# Week3

# Spring Core And Maven

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

**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 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>23</maven.compiler.source>  
 <maven.compiler.target>23</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
 <dependencies>  
 <!-- Spring Core -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.34</version>  
 </dependency>  
 </dependencies>  
  
</project>

**applicationContext.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=”org.example.BookRepository”/>  
  
 <bean id=”bookService” class=”org.example.BookService”>  
 <property name=”bookRepository” ref=”bookRepository”/>  
 </bean>  
</beans>

**Main.java**

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

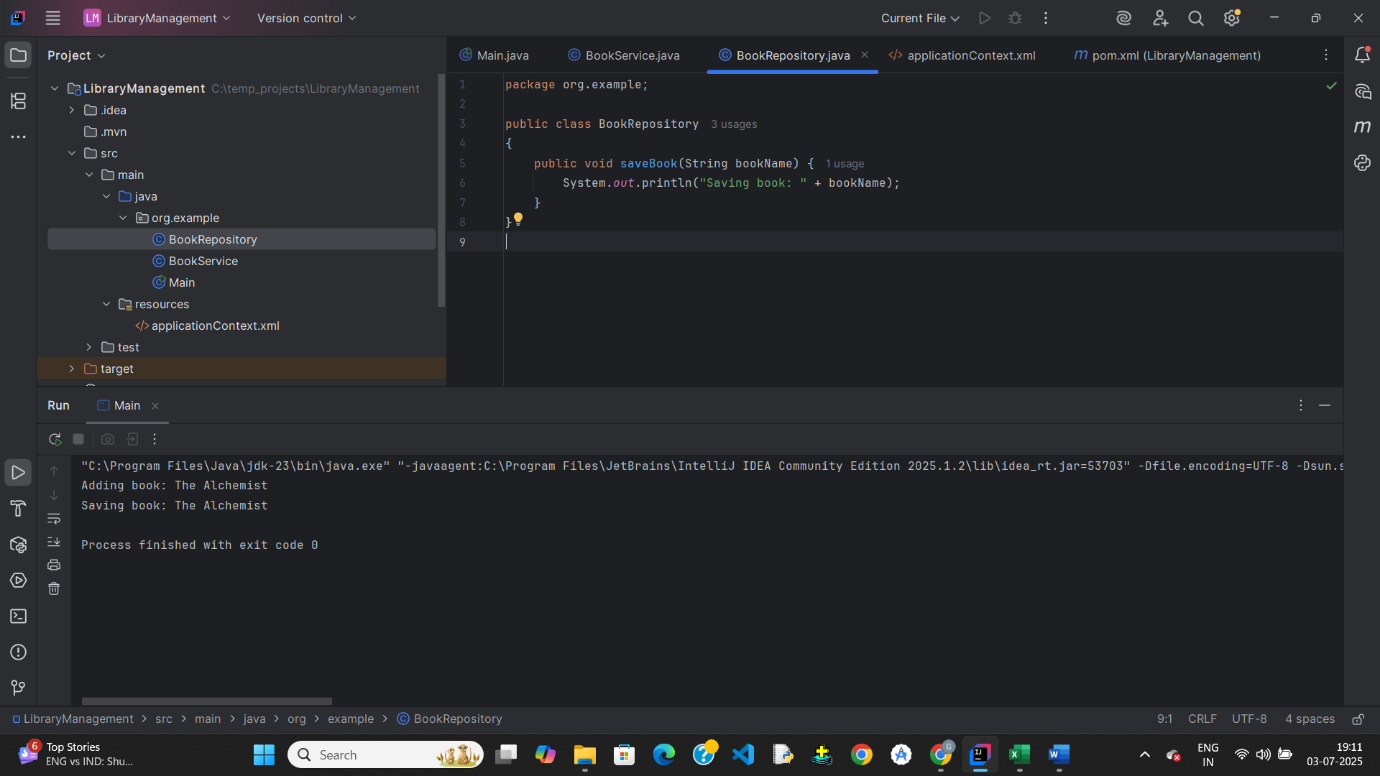
**BookService.java**

package org.example;  
  
public class BookService  
{  
 private BookRepository bookRepository;  
  
