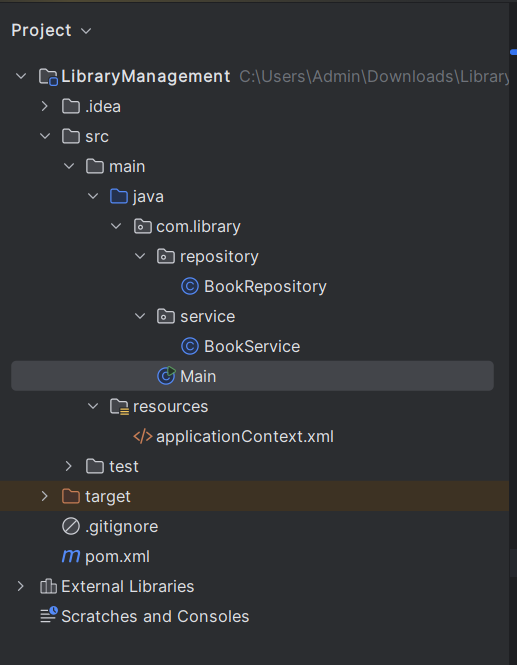
**Spring Core and Maven**

**## *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>com.library</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>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 </dependencies>  
  
</project>

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

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String bookName) {  
 System.*out*.println("Book saved to repository: " + bookName);  
 }  
}

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

package com.library.service;

import com.library.repository.BookRepository;

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);  
 }  
}

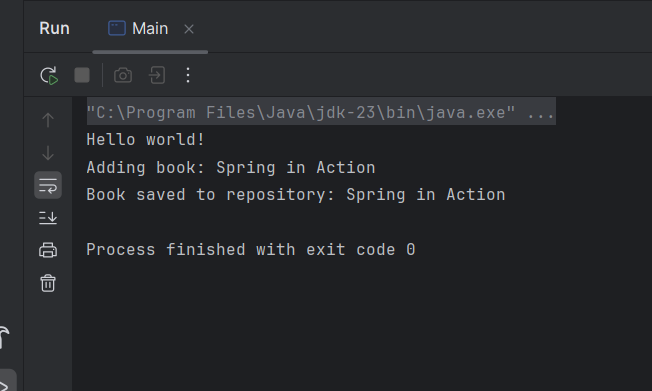
**# resources.applicationContext.xml**

<?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 definitions here -->  
  
 <!-- repository bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository"/>  
  
 <!-- service bean with dependency injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

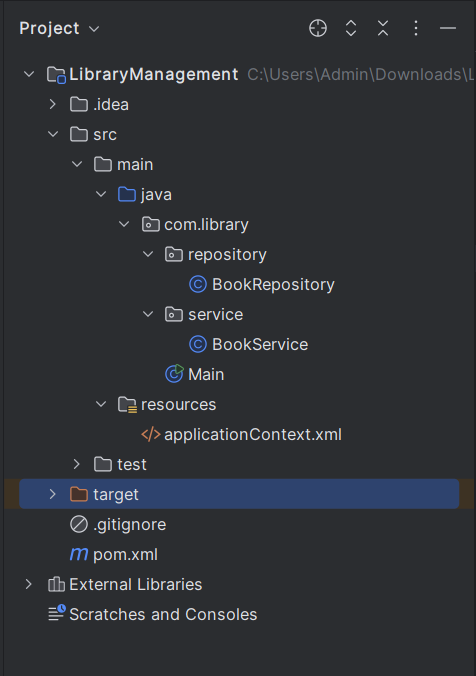
**# com.library.Main.java**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Hello world!");  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
 bookService.addBook("Spring in Action");  
 }  
}

**# 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>com.library</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>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 </dependencies>  
  
</project>

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

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String bookName) {  
 System.*out*.println("Book saved to repository: " + bookName);  
 }  
}

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

package com.library.service;

import com.library.repository.BookRepository;

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);  
 }  
}

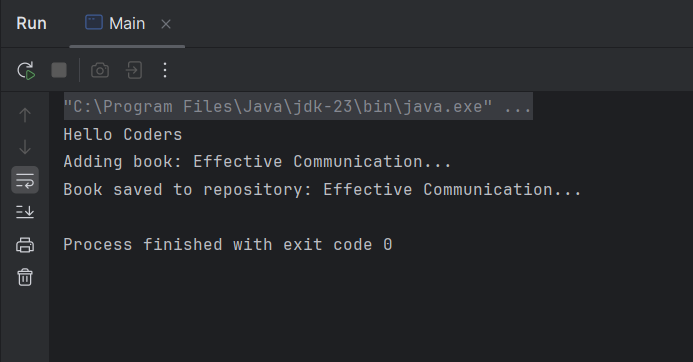
**# resources.applicationContext.xml**

<?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 definitions here -->  
  
 <!-- repository bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository"/>  
  
 <!-- service bean with dependency injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

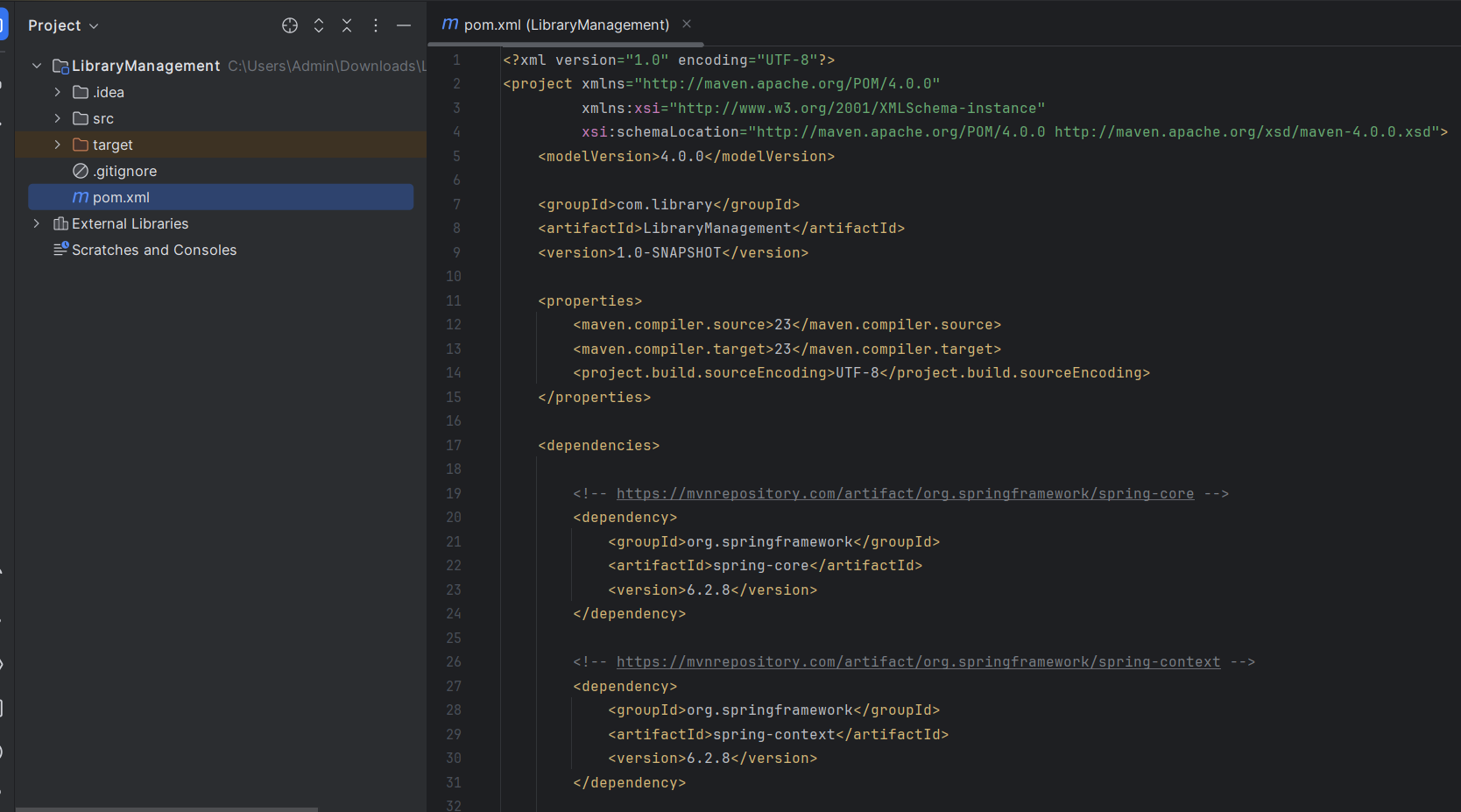
**# com.library.Main.java**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("Hello world!");  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
 bookService.addBook("Effective Communication");  
 }  
}

**# Output**



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

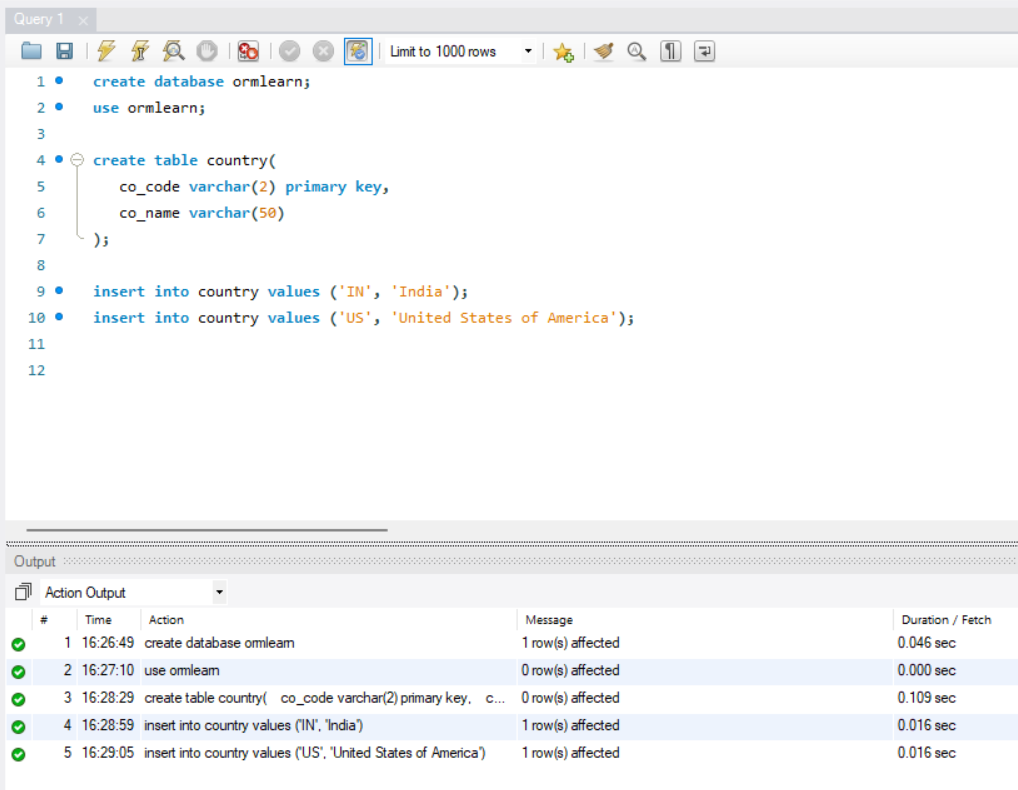
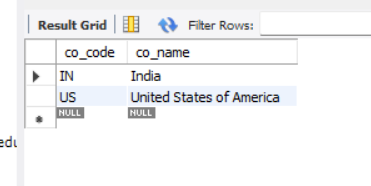
****

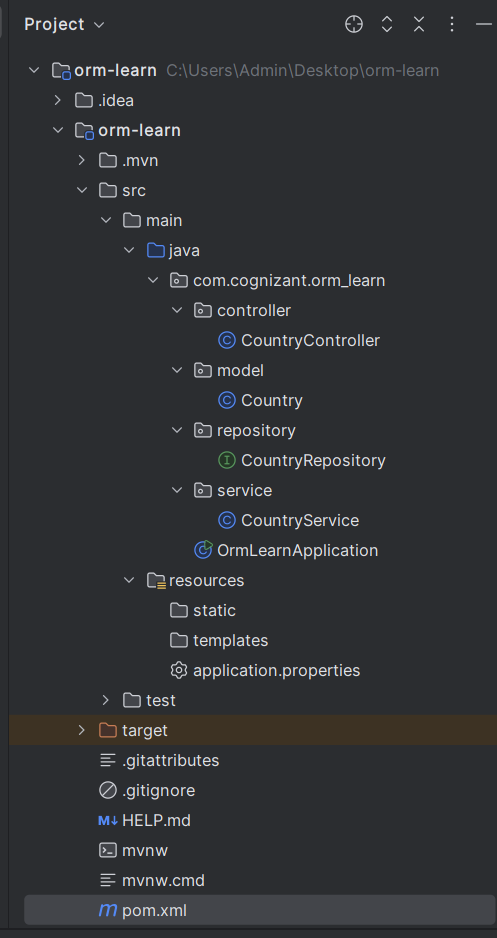
**# pom.xml**

<?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>com.library</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>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-core -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-core</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api -->  
 <dependency>  
 <groupId>javax.servlet</groupId>  
 <artifactId>javax.servlet-api</artifactId>  
 <version>4.0.1</version>  
 <scope>provided</scope>  
 </dependency>  
  
 </dependencies>  
  
 <!-- o Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file. -->  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.0</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**Spring Data JPA with Hibernate**

**## Hands on 1….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>24</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-starter-web</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>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>6.2.8</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
  
</project>

**# com.cognizant.orm\_learn.model.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 + "]";  
 }  
}

**# com.cognizant.orm\_learn.repository.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> {  
}

**# com.cognizant.orm\_learn.service.CountryService.java**

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

**# com.cognizant.orm\_learn.controller.CountryController.java**

package com.cognizant.orm\_learn.controller;  
  
import com.cognizant.orm\_learn.model.Country;  
import com.cognizant.orm\_learn.service.CountryService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.util.List;  
  
@RestController  
public class CountryController {  
  
 @Autowired  
 private CountryService countryService;  
  
 @GetMapping("/")  
 public List<Country> getCountries(){  
 return countryService.getAllCountries();  
 }  
}

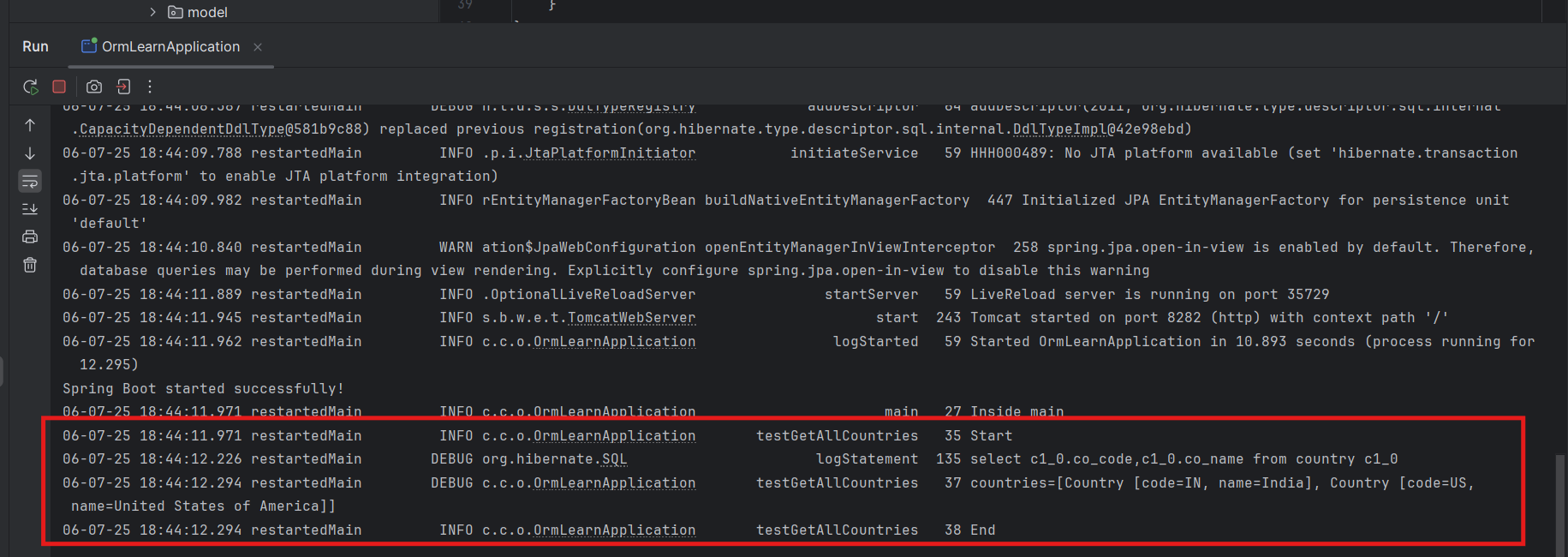
**# application.properties**

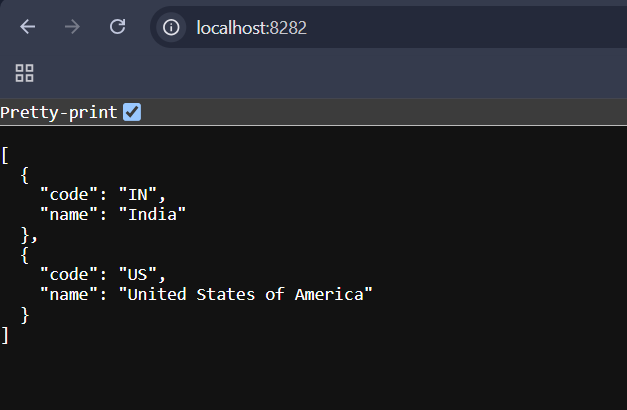
spring.application.name=orm-learn  
server.port = 8282  
  
  
# Spring Framework and application log  
logging.level.org.springframework=info  
logging.level.com.cognizant=debug

# Hibernate logs for displaying executed SQL, input and output  
logging.level.org.hibernate.SQL=trace  
logging.level.org.hibernate.type.descriptor.sql=trace  
  
# Log pattern  
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n  
  
# Database configuration  
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=\*\*\*\*\*\*\*\*\*\*   
  
# Hibernate configuration  
spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.database-platform=org.hibernate.dialect.MySQLDialect

**# com.cognizant.orm\_learn.OrmLearnApplication.java**

package com.cognizant.orm\_learn;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
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.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);  
  
 System.*out*.println("Spring Boot started successfully!");  
 *LOGGER*.info("Inside main");  
  
 *countryService* = context.getBean(CountryService.class);  
  
 *testGetAllCountries*();  
 }  
  
 private static void testGetAllCountries() {  
 *LOGGER*.info("Start");  
 List<Country> countries = *countryService*.getAllCountries();  
 *LOGGER*.debug("countries={}", countries);  
 *LOGGER*.info("End");  
 }  
}

**# Output**

****

**## Hands on 4……Difference between JPA, Hibernate and Spring Data JPA**

| **Feature** | **JPA (Java**  **Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| Type | Specification (interface) | Implementation | Abstraction over JPA |
| Definition | A standard (JSR 338) for ORM in Java | ORM framework that implements JPA | Spring module that simplifies JPA-based data access |
| Provided By | Java EE (Jakarta EE) | Red Hat | Spring Framework |
| Implements JPA? | No (just specification) | Yes | No (depends on implementation like Hibernate) |
| Boilerplate Code | Requires manual implementation | Still requires session/transaction code | Minimizes with Repository interfaces |
| Transactions | Manually handled or via container | Manually handled | Handled automatically using @Transactional |
| Queries | JPQL (Java Persistence Query Language) | HQL (Hibernate Query Language) + JPQL | Derived Queries, JPQL, Native SQL supported |
| Ease of Use | Intermediate | More control, but verbose | Easiest – developer-friendly |
| Common Use Case | Abstract definition for persistence | Full-featured ORM tool | Fast application development using Spring |

Hibernate

---------

-> A framework for persisting/saving java object in a database.

My Application <----> Hibernate <----> Database

(object) ---------------> (object)

-> Hibernate handles all of the low level sql.

-> Minimize the amount of JDBC code .

-> Hibernate provides the Object Relational Mapping(ORM)

Object Relational Mapping

-> The developer defines mapping b/w java class and database table

Java class Database

----------- -------------------

class Student Table: Student

{ column: id INT

int id; column: name VARCHAR

String name;

}

------------------------------------------------------------

JPA

---

-> Jakarta Persistence API

-> It is a standard API for ORM.

-> It consist only specification

|- define a set of interfaces.

|- It requires implementation.

-> By having this API, we are not locked to vendor/framework implementation.

@Entity: Mark class as a JP entity, mapped to the database table.

@Table: Map the entity toa specific table name

Methods of JpaRepository

findAll(), save(), findById() etc.

@Transactional (Method/Class) : Runs method within a database transaction