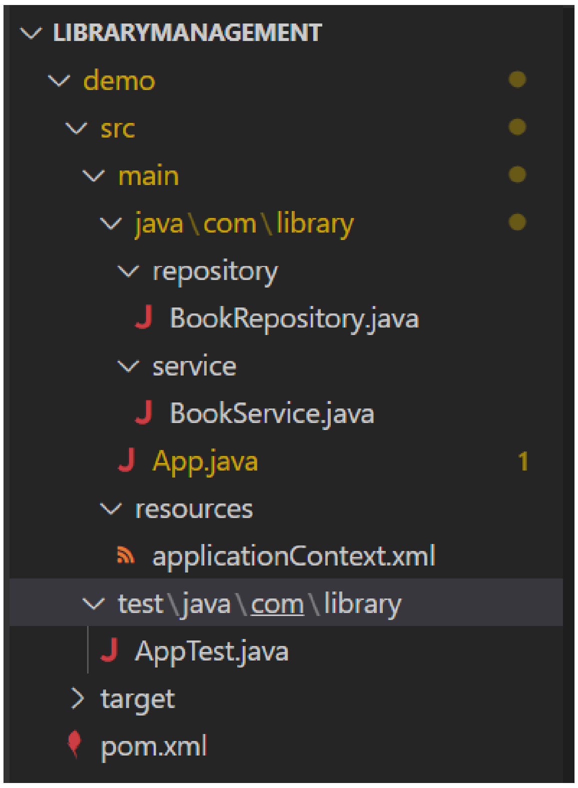
WEEK 3

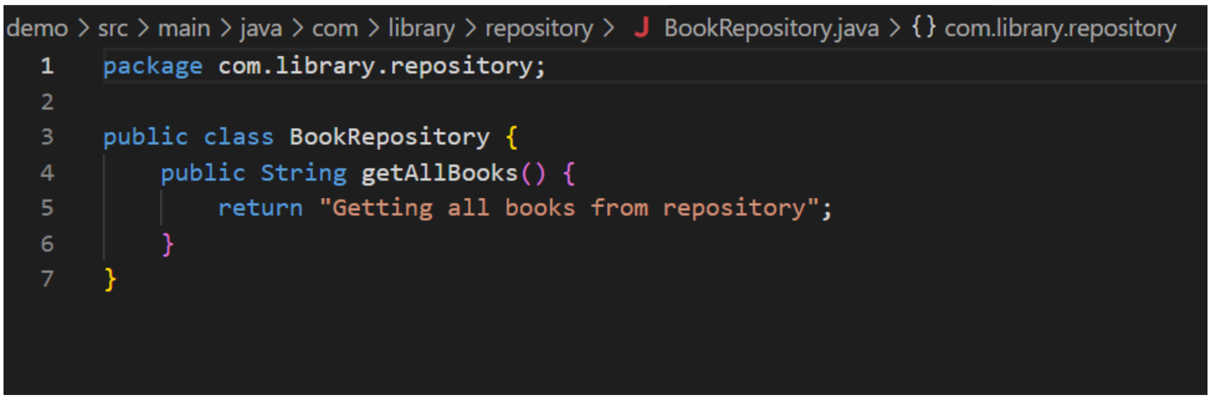
Exercise 1: Configuring a Basic Spring Application

File Structure:



```
LIBRARYMANAGEMENT
  demo
    src
      main
        java\com\library
          repository
            BookRepository.java
          service
            BookService.java
          App.java                1
        resources
          applicationContext.xml
      test\java\com\library
        AppTest.java
    target
    pom.xml
```

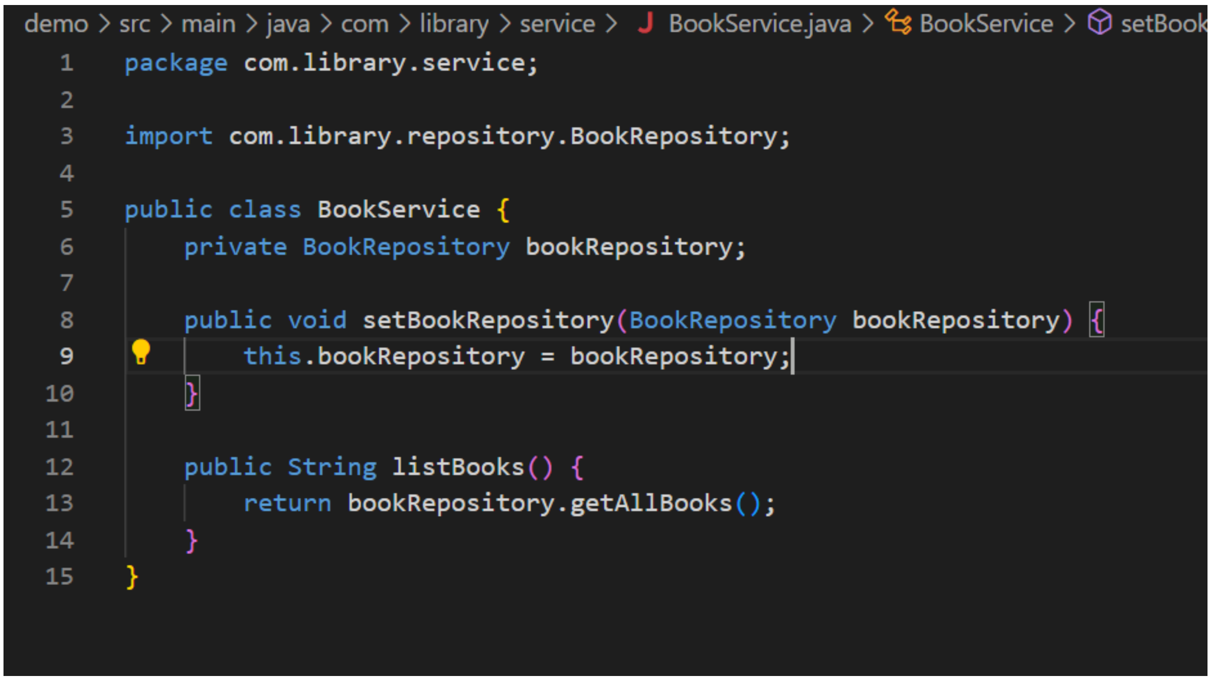
BookRepository.java



demo > src > main > java > com > library > repository > BookRepository.java > {} com.library.repository

```java
package com.library.repository;

public class BookRepository {
    public String getAllBooks() {
        return "Getting all books from repository";
    }
}
```

