**MANDATORY HANDS**

**Spring Data JPA - Quick Example**

**OrmLearnApplication.java**

package com.cognizant.orm\_learn;

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

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.List;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

countries.forEach(country -> LOGGER.debug("Country: {}", country));

LOGGER.info("End");

}

}

**Country.java**

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

**import** jakarta.persistence.\*;

@Entity

@Table(name = "country")

**public** **class** Country {

@Id

@Column(name = "code")

**private** String code;

@Column(name = "name")

**private** String name;

**public** String getCode() { **return** code; }

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

**public** String getName() { **return** name; }

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

@Override

**public** String toString() {

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

}

}

**CountryRepository.java**

package com.cognizant.orm\_learn.repository;

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

import org.springframework.stereotype.Repository;

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

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java**

package com.cognizant.orm\_learn.service;

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

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

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**Application.properties**

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=debug

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

# DB Config

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=SYSTEM

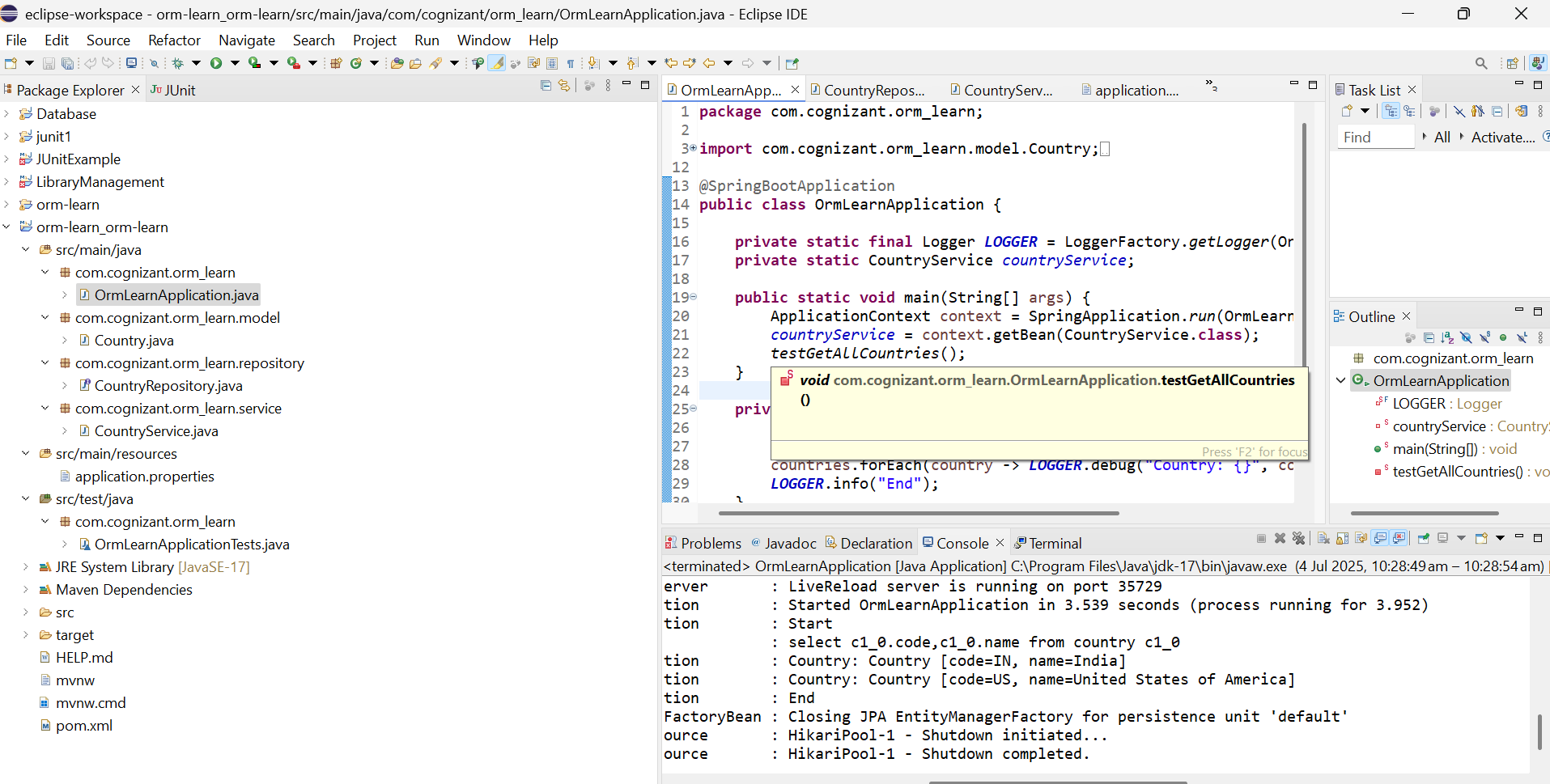
# Hibernate Dialect for Hibernate 6

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

# Schema validation (use validate if table already exists)

spring.jpa.hibernate.ddl-auto=validate

**Output:**

****

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

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification | Implementation of JPA | Abstraction layer on top of JPA |
| **Provides** | API/annotations only | ORM features + native SQL, HQL, caching | Repository interfaces and query methods |
| **Requires Implementation** | Yes – needs a provider like Hibernate | No – it's already a provider | No – uses JPA provider (like Hibernate) underneath |
| **Boilerplate Code** | Requires EntityManager, transactions manually | Slightly better but still requires code for common operations | Reduces boilerplate via JpaRepository, CrudRepository etc. |
| **Integration in Spring** | Manual configuration required | Can be configured easily | Auto-configured in Spring Boot |
| **Query Support** | JPQL | JPQL + HQL + Criteria API | JPQL + method names + @Query support |
| **Ease of Use** | Intermediate | Moderate | Very high |

**Bottom of Form**

**ADDITIONAL HANDS ON**

**Implement services for managing Country**

**CountryService.java**

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

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

**import** java.util.List;

**public** **interface** CountryService {

Country findCountryByCode(String code);

List<Country> getAllCountries();

