**WEEK – 3**

**Spring Core -MAVEN**

**Exercise 1: Configuring a Basic Spring Application**

**pom.xml**

<projectxmlns="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>

<groupId>com.library</groupId>

<artifactId>Librarymanagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</project>

**applicationContext.xml**

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

**BookService.java**

**package**com.library.service;

**import**com.library.repository.BookRepository;

**publicclass**BookService {

**private**BookRepositorybookRepository;

**publicvoid**setBookRepository(BookRepositorybookRepository) {

**this**.bookRepository = bookRepository;

}

**publicvoid**displayBooks() {

bookRepository.getBookList();

}

}

**BookRepository.java**

package com.library.repository;

public class BookRepository {

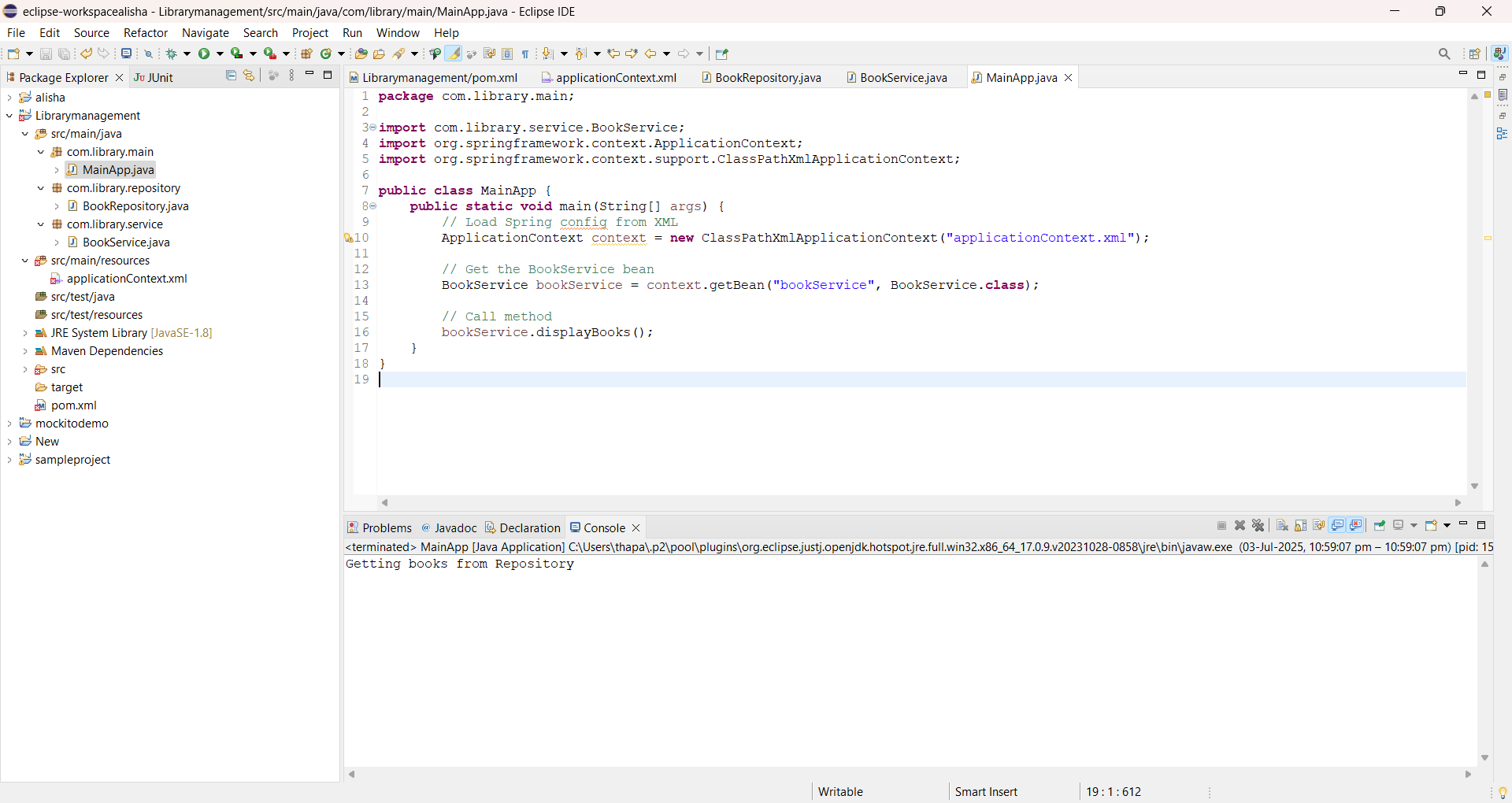
public void getBookList() {

System.*out*.println("Getting books from Repository");

