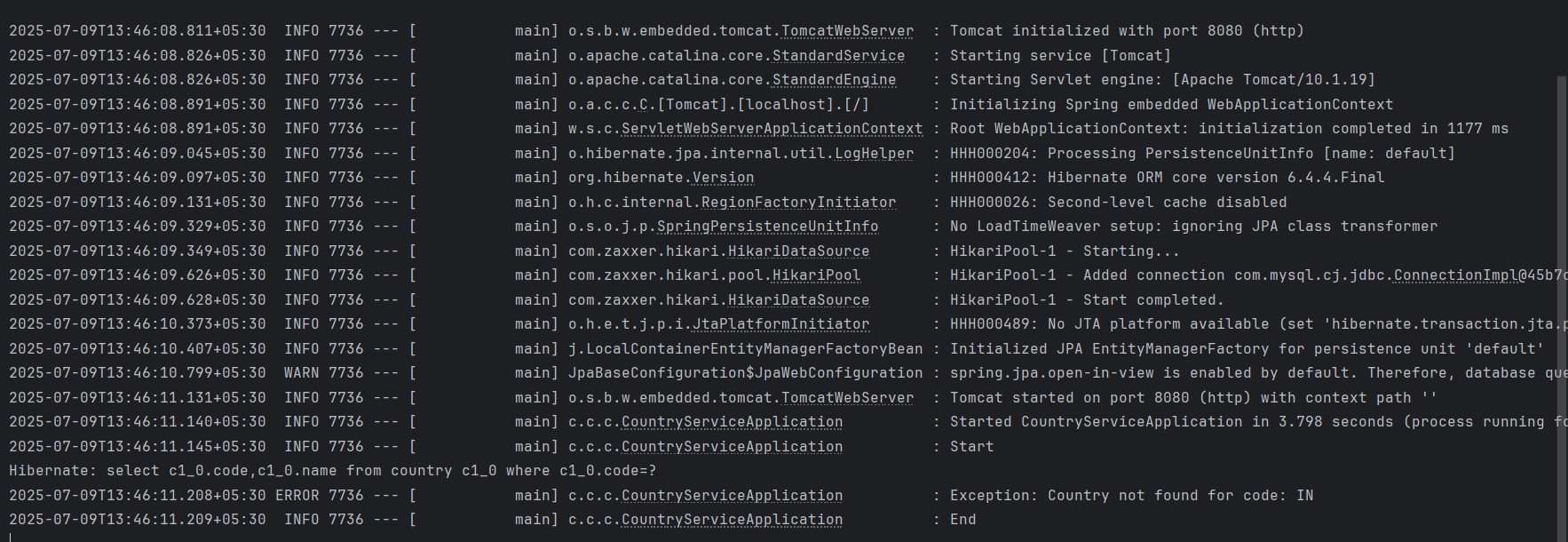
package com.cts.countryservice.exception;  
  
public class CountryNotFoundException extends RuntimeException {  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

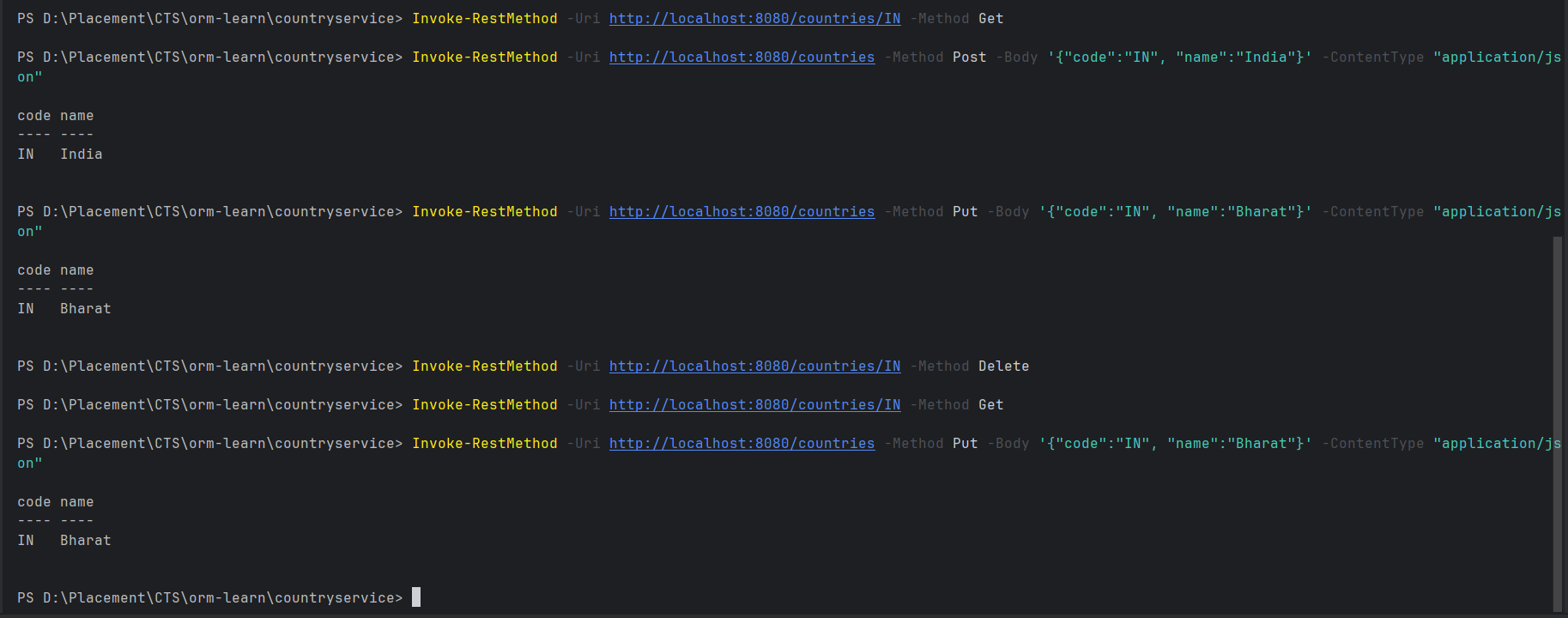
* Create a file named CountryServiceApplication.java in src\main\java\com\cts\countryservice\CountryServiceApplication.java
* **CountryServiceApplication.java:**

package com.cts.countryservice;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.CommandLineRunner;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import com.cts.countryservice.service.CountryService;  
import com.cts.countryservice.entity.Country;  
import com.cts.countryservice.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
  
  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class CountryServiceApplication implements CommandLineRunner {  
  
 @Autowired  
 private CountryService countryService;  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(CountryServiceApplication.class);  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(CountryServiceApplication.class, args);  
 }  
  
 @Override  
 public void run(String... args) throws Exception {  
 getAllCountriesTest();  
 }  
  
 private void getAllCountriesTest() {  
 *LOGGER*.info("Start");  
 try {  
 Country country = countryService.findCountryByCode("IN");  
 *LOGGER*.debug("Country: {}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Exception: {}", e.getMessage());  
 }  
 *LOGGER*.info("End");  
 }  
}

* Build the project and run the CountryServiceApplication.java.
* **Output:**

****

* **=**
* Adding the country and getting the country by the country code

****

# File name: 2. spring-data-jpa-handson

# i)Demonstrate implementation of Query Methods feature of Spring Data JPA

* **Program:**
* Create a new maven project and create package src/main/java/com/cognizant/ormlearn
* Update the pom.xml.
* **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>  
  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.2.0</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <groupId>com.cognizant</groupId>  
 <artifactId>orm-learn</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>orm-learn</name>  
 <description>Spring Boot ORM Learn Project</description>  
 <packaging>jar</packaging>  
  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
  
 <dependencies>  
 <!-- Spring Boot Core Starter -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
  
 <!-- Spring Boot Web Starter -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
  
 <!-- Spring Boot Data JPA -->  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
  
 <!-- MySQL JDBC Driver -->  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 </dependency>  
  
 <!-- Lombok (Optional) -->  
 <dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <optional>true</optional>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Spring Boot Maven Plugin -->  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

* Configure the application.properties in the path src/main/resources/application.properties.
* **application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn1  
spring.datasource.username=root  
spring.datasource.password=Data@1234\_  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true

* Create file named Country.java in the path com.cognizant.ormlearn.model
* **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 + "]";  
 }  
}

* Create file named Stock.java in the path com.cognizant.ormlearn.model
* **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;  
  
 // Getters and Setters  
  
 @Override  
 public String toString() {  
 return "Stock{" +  
 "id=" + id +  
 ", code='" + code + '\'' +  
 ", date=" + date +  
 ", open=" + open +  
 ", close=" + close +  
 ", volume=" + volume +  
 '}';  
 }  
}

* Create file named CountryRepository.java in the path com.cognizant.ormlearn.repository
* **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);  
}

* Create file named StockRepository.java in the path com.cognizant.ormlearn.repository
* **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);  
}

* Create file named OrmLearnApplication.java in the path com.cognizant.ormlearn
* **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()));  
 }  
}

* In MySQL database create a table and insert the sample data

CREATE DATABASE ormlearn1;

USE ormlearn1;

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(100)

);

CREATE TABLE stock (

st\_id INT PRIMARY KEY AUTO\_INCREMENT,

st\_code VARCHAR(10),

st\_date DATE,

st\_open DECIMAL(10,2),

st\_close DECIMAL(10,2),

st\_volume BIGINT

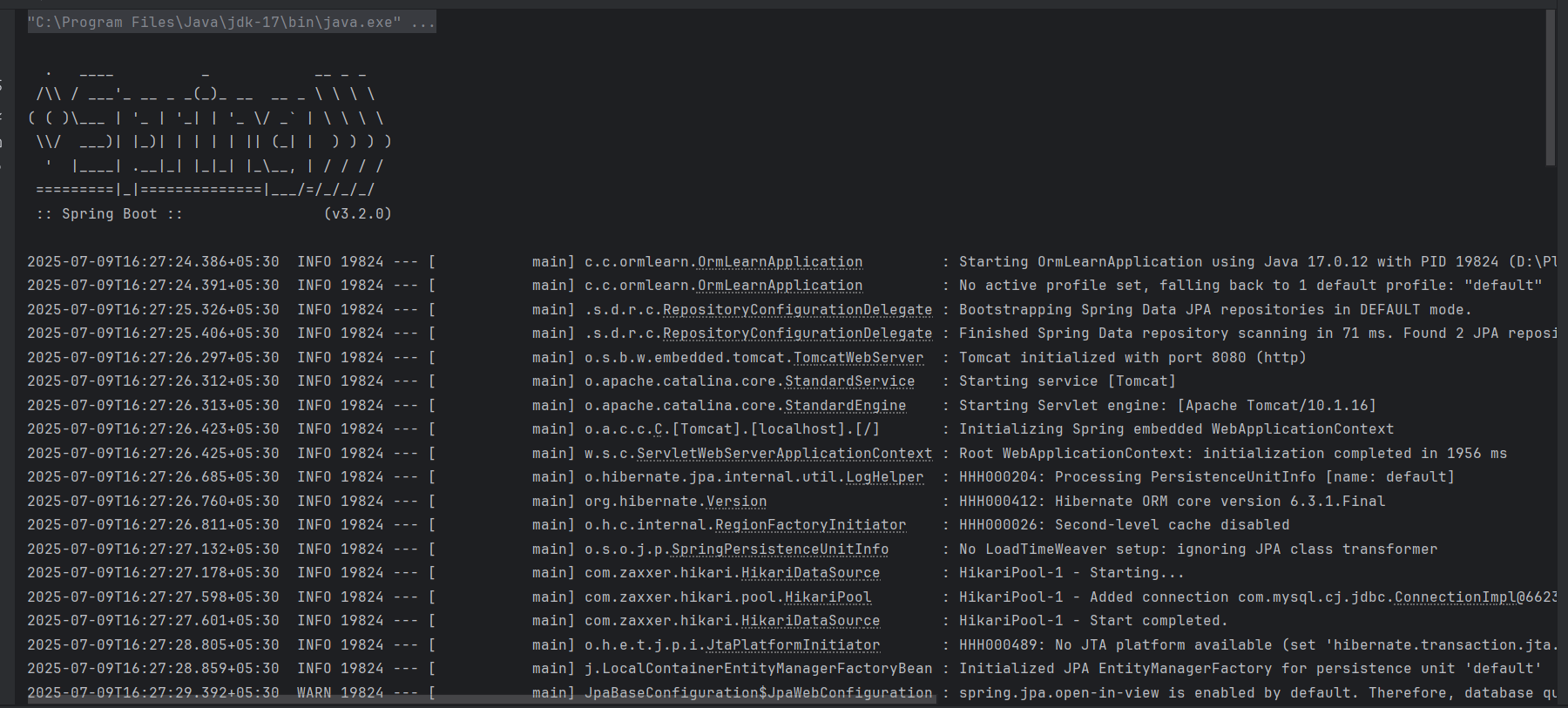
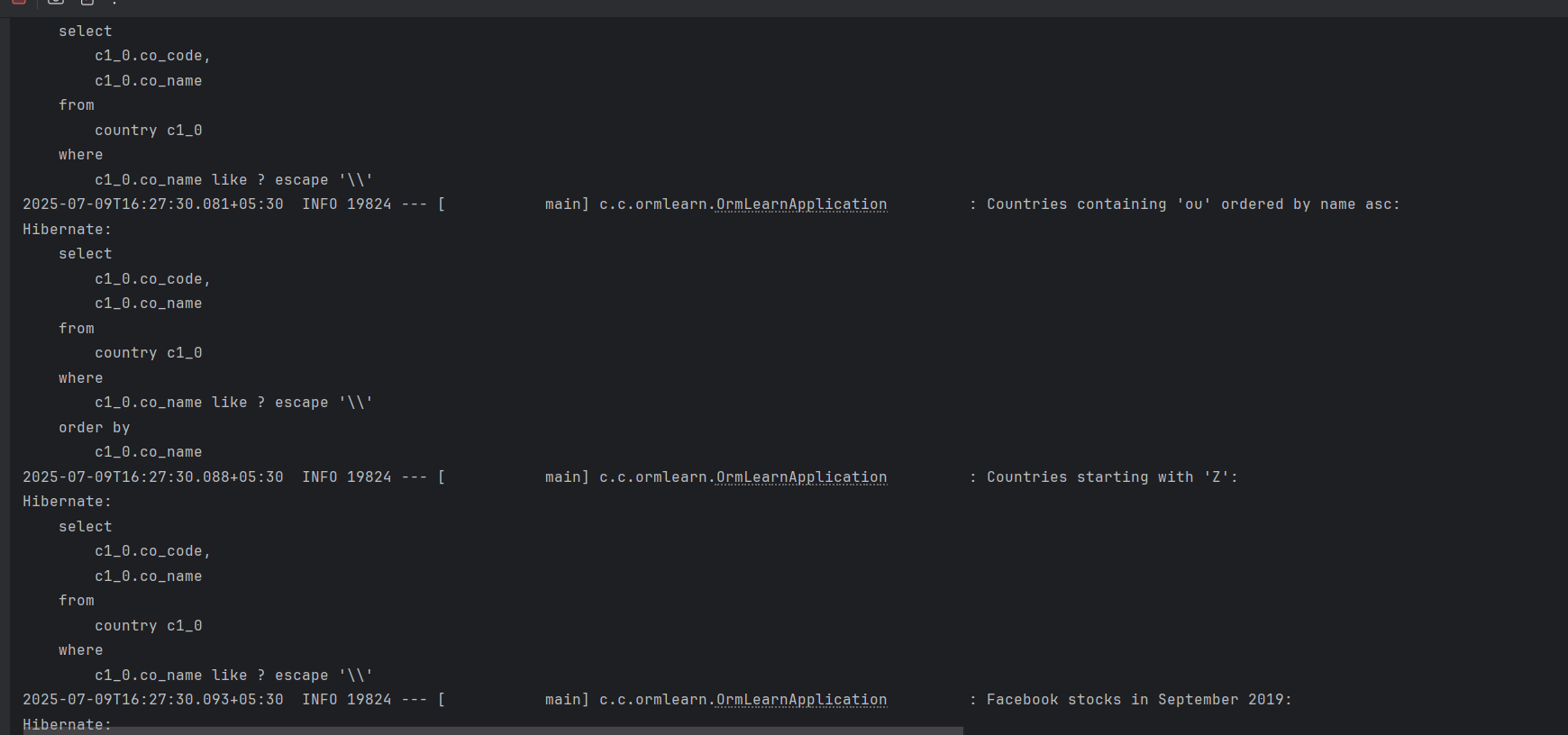
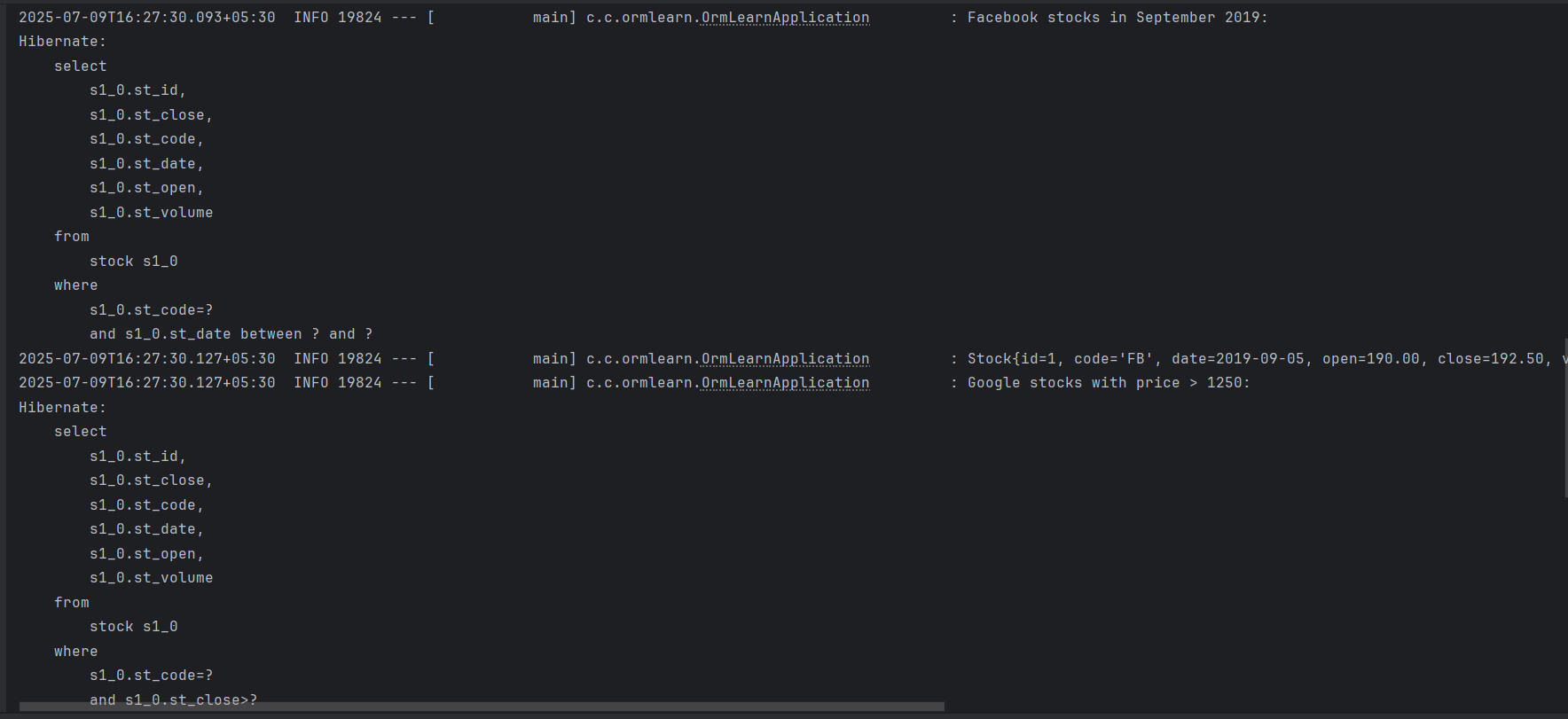
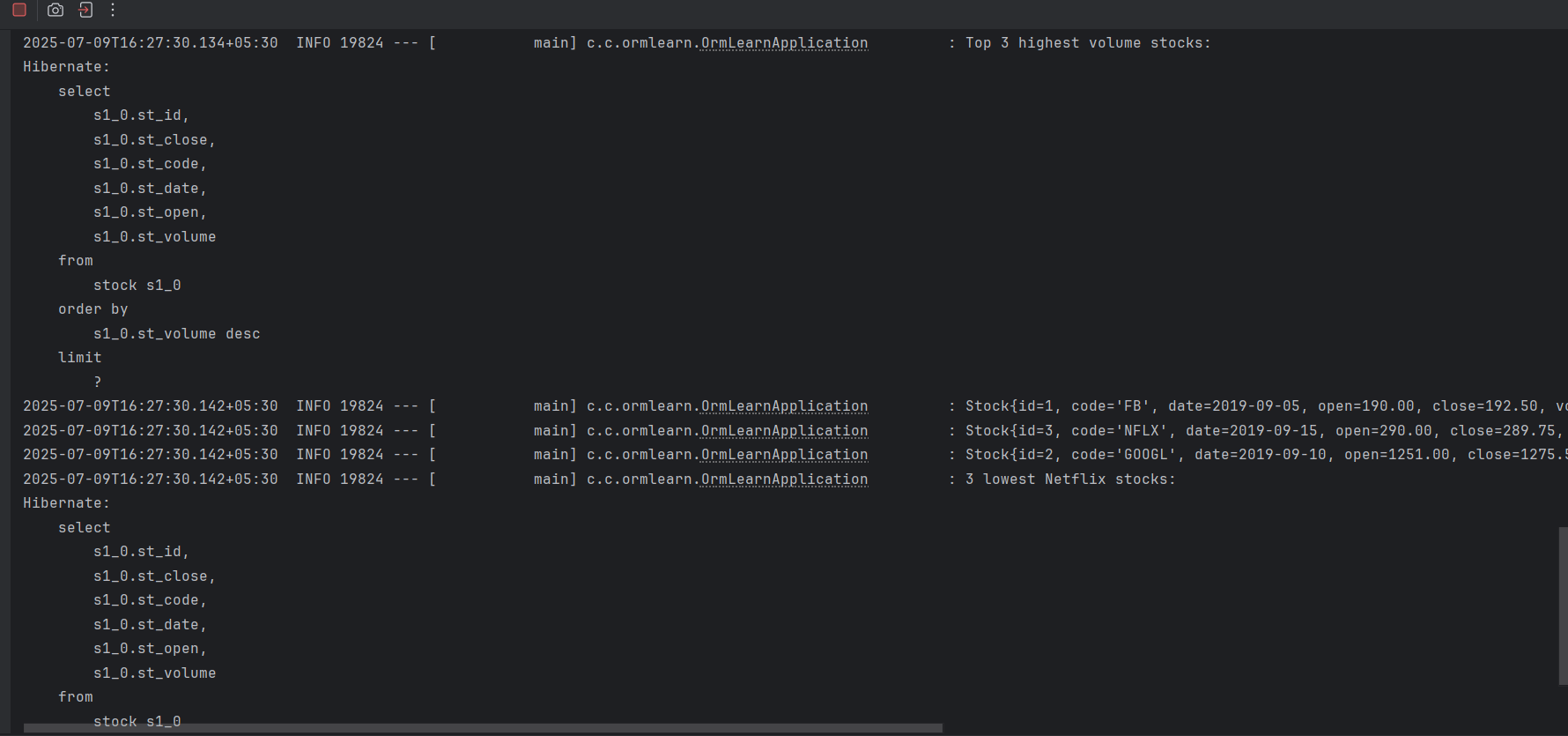
);