}

**CountryServiceImpl.java**

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

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

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

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

**import** org.springframework.stereotype.Service;

**import** java.util.List;

@Service

**public** **class** CountryServiceImpl **implements** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Override

**public** Country findCountryByCode(String code) {

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

}

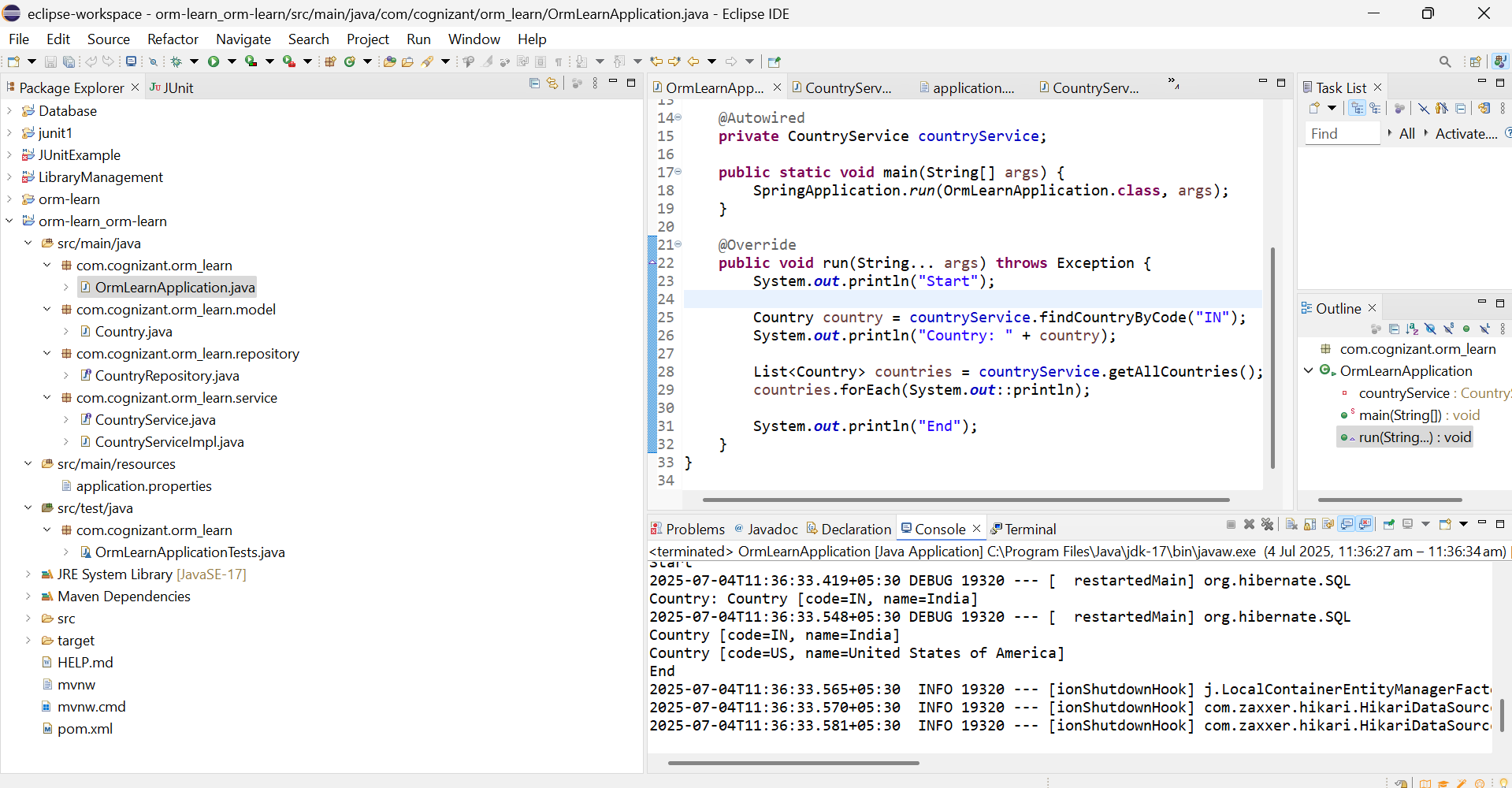
@Override

**public** List<Country> getAllCountries() {

**return** countryRepository.findAll();

}

}

