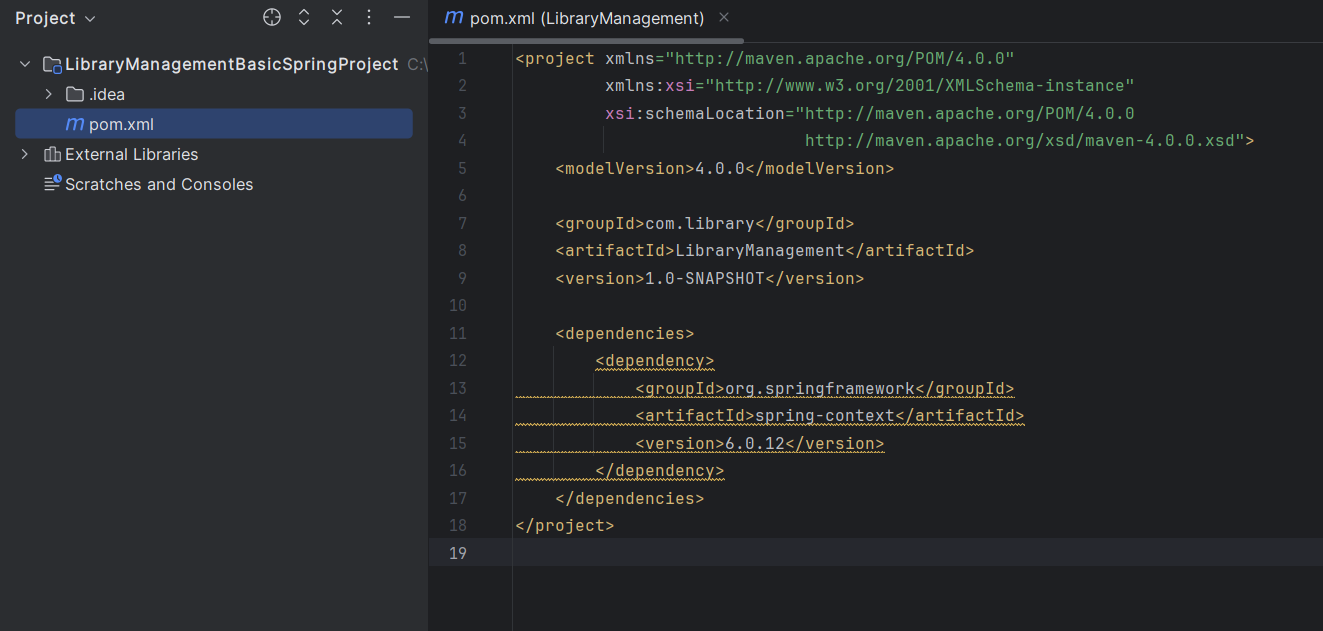
**Cognizant - DN 4.0 Deep Skilling Stage**

**Mandatory Hands-On Week 3 Solutions**

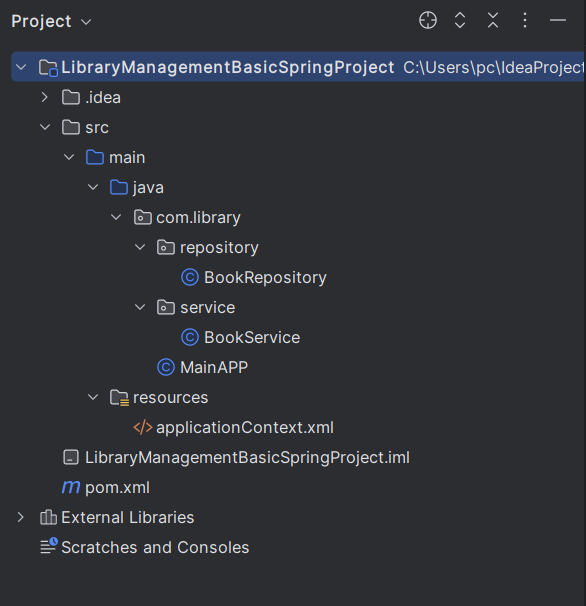
1. **Spring Core Maven**

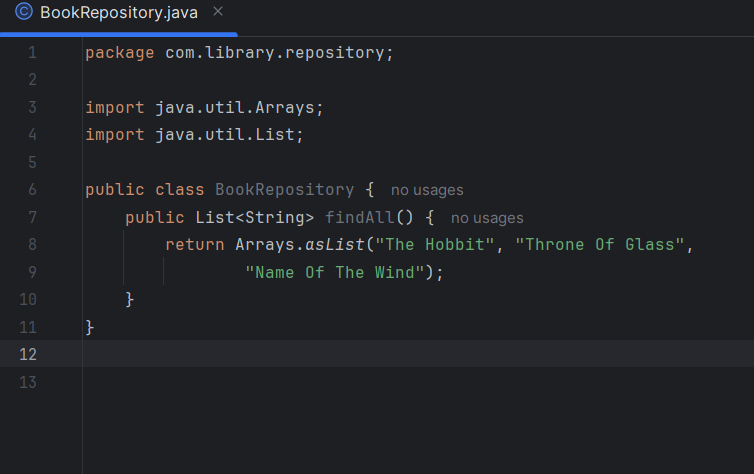
**Exercise 1: Configuring a Basic Spring Application**

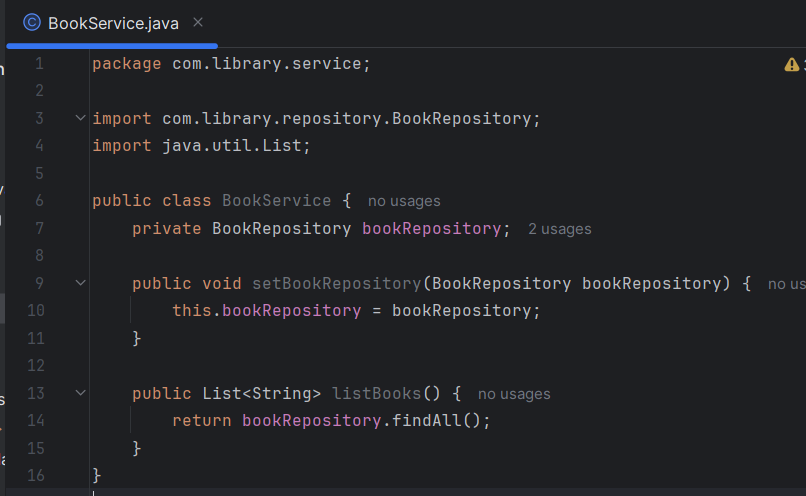
Pom.xml:

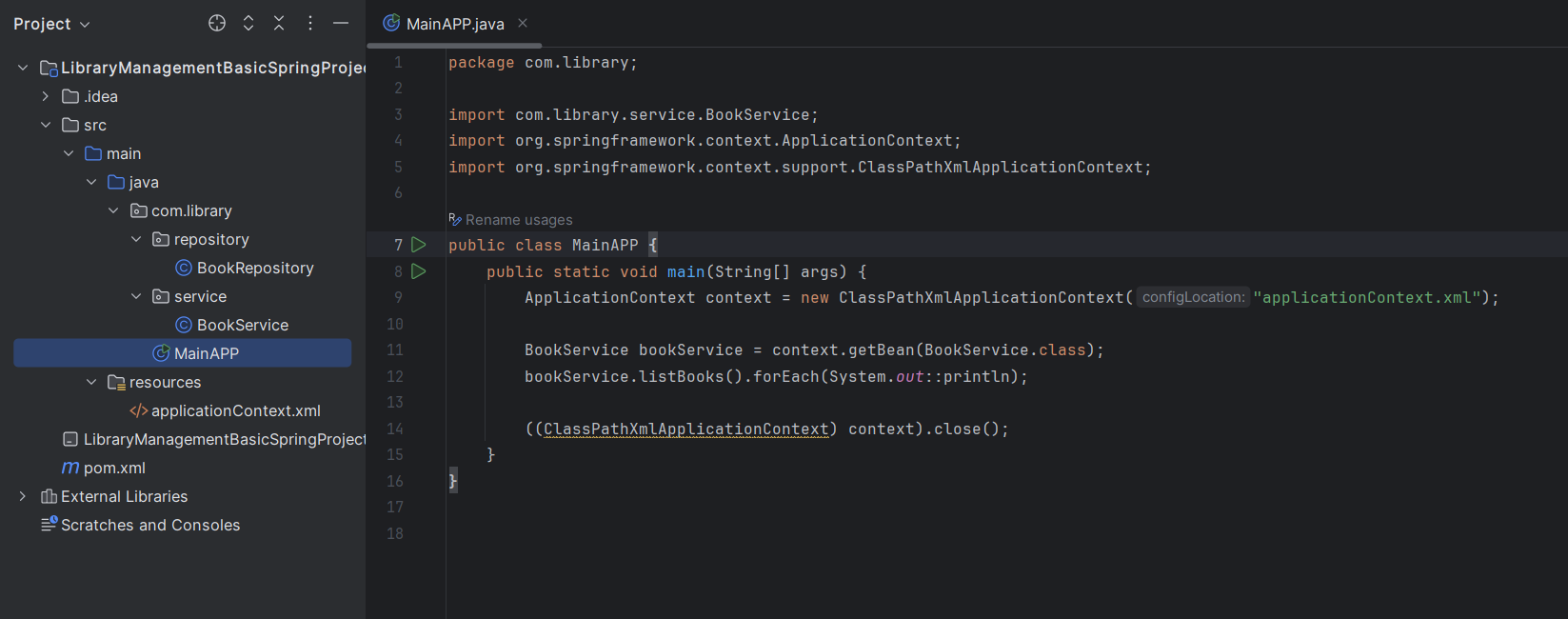


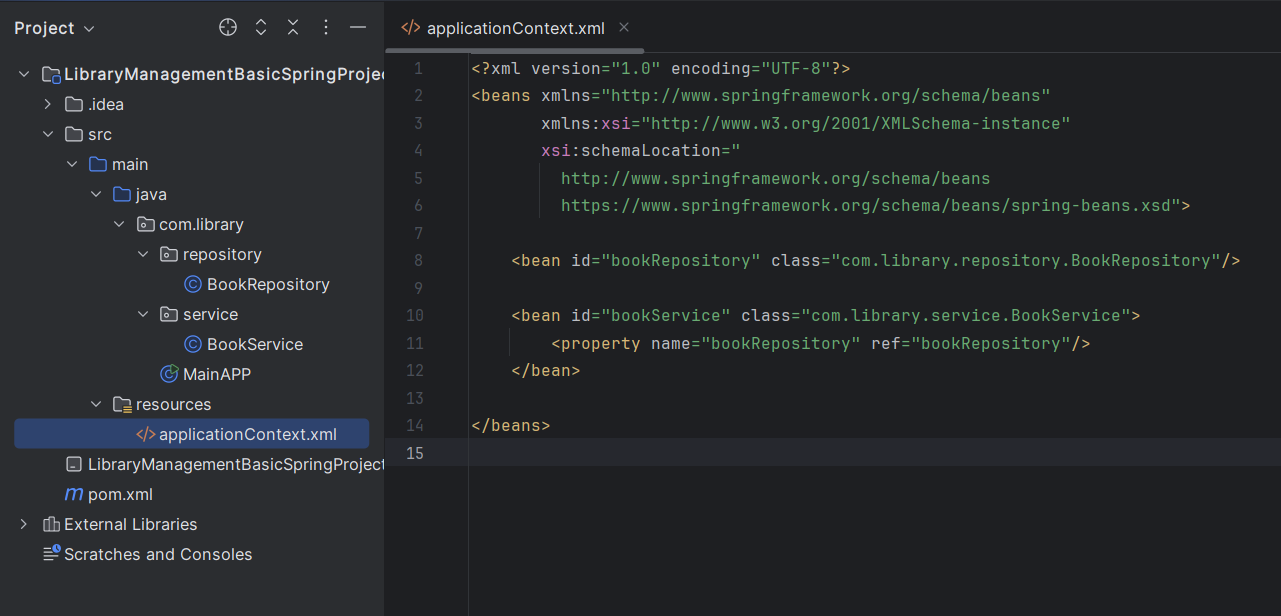
Project Structure:



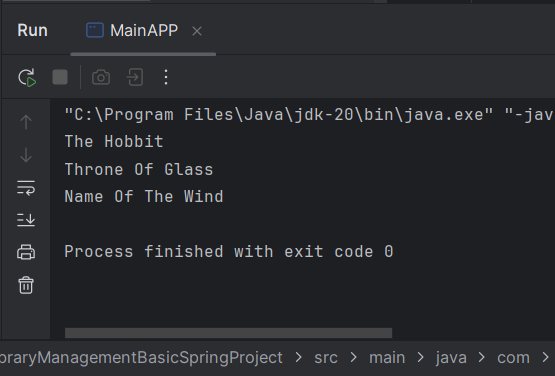






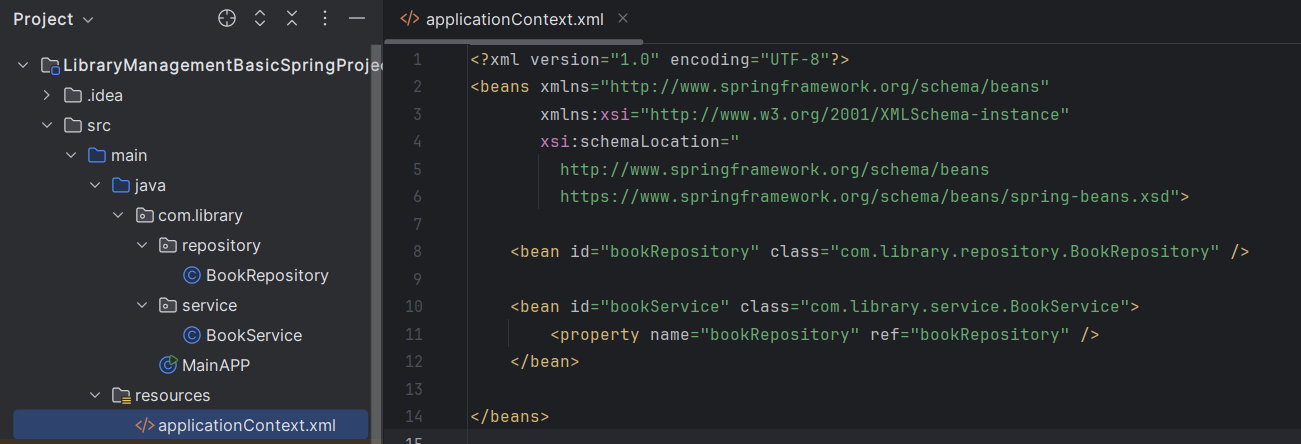
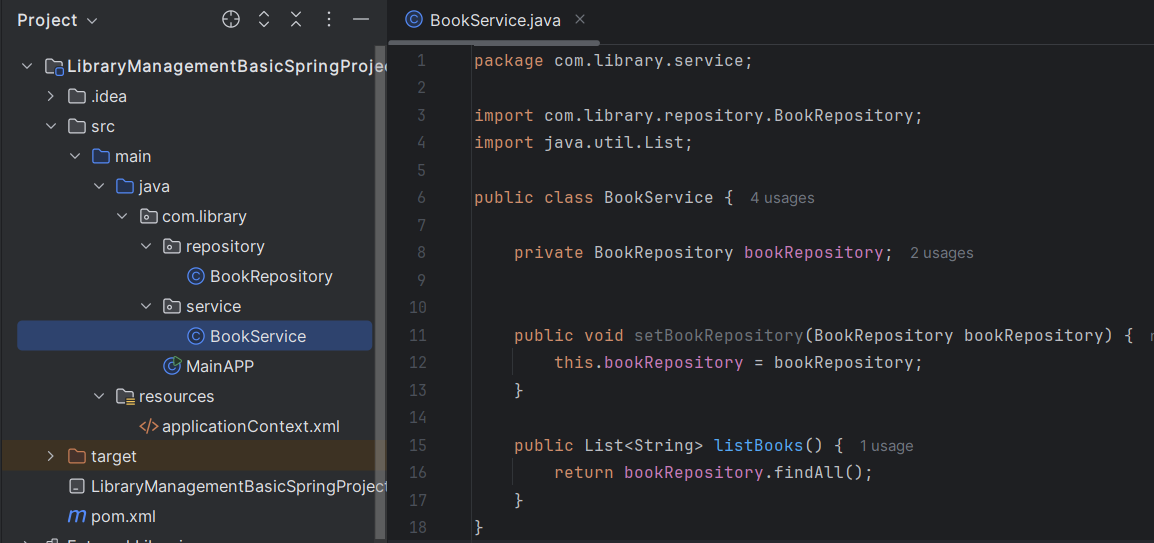