}

}

**Output:**



**Exercise 2: Implementing Dependency Injection**

**pom.xml**

<projectxmlns="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>

<groupId>com.library</groupId>

<artifactId>Librarymanagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

</dependencies>

</project>

**applicationContext.xml**

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="bookRepository" class="com.library.repository.BookRepository" />

<bean id="bookService" class="com.library.service.BookService">

<property name="bookRepository" ref="bookRepository" />

</bean>

</beans>

**BookRepository.java**

**package**com.library.repository;

**publicclass**BookRepository {

**publicvoid**getBookList() {

System.***out***.println("Getting books from Repository");

}

}

**BookService.java**

**package**com.library.service;

**import**com.library.repository.BookRepository;

**publicclass**BookService {

**private**BookRepositorybookRepository;

**publicvoid**setBookRepository(BookRepositorybookRepository) {

**this**.bookRepository = bookRepository;

}

**publicvoid**displayBooks() {

System.***out***.println("BookService is working...");

bookRepository.getBookList();

}

}

**MainApp.java**

**package**com.library.main;

**import**com.library.service.BookService;

**import**org.springframework.context.ApplicationContext;

**import**org.springframework.context.support.ClassPathXmlApplicationContext;

**publicclass**MainApp {

**publicstaticvoid**main(String[] args) {

ApplicationContextcontext = **new**ClassPathXmlApplicationContext("applicationContext.xml");

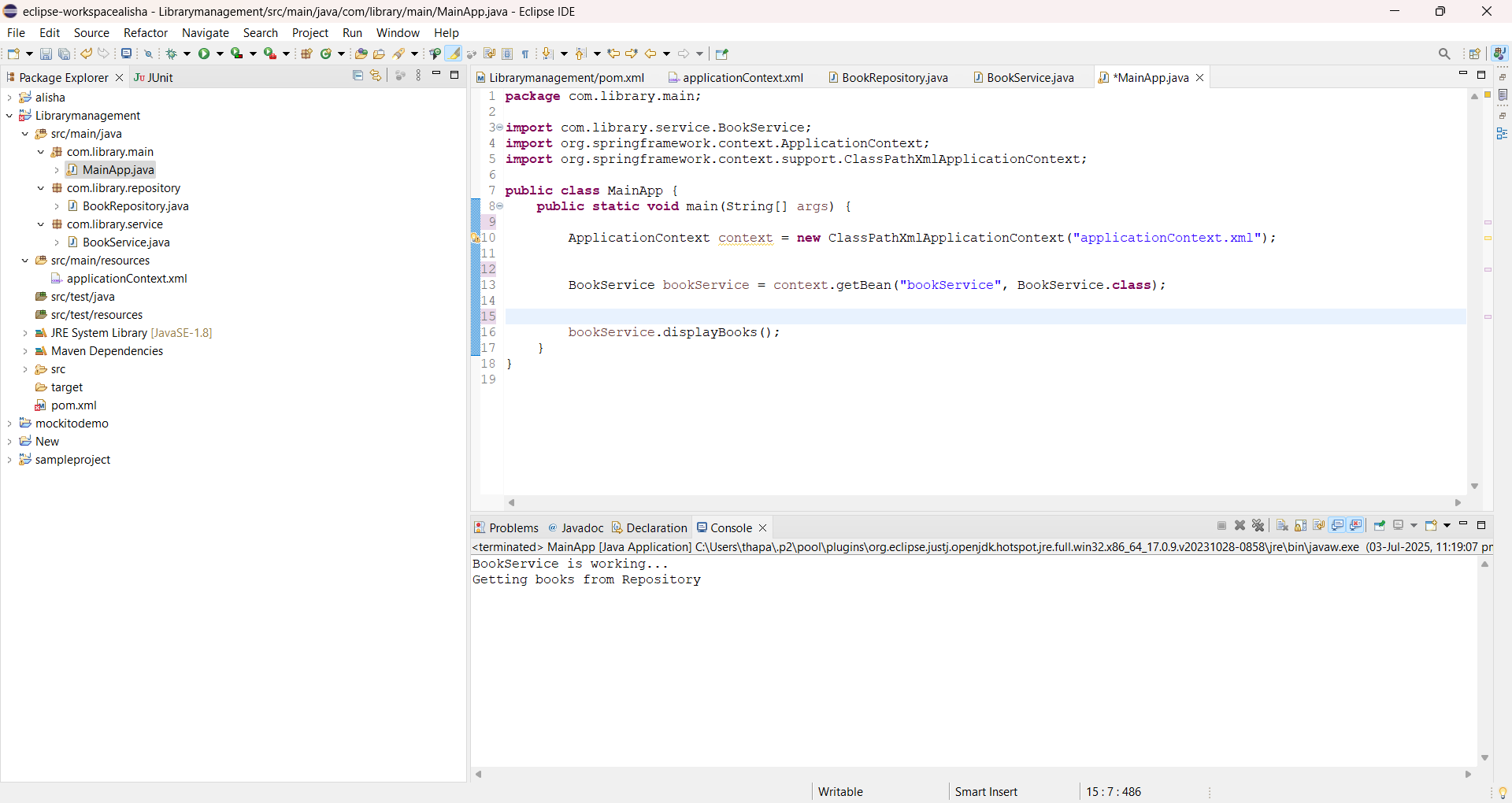
BookServicebookService = context.getBean("bookService", BookService.**class**);