**Output**

**Find a country based on country code**

**CountryServiceImpl.java**

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

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

**import** org.springframework.stereotype.Service;

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

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

@Service

**public** **class** CountryServiceImpl **implements** CountryService {

@Autowired

**private** CountryRepository countryRepository;

@Override

**public** Country findCountryByCode(String code) {

**return** countryRepository.findByCode(code);

}

}

**CountryService.java**

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

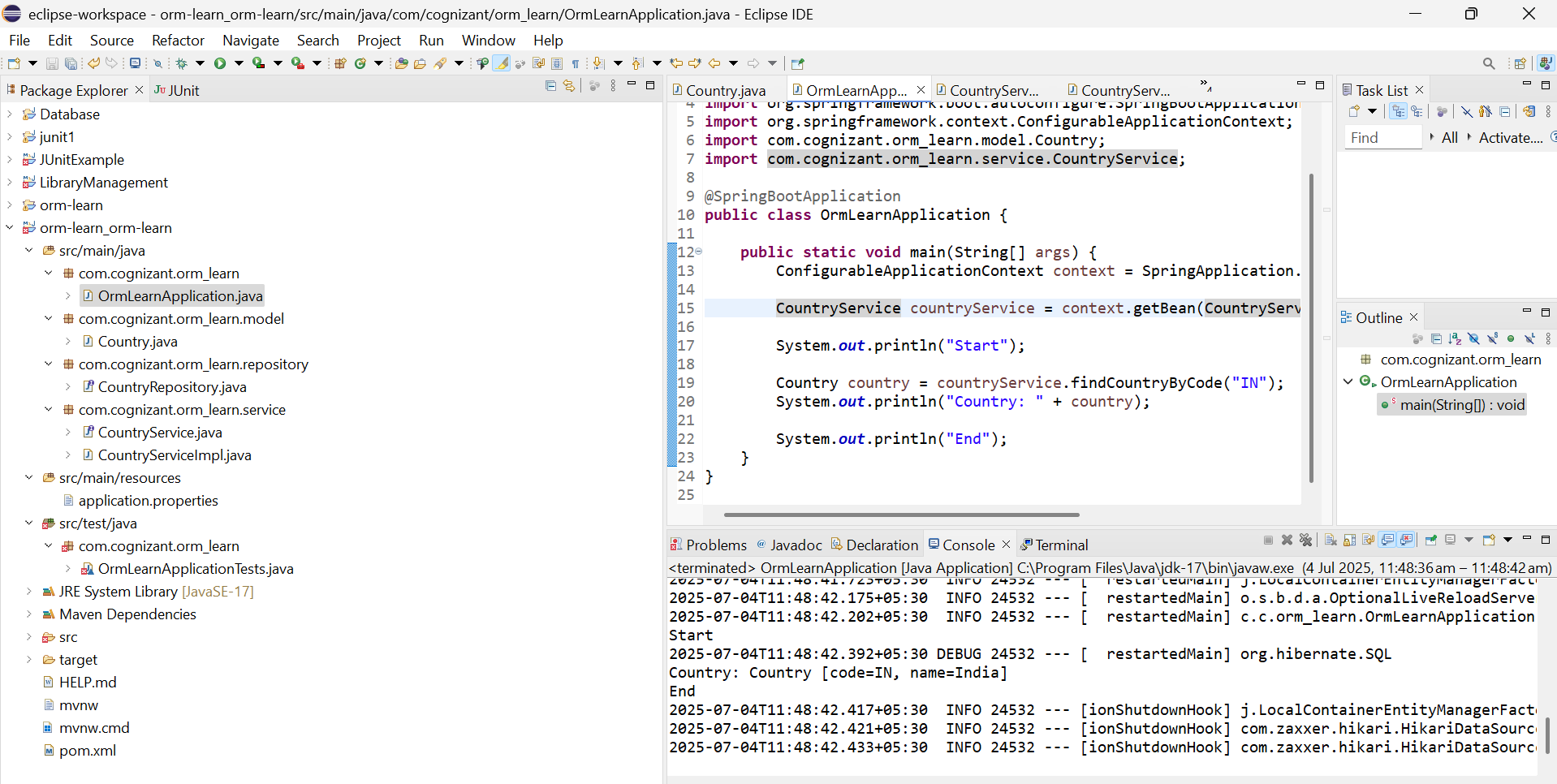
**import** com.cognizant.orm\_learn.model.Country;