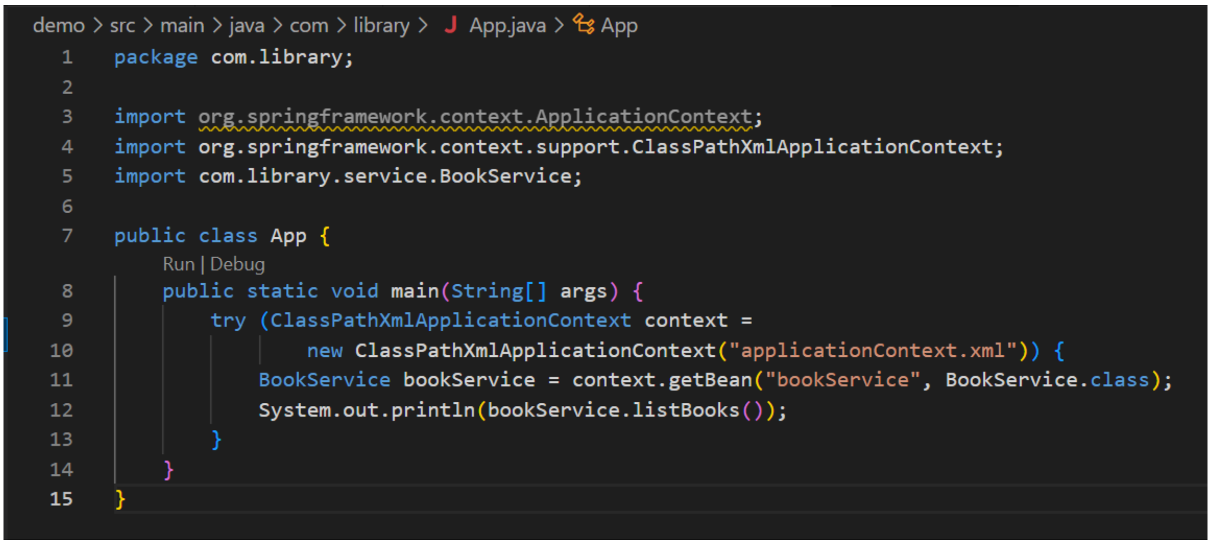
BookService.java



demo > src > main > java > com > library > service > BookService.java > BookService > setBook

```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public String listBooks() {
        return bookRepository.getAllBooks();
    }
}
```

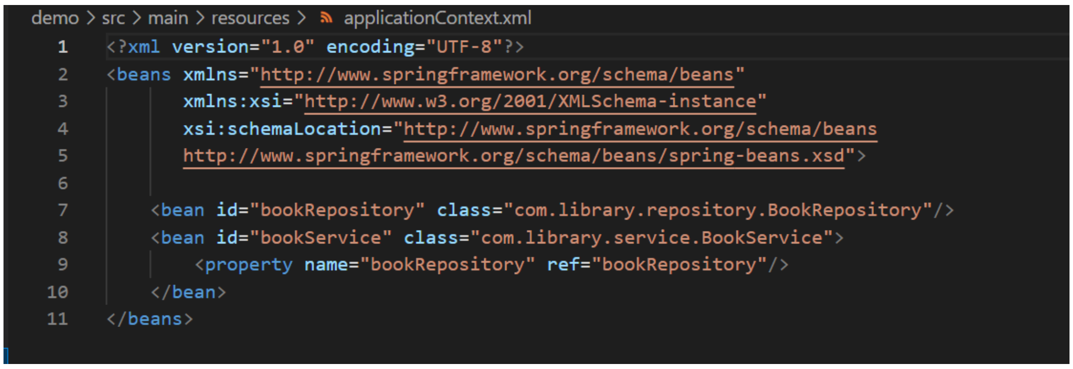
App.java



demo > src > main > java > com > library > App.java > App

```java
package com.library;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;

public class App {
    Run | Debug
    public static void main(String[] args) {
        try (ClassPathXmlApplicationContext context =
                new ClassPathXmlApplicationContext("applicationContext.xml")) {
            BookService bookService = context.getBean("bookService", BookService.class);
            System.out.println(bookService.listBooks());
        }
    }
}
```

