Spring Core_Maven

Exercise 1: Configuring a Basic Spring Application

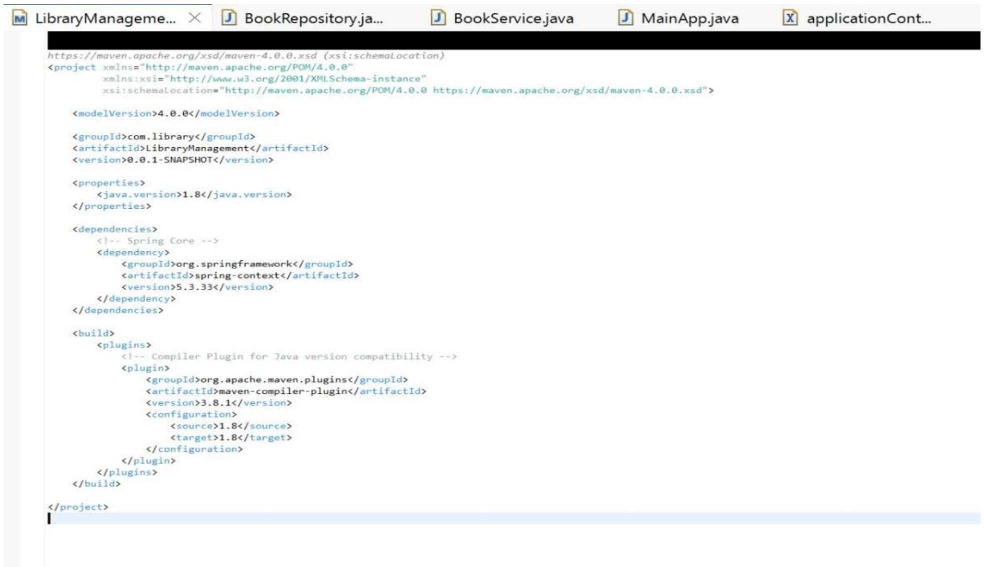
Scenario:

Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.

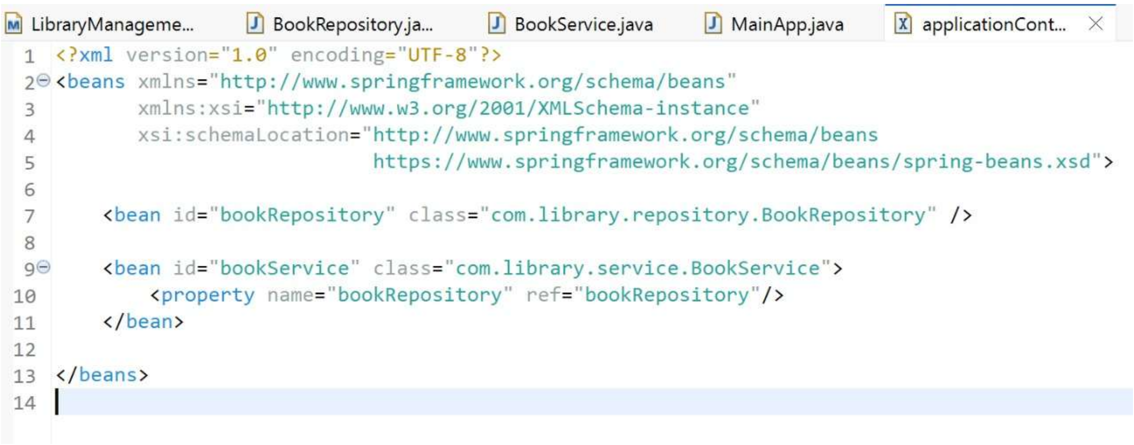
Steps:

1. **Set Up a Spring Project:**
   - Create a Maven project named **LibraryManagement**.
   - Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   - Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   - Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   - Create a package **com.library.service** and add a class **BookService**.
   - Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**
   - Create a main class to load the Spring context and test the configuration.

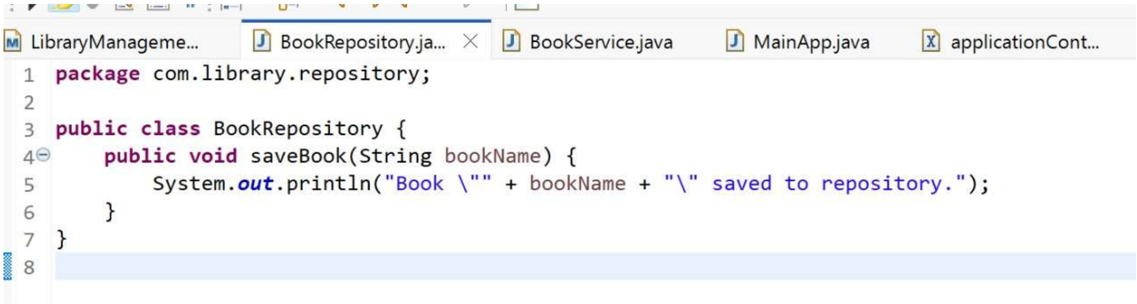
CODES:



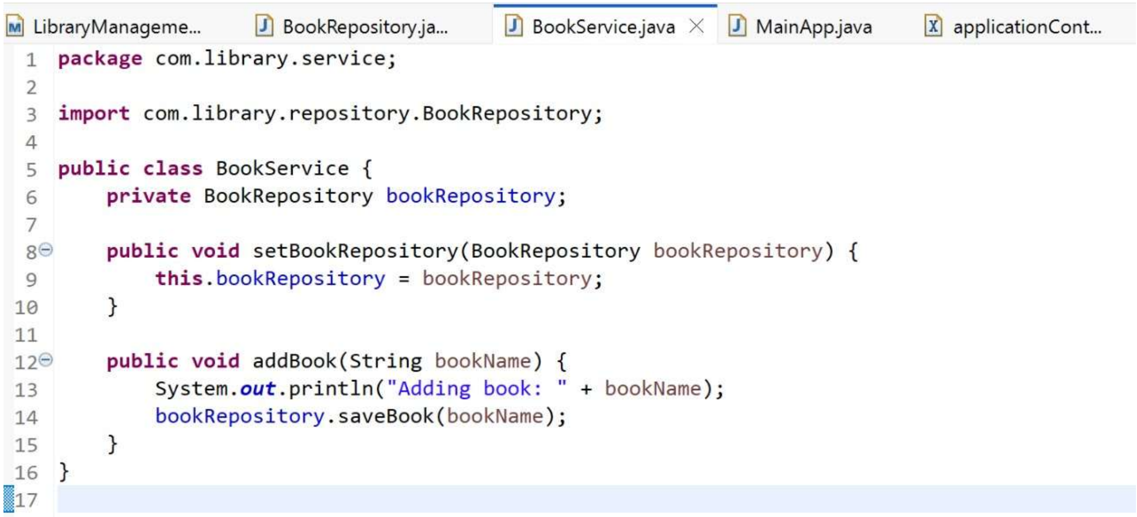
```
[redacted]
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <!-- Spring Core -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.33</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- Compiler Plugin for Java version compatibility -->
            <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>

</project>
```