**public** **interface** CountryService {

Country findCountryByCode(String code);

}

**Output:**

****

**Add a new country**

**OrmLearnApplication.java**

package com.cognizant.springlearn;

import com.cognizant.springlearn.model.Country;

import com.cognizant.springlearn.service.CountryService;

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

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.transaction.annotation.Transactional;

@SpringBootApplication

public class OrmLearnApplication {

private static CountryService countryService;

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

@Autowired

public void setCountryService(CountryService service) {

OrmLearnApplication.countryService = service;

}

public static void main(String[] args) {

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

testFindCountryByCode(); // 👈 Call the test method here

}

private static void testFindCountryByCode() {

LOGGER.info("Start");

try {

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

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

} catch (CountryNotFoundException e) {

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

}

LOGGER.info("End");

}

}

**CountryService.java**

package com.cognizant.springlearn.service;

import com.cognizant.springlearn.model.Country;

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

import com.cognizant.springlearn.repository.CountryRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.Optional;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

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

if (!result.isPresent()) {

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

}

return result.get();

}

}

**CountryRepository.java**

package com.cognizant.springlearn.repository;

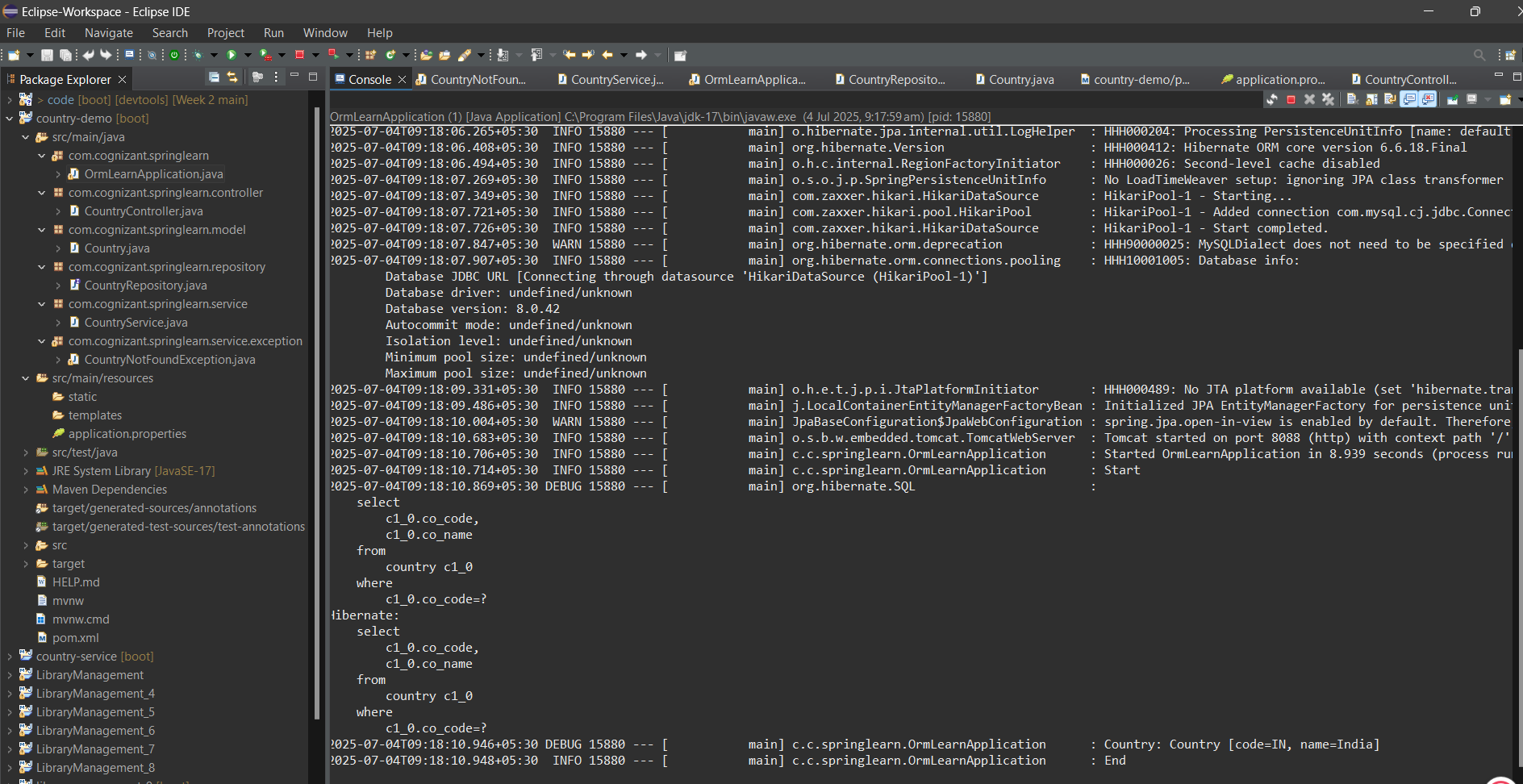
import com.cognizant.springlearn.model.Country;

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

public interface CountryRepository extends JpaRepository<Country, String> {

}

**Output**



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

**Country.java**

package com.example.ormlearn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_name")

private String name;

// Getters and Setters

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 code + " - " + name;

}

}

**CountryRepository.java**

package com.example.ormlearn.repository;

import java.util.List;

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

import com.example.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

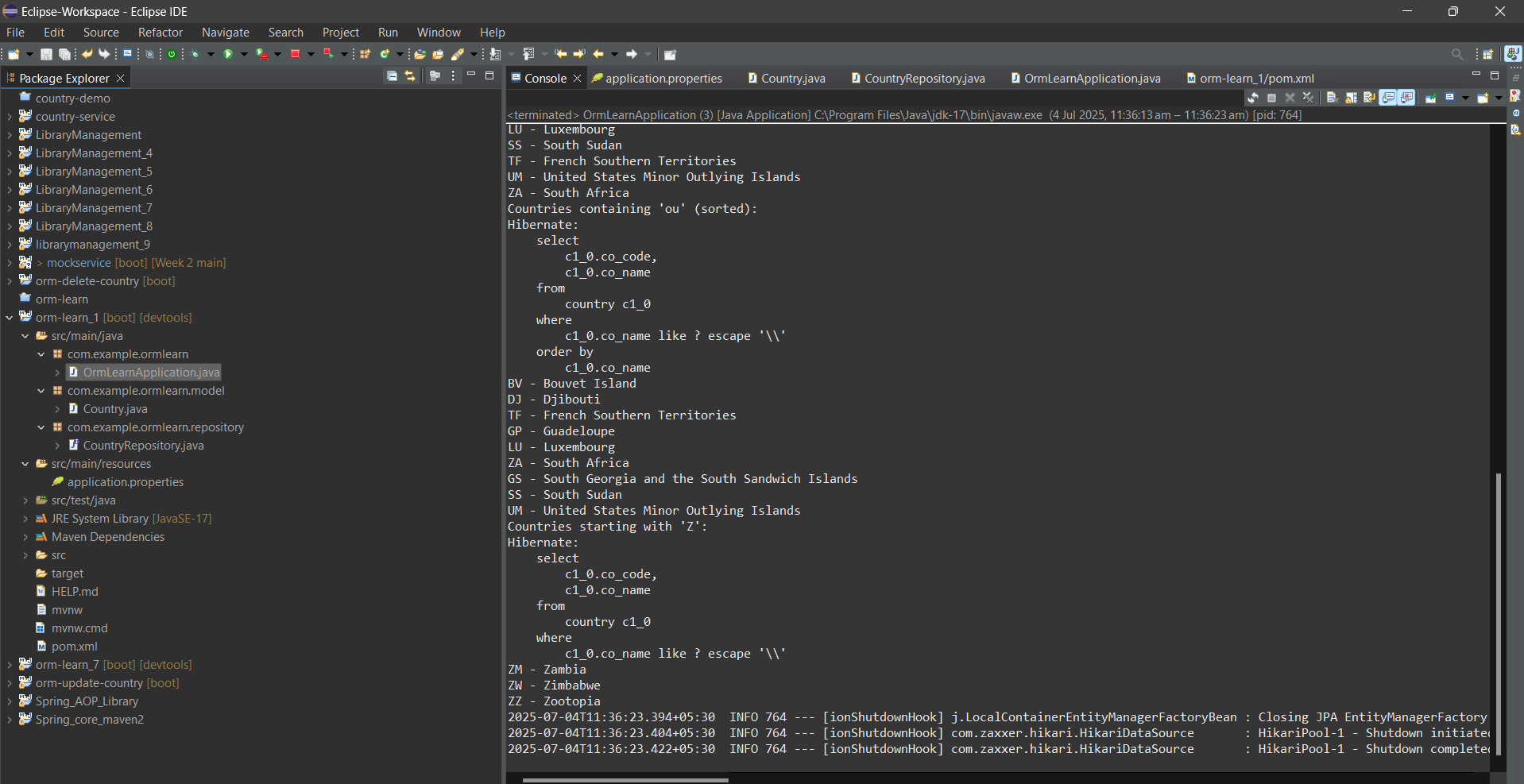
List<Country> findByNameContaining(String text);

List<Country> findByNameContainingOrderByNameAsc(String text);

List<Country> findByNameStartingWith(String letter);

}

**Output**



**Demonstrate implementation of O/R Mapping**

**Stock.java**

package com.cognizant.ormlearn.model;

import javax.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;

@Column(name = "st\_date")

@Temporal(TemporalType.DATE)

private Date date;

@Column(name = "st\_open")

private BigDecimal open;

@Column(name = "st\_close")

private BigDecimal close;

@Column(name = "st\_volume")

private BigDecimal volume;

}

**StockRepository.java**

package com.cognizant.ormlearn.repository;

import java.util.Date;

import java.util.List;

import java.math.BigDecimal;

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

import com.cognizant.ormlearn.model.Stock;

public interface StockRepository extends JpaRepository<Stock, Integer> {

// 1. Facebook stocks in Sep 2019

List<Stock> findByCodeAndDateBetween(String code, Date startDate, Date endDate);

// 2. Google stocks where price > 1250

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

// 3. Top 3 volumes

List<Stock> findTop3ByOrderByVolumeDesc();

// 4. Netflix lowest 3 close prices

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**Introduction to HQL and JPQL**

* HQL (Hibernate Query Language) is an object-based query language that is part of the Hibernate ORM framework, designed to work with Java entity objects.
* JPQL (Java Persistence Query Language) is a standardized query language defined by the JPA (Java Persistence API) specification, enabling interaction with databases in an object-oriented manner.

**Key Similarities Between HQL and JPQL:**

* Both HQL and JPQL share SQL-like syntax, but instead of targeting database tables and columns, they operate on Java classes and their fields (entities and attributes).

| **Feature** | **HQL** | **JPQL** |
| --- | --- | --- |
| Ownership | Part of Hibernate | Part of JPA specification |
| Syntax Base | Object-oriented SQL-like | Subset of HQL |
| INSERT Operation | Supported | **Not supported** |
| Native SQL Conversion | Hibernate handles SQL conversion | JPA providers (e.g., Hibernate) do it |

* They support key operations such as SELECT, UPDATE, and DELETE.
* Queries are written using Java entity class names and their properties, rather than referring directly to database-specific elements.

**Key Differences:**

* JPQL is a subset of HQL, meaning every JPQL query is a valid HQL query, but not all HQL queries are valid JPQL.
* INSERT operations are allowed in HQL but not in JPQL.

**Example Syntax:**

// JPQL / HQL SELECT example

Query q = entityManager.createQuery("SELECT e FROM Employee e WHERE e.salary > :minSal");

// HQL INSERT example (not allowed in JPQL)

Query q = session.createQuery("INSERT INTO Employee(name, salary) SELECT e.name, e.salary FROM OldEmployee e");