bookService.displayBooks();

}

}

**Output:**

****

**Exercise 4: Creating and Configuring a Maven Project**

**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 https://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>0.0.1-SNAPSHOT</version>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</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>

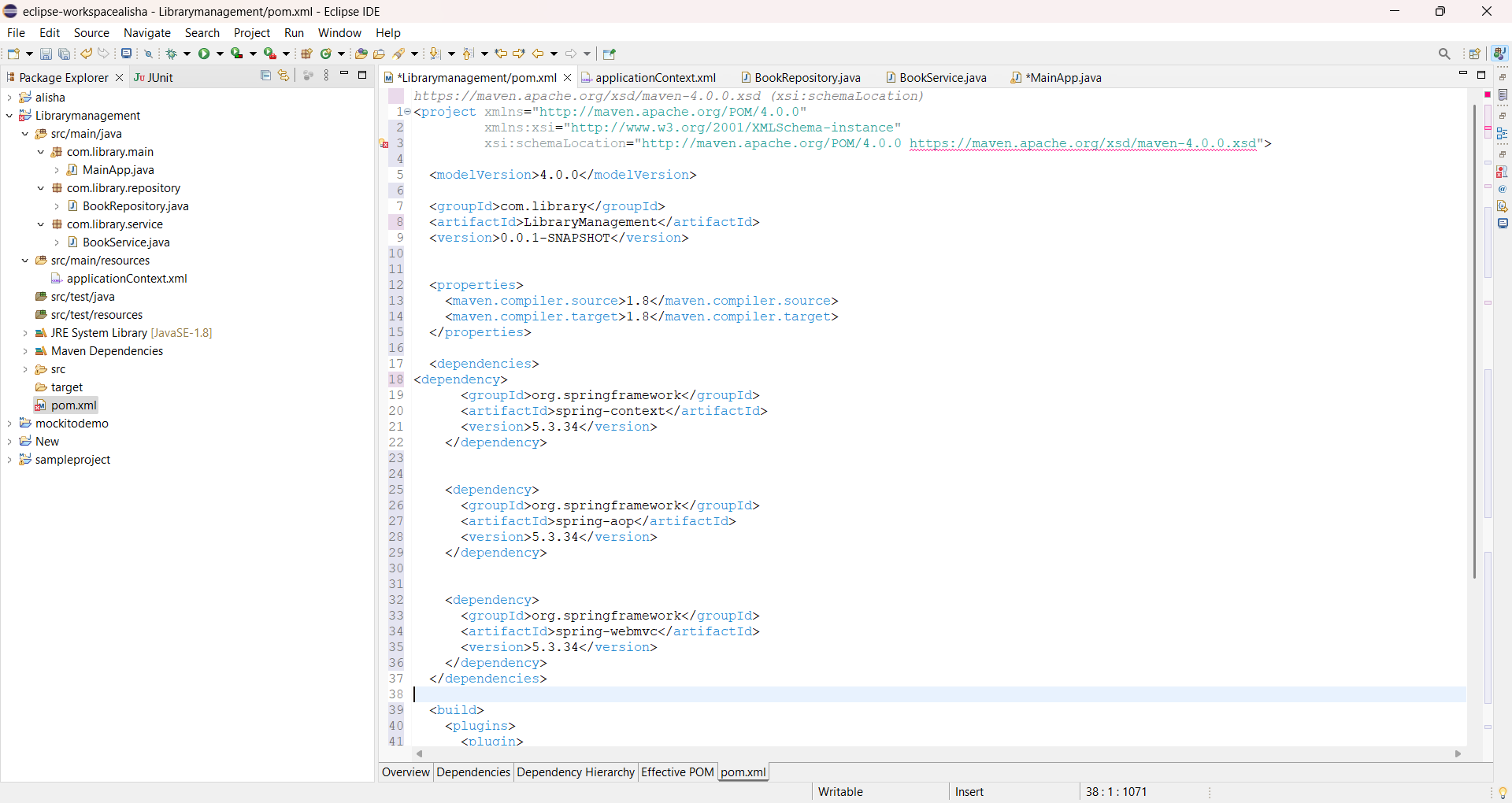
</plugin>

</plugins>

</build>

</project>

**Output:**

****

**Spring Data JPA**

* **Spring Data JPA - Quick Example**

**Software Pre-requisites**

* MySQL Server 8.0
* MySQL Workbench 8
* Eclipse IDE for Enterprise Java Developers 2019-03 R
* Maven 3.6.2

**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>

<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>

**application.properties**

**# Logging**

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=debug

logging.level.org.hibernate.orm.jdbc.bind=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=root

**# JPA / Hibernate**

spring.jpa.hibernate.ddl-auto=validate

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

OrmLearnApplication.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);

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info(&quot;Start&quot;);

List&lt;Country&gt; countries = countryService.getAllCountries();

LOGGER.debug(&quot;countries={}&quot;, countries);

LOGGER.info(&quot;End&quot;);

}

}

**Country.java**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity // Tells Spring this is an Entity class (maps to DB table)

@Table(name = &quot;country&quot;) public class Country {

@Id

@Column(name = &quot;code&quot;)

private String code;

@Column(name = &quot;name&quot;)

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 &quot;Country [code=&quot; + code + &quot;, name=&quot; + name + &quot;]&quot;;