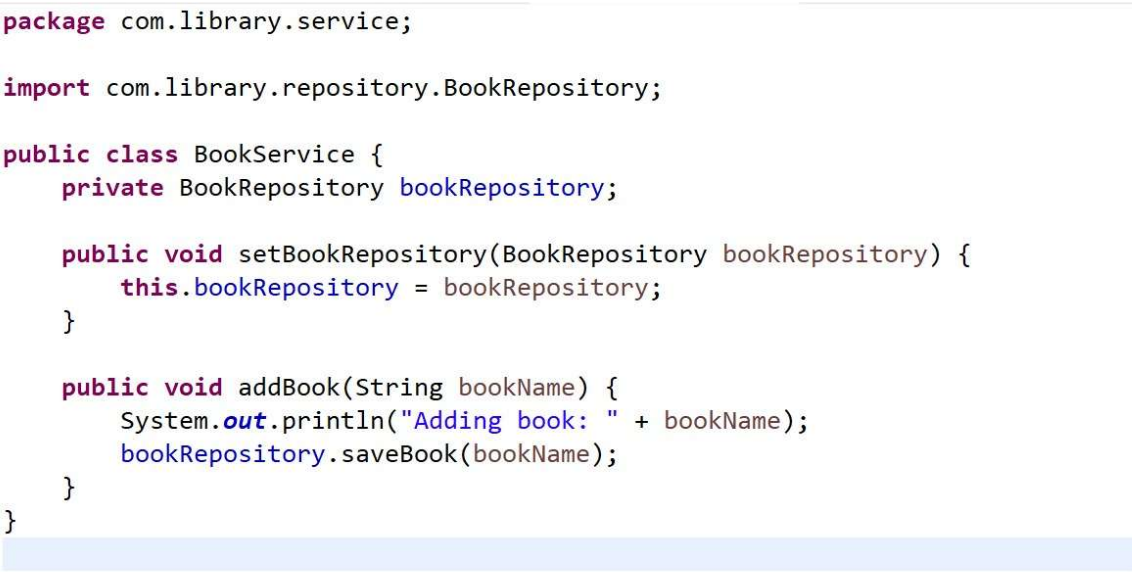
```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           https://www.springframework.org/schema/beans/spring-beans.xsd">

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

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

</beans>
```



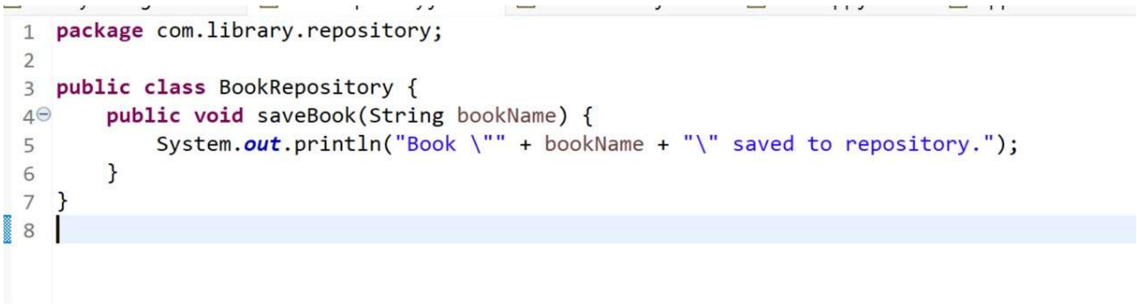
```java
package com.library.repository;

public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Book \"" + bookName + "\" saved to repository.");
    }
}
```



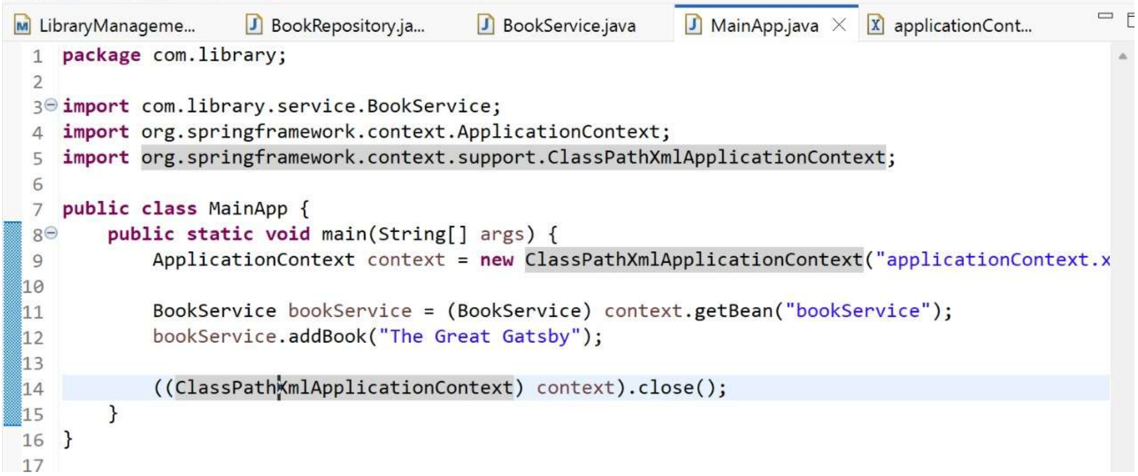
```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

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

    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```



```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(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

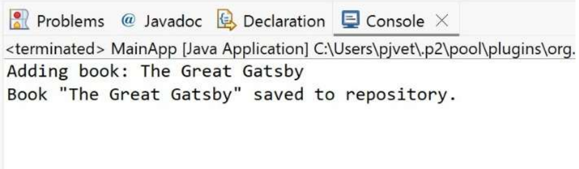


```java
package com.library.repository;

public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Book \"" + bookName + "\" saved to repository.");
    }
}

```



```java
package com.library;

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

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.x

        BookService bookService = (BookService) context.getBean("bookService");
        bookService.addBook("The Great Gatsby");

        ((ClassPathXmlApplicationContext) context).close();
    }
}

```

OUTPUT:



```
<terminated> MainApp [Java Application] C:\Users\pjvet\.p2\pool\plugins\org.
Adding book: The Great Gatsby
Book "The Great Gatsby" saved to repository.
```

Exercise 2: Implementing Dependency Injection

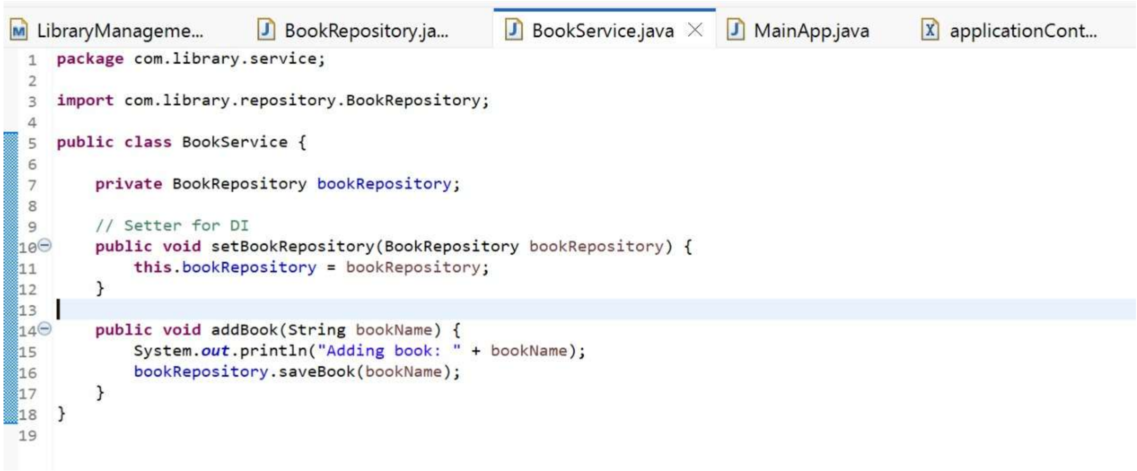
Scenario:

In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

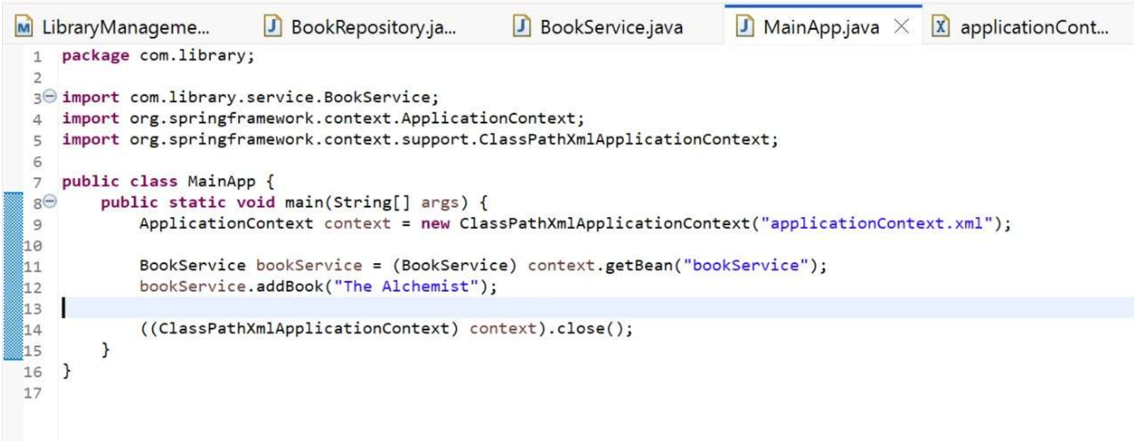
Steps:

1. **Modify the XML Configuration:**
   - **Update applicationContext.xml to wire BookRepository into BookService.**
2. **Update the BookService Class:**
   - **Ensure that BookService class has a setter method for BookRepository.**
3. **Test the Configuration:**
   - **Run the LibraryManagementApplication main class to verify the dependency injection.**

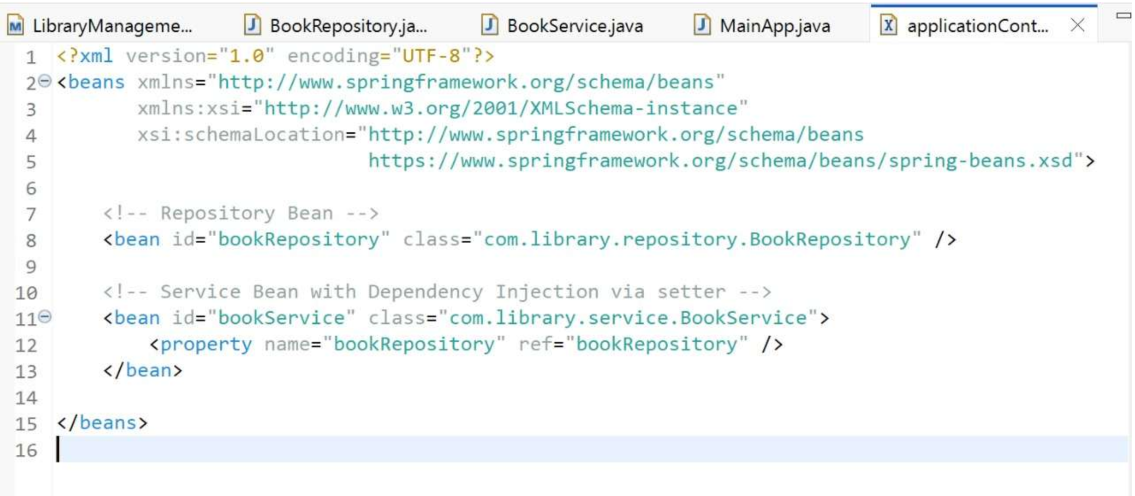
CODES:



```java
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for DI
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void addBook(String bookName) {
        System.out.println("Adding book: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

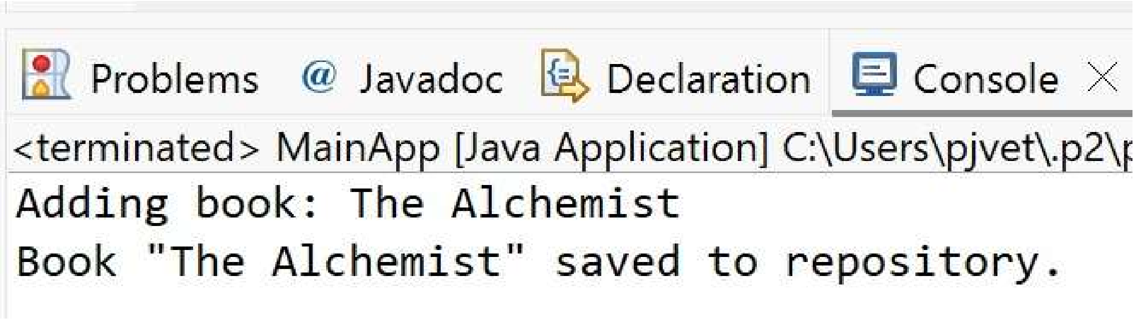


```java
package com.library;

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

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

        BookService bookService = (BookService) context.getBean("bookService");
        bookService.addBook("The Alchemist");

        ((ClassPathXmlApplicationContext) context).close();
    }
}

```



```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">

    <!-- Repository Bean -->
    <bean id="bookRepository" class="com.library.repository.BookRepository" />

    <!-- Service Bean with Dependency Injection via setter -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository" />
    </bean>

</beans>

```

OUTPUT:



```
<terminated> MainApp [Java Application] C:\Users\pjvet\.p2\p
Adding book: The Alchemist
Book "The Alchemist" saved to repository.
```

Exercise 4: Creating and Configuring a Maven Project

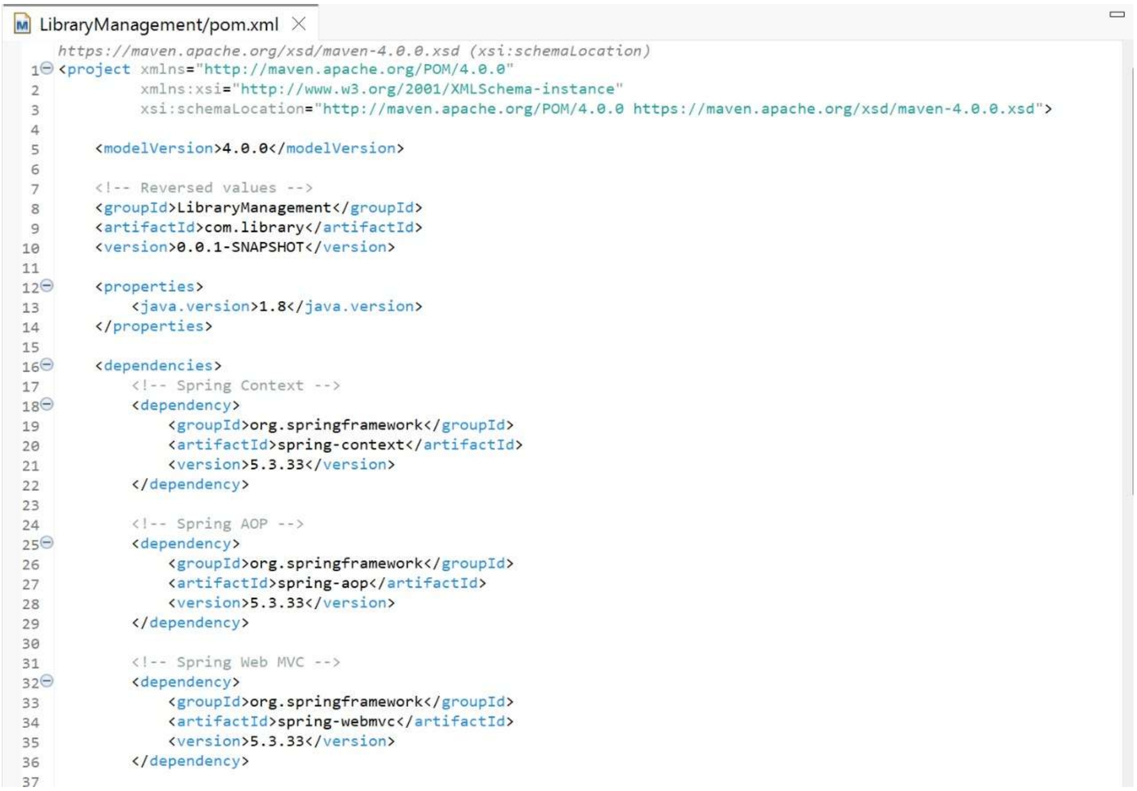
Scenario:

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

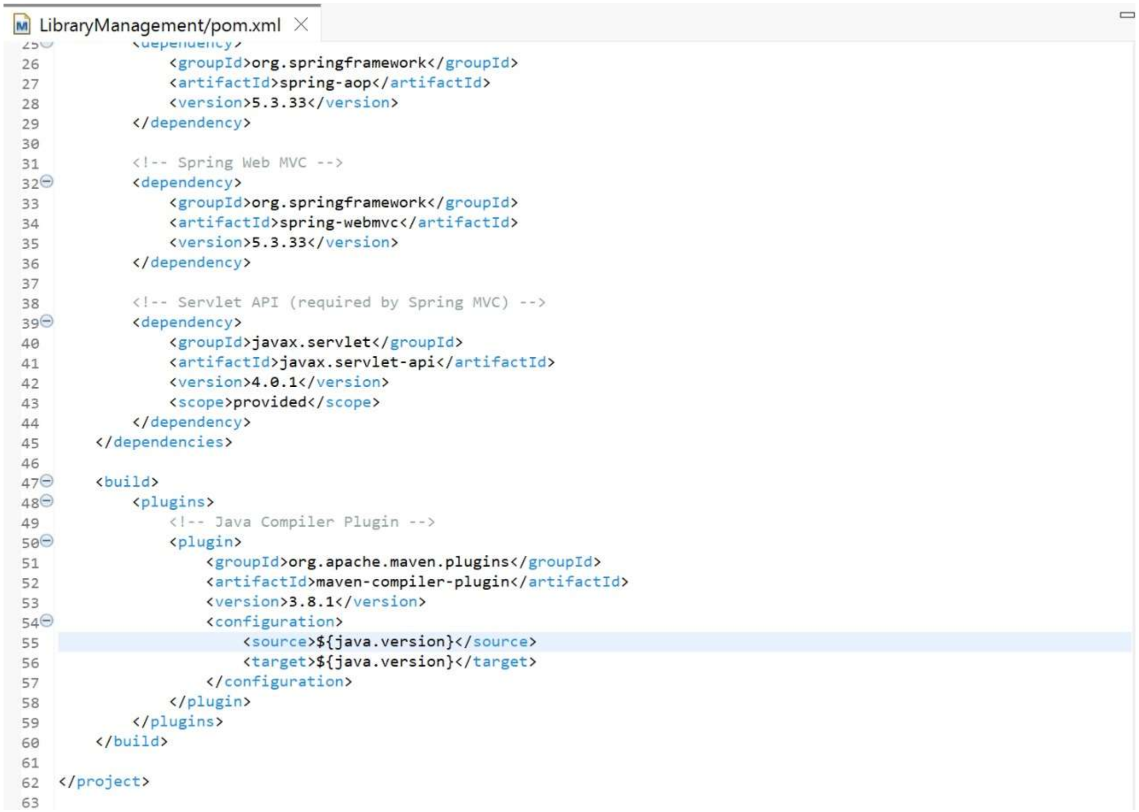
Steps:

1. **Create a New Maven Project:**
   - Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
   - Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   - Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

CODES:

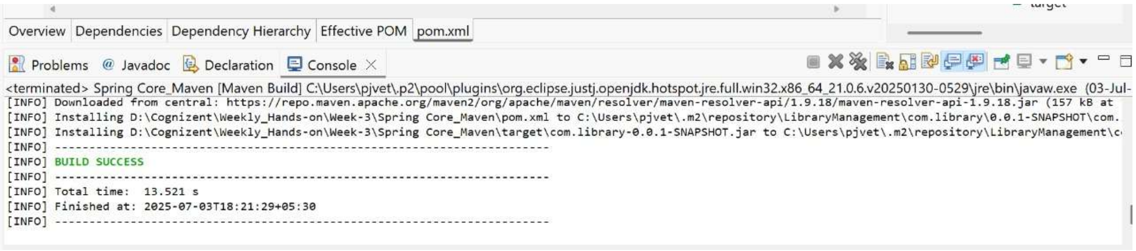


```
https://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">

    <modelVersion>4.0.0</modelVersion>

    <!-- Reversed values -->
    <groupId>LibraryManagement</groupId>
    <artifactId>com.library</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <properties>
        <java.version>1.8</java.version>
    </properties>

    <dependencies>
        <!-- Spring Context -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.33</version>
        </dependency>

        <!-- Spring AOP -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.3.33</version>
        </dependency>

        <!-- Spring Web MVC -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.3.33</version>
        </dependency>

```



```xml
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.3.33</version>
        </dependency>

        <!-- Spring Web MVC -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.3.33</version>
        </dependency>

        <!-- Servlet API (required by Spring MVC) -->
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- Java 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>
        </plugins>
    </build>

</project>
```

OUTPUT:

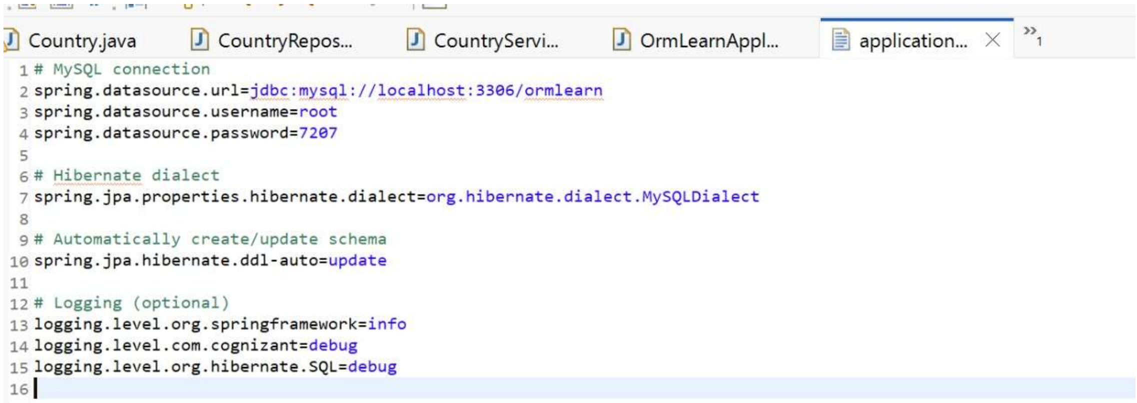


```
<terminated> Spring Core_Maven [Maven Build] C:\Users\pjvet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe  (03-Jul-
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/resolver/maven-resolver-api/1.9.18/maven-resolver-api-1.9.18.jar (157 kB at
[INFO] Installing D:\Cognizent\Weekly_Hands-on\Week-3\Spring Core_Maven\pom.xml to C:\Users\pjvet\.m2\repository\LibraryManagement\com.library\0.0.1-SNAPSHOT\com.
[INFO] Installing D:\Cognizent\Weekly_Hands-on\Week-3\Spring Core_Maven\target\com.library-0.0.1-SNAPSHOT.jar to C:\Users\pjvet\.m2\repository\LibraryManagement\c
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  13.521 s
[INFO] Finished at: 2025-07-03T18:21:29+05:30
[INFO] ------------------------------------------------------------------------
```

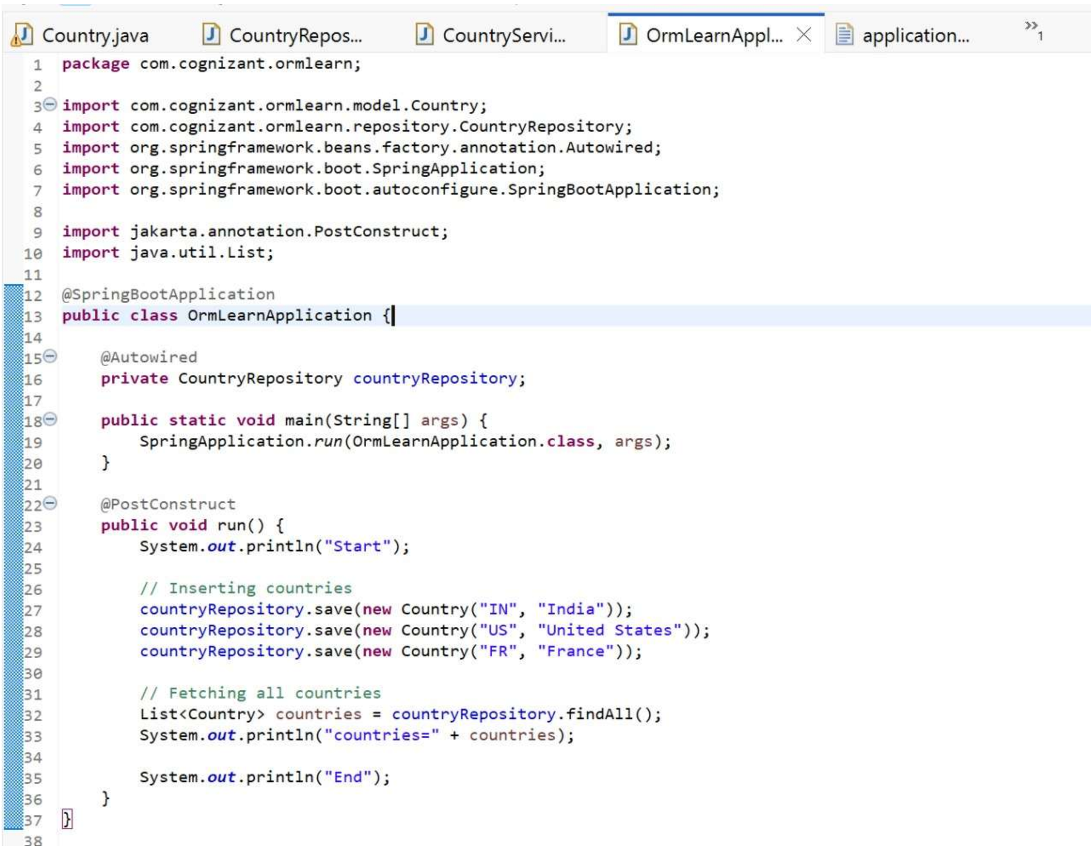
Spring-data-jpa-handson

Spring Data JPA

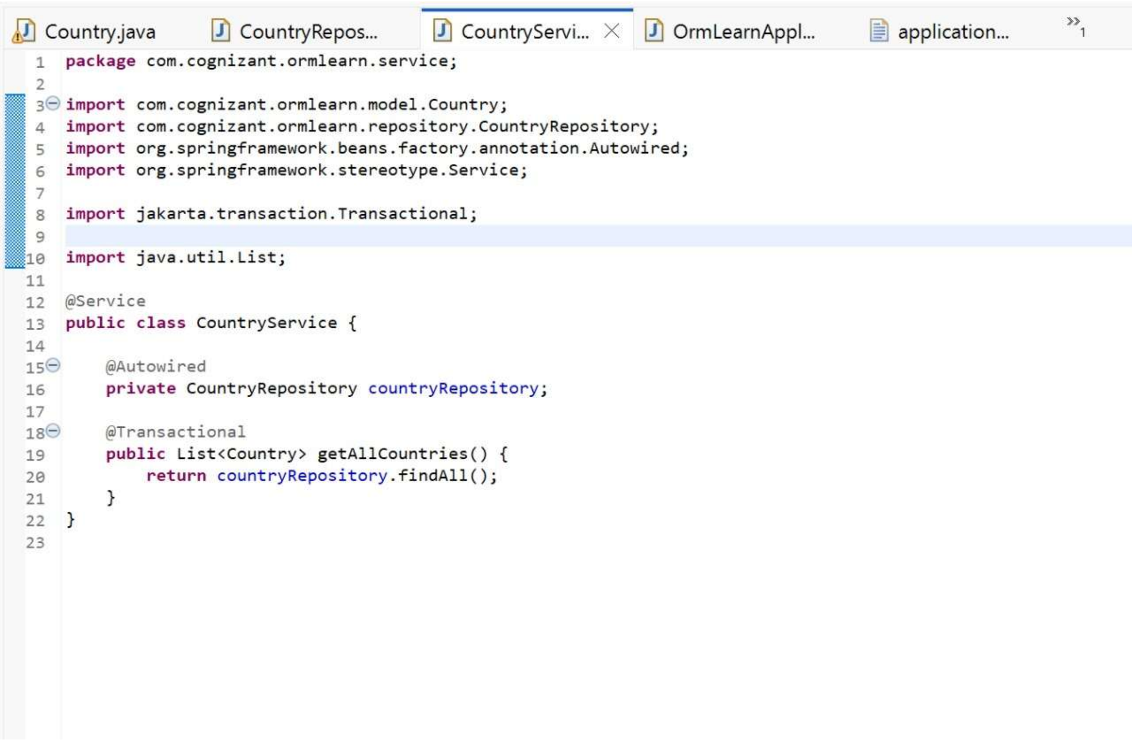
CODES:



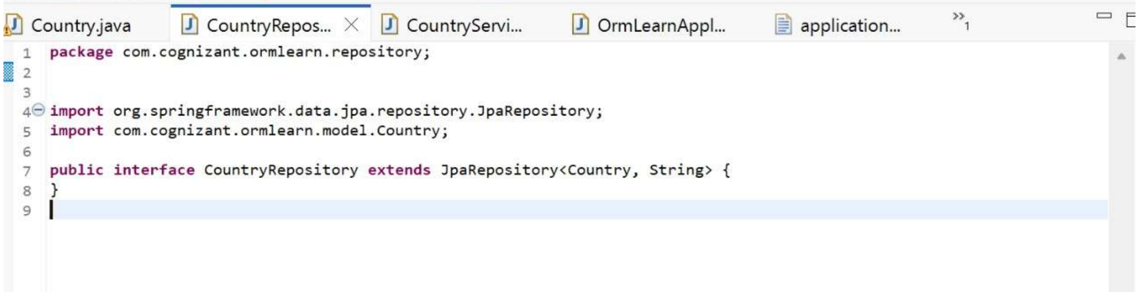
```
# MySQL connection
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=7207

# Hibernate dialect
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

# Automatically create/update schema
spring.jpa.hibernate.ddl-auto=update

# Logging (optional)
logging.level.org.springframework=info
logging.level.com.cognizant=debug
logging.level.org.hibernate.SQL=debug

```



```java
package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

import jakarta.annotation.PostConstruct;
import java.util.List;

@SpringBootApplication
public class OrmLearnApplication {

    @Autowired
    private CountryRepository countryRepository;

    public static void main(String[] args) {
        SpringApplication.run(OrmLearnApplication.class, args);
    }

    @PostConstruct
    public void run() {
        System.out.println("Start");

        // Inserting countries
        countryRepository.save(new Country("IN", "India"));
        countryRepository.save(new Country("US", "United States"));
        countryRepository.save(new Country("FR", "France"));

        // Fetching all countries
        List<Country> countries = countryRepository.findAll();
        System.out.println("countries=" + countries);

        System.out.println("End");
    }
}

```



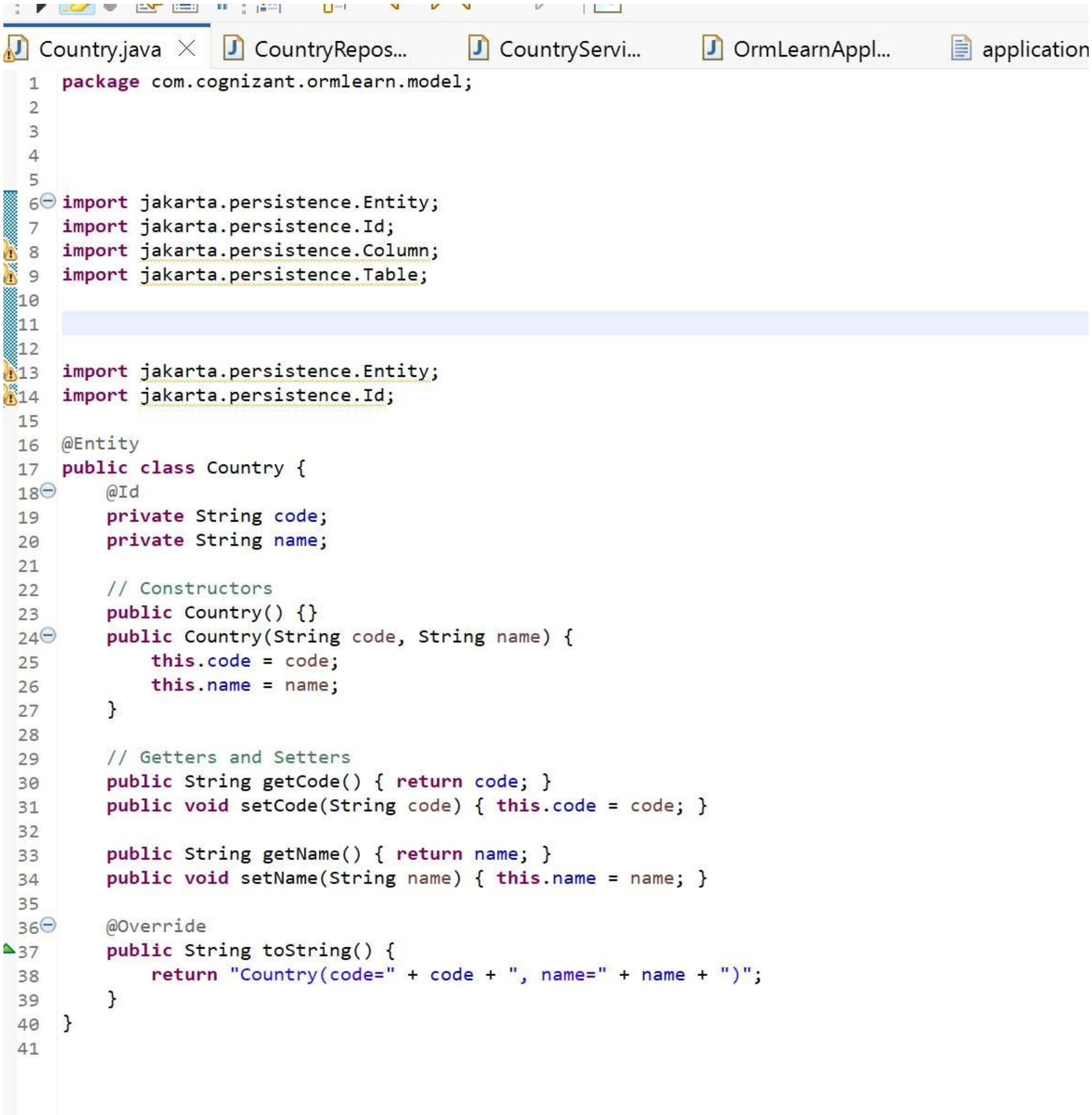
```java
package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Country;
import com.cognizant.ormlearn.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import java.util.List;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public List<Country> getAllCountries() {
        return countryRepository.findAll();
    }
}

```



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


import org.springframework.data.jpa.repository.JpaRepository;
import com.cognizant.ormlearn.model.Country;

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

```

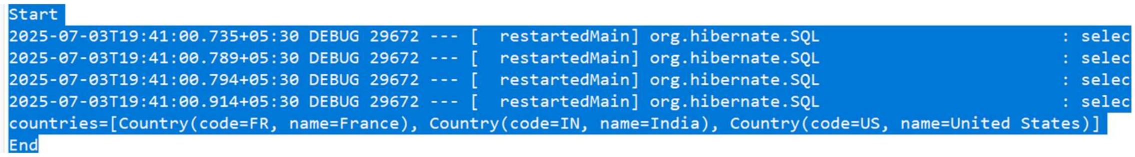
MYSQL:

```sql
USE omlearn;
CREATE TABLE country (
   code CHAR(2) PRIMARY KEY,
   name VARCHAR(255) NOT NULL
);

INSERT INTO country (code, name)
VALUES ('IN', 'India'), ('US', 'United States'), ('FR', 'France');
```

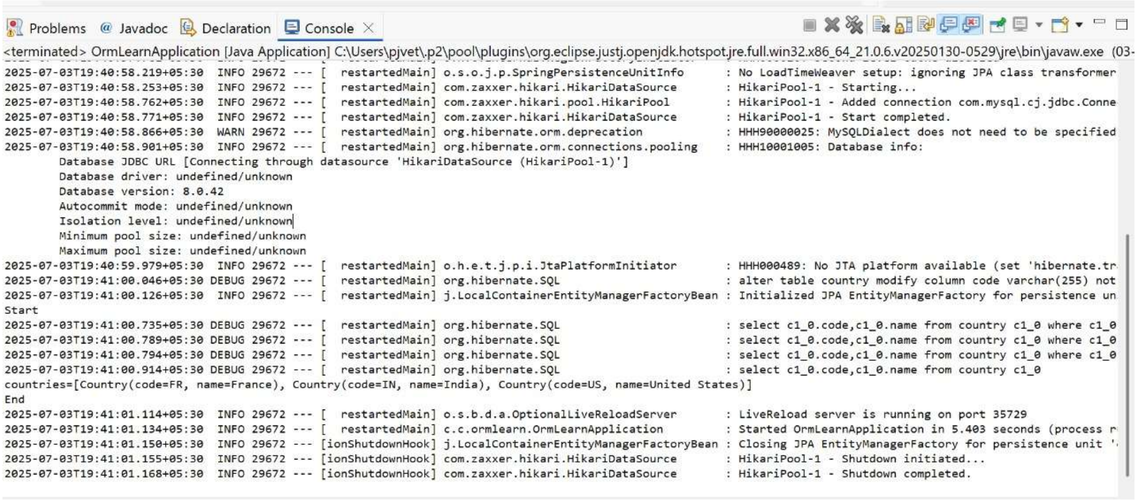


```java
package com.cognizant.ormlearn.model;




import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Column;
import jakarta.persistence.Table;



import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Country {
    @Id
    private String code;
    private String name;

    // Constructors
    public Country() {}
    public Country(String code, String name) {
        this.code = code;
        this.name = 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 + ")";
    }
}
```

OUTPUT:



```
Start
2025-07-03T19:41:00.735+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : selec
2025-07-03T19:41:00.789+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : selec
2025-07-03T19:41:00.794+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : selec
2025-07-03T19:41:00.914+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : selec
countries=[Country(code=FR, name=France), Country(code=IN, name=India), Country(code=US, name=United States)]
End
```