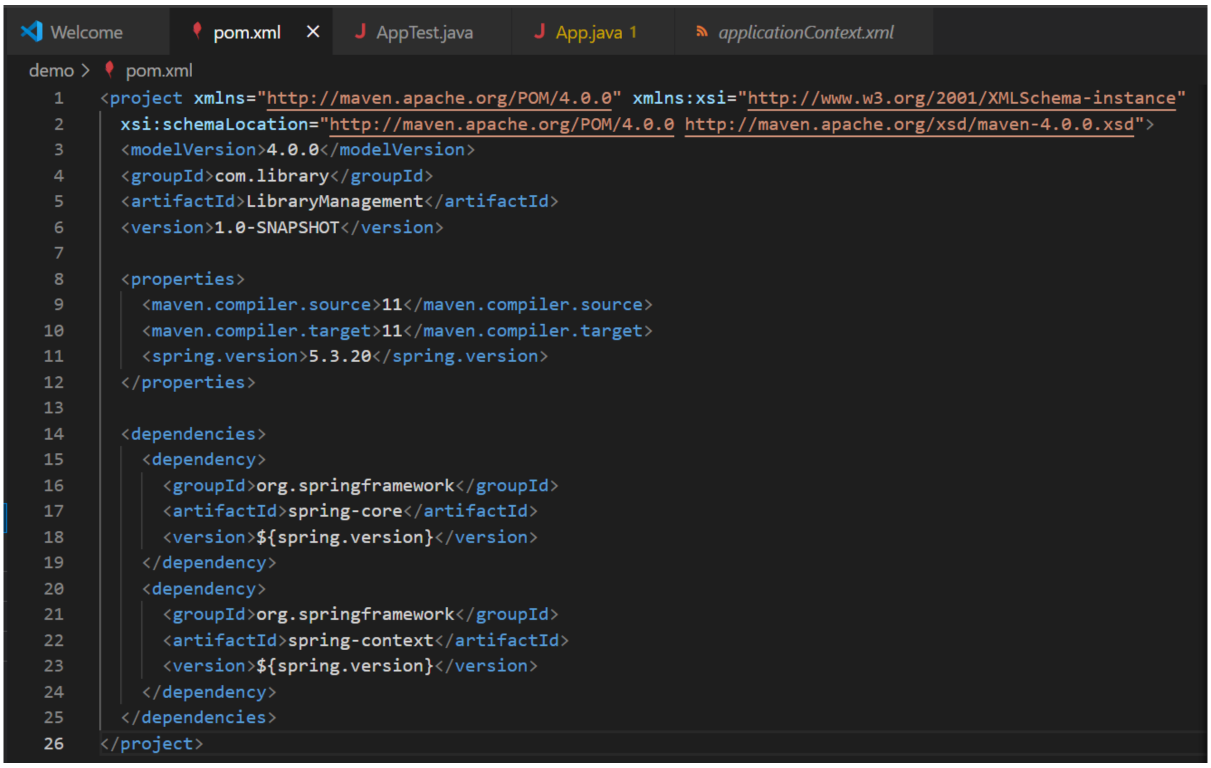
applicationContext.xml



demo > src > main > resources > applicationContext.xml

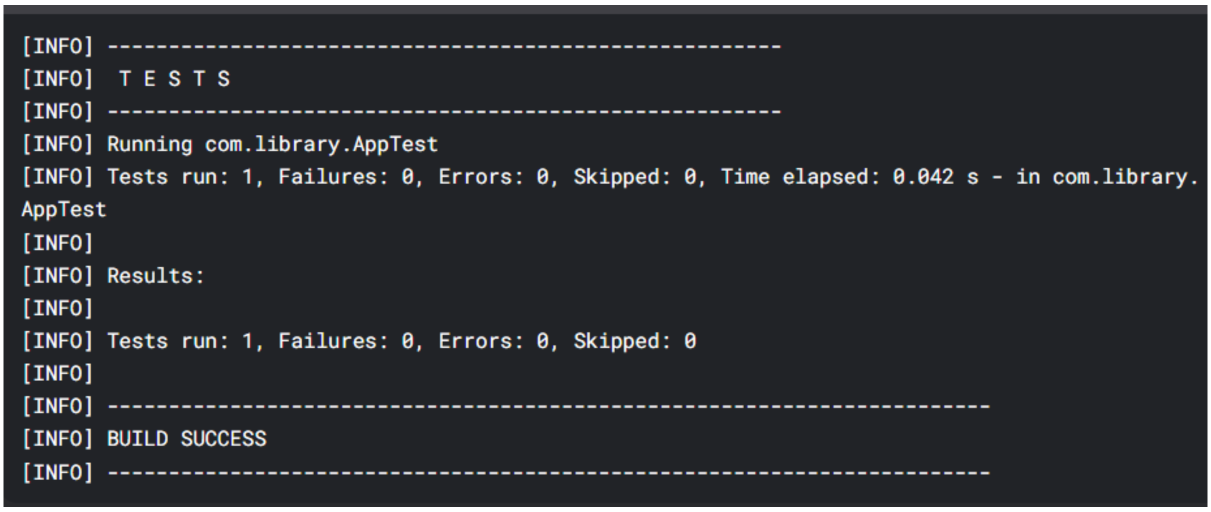
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.library.repository.BookRepository"/>
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>
</beans>
```

pom.xml



```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.library</groupId>
  <artifactId>LibraryManagement</artifactId>
  <version>1.0-SNAPSHOT</version>

  <properties>
    <maven.compiler.source>11</maven.compiler.source>
    <maven.compiler.target>11</maven.compiler.target>
    <spring.version>5.3.20</spring.version>
  </properties>

  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>${spring.version}</version>
    </dependency>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>${spring.version}</version>
    </dependency>
  </dependencies>
</project>
```

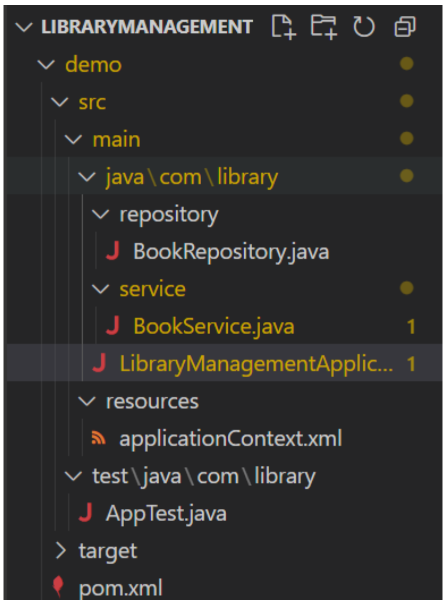
Output:



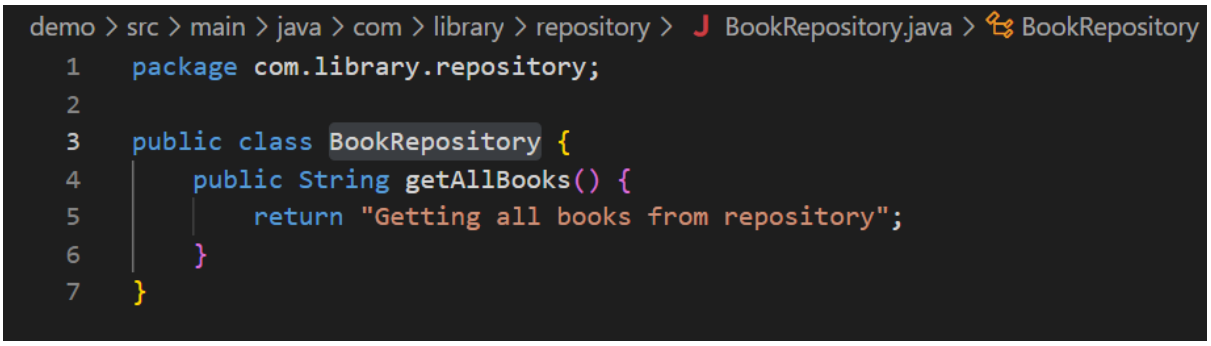
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.library.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.042 s - in com.library.
AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
```

Exercise 2: Implementing Dependency Injection

File structure:

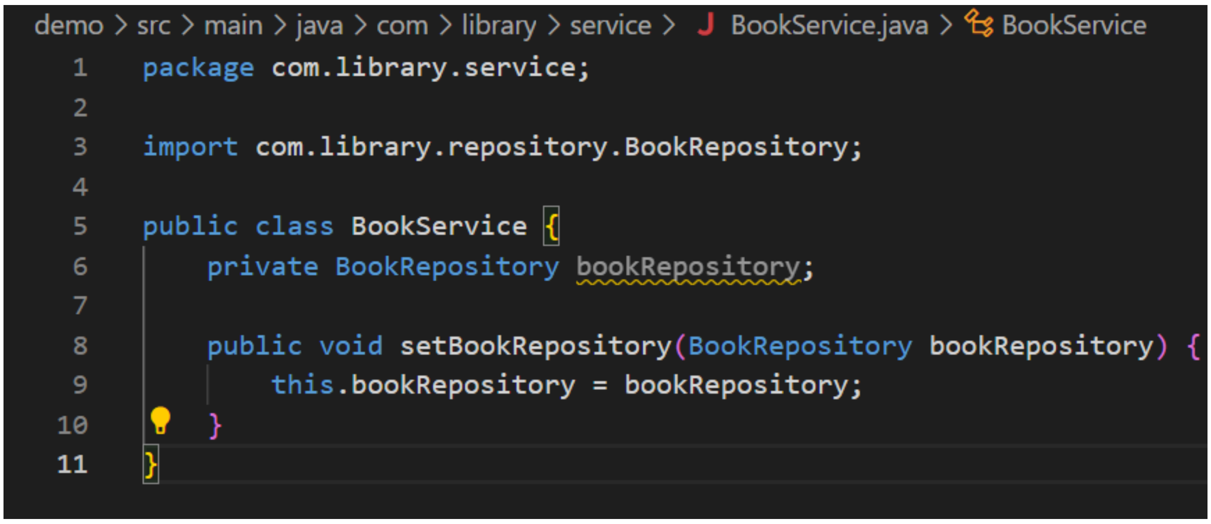

```
LIBRARYMANAGEMENT
  demo
    src
      main
        java\com\library
          repository
            BookRepository.java
          service
            BookService.java        1
          LibraryManagementApplic... 1
        resources
          applicationContext.xml
      test\java\com\library
        AppTest.java
    target
    pom.xml
```