}

}

CountryRepository.java

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&lt;Country, String&gt; {

}

CountryService.java

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

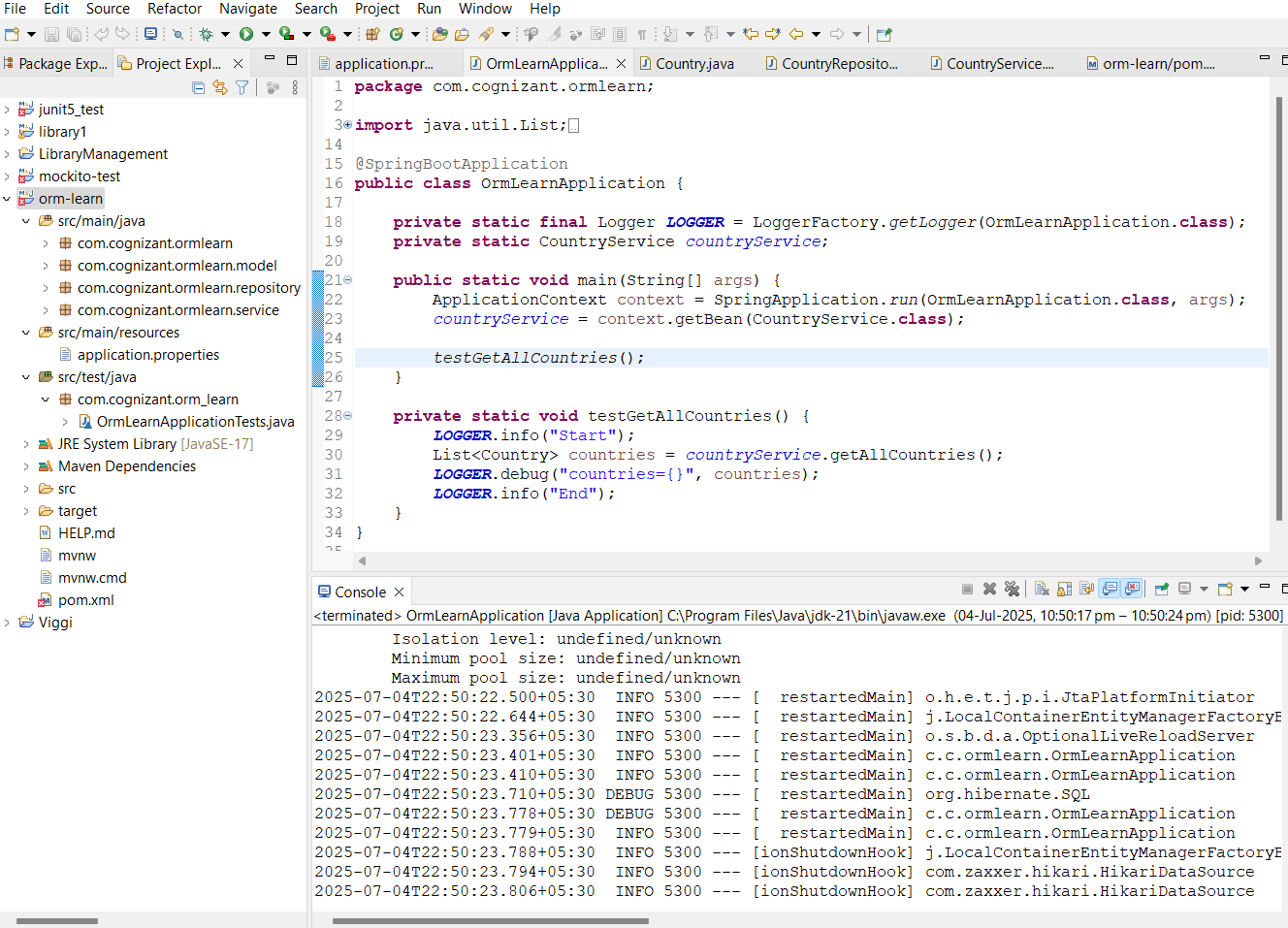
public List&lt;Country&gt; getAllCountries() {

return countryRepository.findAll();

}

}

**Output:**

****

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

1. JPA (Java Persistence API)

* A *specification* (interface) defined by Java EE (now Jakarta EE).  
  Think of it as a *set of guidelines* or *rules* for how Java objects should be mapped to a relational database.
* JPA does not provide implementation itself. It only defines APIs and annotations like:

@Entity @Table @Id @GeneratedValue

* Example Implementations:
  + Hibernate
  + EclipseLink
  + OpenJPA

Analogy:  
JPA is like a *contract or interface* that says “here’s how persistence should work.”

2. Hibernate

* A *framework* (library) and one of the most popular implementations of JPA.
  + Provides the actual working code that fulfills JPA’s contracts.
  + Also offers *extra features beyond JPA* (e.g., better caching, performance tuning, native APIs.  
    You can use Hibernate:
  + As a JPA implementation (using only the standard JPA APIs).
  + Or natively (using Hibernate’s specific APIs like Session, Criteria, etc.).
* Example:

Session session = sessionFactory.openSession(); // Hibernate native API

3. Spring Data JPA  
 A *Spring project* that sits on top of JPA (and an implementation like Hibernate).

To simplify the development of repository (DAO) layers by:

* + Automatically generating query methods.
  + Reducing boilerplate code.
  + Integrating easily with Spring Boot.
  + CrudRepository, JpaRepository, PagingAndSortingRepository.
  + Derived queries by method names.
  + Custom queries with @Query.
* Example Repository:

public interface UserRepository extends JpaRepository<User, Long>{ List<User>findByLastName(String lastName); }

|  |  |  |  |
| --- | --- | --- | --- |
| Aspect | JPA | Hibernate | Spring Data JPA |
| Type | Specification | Implementation / Framework | Spring abstraction over JPA |
| Provides | Interfaces, annotations | Concrete code + extra features | Auto-generated Repositories, Queries |
| Dependency | No runtime code | Hibernate library | Spring Data JPA + Hibernate |
| Usage | EntityManager | Session (native) or JPA | JpaRepository, Spring repositories |

Quick takeaway:

* JPA = *standard* interface
* Hibernate = *implementation* of JPA + more
* Spring Data JPA = *Spring helper* that makes JPA/Hibernate easier to use