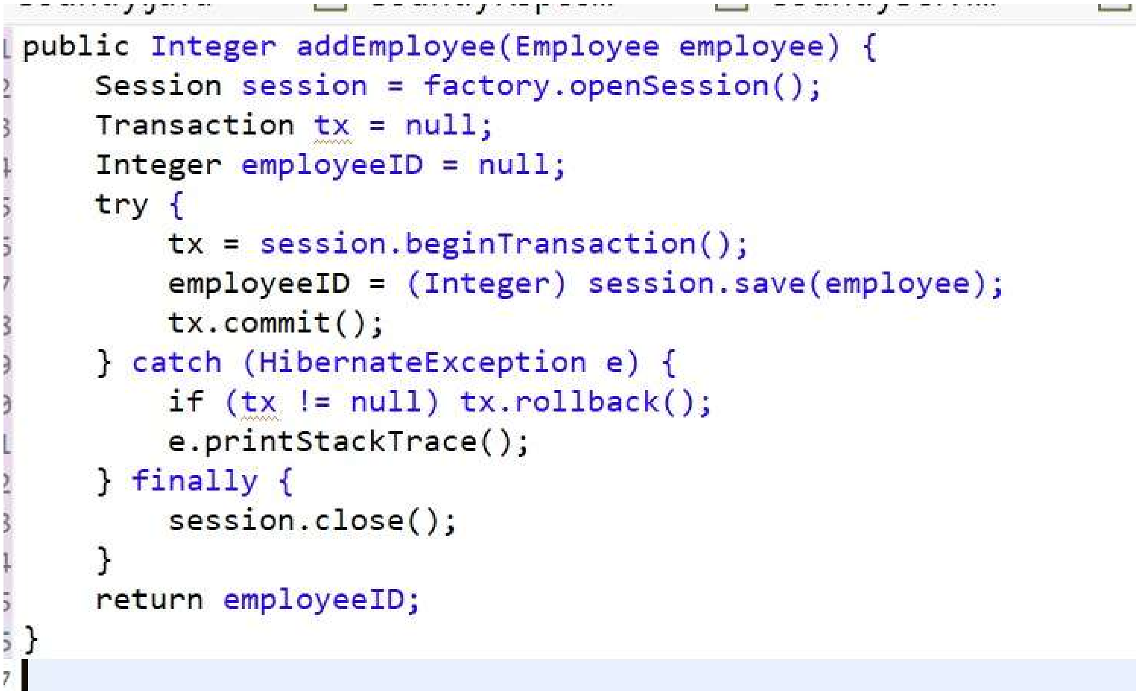
```
<terminated> OrmLearnApplication [Java Application] C:\Users\pjvet\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.6.v20250130-0529\jre\bin\javaw.exe  (03-
2025-07-03T19:40:58.219+05:30  INFO 29672 --- [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-03T19:40:58.253+05:30  INFO 29672 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2025-07-03T19:40:58.762+05:30  INFO 29672 --- [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.cj.jdbc.Conne
2025-07-03T19:40:58.771+05:30  INFO 29672 --- [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2025-07-03T19:40:58.866+05:30  WARN 29672 --- [  restartedMain] org.hibernate.orm.deprecation            : HHH90000025: MySQLDialect does not need to be specified
2025-07-03T19:40:58.901+05:30  INFO 29672 --- [  restartedMain] org.hibernate.orm.connections.pooling    : HHH10001005: Database info:
	Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
	Database driver: undefined/unknown
	Database version: 8.0.42
	Autocommit mode: undefined/unknown
	Isolation level: undefined/unknown
	Minimum pool size: undefined/unknown
	Maximum pool size: undefined/unknown
2025-07-03T19:40:59.979+05:30  INFO 29672 --- [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.tr
2025-07-03T19:41:00.046+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : alter table country modify column code varchar(255) not
2025-07-03T19:41:00.126+05:30  INFO 29672 --- [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence un
Start
2025-07-03T19:41:00.735+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : select c1_0.code,c1_0.name from country c1_0 where c1_0
2025-07-03T19:41:00.789+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : select c1_0.code,c1_0.name from country c1_0 where c1_0
2025-07-03T19:41:00.794+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : select c1_0.code,c1_0.name from country c1_0 where c1_0
2025-07-03T19:41:00.914+05:30 DEBUG 29672 --- [  restartedMain] org.hibernate.SQL                        : select c1_0.code,c1_0.name from country c1_0
countries=[Country(code=FR, name=France), Country(code=IN, name=India), Country(code=US, name=United States)]
End
2025-07-03T19:41:01.114+05:30  INFO 29672 --- [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
2025-07-03T19:41:01.134+05:30  INFO 29672 --- [  restartedMain] c.c.ormlearn.OrmLearnApplication         : Started OrmLearnApplication in 5.403 seconds (process r
2025-07-03T19:41:01.150+05:30  INFO 29672 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit '
2025-07-03T19:41:01.155+05:30  INFO 29672 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown initiated...
2025-07-03T19:41:01.168+05:30  INFO 29672 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Shutdown completed.
```

Difference between JPA, Hibernate and Spring Data JPA

Feature	**JPA**	**Hibernate**	**Spring Data JPA**
Type	Specification	Implementation	Abstraction over JPA
Provides implementation?	No	Yes	No (Uses Hibernate)
Session management	Manual	Manual	Automatic
Boilerplate code	Medium	High	Very Low
Repository support	No	No	Yes

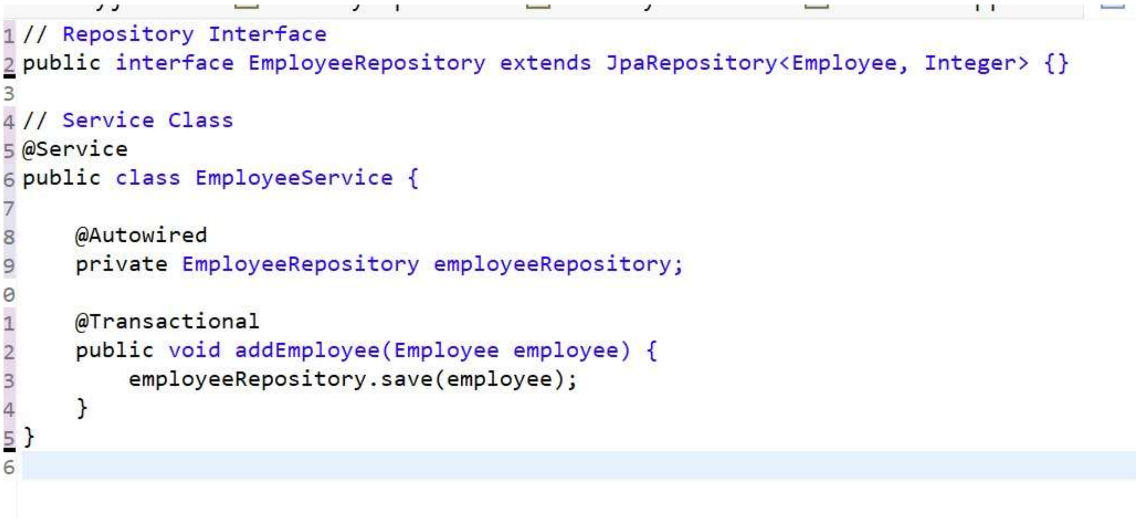
Code Comparison:

Hibernate Code:



```
public Integer addEmployee(Employee employee) {
    Session session = factory.openSession();
    Transaction tx = null;
    Integer employeeID = null;
    try {
        tx = session.beginTransaction();
        employeeID = (Integer) session.save(employee);
        tx.commit();
    } catch (HibernateException e) {
        if (tx != null) tx.rollback();
        e.printStackTrace();
    } finally {
        session.close();
    }
    return employeeID;
}
```

Spring Data JPA Code:



```
// Repository Interface
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

// Service Class
@Service
public class EmployeeService {

    @Autowired
    private EmployeeRepository employeeRepository;

    @Transactional
    public void addEmployee(Employee employee) {
        employeeRepository.save(employee);
    }
}
```