BookRepository.java


demo > src > main > java > com > library > repository > BookRepository.java > BookRepository

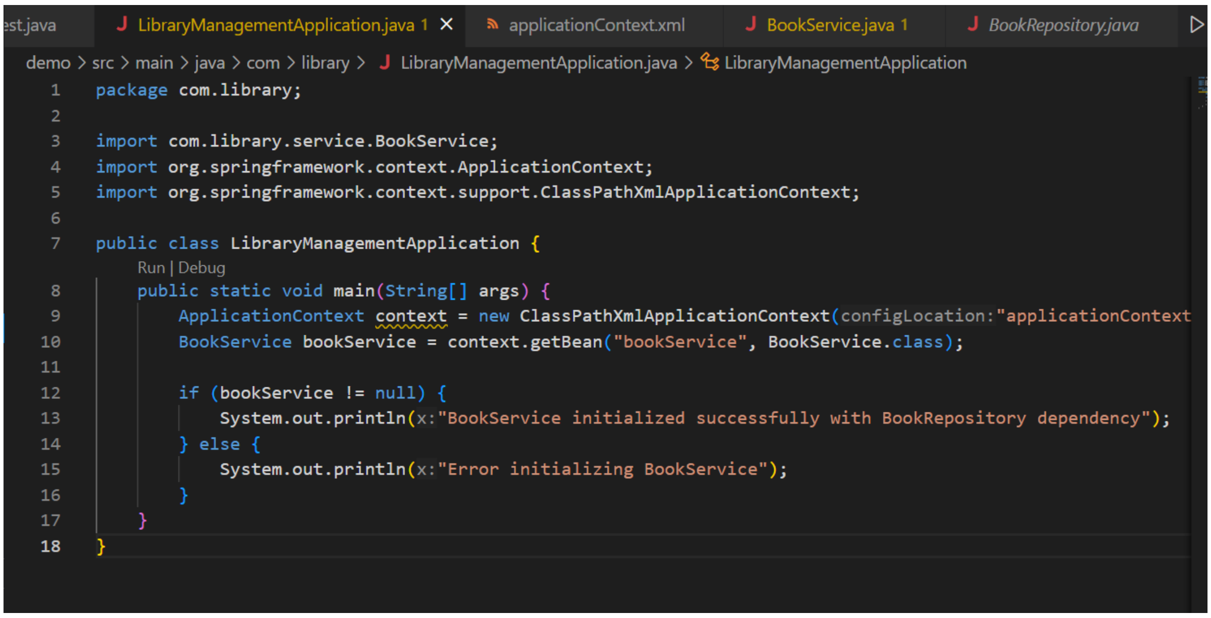
```java
package com.library.repository;

public class BookRepository {
    public String getAllBooks() {
        return "Getting all books from repository";
    }
}
```

BookService.java


demo > src > main > java > com > library > service > BookService.java > BookService

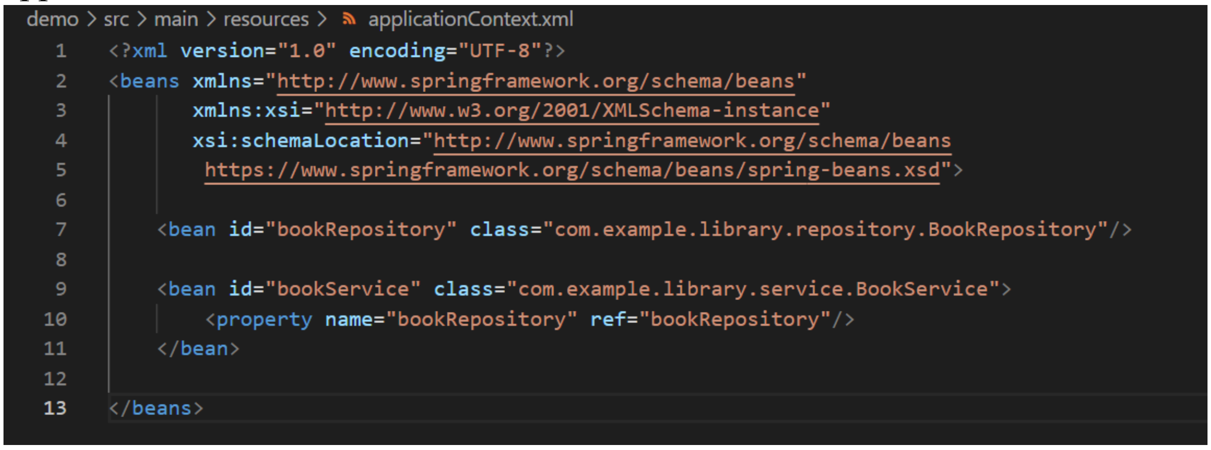
```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }
}
```

LibraryManagementApplication.java


est.java
LibraryManagementApplication.java 1
applicationContext.xml
BookService.java 1
BookRepository.java
demo > src > main > java > com > library > LibraryManagementApplication.java > LibraryManagementApplication
Run | Debug

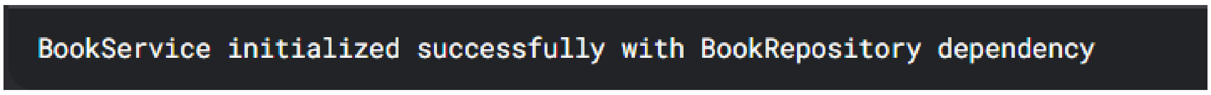
```java
package com.library;

import com.library.service.BookService;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext(configLocation:"applicationContext
        BookService bookService = context.getBean("bookService", BookService.class);

        if (bookService != null) {
            System.out.println(x:"BookService initialized successfully with BookRepository dependency");
        } else {
            System.out.println(x:"Error initializing BookService");
        }
    }
}
```

applicationContext.xml



```xml
<?xml version="1.0" encoding="UTF-8"?>
<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
        https://www.springframework.org/schema/beans/spring-beans.xsd">

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

    <bean id="bookService" class="com.example.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>
```

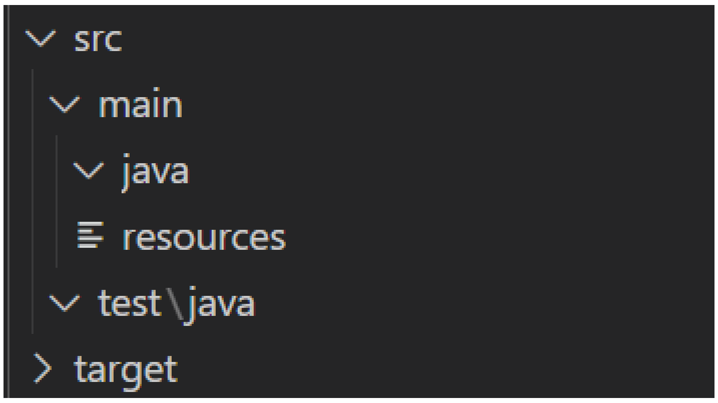
output:



```
BookService initialized successfully with BookRepository dependency
```

