**DIGITAL NURTURE 4.0 JAVA -FSE**

**Submitted By:**

**THAMIMUL ANSARI M**

**Superset ID: 6400989**

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

**Scenario:**  
A company is developing a web application for managing a library. I need to use the Spring Framework to handle the backend operations.

#### Steps & Output:

1. **Set Up a Spring Project:**
   * Created a Maven project named LibraryManagement
   * Added Spring Core dependency in pom.xml:

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.36</version>  
</dependency>

1. **Configured the Application Context:**
   * Created applicationContext.xml in src/main/resources

<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>

1. **Defined Service and Repository Classes:**

* **com.library.repository.BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void save() {  
 System.out.println("Book saved to repository.");  
 }  
}

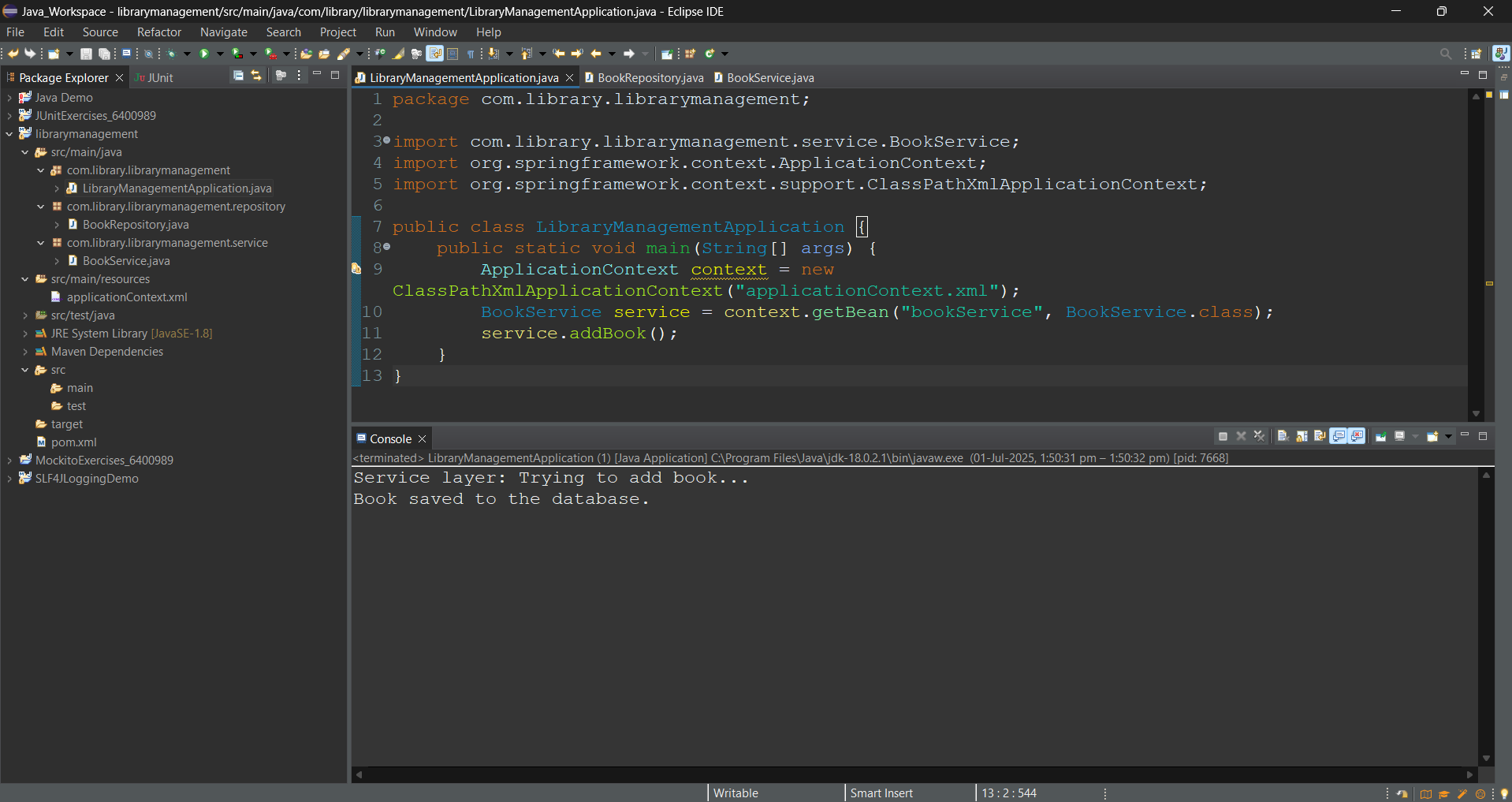
* **com.library.service.BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook() {  
 bookRepository.save();  
 System.out.println("Book added using BookService.");  
 }  
}