Output:

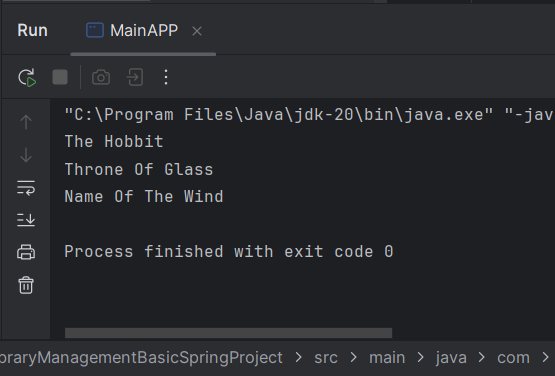


**Exercise 2: Implementing Dependency Injection**

Adding setter:

****

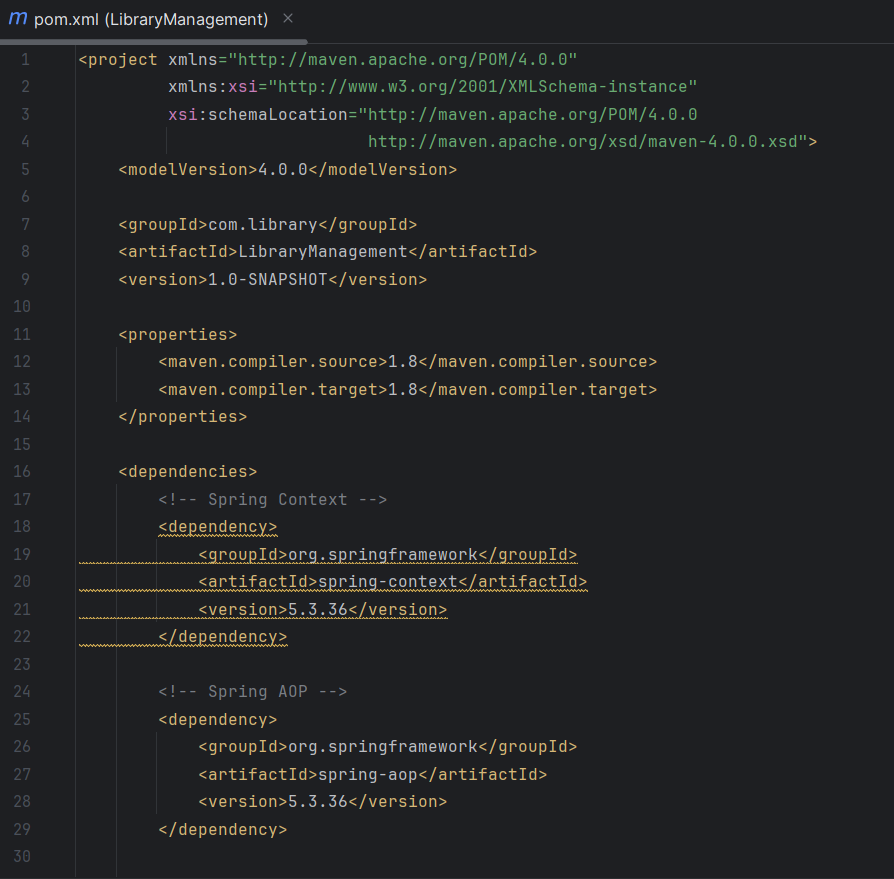
Output:

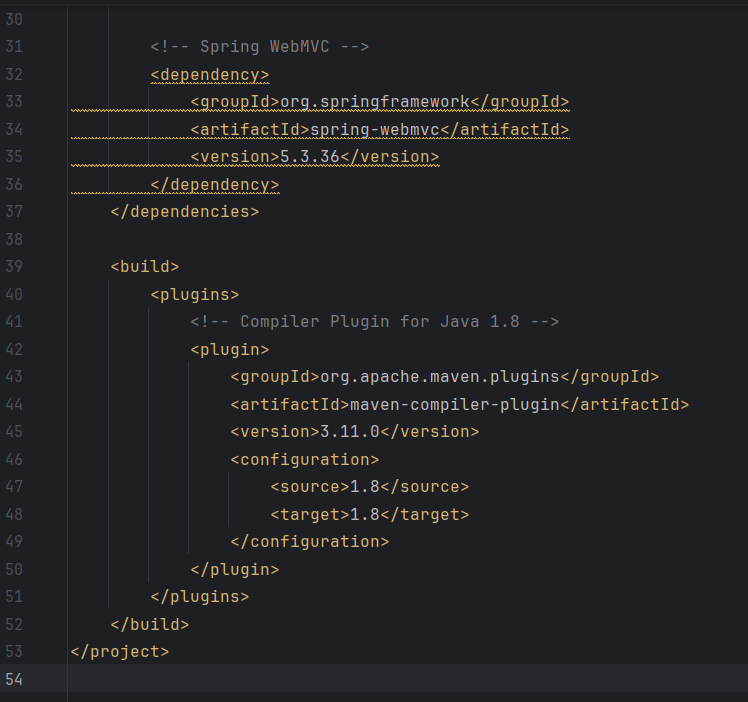


**Exercise 3: Creating and Configuring a Maven Project**

updated pom.xml with:

* Spring Context
* Spring AOP
* Spring WebMVC
* Maven Compiler Plugin (Java 1.8

****

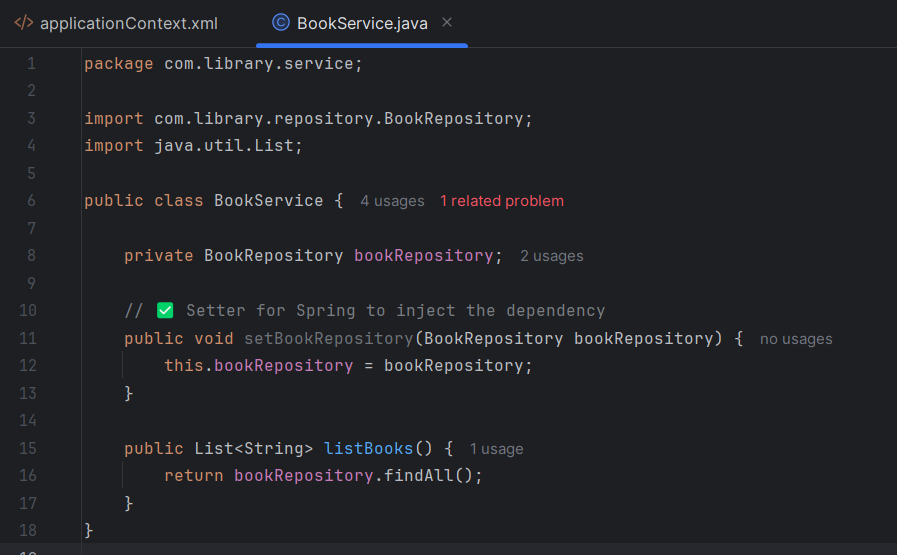
****

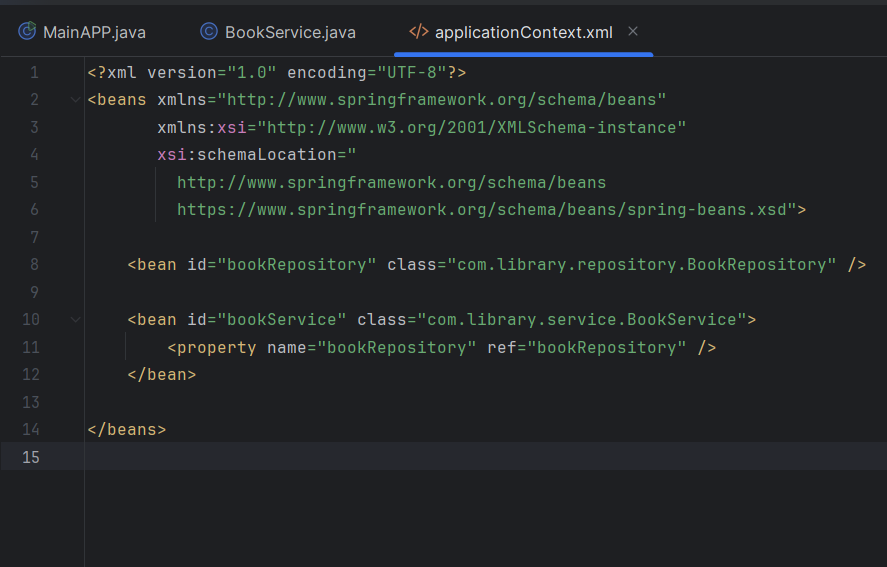
**Additional Hands-On**

**Exercise 5: Configuring the Spring IoC Container**

* Created applicationContext.xml in src/main/resources
* Defined BookService and BookRepository beans
* Used setter injection in BookService
* Loaded Spring context in MainApp.java
* Retrieved and used BookService bean to print books