INSERT INTO stock (st\_code, st\_date, st\_open, st\_close, st\_volume) VALUES

('FB', '2019-09-05', 190.00, 192.50, 12000000),

('GOOGL', '2019-09-10', 1251.00, 1275.50, 8000000),

('NFLX', '2019-09-15', 290.00, 289.75, 9500000);

* Build the project and run the OrmLearnApplication.java.
* **Output:**
* ****
* ****
* ****
* ****

# (ii) Demonstrate Implementation of O/R Mapping in Spring Data JPA

* **Program:**
* Create MySQL Schema and tables .

CREATE DATABASE ormlearn2;

USE ormlearn2;

CREATE TABLE department (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(50)

);

CREATE TABLE employee (

id INT PRIMARY KEY AUTO\_INCREMENT,

name VARCHAR(100),

salary DECIMAL(10,2),

dept\_id INT,

FOREIGN KEY (dept\_id) REFERENCES department(id)

);

* Insert the sample data

INSERT INTO department (name) VALUES ('HR'), ('Engineering');

INSERT INTO employee (name, salary, dept\_id) VALUES

('Hari', 50000, 1),

('Ravi', 60000, 2),

('Viknesh', 65000, 2);

* Configure application.properties in src/main/resources/application.properties
* **Application.properties:**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn2  
spring.datasource.username=root  
spring.datasource.password=Data@1234\_  
spring.jpa.hibernate.ddl-auto=none  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true

* Create entity class Department.java, Employee.java in com.cognizant.ormlearn.model package
* **Department.java:**

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "department")  
public class Department {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 private String name;  
  
 @OneToMany(mappedBy = "department")  
 private List<Employee> employees;  
  
 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 List<Employee> getEmployees() { return employees; }  
 public void setEmployees(List<Employee> employees) { this.employees = employees; }  
  
 @Override  
 public String toString() {  
 return "Department [id=" + id + ", name=" + name + "]";  
 }  
}

* **Employee.java:**

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.\*;  
  
import java.math.BigDecimal;  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 private String name;  
  
 private BigDecimal salary;  
  
 @ManyToOne  
 @JoinColumn(name = "dept\_id")  
 private Department department;  
  
 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 BigDecimal getSalary() { return salary; }  
 public void setSalary(BigDecimal salary) { this.salary = salary; }  
  
 public Department getDepartment() { return department; }  
 public void setDepartment(Department department) { this.department = department; }  
  
 @Override  
 public String toString() {  
 return "Employee [id=" + id + ", name=" + name + ", salary=" + salary +  
 ", department=" + department.getName() + "]";  
 }  
}

* Create repository interfaces EmployeeRepository.java,DepartmentRepository.java in com.cognizant.ormlearn.repository package.
* **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> {  
}

