**WEEK - 3**

**Spring Data with JPA**

**Mandatory**

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

**ormLearnApplication.java**

package com.cognizant.orm.learn;

import java.util.ArrayList;

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.orm.learn.model.Country;

import com.cognizant.orm.learn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication {

private static CountryService countryService;

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

public static void main(String[] args) {

//SpringApplication.run(OrmLearnApplication.class, args);

LOGGER.info("Inside main");

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

countryService=context.getBean(CountryService.class);

testGetallCountries();

testSaveallCountries();

}

private static void testGetallCountries() {

List<Country> countrylist=countryService.getAllcountries();

LOGGER.info("List of countries:::::"+countrylist.size());

}

private static void testSaveallCountries() {

Country country=new Country();

country.setCode("8");

country.setName("India1");

LOGGER.info("List of countries::Saved:::"+countryService.saveCountry(country));

}

}

**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;

public interface CountryRepository extends JpaRepository <Country, String> {

}

Application.properties

spring.application.name=orm-learn

# Spring Framework and application log

logging.level.org.springframework=info

logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output

logging.level.org.hibernate.SQL=trace

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

# Log pattern

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

# Database 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=saibaba

# Hibernate configuration

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**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 Data JPA and Hibernate</description>

<url/>

<licenses>

<license/>

</licenses>

<developers>

<developer/>

</developers>

<scm>

<connection/>

<developerConnection/>

<tag/>

<url/>

</scm>

<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-devtools</artifactId>

<scope>runtime</scope>

<optional>true</optional>

</dependency>

<dependency>

<groupId>com.mysql</groupId>

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

<version>8.0.33</version>

<scope>runtime</scope>

</dependency>

<dependency>

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

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

<scope>test</scope>

</dependency>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

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

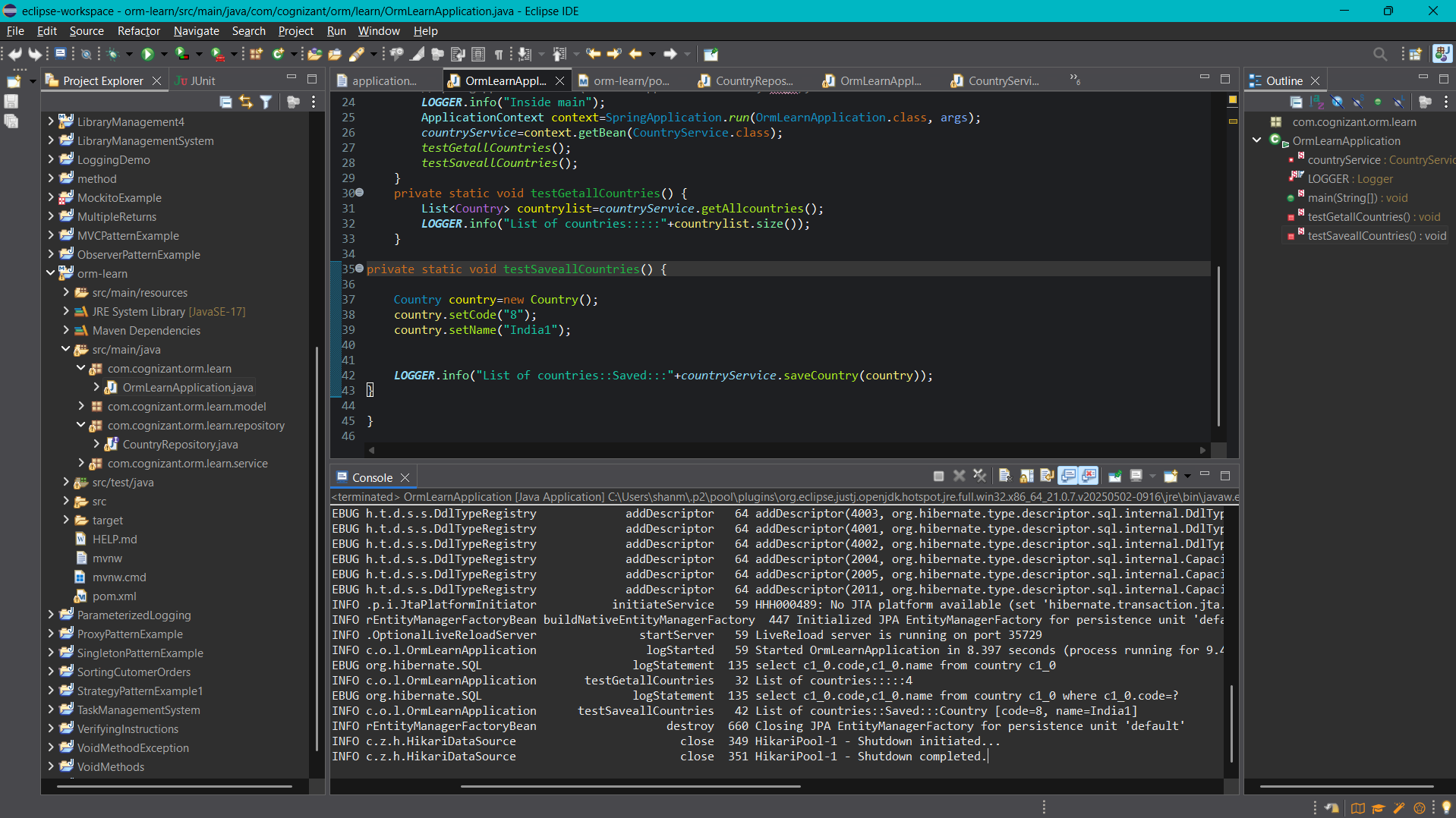
<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>



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

**1. JPA (Java Persistence API)**

* JPA is a specification, not an implementation.
* It defines a set of interfaces and annotations for object-relational mapping (ORM) in Java.
* It allows Java developers to map Java objects (entities) to relational database tables.
* JPA is part of the Java EE (Jakarta EE) standard.
* To actually work, JPA requires a provider/implementation (like Hibernate, EclipseLink, etc.).
* Think of JPA as a set of rules or a contract for ORM.

**2. Hibernate**

* Hibernate is a popular ORM framework and the most widely used implementation of JPA.
* It provides all the functionality defined in JPA and also offers many additional features not covered by the specification.
* Hibernate can be used in two ways:
* As a JPA provider (standardized approach)
* Using native Hibernate APIs (non-standard but powerful)
* It supports advanced ORM capabilities like lazy loading, caching, and custom query languages (HQL).
* Hibernate is the engine that powers the JPA specification in most Java projects.

**3. Spring Data JPA**

* Spring Data JPA is a framework built on top of JPA (and typically uses Hibernate under the hood).
* It simplifies data access layers by reducing boilerplate code.
* It provides ready-to-use repository interfaces like JpaRepository, CrudRepository, etc.
* Spring Data JPA allows you to write custom queries using method names or annotations like @Query.
* You don’t have to write implementation classes for your DAOs manually.
* Spring Data JPA is an abstraction that makes JPA easy and fast to use within Spring applications.

**Summary**

* JPA is the standard (interface).
* Hibernate is a tool (implementation of JPA).
* Spring Data JPA is a Spring module that builds on JPA and Hibernate to reduce effort and complexity.