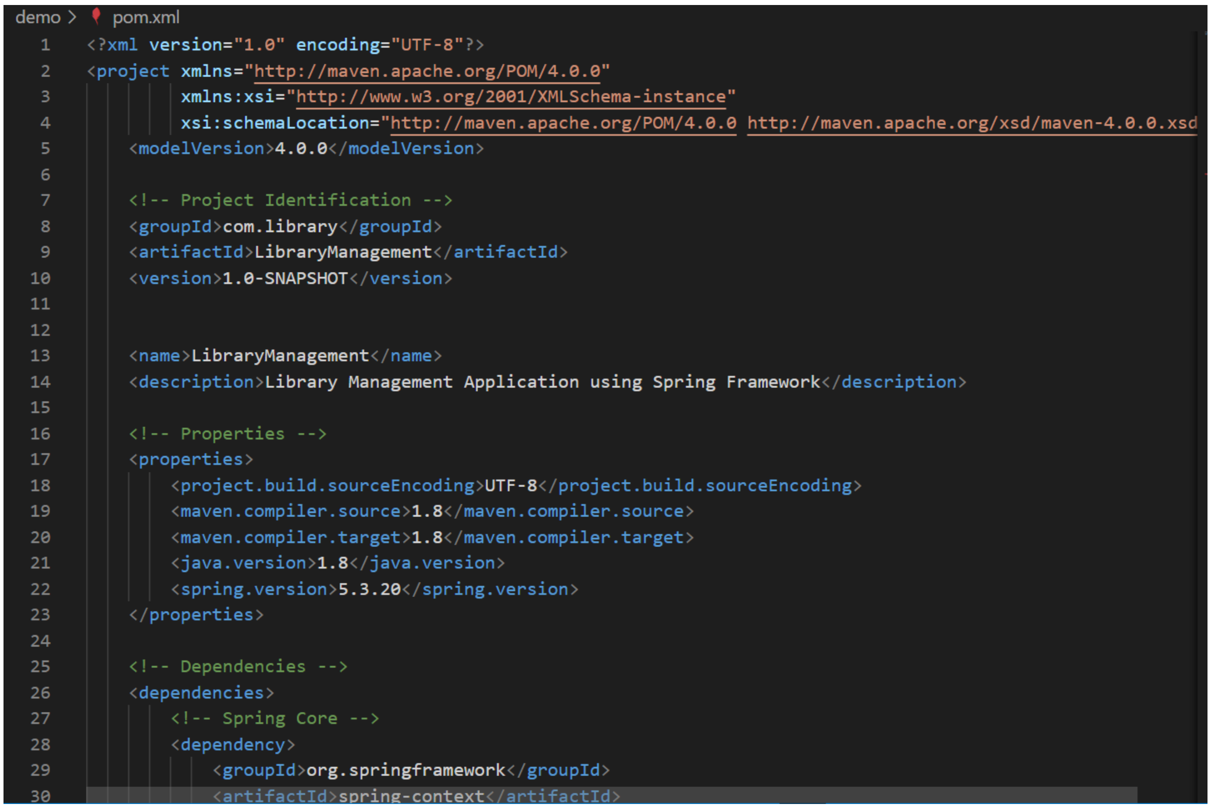
Exercise 4: Creating and Configuring a Maven Project

File structure:



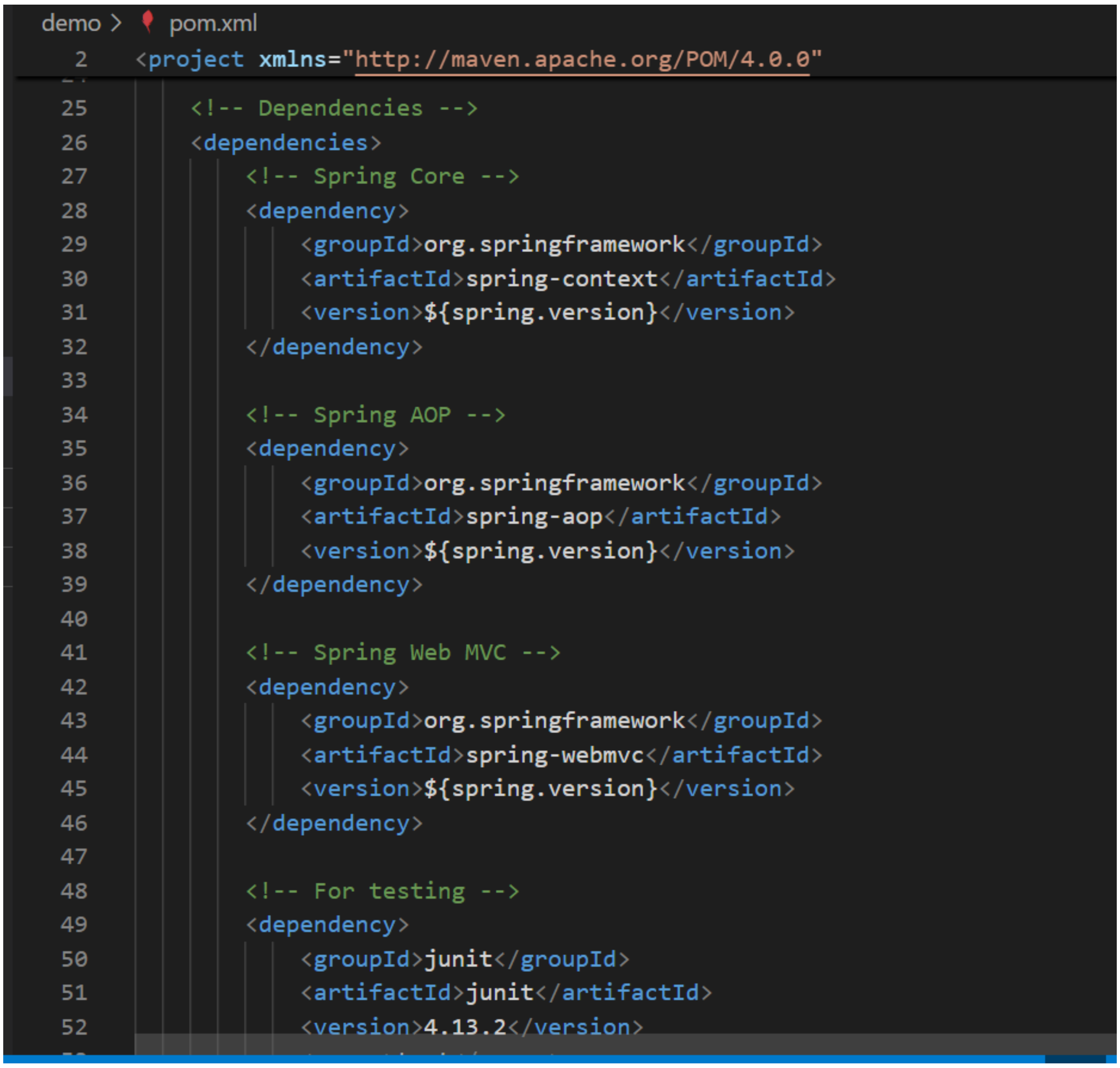
```
∨ src
  ∨ main
    ∨ java
      resources
  ∨ test\java
> target
```