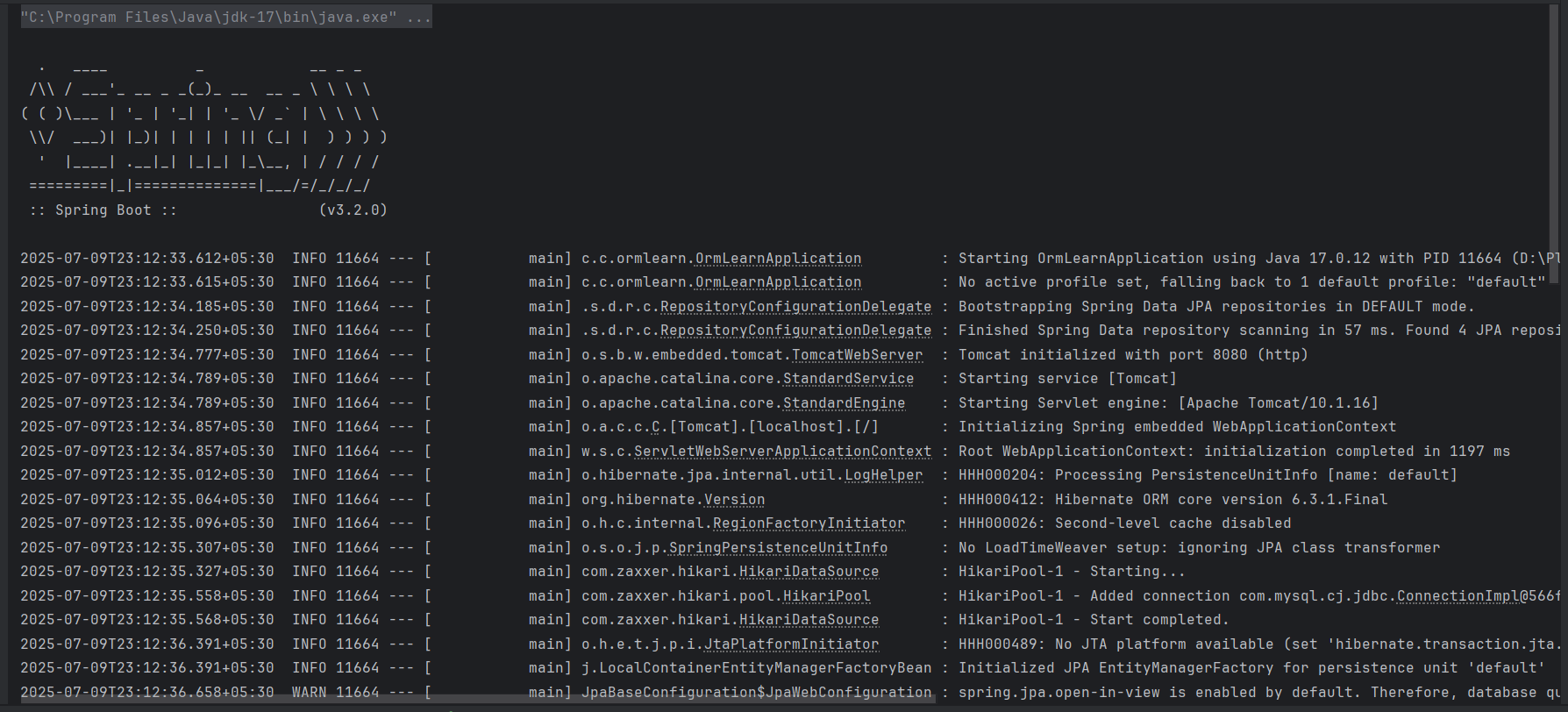
* **DepartmentRepository.java:**

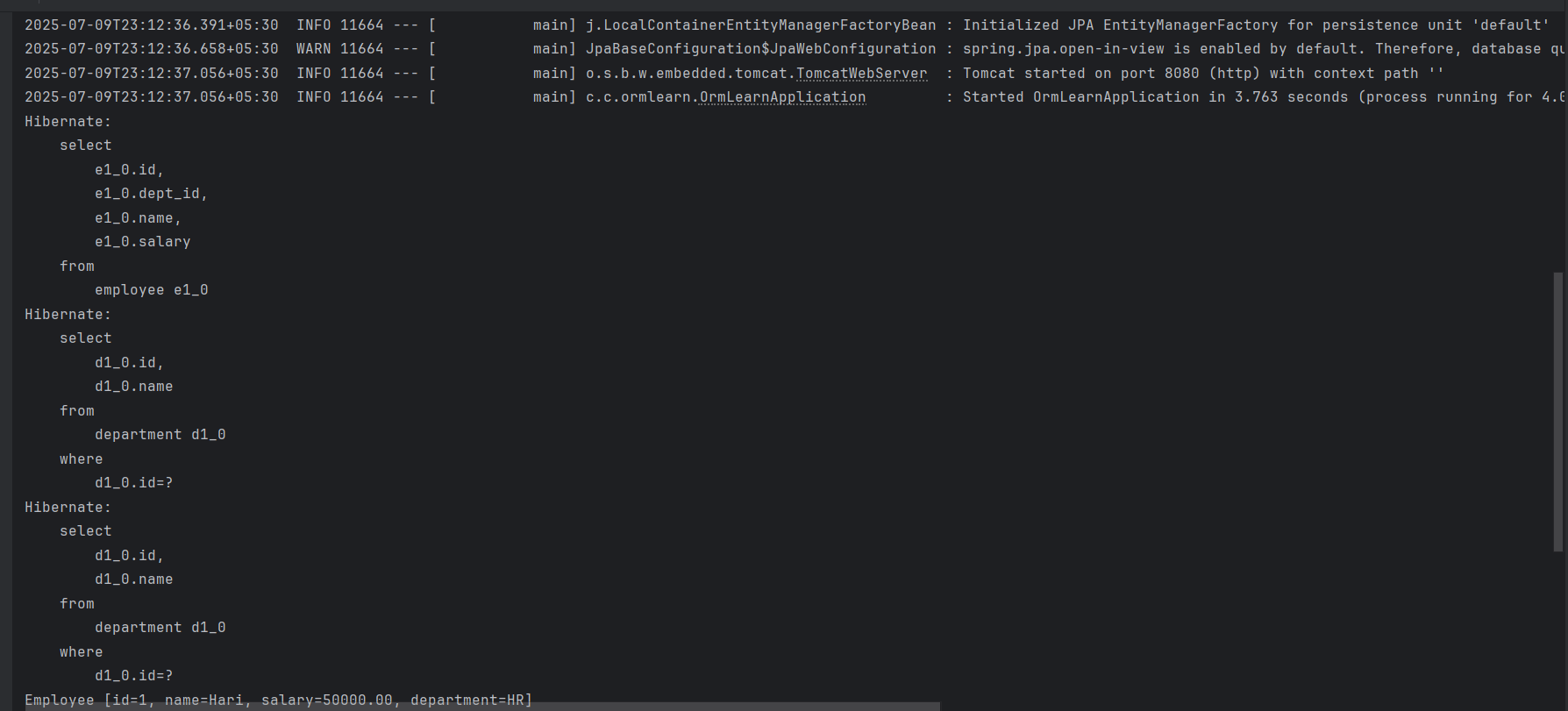
package com.cognizant.ormlearn.repository;  
  
import com.cognizant.ormlearn.model.Department;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface DepartmentRepository extends JpaRepository<Department, Integer> {  
}

* Create the main class OrmLearnApplication.java in src/main/java/com/cognizant/ormlearn/OrmLearnApplication.java
* **OrmLearnApplication.java:**

package com.cognizant.ormlearn;  
  
import com.cognizant.ormlearn.model.Department;  
import com.cognizant.ormlearn.model.Employee;  
import com.cognizant.ormlearn.repository.DepartmentRepository;  
import com.cognizant.ormlearn.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
import org.springframework.transaction.annotation.Transactional;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
import java.util.List;  
  