****

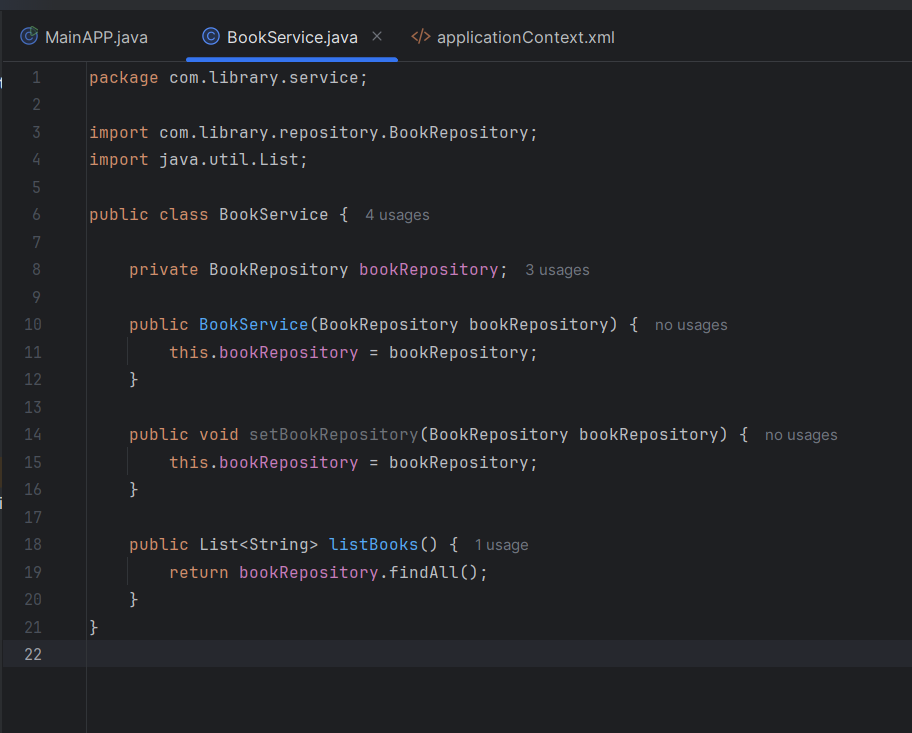
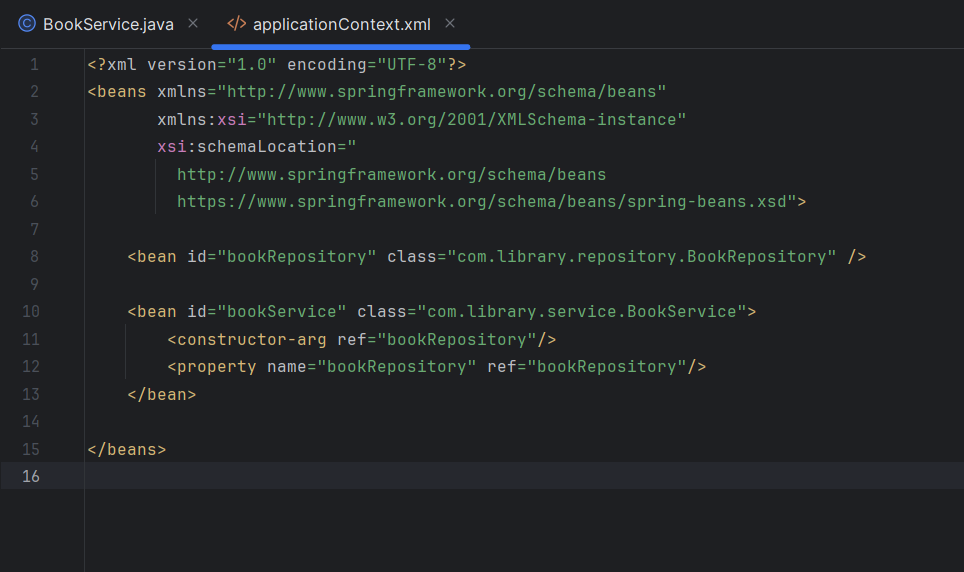
****

****

**Exercise 7: Implementing Constructor and Setter Injection**

### 

* Modified BookService to support both:  
  + Constructor injection
  + Setter injection
* Updated applicationContext.xml to:  
  + Inject bookRepository using both <constructor-arg> and <property>
* Verified it by running MainApp.java, confirming Spring injected dependencies successfully.

****

1. **spring-data-jpa-handson**

**Exercise 2: Difference between JPA, Hibernate and Spring Data JPA**

### JPA (Java Persistence API)

JPA is a specification provided by Java for object-relational mapping (ORM). It defines a standard way to map Java objects to database tables and manage data using annotations like @Entity, @Id, and @OneToMany. But JPA itself doesn’t provide any implementation — it's just a set of interfaces and rules.

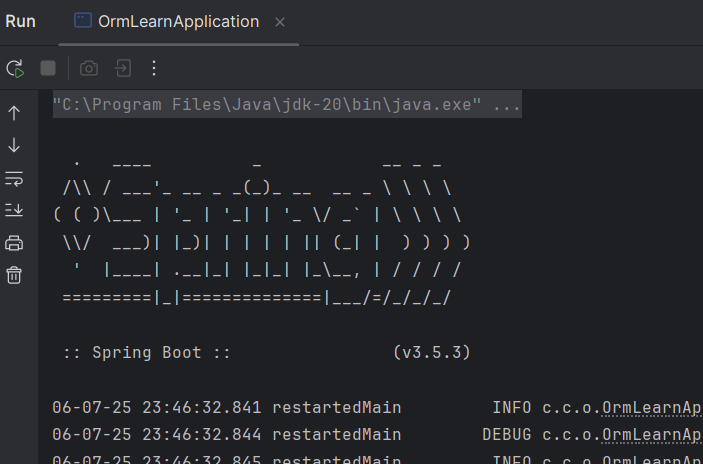
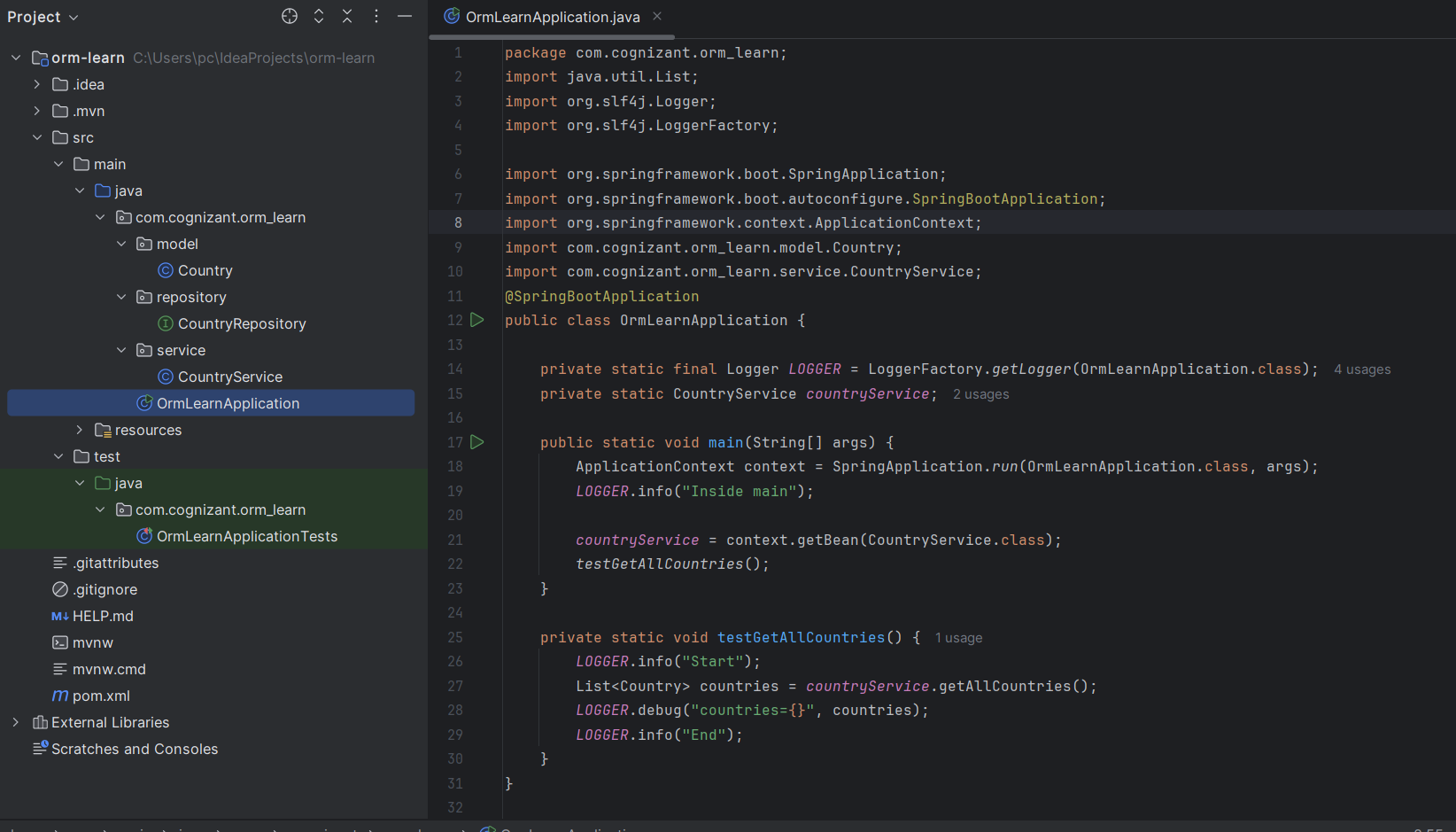
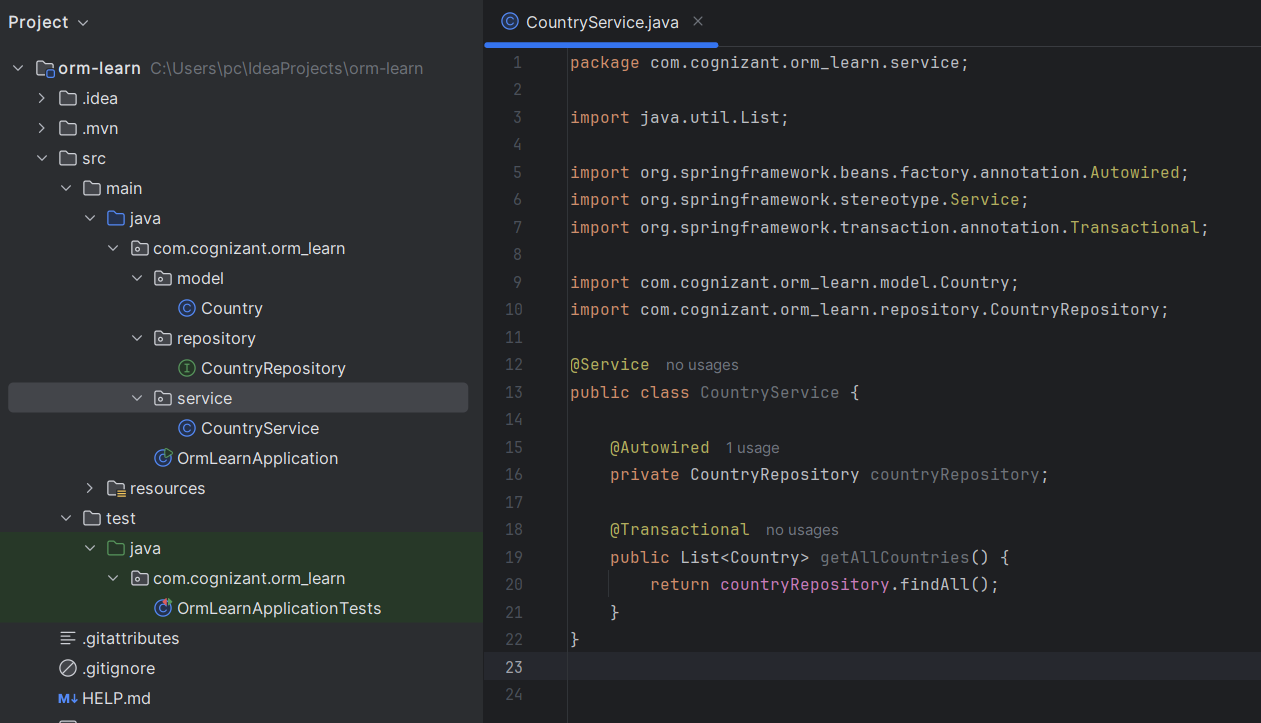
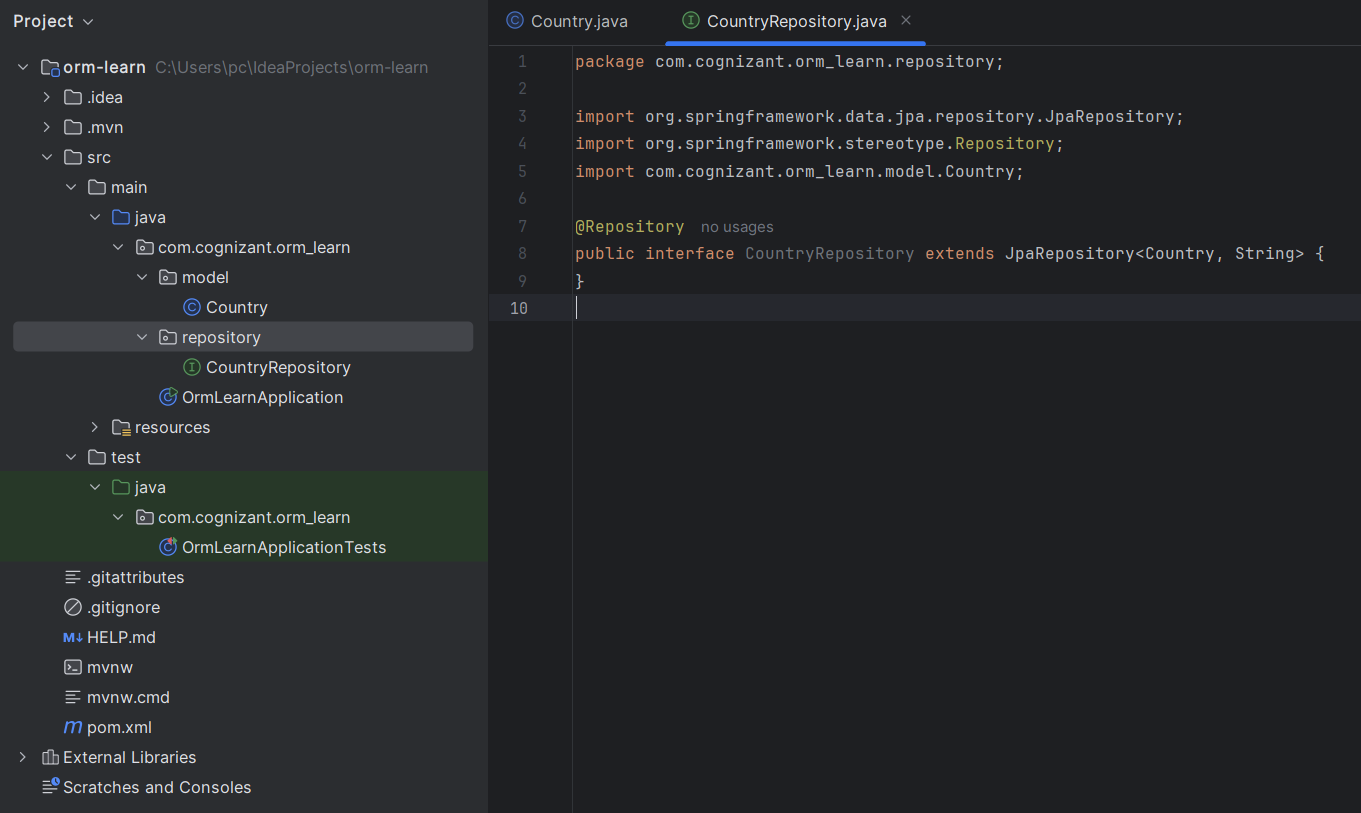
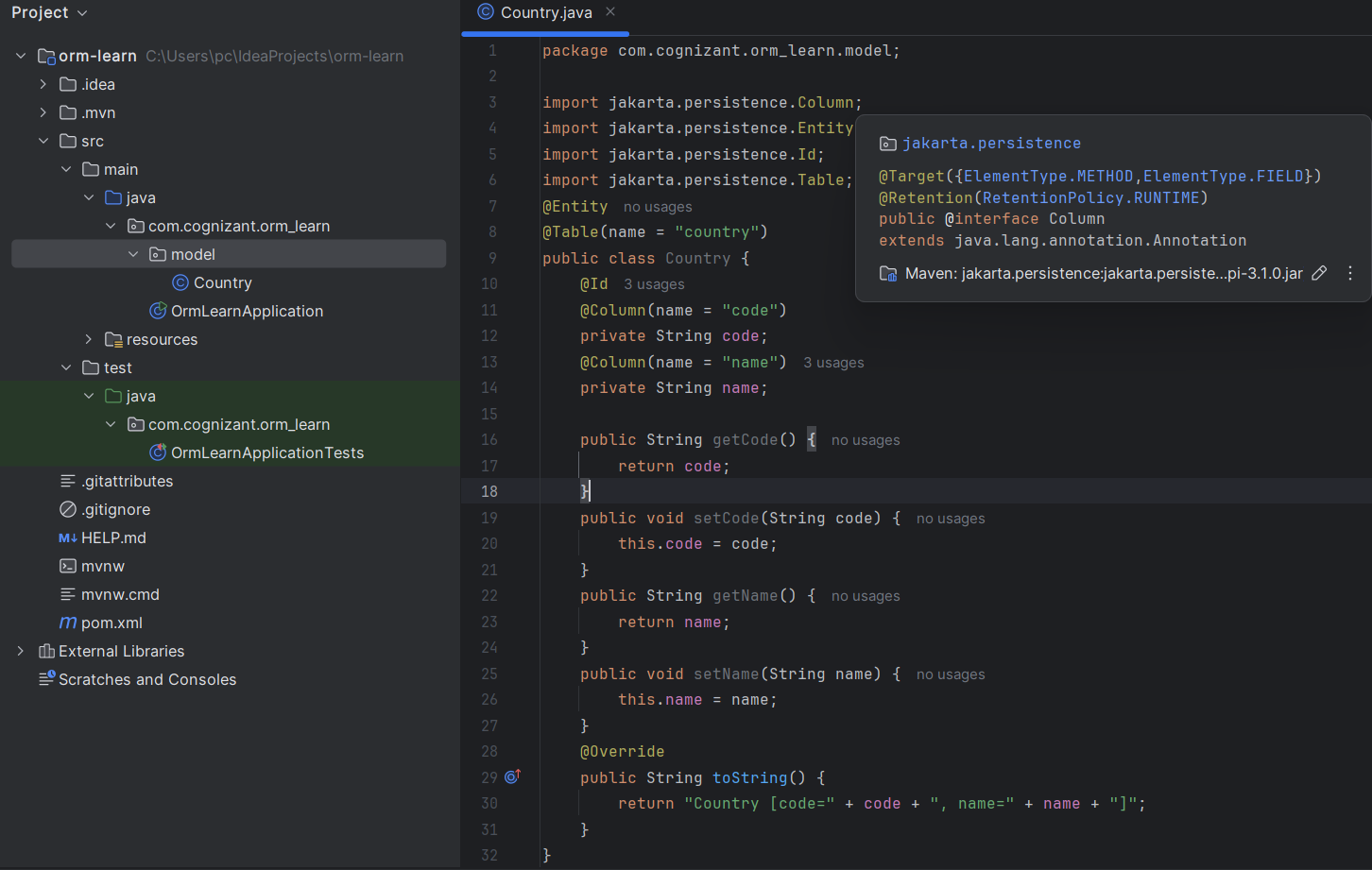
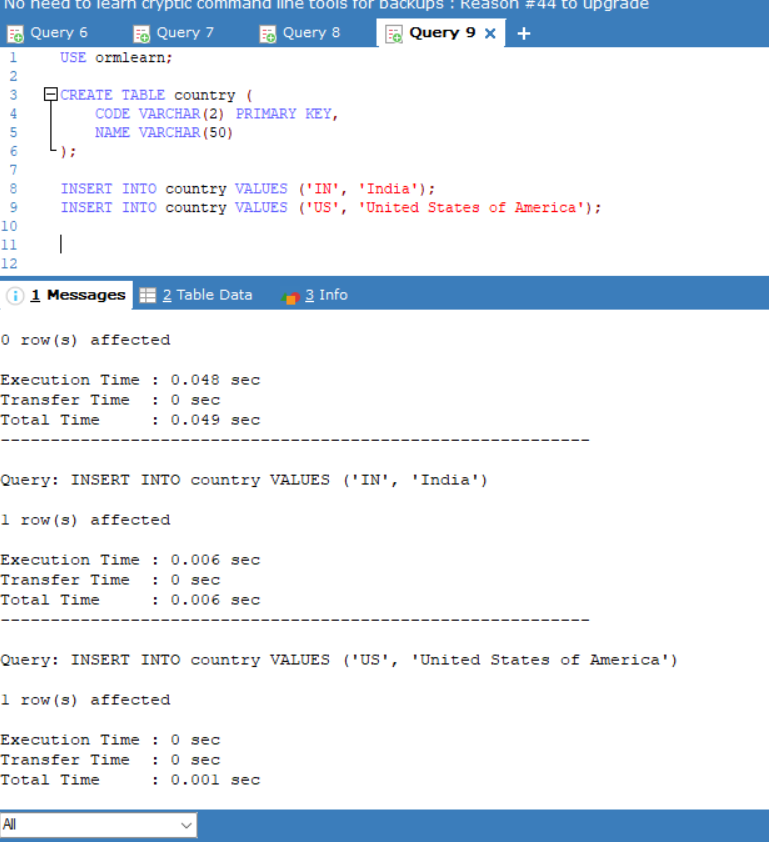
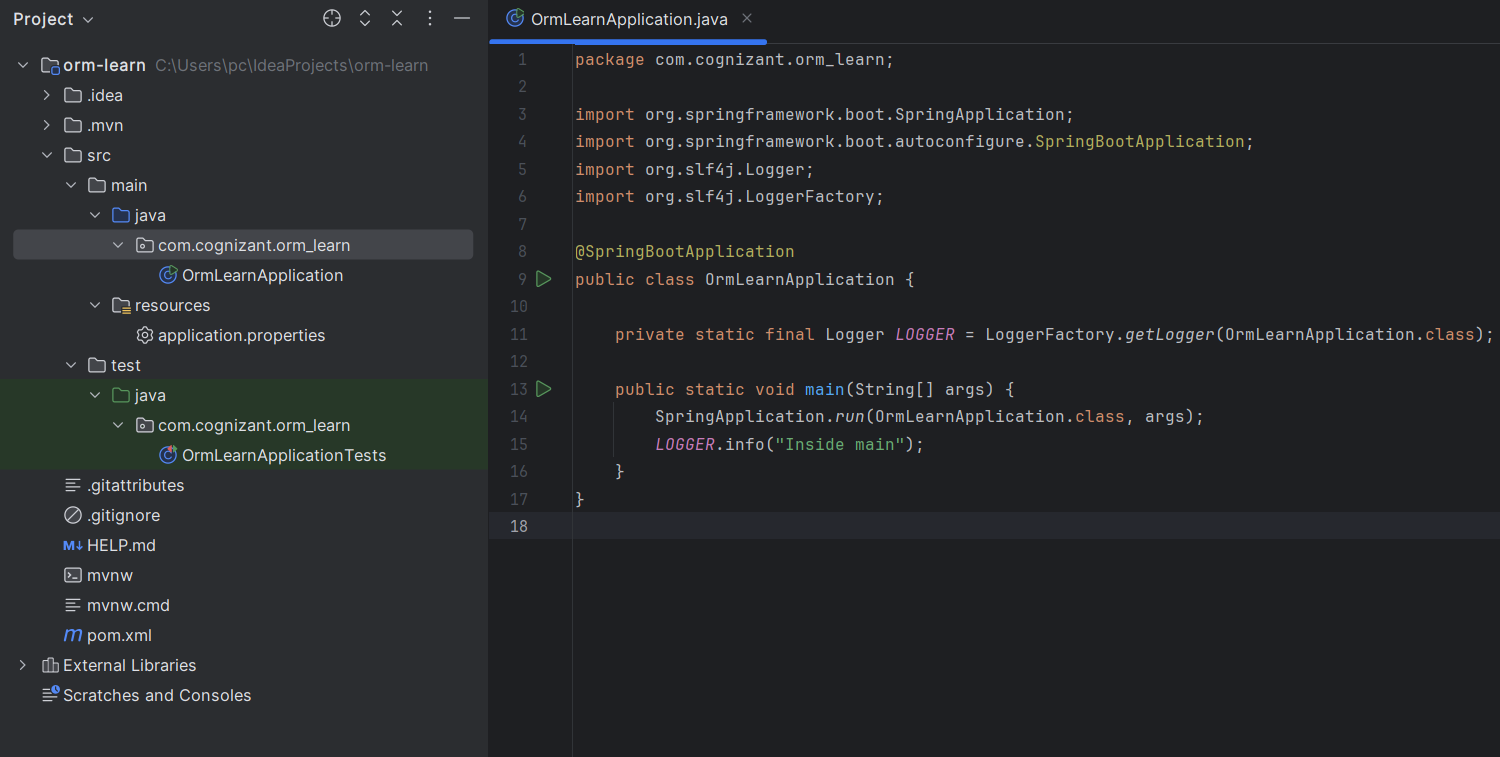
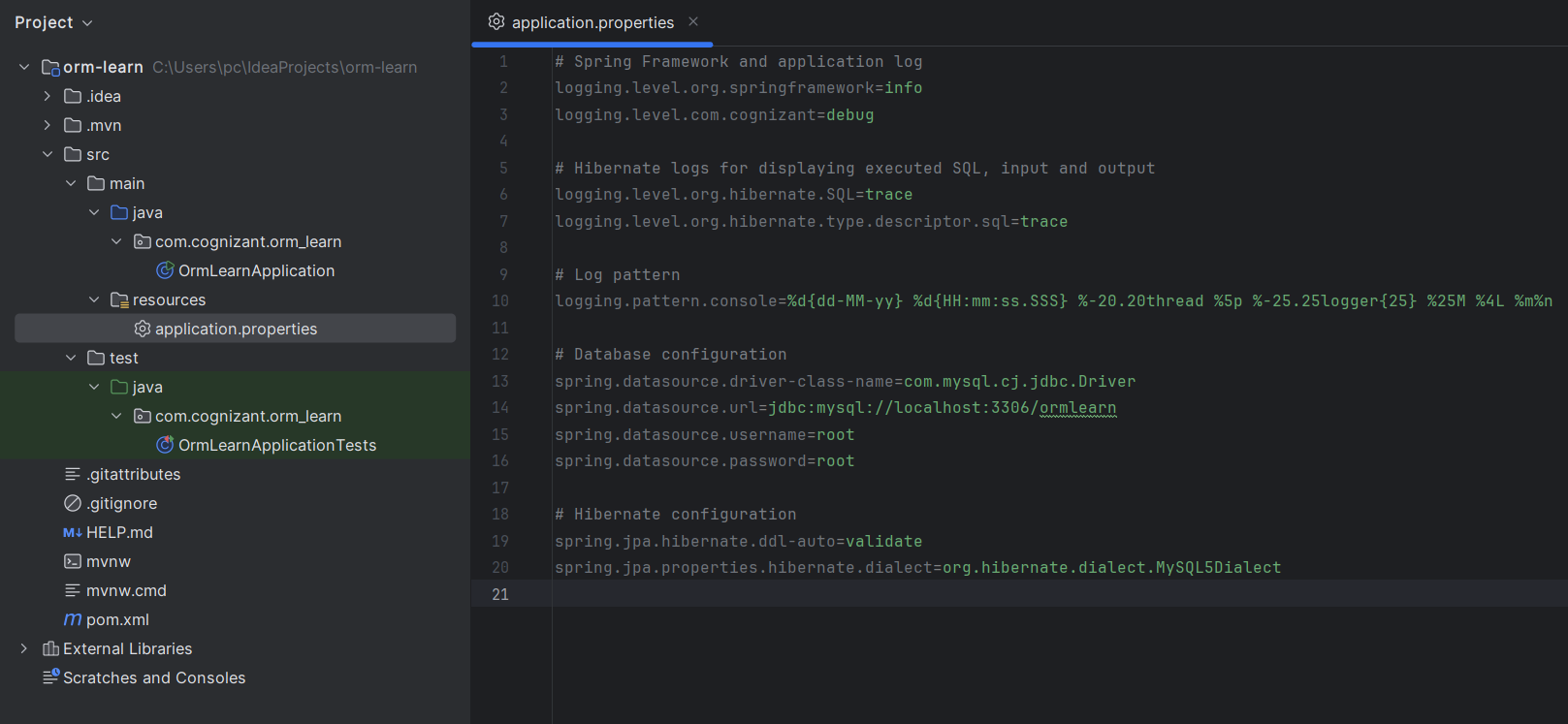
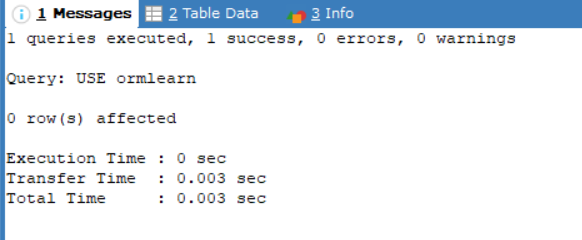
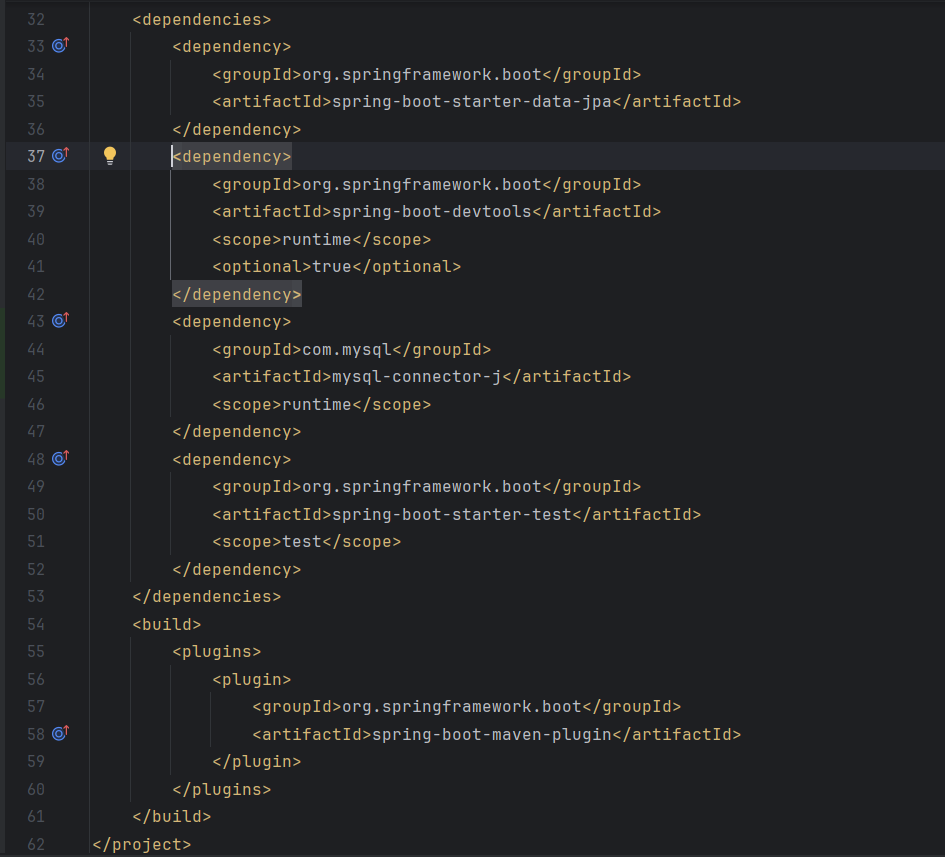
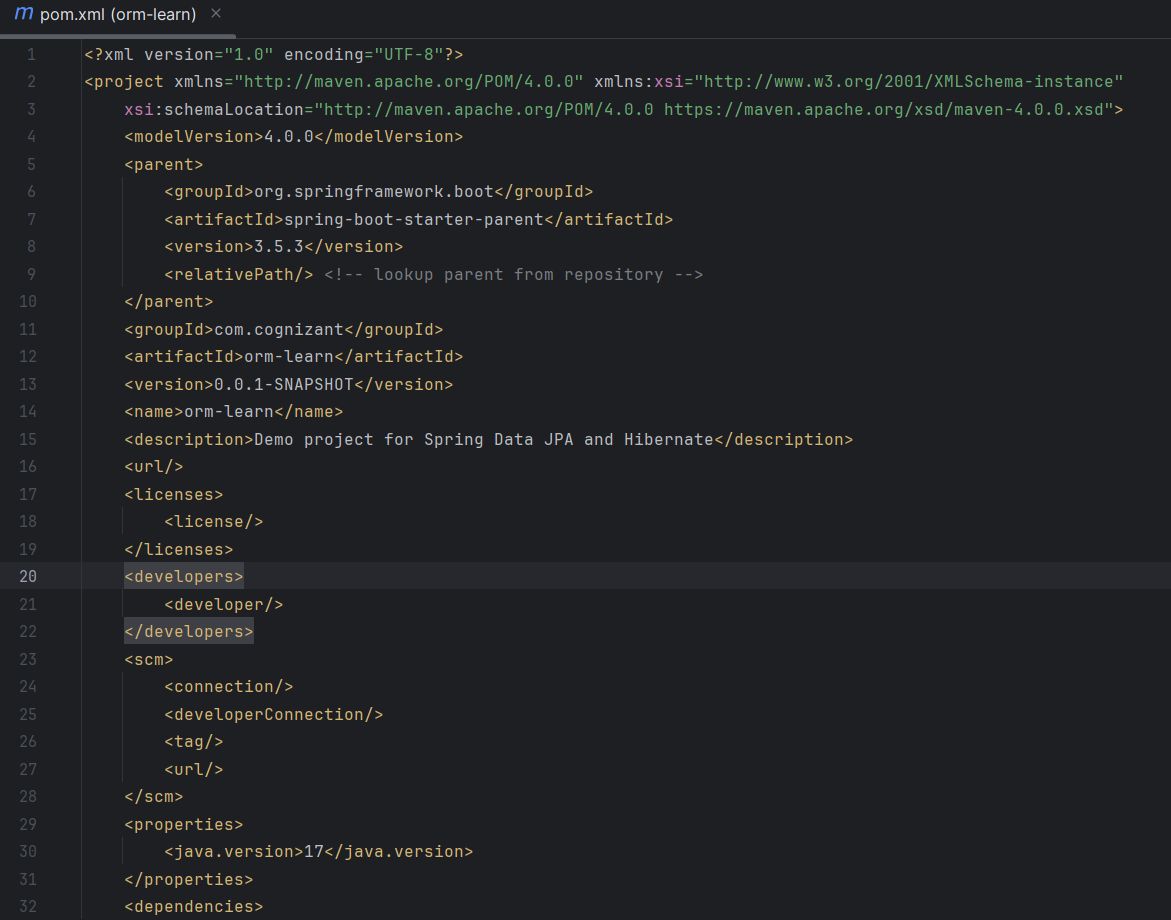
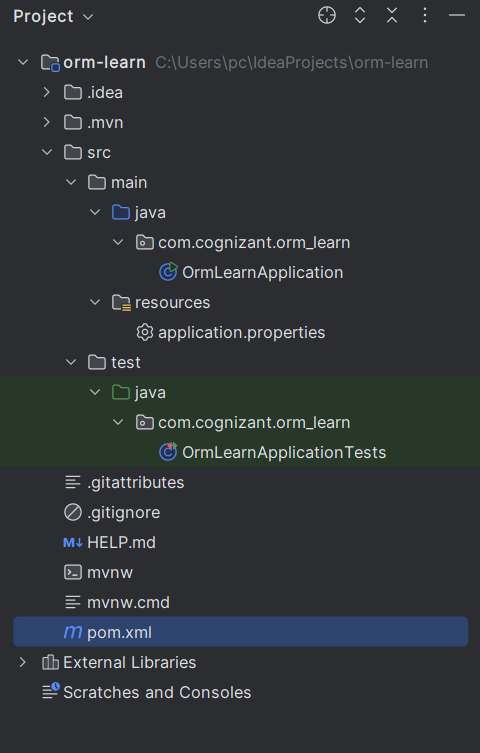
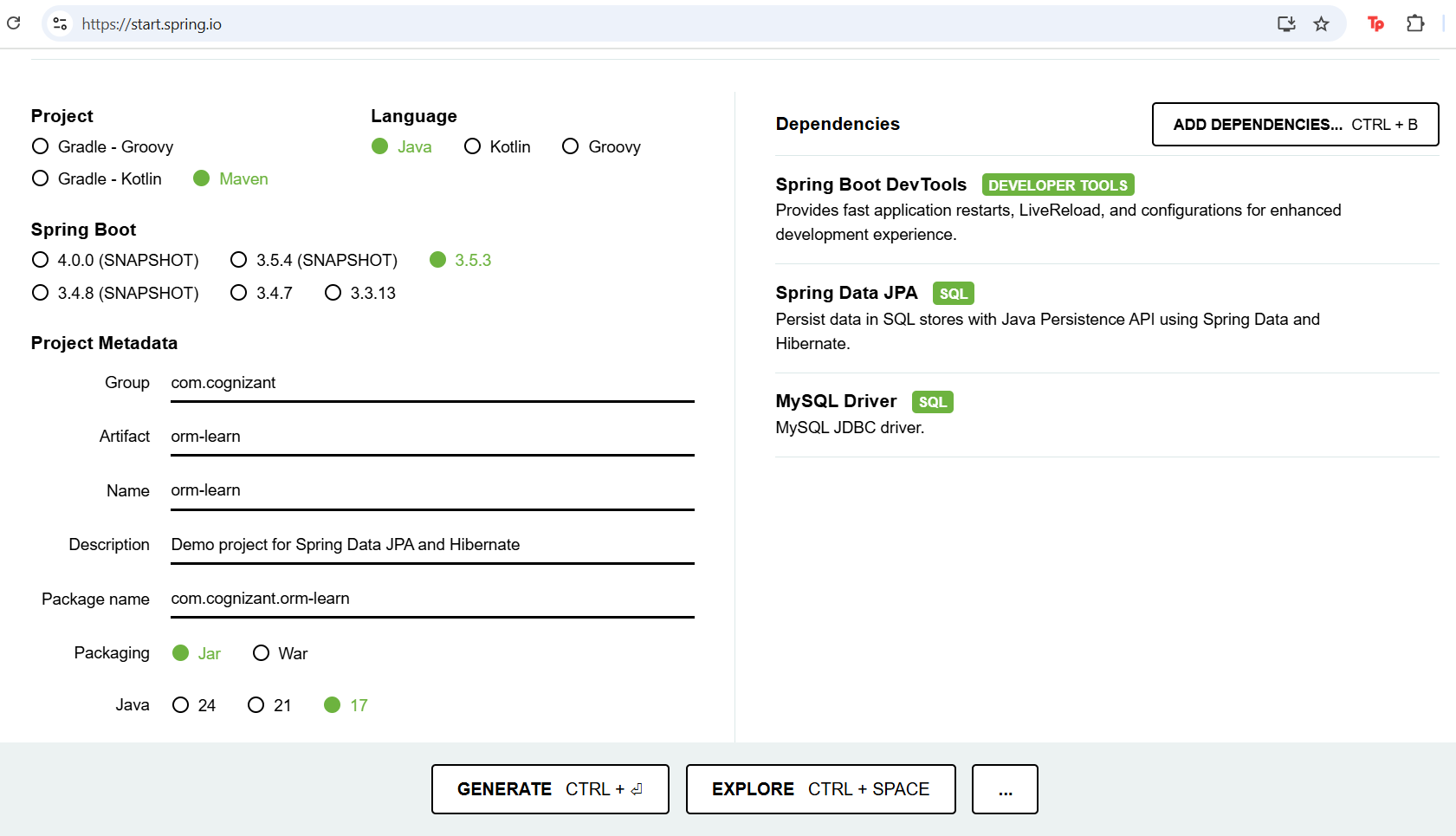
### Hibernate

Hibernate is the most popular implementation of JPA. It provides the actual functionality to persist Java objects to the database. You can use Hibernate directly (even without JPA), and it offers many features beyond the JPA specification like better caching, lazy loading, and advanced querying.

### Spring Data JPA

Spring Data JPA builds on top of JPA and Hibernate and simplifies data access even further. It allows you to write almost no SQL or JPQL — just define interfaces like BookRepository, and Spring will automatically create implementations for common operations like save(), findById(), and delete().

**Exercise 1: Spring Data JPA - Quick Example**

****