pom.xml

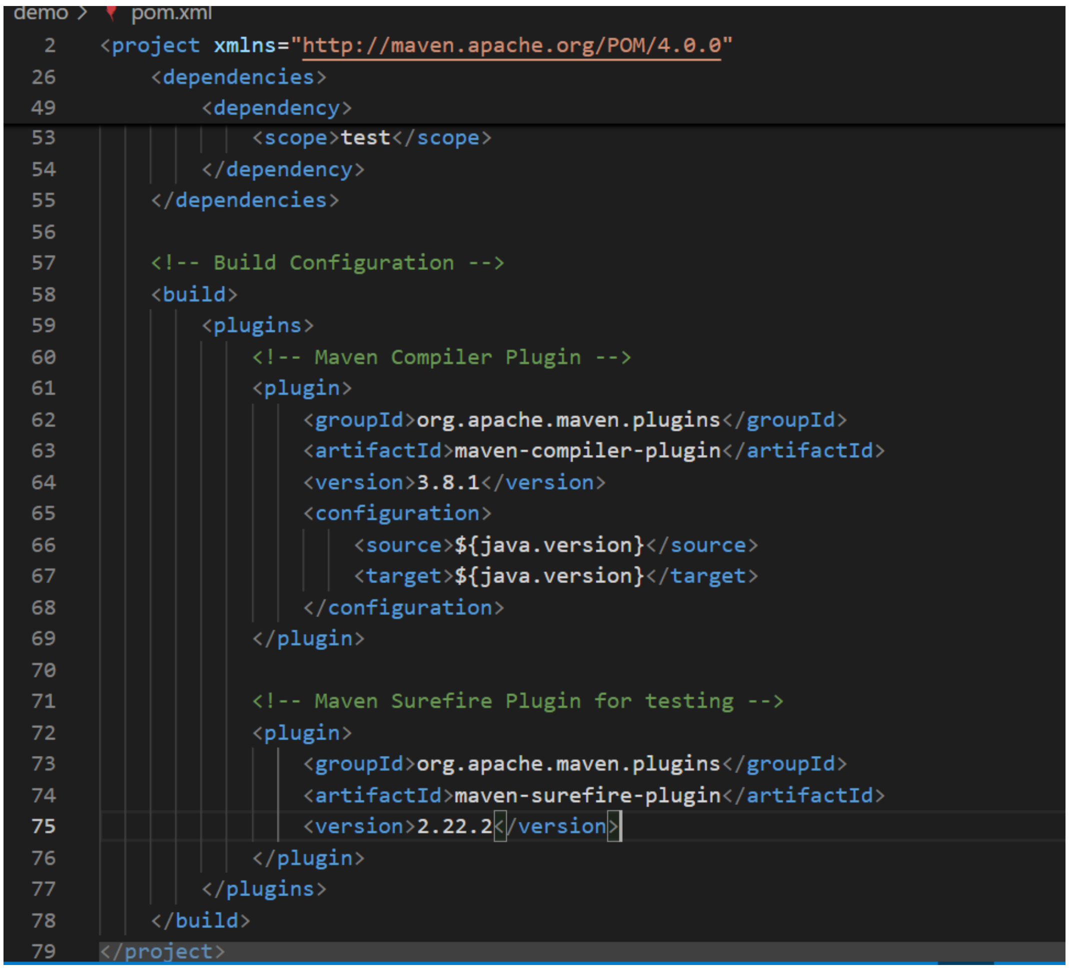


demo > pom.xml

```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 http://maven.apache.org/xsd/maven-4.0.0.xsd
    <modelVersion>4.0.0</modelVersion>

    <!-- Project Identification -->
    <groupId>com.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0-SNAPSHOT</version>


    <name>LibraryManagement</name>
    <description>Library Management Application using Spring Framework</description>

    <!-- Properties -->
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <java.version>1.8</java.version>
        <spring.version>5.3.20</spring.version>
    </properties>

    <!-- Dependencies -->
    <dependencies>
        <!-- Spring Core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
```

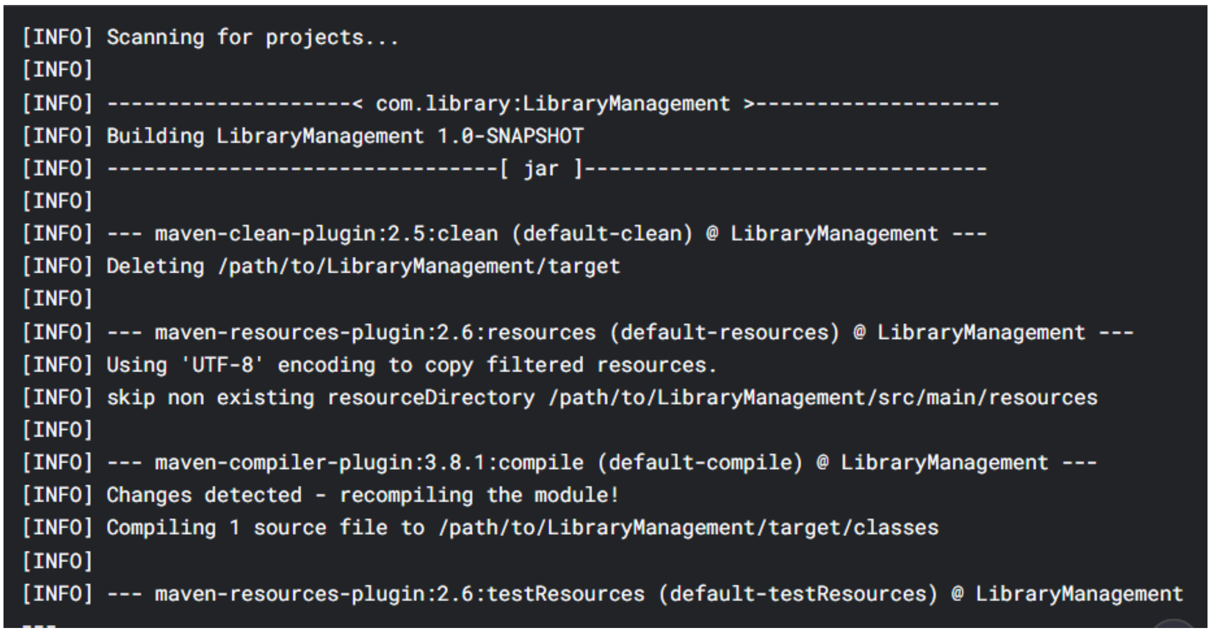


```xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    <!-- Dependencies -->
    <dependencies>
        <!-- Spring Core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- Spring AOP -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- Spring Web MVC -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>${spring.version}</version>
        </dependency>

        <!-- For testing -->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.13.2</version>
```



```
demo > pom.xml
<project xmlns="http://maven.apache.org/POM/4.0.0"
    <dependencies>
        <dependency>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <!-- Build Configuration -->
    <build>
        <plugins>
            <!-- Maven Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                </configuration>
            </plugin>

            <!-- Maven Surefire Plugin for testing -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>2.22.2</version>
            </plugin>
        </plugins>
    </build>
</project>
```

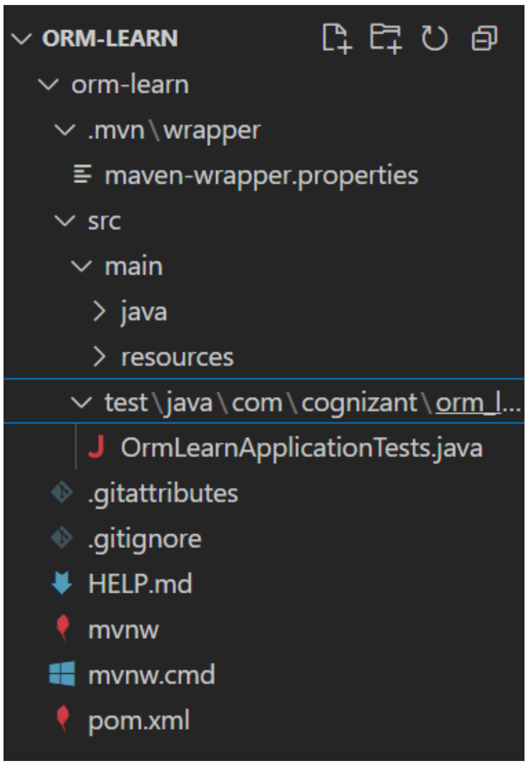
Output:



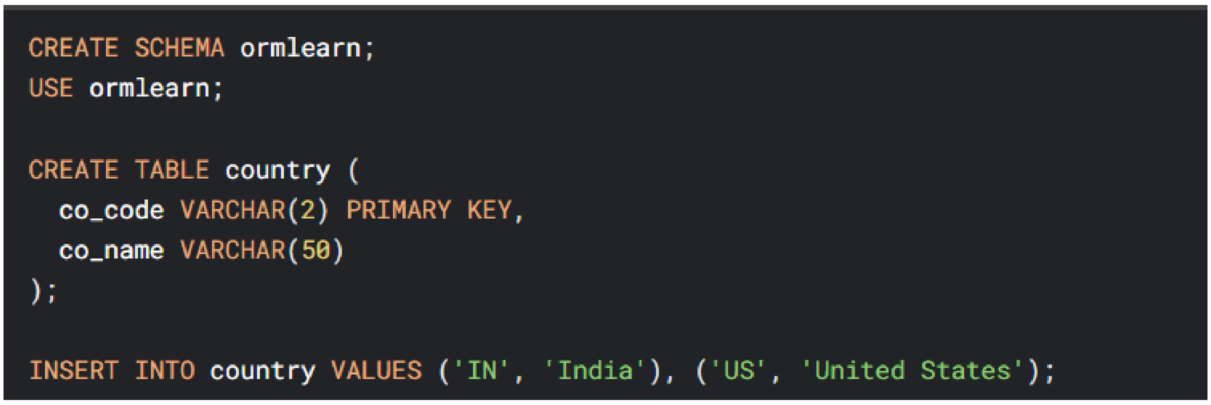
```
[INFO] Scanning for projects...
[INFO]
[INFO] --------------------< com.library:LibraryManagement >--------------------
[INFO] Building LibraryManagement 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ LibraryManagement ---
[INFO] Deleting /path/to/LibraryManagement/target
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ LibraryManagement ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /path/to/LibraryManagement/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ LibraryManagement ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to /path/to/LibraryManagement/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ LibraryManagement
---
```

Spring Data JPA - Quick Example

File Structure:

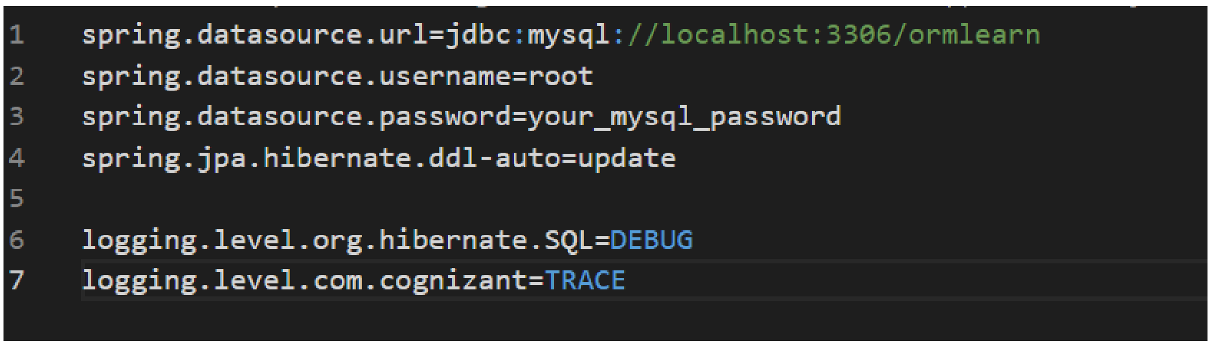


```
ORM-LEARN
  orm-learn
    .mvn\wrapper
      maven-wrapper.properties
    src
      main
        java
        resources
      test\java\com\cognizant\orm_l...
        OrmLearnApplicationTests.java
    .gitattributes
    .gitignore
    HELP.md
    mvnw
    mvnw.cmd
    pom.xml
```



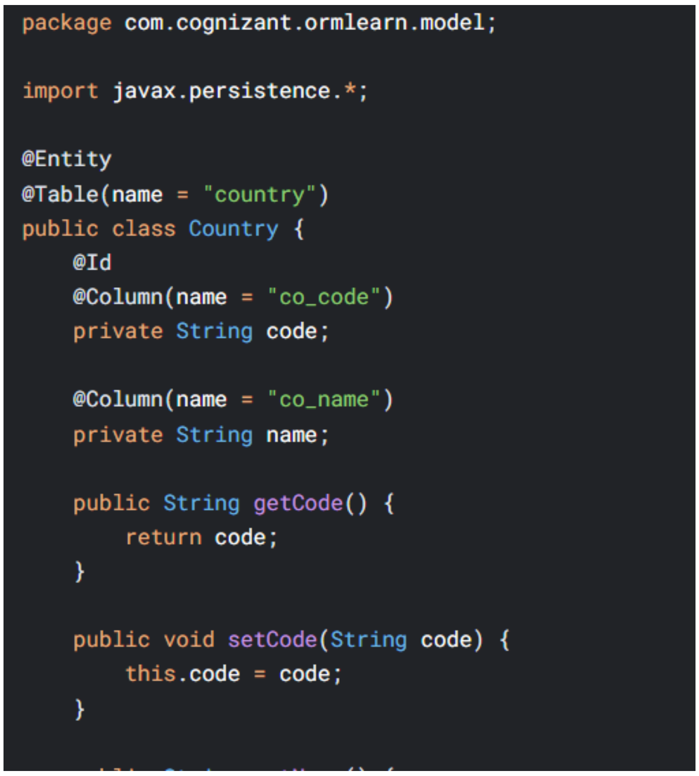
```sql
CREATE SCHEMA ormlearn;
USE ormlearn;

CREATE TABLE country (
  co_code VARCHAR(2) PRIMARY KEY,
  co_name VARCHAR(50)
);

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

application.properties

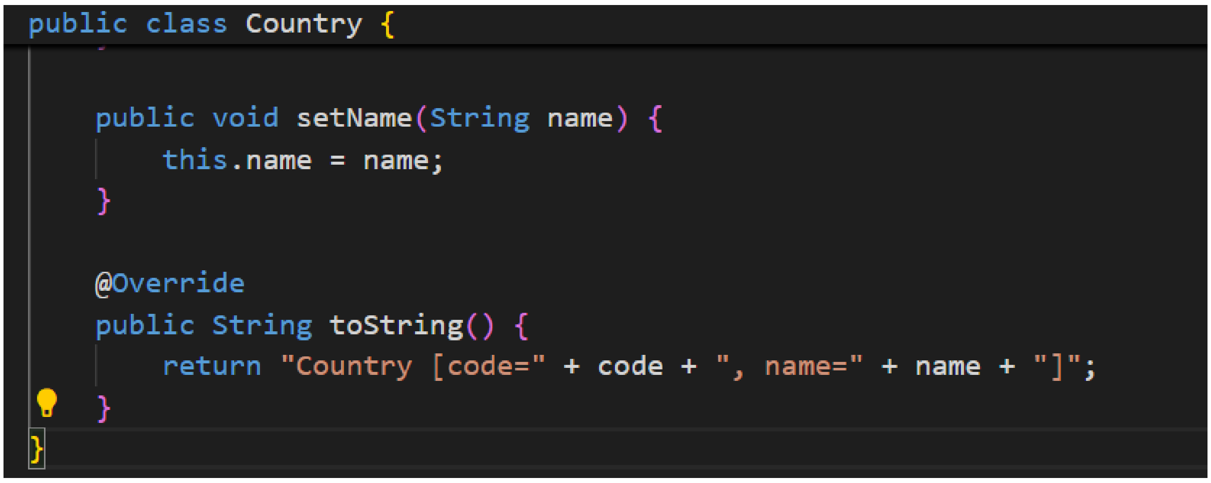


```
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=your_mysql_password
spring.jpa.hibernate.ddl-auto=update

logging.level.org.hibernate.SQL=DEBUG
logging.level.com.cognizant=TRACE
```

Country.java

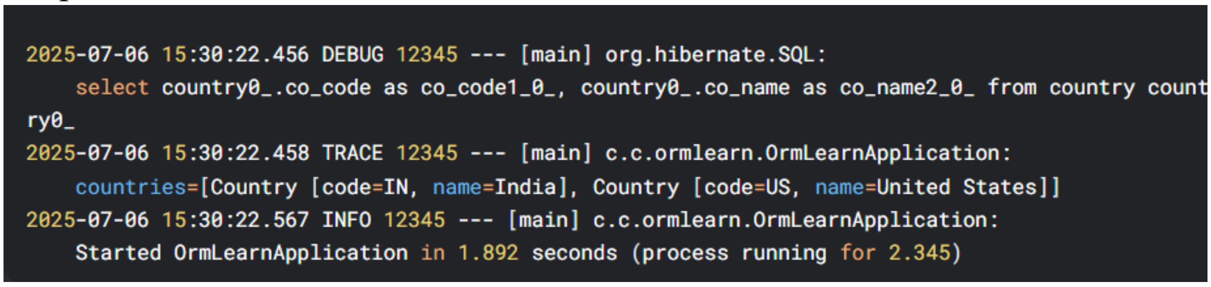


```java
package com.cognizant.ormlearn.model;

import javax.persistence.*;

@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;
    }
```



```java
public class Country {

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

    @Override
    public String toString() {
        return "Country [code=" + code + ", name=" + name + "]";
    }
}
```

Output:



```
2025-07-06 15:30:22.456 DEBUG 12345 --- [main] org.hibernate.SQL:
    select country0_.co_code as co_code1_0_, country0_.co_name as co_name2_0_ from country count
ry0_
2025-07-06 15:30:22.458 TRACE 12345 --- [main] c.c.ormlearn.OrmLearnApplication:
    countries=[Country [code=IN, name=India], Country [code=US, name=United States]]
2025-07-06 15:30:22.567 INFO 12345 --- [main] c.c.ormlearn.OrmLearnApplication:
    Started OrmLearnApplication in 1.892 seconds (process running for 2.345)
```

Difference between JPA, Hibernate and Spring Data JPA

Java Persistence API (JPA)

- **Standard specification** (JSR 338) for object-relational mapping (ORM) in Java
- Defines interfaces and annotations for persisting Java objects to a database
- **Not an implementation** - just provides the API specification
- Part of Java EE (now Jakarta EE)
- Common annotations: @Entity, @Table, @Id, @Column

Hibernate

- **Most popular implementation** of the JPA specification
- Provides all the JPA features plus additional proprietary features
- Handles all the low-level database interactions
- Manages database connections, transactions, and SQL generation
- More verbose configuration and coding compared to Spring Data JPA

Spring Data JPA

- **Abstraction layer** on top of JPA providers (like Hibernate)
- Reduces boilerplate code through repository interfaces
- Provides powerful features like:
  - Automatic query generation from method names
  - Pagination and sorting support
  - Custom query annotations (@Query)
- Still requires a JPA provider (Hibernate, EclipseLink, etc.) underneath
- Integrates seamlessly with Spring's transaction management

Aspect	**JPA**	**Hibernate**	**Spring Data JPA**
Nature	Specification	Implementation	Abstraction Layer
Boilerplate	Medium	High	Low
Configuration	Standard	Proprietary extensions	Spring-style
Query Creation	JPQL / Criteria API	HQL / Criteria	Method name conventions
Transaction Mgmt	Depends on implementation	Manual or JTA	Spring @Transactional