@SpringBootApplication  
public class OrmLearnApplication implements CommandLineRunner {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Autowired  
 private DepartmentRepository departmentRepository;  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(OrmLearnApplication.class, args);  
 }  
  
 @Override  
 @Transactional  
 public void run(String... args) {  
 List<Employee> employees = employeeRepository.findAll();  
 for (Employee emp : employees) {  
 System.*out*.println(emp);  
 }  
  
 Department dept = departmentRepository.findById(2).get();  
 System.*out*.println("Department: " + dept.getName());  
 for (Employee e : dept.getEmployees()) {  
 System.*out*.println(e.getName());  
 }  
 }  
}

* Run the **OrmLearnApplication.java.**
* **Output:**

****

****

****

# File:3. spring-data-jpa-handson

# Demonstrate writing Hibernate Query Language and Native Query

* **Program:**
* Configure the application.properties

Application.properties:

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn2  
spring.datasource.username=root  
spring.datasource.password=Data@1234\_  
spring.jpa.hibernate.ddl-auto=none  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.format\_sql=true

* Creat the entity classes Employee.java,Department.java,Skill.java.

**Employee.java:**

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.\*;  
import java.math.BigDecimal;  
import java.util.List;  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 private String name;  
 private BigDecimal salary;  
 private Boolean permanent;  
  
 @ManyToOne  
 @JoinColumn(name = "dept\_id")  
 private Department department;  
  
 @ManyToMany  
 @JoinTable(name = "employee\_skill",  
 joinColumns = @JoinColumn(name = "employee\_id"),  
 inverseJoinColumns = @JoinColumn(name = "skill\_id"))  
 private List<Skill> skillList;  
  
 public int getId() {  
 return id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public BigDecimal getSalary() {  
 return salary;  
 }  
  
 public boolean isPermanent() {  
 return permanent;  
 }  
  
 public Department getDepartment() {  
 return department;  
 }  
  
 public List<Skill> getSkillList() {  
 return skillList;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setSalary(BigDecimal salary) {  
 this.salary = salary;  
 }  
  
 public void setPermanent(boolean permanent) {  
 this.permanent = permanent;  
 }  
  
 public void setDepartment(Department department) {  
 this.department = department;  
 }  
  
 public void setSkillList(List<Skill> skillList) {  
 this.skillList = skillList;  
 }  
  
 @Override  
 public String toString() {  
 return "Employee{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", salary=" + salary +  
 ", permanent=" + permanent +  
 ", department=" + (department != null ? department.getName() : null) +  
 '}';  
 }  
}

* **Department.java:**

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "department")  
public class Department {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 private String name;  
  
 @OneToMany(mappedBy = "department")  
 private List<Employee> employees;  
  
 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 List<Employee> getEmployees() { return employees; }  
 public void setEmployees(List<Employee> employees) { this.employees = employees; }  
  
 @Override  
 public String toString() {  
 return "Department [id=" + id + ", name=" + name + "]";  
 }  
}

* **Skill.java:**

package com.cognizant.ormlearn.model;  
  
import jakarta.persistence.\*;  
import java.util.List;  
  
@Entity  
@Table(name = "skill")  
public class Skill {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 private String name;  
  
 @ManyToMany(mappedBy = "skillList")  
 private List<Employee> employeeList;  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public List<Employee> getEmployeeList() {  
 return employeeList;  
 }  
  
 public void setEmployeeList(List<Employee> employeeList) {  
 this.employeeList = employeeList;  
 }  
  
 @Override  
 public String toString() {  
 return "Skill{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 '}';  
 }  
}

* Create the reposiotories named EmployeeRepository.java

**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 LEFT JOIN FETCH e.department d LEFT JOIN FETCH e.skillList WHERE e.permanent = true")  
 List<Employee> getAllPermanentEmployees();  
  
  
 @Query("SELECT AVG(e.salary) FROM Employee e WHERE e.department.id = :id")  
 double getAverageSalary(@Param("id") int departmentId);  
  
 @Query(value = "SELECT \* FROM employee", nativeQuery = true)  
 List<Employee> getAllEmployeesNative();  
}

* Create the interface named EmployeeService.java and class named EmployeeServiceImpl.java.

**EmployeeService.java:**

package com.cognizant.ormlearn.service;  
  
import com.cognizant.ormlearn.model.Employee;  
import java.util.List;  
  
public interface EmployeeService {  
 List<Employee> getAllPermanentEmployees();  
 double getAverageSalary(int departmentId);  
 List<Employee> getAllEmployeesNative();  
}

* **EmployeeServiceImpl.java:**

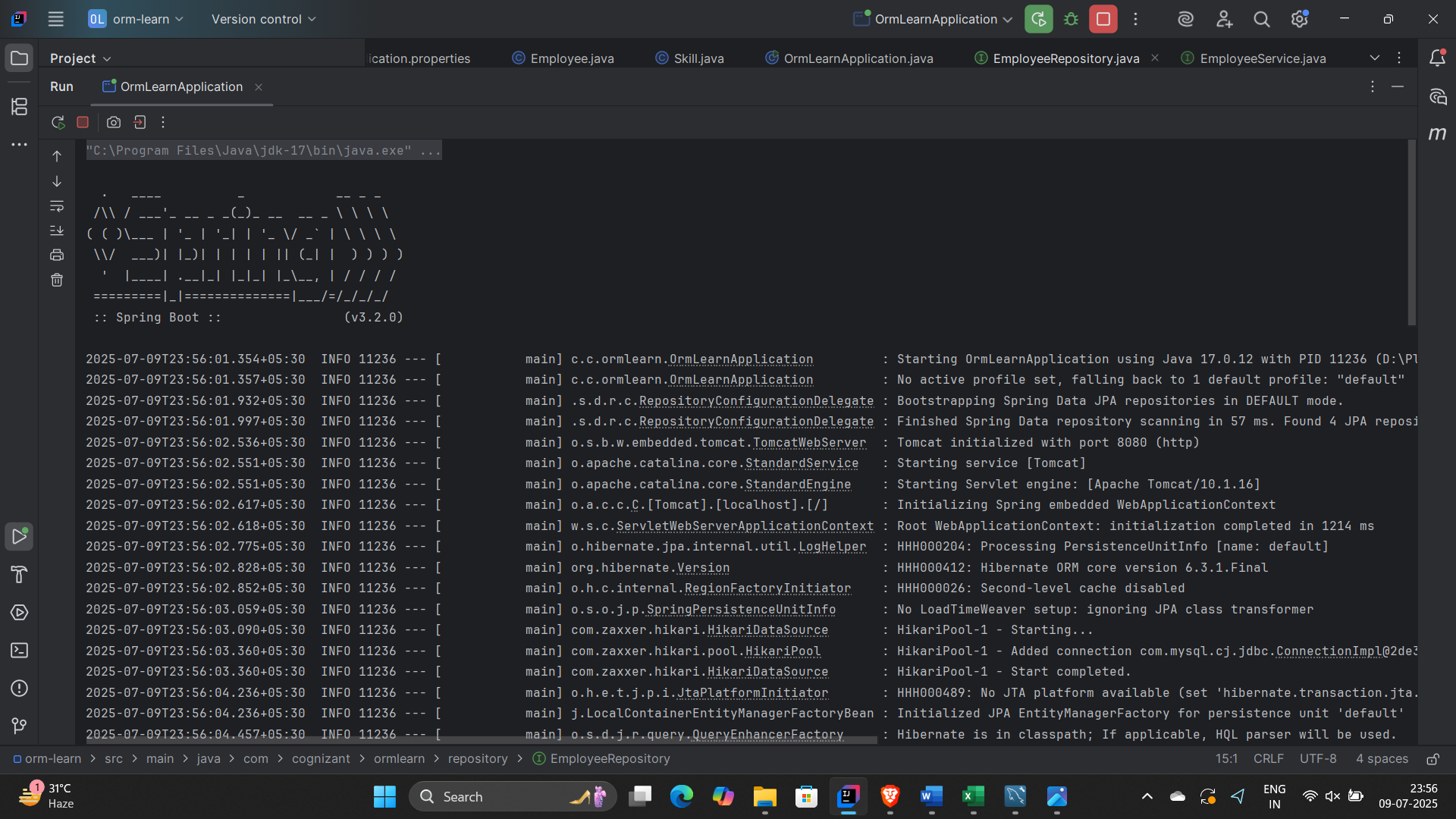
package com.cognizant.ormlearn.service;  
  
import com.cognizant.ormlearn.model.Employee;  
import com.cognizant.ormlearn.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class EmployeeServiceImpl implements EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Override  
 public List<Employee> getAllPermanentEmployees() {  
 return employeeRepository.getAllPermanentEmployees();  
 }  
  
 @Override  
 public double getAverageSalary(int departmentId) {  
 return employeeRepository.getAverageSalary(departmentId);  
 }  
  
 @Override  
 public List<Employee> getAllEmployeesNative() {  
 return employeeRepository.getAllEmployeesNative();  
 }  
}

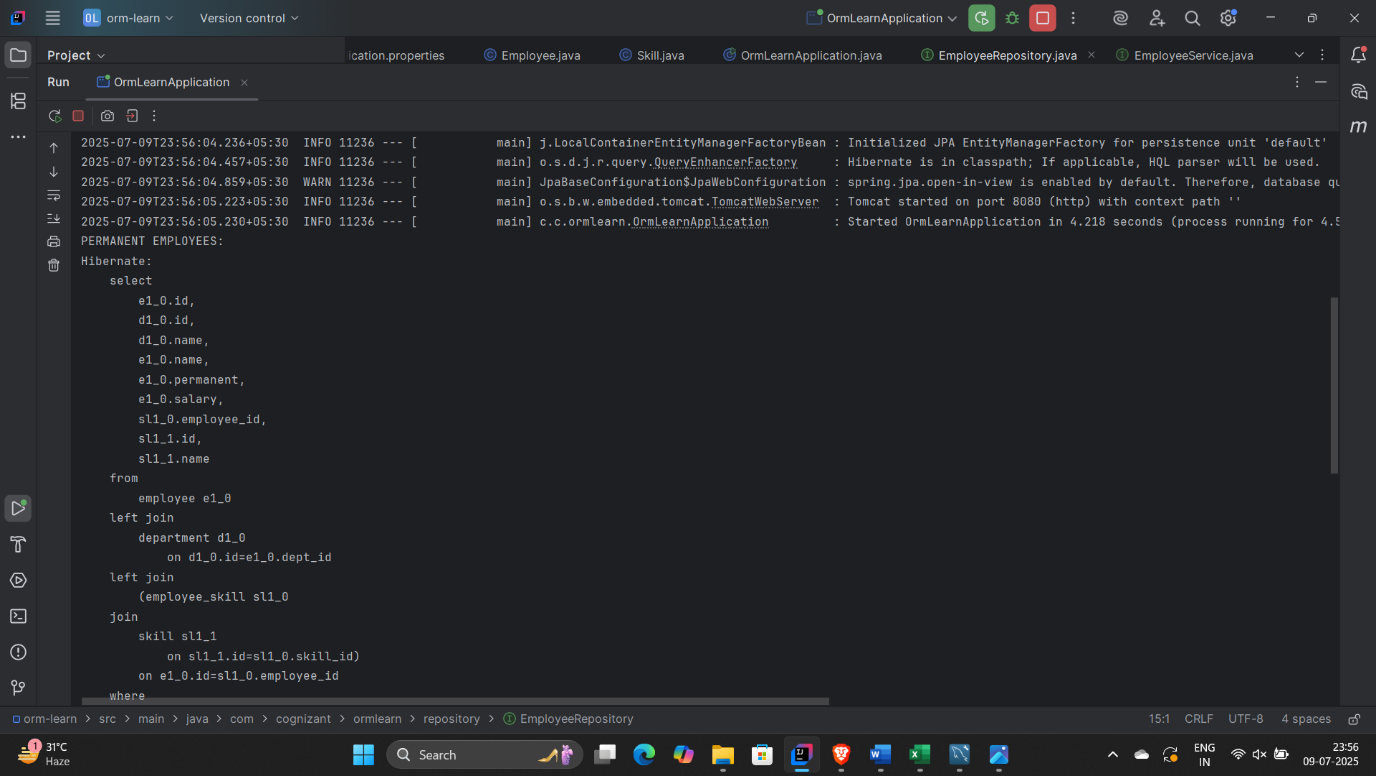
* Create the main class OrmLearnApplication.java.

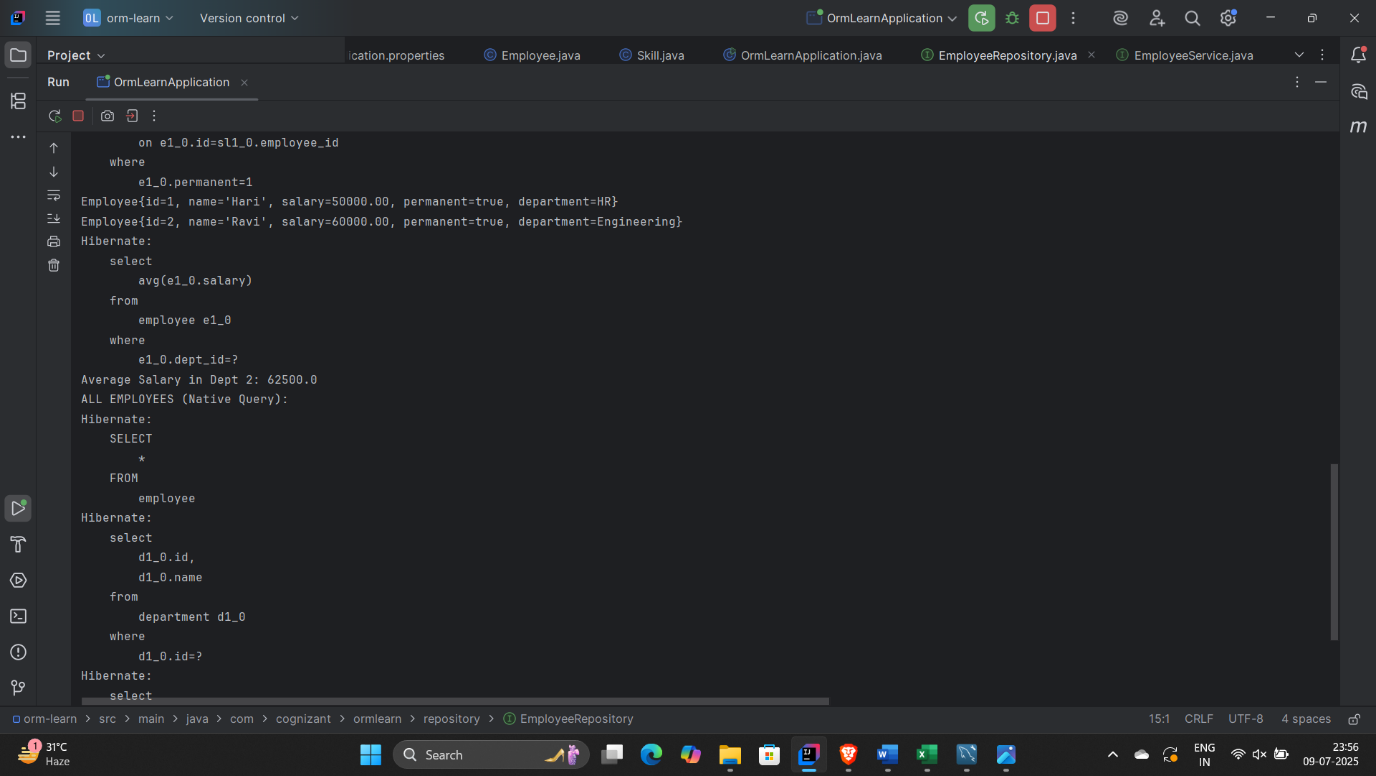
**OrmLearnApplication.java:**

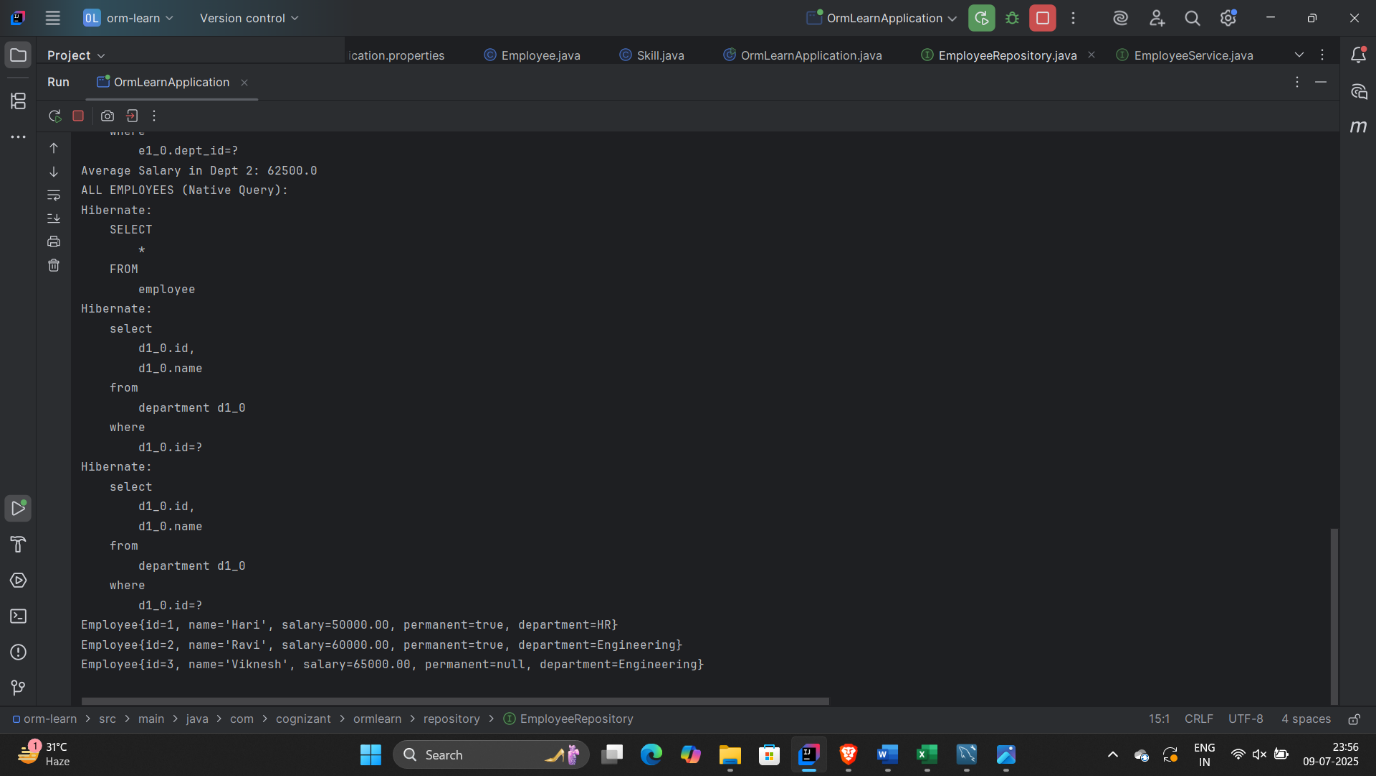
package com.cognizant.ormlearn;  
  
import com.cognizant.ormlearn.model.Employee;  
import com.cognizant.ormlearn.service.EmployeeService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.boot.CommandLineRunner;  
import org.springframework.boot.SpringApplication;  
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 {  
 testGetAllPermanentEmployees();  
 testGetAverageSalary();  
 testGetAllEmployeesNative();  
 }  
  
 public void testGetAllPermanentEmployees() {  
 System.*out*.println("PERMANENT EMPLOYEES:");  
 employeeService.getAllPermanentEmployees()  
 .forEach(e -> {  
 System.*out*.println(e);  
 e.getSkillList().forEach(skill -> System.*out*.println("Skill: " + skill.getName()));  
 });  
 }  
  
 public void testGetAverageSalary() {  
 double avg = employeeService.getAverageSalary(2);  
 System.*out*.println("Average Salary in Dept 2: " + avg);  
 }  
  
 public void testGetAllEmployeesNative() {  
 System.*out*.println("ALL EMPLOYEES (Native Query):");  
 employeeService.getAllEmployeesNative()  
 .forEach(System.*out*::println);  
 }  
}

**Output:**

****

****

****

****