 // Setter for dependency injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Adding book: " + bookName);  
 bookRepository.saveBook(bookName);  
 }  
}

**BookRepository.xml**

package org.example;  
  
public class BookRepository  
{  
 public void saveBook(String bookName) {  
 System.*out*.println("Saving book: " + bookName);  
 }  
}

**Output:**

****

**Exercise 2: Implementing Dependency Injection**

**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 http://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
  
 <groupId>org.example</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>1.0-SNAPSHOT</version>  
  
 <properties>  
 <maven.compiler.source>23</maven.compiler.source>  
 <maven.compiler.target>23</maven.compiler.target>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
 <dependencies>  
 <!-- Spring Core -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.34</version>  
 </dependency>  
 </dependencies>  
  
</project>

**applicationContext.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">  
  
 <!-- Repository Bean -->  
 <bean id="bookRepository" class="org.example.BookRepository" />  
  
 <!-- Service Bean: constructor for bookRepository, setter for libraryName -->  
 <bean id="bookService" class="org.example.BookService">  
 <!-- Constructor injection -->  
 <constructor-arg ref="bookRepository"/>  
  
 <!-- Setter injection -->  
 <property name="libraryName" value="City Central Library"/>  
 </bean>  
  
</beans>

**Main.java**

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

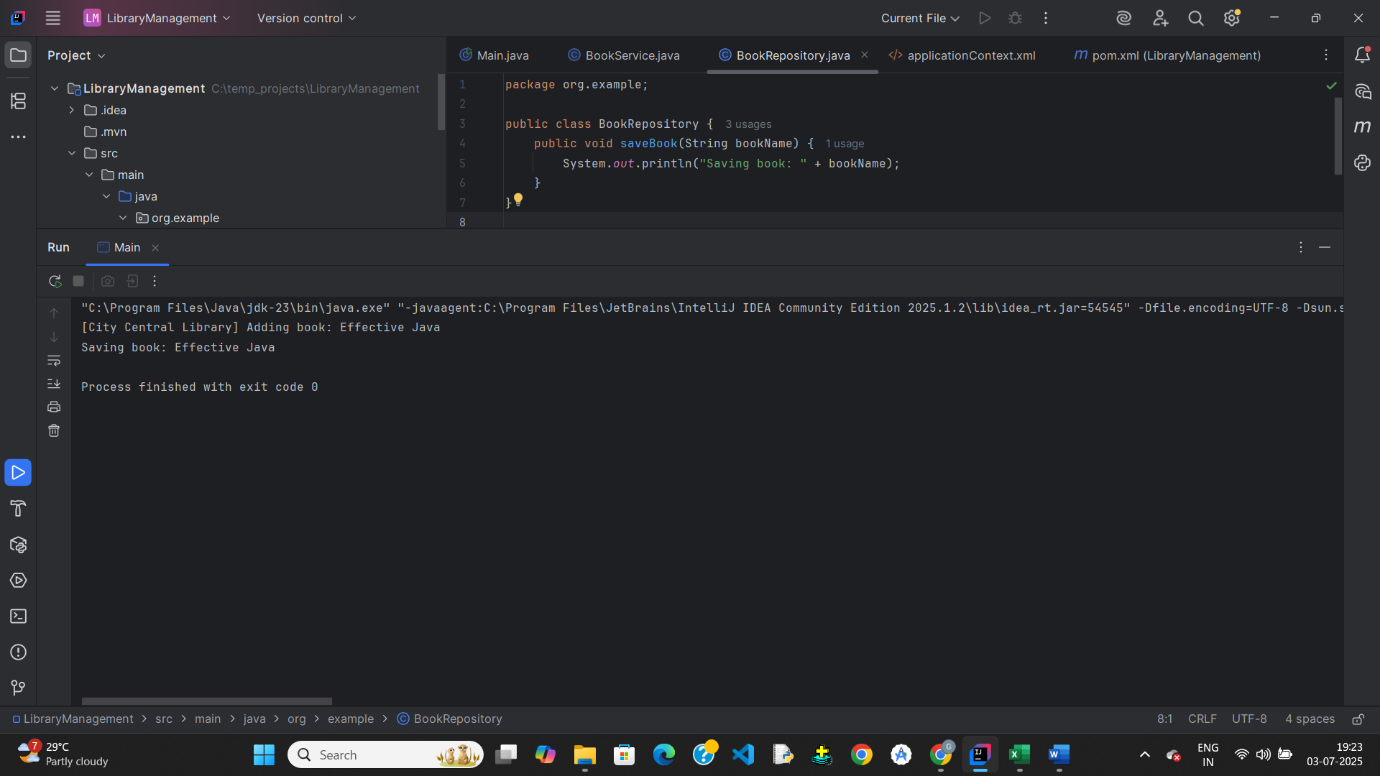
**BookService.java**

package org.example;  
  
public class BookService {  
 private BookRepository bookRepository; // injected via constructor  
 private String libraryName; // injected via setter  
  
 // Constructor-based injection  
 public BookService(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 // Setter-based injection  
 public void setLibraryName(String libraryName) {  
 this.libraryName = libraryName;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("[" + libraryName + "] Adding book: " + bookName);  
 bookRepository.saveBook(bookName);  
 }  
}

**BookRepository.java**

package org.example;  
  
public class BookRepository {  
 public void saveBook(String bookName) {  
 System.*out*.println("Saving book: " + bookName);  
 }  
}

**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

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>

<packaging>jar</packaging>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.34</version>

</dependency>

<!-- Step 3: Configure Maven Compiler Plugin -->

<build>

<plugins>

<plugin>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

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

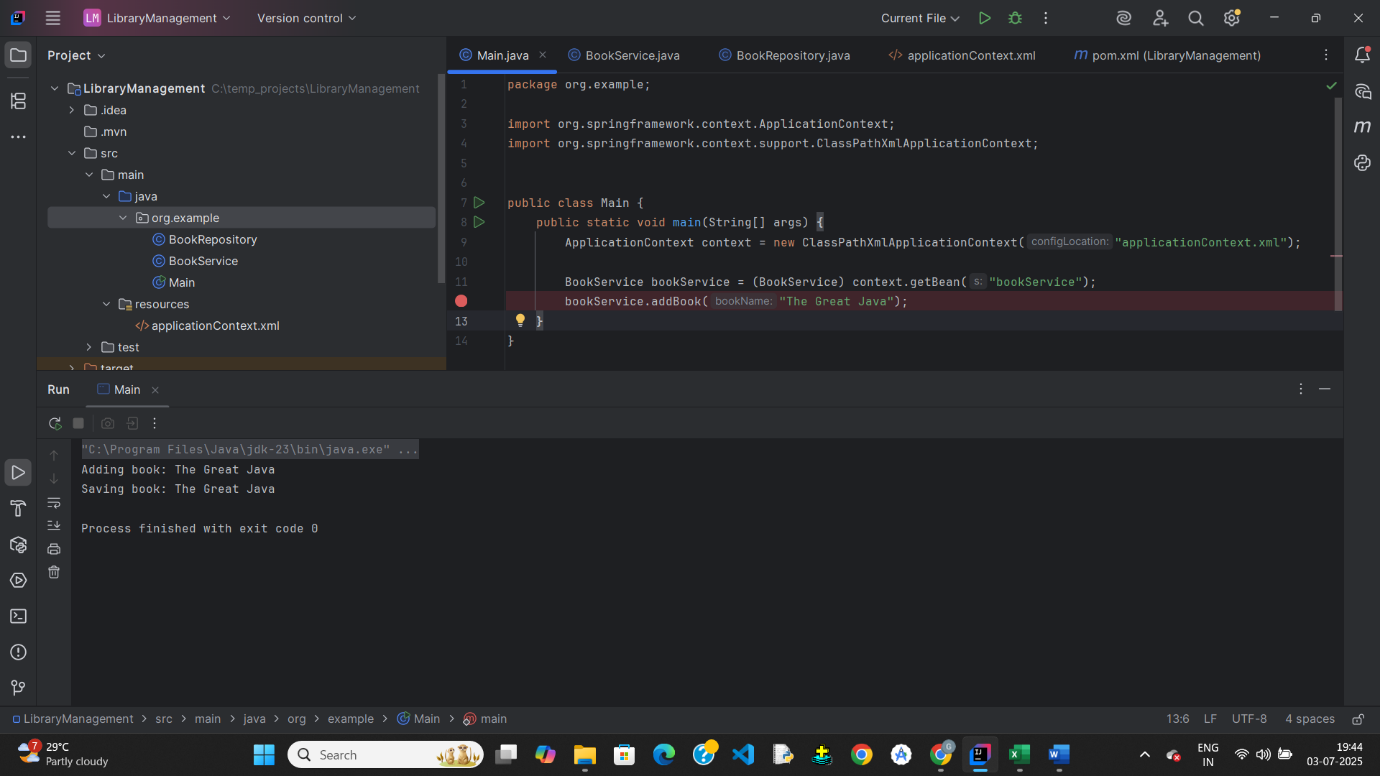
<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
 xsi:schemaLocation="http://www.springframework.org/schema/beans  
 http://www.springframework.org/schema/beans/spring-beans.xsd">  
  
 <!-- Define BookRepository Bean -->  
 <bean id="bookRepository" class="org.example.BookRepository" />  
  
 <!-- Define BookService Bean with setter injection -->  
 <bean id="bookService" class="org.example.BookService">  
 <property name="bookRepository" ref="bookRepository" />  
 </bean>  
  
</beans>

package org.example;  
  
public class BookRepository {  
 public void saveBook(String bookName) {  
 System.*out*.println("Saving book: " + bookName);  
 }  
}

package org.example;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter method for Spring DI  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Adding book: " + bookName);  
 bookRepository.saveBook(bookName);  
 }  
  
}

package org.example;  
  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
  
public class Main {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.addBook("The Great Java");  
 }  
}

**Output:**



# Week-3

# Spring Data JPA with Spring Boot, Hibernate

**Spring Data JPA - Quick Example**

**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**

spring.application.name=orm-learn  
# Logging  
logging.level.org.springframework=info  
logging.level.com.cognizant=debug  
logging.level.org.hibernate.SQL=trace  
logging.level.org.hibernate.type.descriptor.sql=trace  
  
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  
  
# DB Configuration  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
spring.datasource.url=jdbc:mysql://localhost:3306/cognizant  
spring.datasource.username=root  
spring.datasource.password=root   
  
# Hibernate  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

**main.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
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 java.util.List;  
  
@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);  
 *LOGGER*.info("Inside main");  
  
 *countryService* = context.getBean(CountryService.class);  
  
 *testGetAllCountries*();  
 }  
  
 private static void testGetAllCountries() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.getAllCountries();  
 *LOGGER*.info("countries={}", countries);  
 *LOGGER*.info("End");  
 }  
}

**Country.java**

package com.cognizant.orm\_learn.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 = "co\_code")  
 private String code;  
  
 @Column(name = "co\_name")  
 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 "Country [code=" + code + ", name=" + name + "]";  
 }  
}

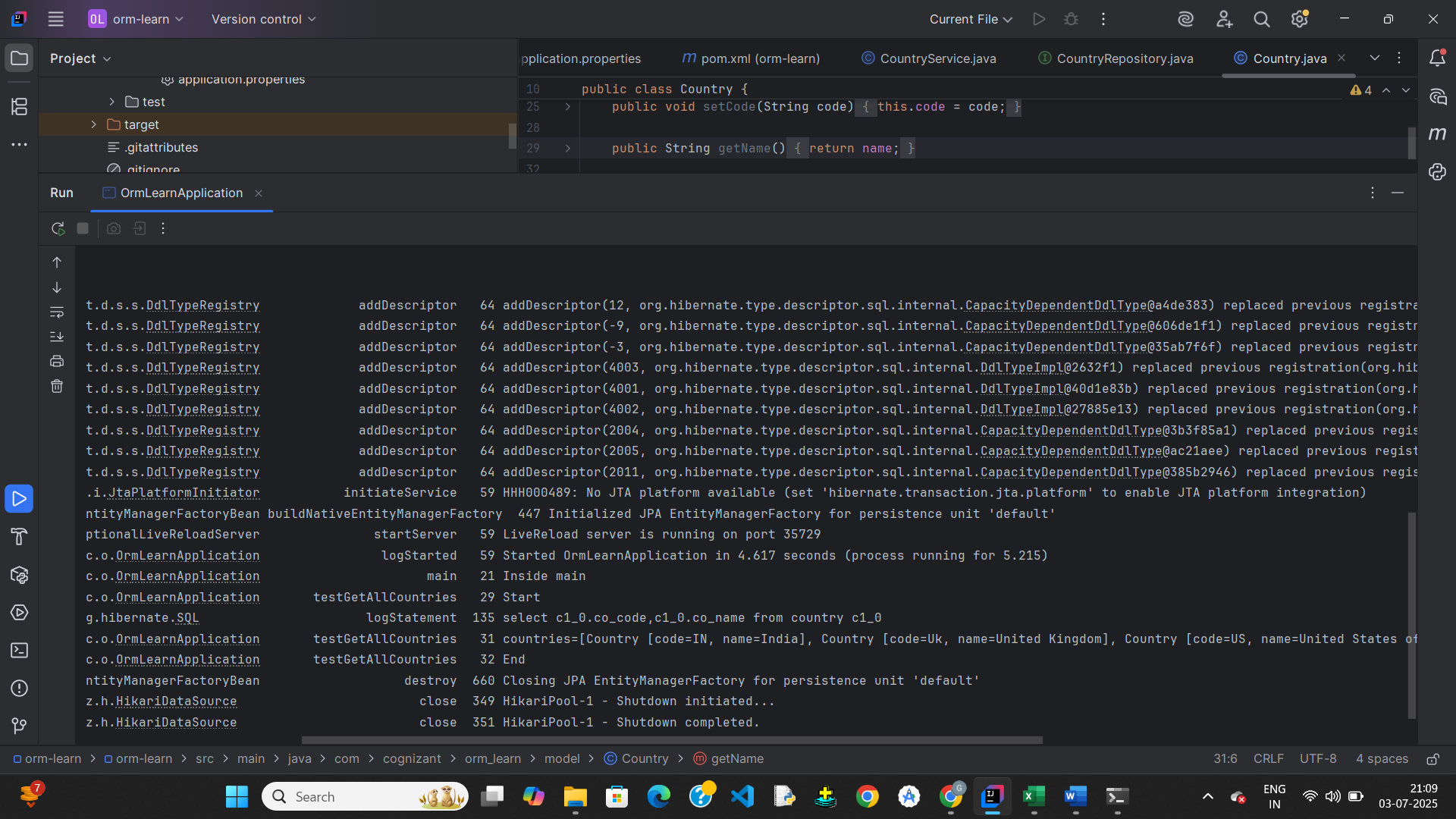
**CountryRepository.java**

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;  
  
@Repository  
public interface CountryRepository extends JpaRepository<Country, String>  
{  
  
}

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

**Output:**

****

**Explain the difference between Java Persistence API, Hibernate and Spring Data JPA**

1. **JPA (Java Persistence API) — *Specification / Interface***

***What it is:*** A Java specification for ORM (Object-Relational Mapping).

It defines interfaces and annotations for mapping Java objects to database tables.

***Who provides it:*** Part of Jakarta EE (previously Java EE). It’s not an implementation.

***What it defines:***

* Entity mappings using annotations like @Entity, @Id, @OneToMany, etc.
* Entity Manager API (EntityManager, Query, PersistenceContext, etc.)
* JPQL (Java Persistence Query Language)

***Need:*** Requires an implementation to actually work.

**Example:**

@Entity

public class Country {

@Id

private String code;

private String name;

}

**2. Hibernate — *Implementation of JPA (and more)***

* ***What it is****:* A **popular ORM framework** that **implements JPA**.
* ***Extra features****:* Provides many advanced features **beyond JPA**, like:
  + Caching
  + Batch processing
  + Custom SQL
* ***Usage****:* You can use Hibernate through the JPA interfaces (standard way) or directly with its native APIs (non-standard way).

***Hibernate extends JPA like this:***

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

**3.Spring Data JPA — *Abstraction / Simplification***

* ***What it is***: A **Spring project** that builds on **JPA (and uses Hibernate underneath)**.
* ***Purpose****:* It makes working with JPA easier by **removing boilerplate code**.
* ***Key feature****:* You define just interfaces (like CountryRepository) and Spring generates implementations for common operations.

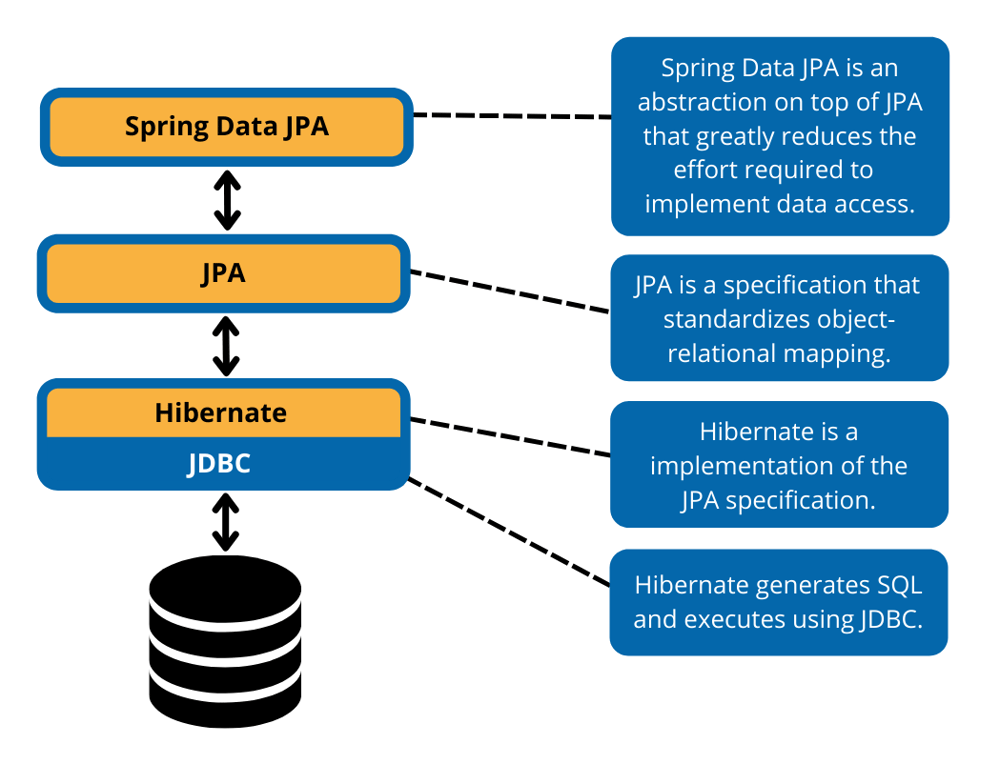
Example:

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String keyword);

}

* **No need to write SQL or EntityManager code** for basic queries — Spring handles it.
* Uses **Spring Boot**, **Spring Context**, and **Spring Transaction Management** for full integration.



**Find a country based on country code**

**Main.java**

package com.cognizant.orm\_learn;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
  
@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);  
 *LOGGER*.info("Inside main");  
  
 *countryService* = context.getBean(CountryService.class);  
  
 *testFindCountryByCode*();  
 }  
  
 private static void testFindCountryByCode() {  
 *LOGGER*.info("Start");  
 try {  
 Country country = *countryService*.findCountryByCode("IN");  
 *LOGGER*.debug("Country: {}", country);  
 } catch (CountryNotFoundException e) {  
 *LOGGER*.error("Exception: {}", e.getMessage());  
 }  
 *LOGGER*.info("End");  
 }  
}

**Country.java**

package com.cognizant.orm\_learn.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 = "co\_code")  
 private String code;  
  
 @Column(name = "co\_name")  
 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 "Country [code=" + code + ", name=" + name + "]";  
 }  
}

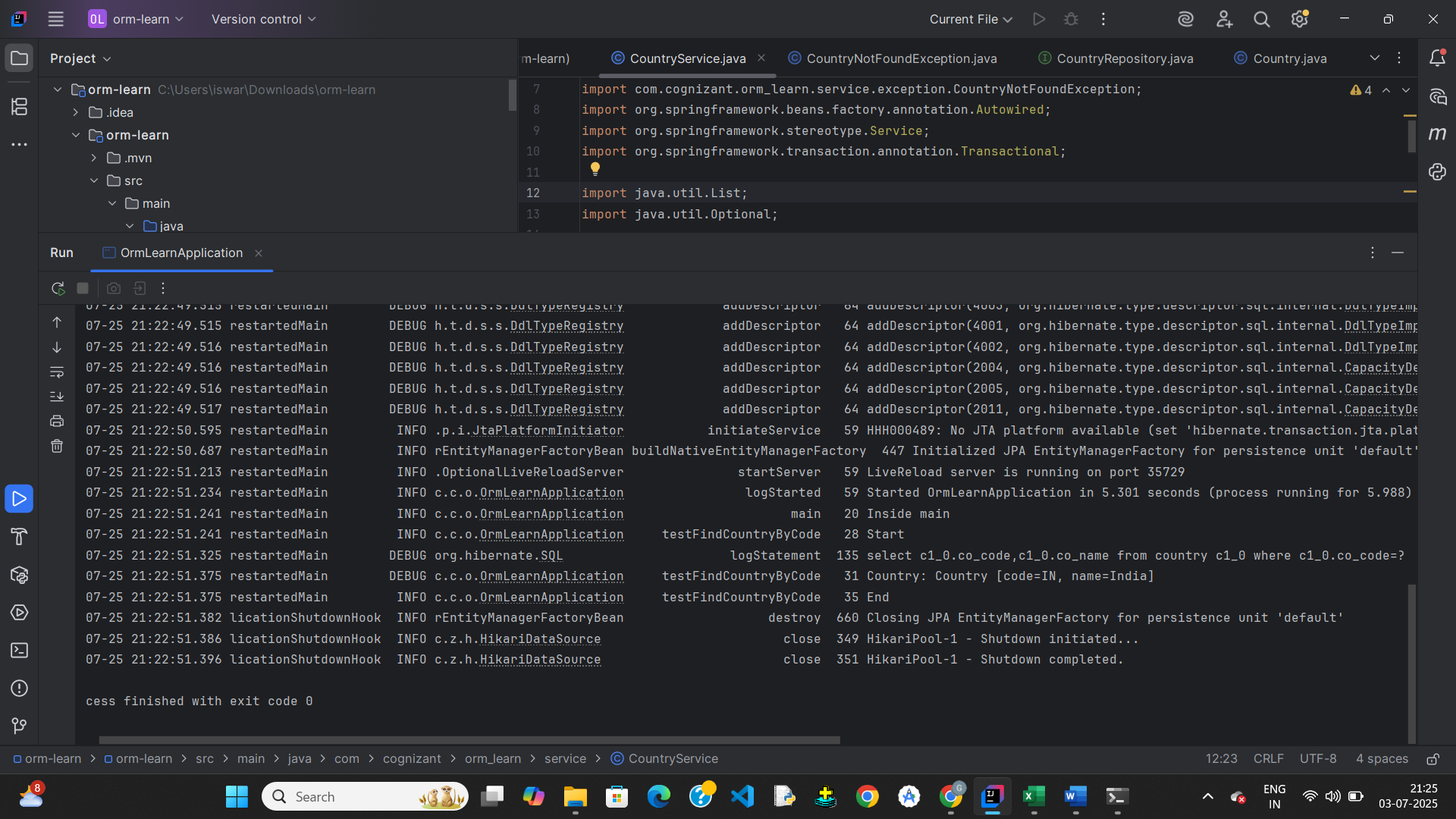
**CountryNotFoundException.java**

package com.cognizant.orm\_learn.service.exception;  
  
public class CountryNotFoundException extends Exception  
{  
 public CountryNotFoundException(String message) {  
 super(message);  
 }  
}

**CountryService.java**

package com.cognizant.orm\_learn.service;  
  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
import com.cognizant.orm\_learn.service.exception.CountryNotFoundException;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;  
  
import java.util.List;  
import java.util.Optional;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
  
 public List<Country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
 @Transactional  
 public Country findCountryByCode(String countryCode) throws CountryNotFoundException {  
 Optional<Country> result = countryRepository.findById(countryCode);  
 if (!result.isPresent()) {  
 throw new CountryNotFoundException("Country not found with code: " + countryCode);  
 }  
 return result.get();  
 }  
}

**Output:**

****