1. **Main Class to Run:**

package com.library.main;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
import com.library.service.BookService;  
  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = (BookService) context.getBean("bookService");  
 service.addBook();  
 }  
}

**Output:**



✅ **Exercise 2: Implementing Dependency Injection**

**Scenario:**  
I have implemented dependency injection in the Book DI App using Spring Framework's XML configuration and setter injection.

#### Steps & Output:

1. **Created `BookProcessor` class with a setter for `BookStorage`.**

package com.thamimul.bookdi.service;

import com.thamimul.bookdi.repo.BookStorage;

public class BookProcessor {

private BookStorage bookStorage;

// Setter injection method

public void setBookStorage(BookStorage bookStorage) {

this.bookStorage = bookStorage;

}

public void process() {

System.out.println("Preparing to process a book...");

bookStorage.stashBook();

}

}

1. **Defined bean wiring in `applicationContext.xml`:**

<bean id="bookStorage" class="com.thamimul.bookdi.repo.BookStorage" />

<bean id="bookProcessor" class="com.thamimul.bookdi.service.BookProcessor">

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

</bean>

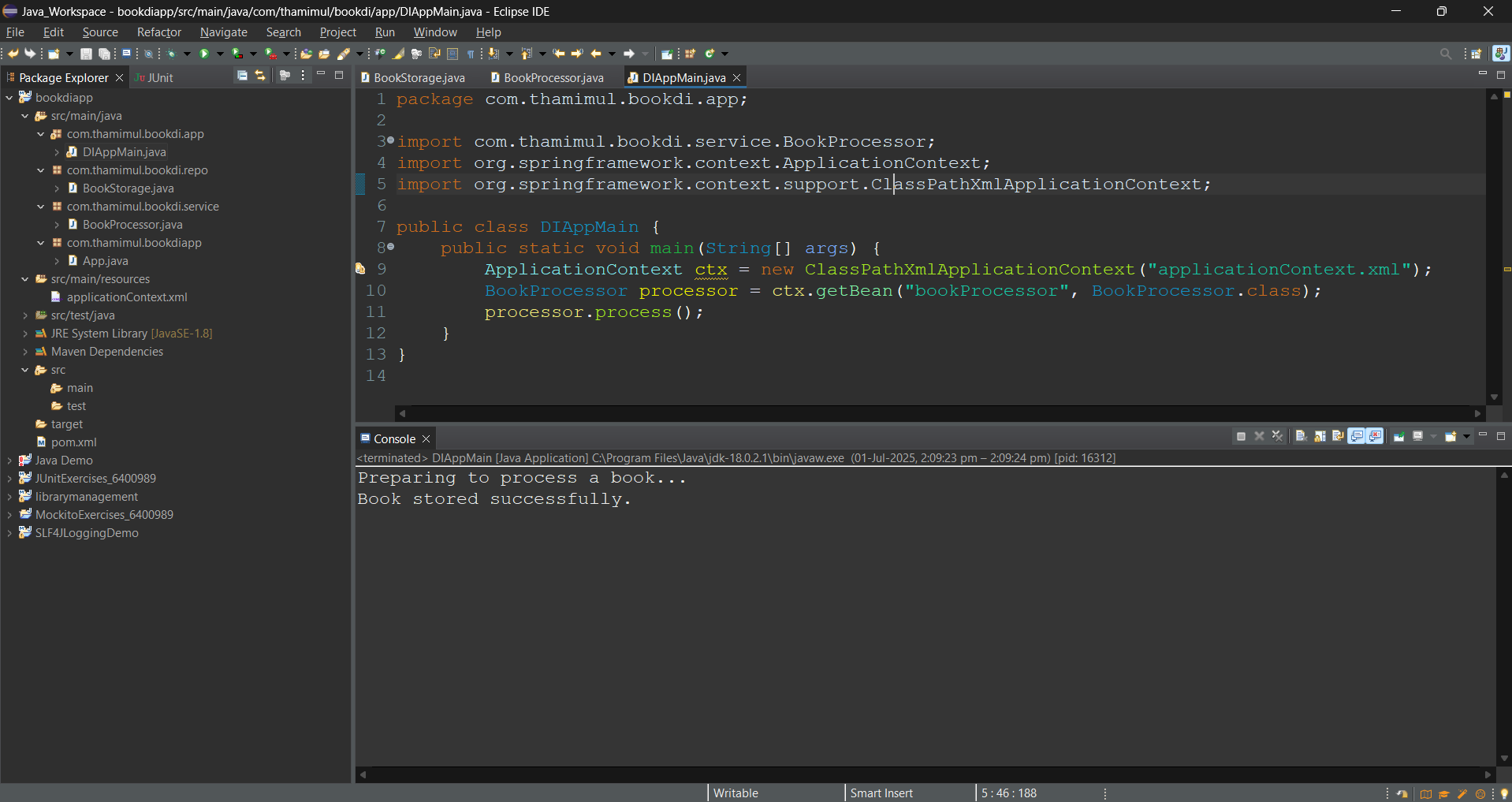
1. **Ran the application using `DIAppMain.java`:**

ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");

BookProcessor processor = ctx.getBean("bookProcessor", BookProcessor.class);

processor.process();

**Output:**



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

**Scenario:**  
You need to set up a new Maven project for the library management application and add Spring dependencies.

#### Steps:

1. **Created Maven Project:** LibraryManagement\_004
2. **Added Spring Dependencies in** ``**:**

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.36</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.36</version>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.36</version>  
 </dependency>  
</dependencies>

1. **Configured Maven Compiler Plugin:**

<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>

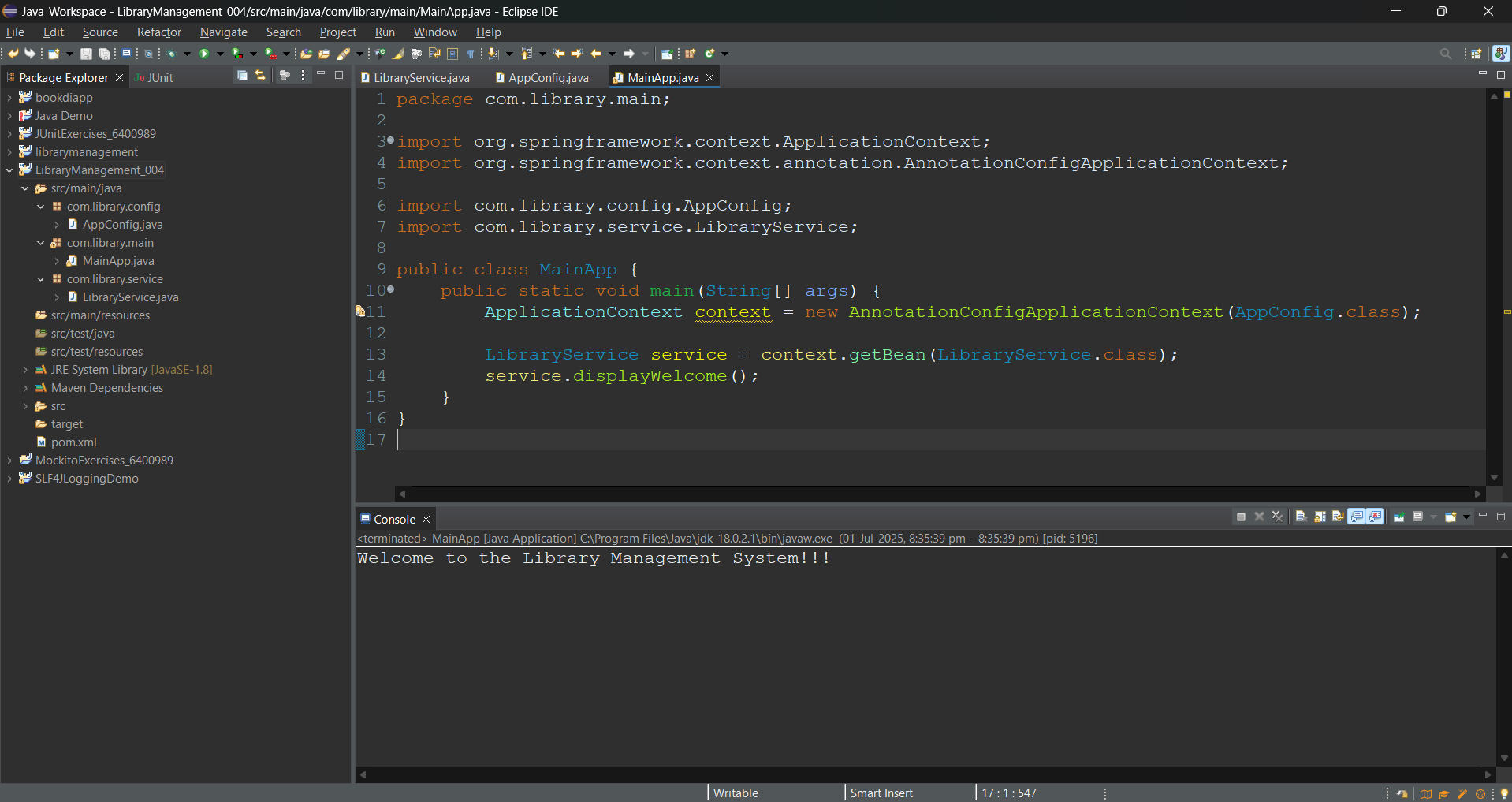
1. **Created a basic Spring test using annotation-based configuration:**

**AppConfig.java:**  
package com.library.config;  
  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
@ComponentScan(basePackages = "com.library")  
public class AppConfig {}

**LibraryService.java:**  
package com.library.service;  
  
import org.springframework.stereotype.Component;  
  
@Component  
public class LibraryService {  
 public void displayWelcome() {  
 System.out.println("📚 Welcome to the Library Management System 📚");  
 }  
}

**MainApp.java:**  
package com.library.main;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.annotation.AnnotationConfigApplicationContext;  
import com.library.config.AppConfig;  
import com.library.service.LibraryService;  
  
public class MainApp {  
 public static void main(String[] args) {  
 ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);  
 LibraryService service = context.getBean(LibraryService.class);  
 service.displayWelcome();  
 }  
}

**Output**:



## ✅ Spring Data JPA - Quick Example

### 🔸 Demonstrate the need and benefit of Spring Data JPA

(evolution of ORM → Spring Data JPA using H2 DB)

### ✅ Exercise: Spring Data JPA with Country Entity

#### 🔹 Country.java

package com.thamimul.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="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.thamimul.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.thamimul.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

#### 🔹 CountryService.java

package com.thamimul.ormlearn.service;

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

import org.springframework.stereotype.Service;

import com.thamimul.ormlearn.model.Country;

import com.thamimul.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country findCountryByCode(String code) {

return countryRepository.findById(code).get();

}

}

#### 🔹 OrmLearnApplication.java

package com.thamimul.ormlearn;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.thamimul.ormlearn.model.Country;

import com.thamimul.ormlearn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryService countryService;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

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

System.out.println(country);

}

}

### 📸 Output: