**pom.xml**

xml

CopyEdit

<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>

</properties>

<dependencies>

<!-- Spring Core Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

**📄 applicationContext.xml (in src/main/resources)**

xml

CopyEdit

<?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="com.library.repository.BookRepository"/>

<!-- Service Bean with Dependency Injection -->

<bean id="bookService" class="com.library.service.BookService">

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

</bean>

</beans>

**📄 LibraryManagementApp.java**

java

CopyEdit

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApp {

public static void main(String[] args) {

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

BookService service = context.getBean("bookService", BookService.class);

service.listBooks();

}

}

**📄 BookRepository.java**

java

CopyEdit

package com.library.repository;

public class BookRepository {

public void display() {

System.out.println("BookRepository: fetching books...");

}

}

**📄 BookService.java**

java

CopyEdit

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 listBooks() {

System.out.println("BookService: Listing books...");

bookRepository.display();

}

}

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

**pom.xml**

xml

CopyEdit

<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>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<!-- Spring Core (ApplicationContext, Beans, DI) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring AOP (for logging/aspects) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<!-- Spring WebMVC (for future exercises) -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

<!-- SLF4J (optional but common for AOP logging) -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<!-- Logback for SLF4J logging -->

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin for Java 1.8 -->

<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>

**Spring Data JPA - Quick Example**

**Difference between JPA, Hibernate and Spring Data JPA**

**Book.java**

java

CopyEdit

package com.library.entity;

import jakarta.persistence.\*;

@Entity

public class Book {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String title;

private String author;

// Getters and Setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getTitle() { return title; }

public void setTitle(String title) { this.title = title; }

public String getAuthor() { return author; }

public void setAuthor(String author) { this.author = author; }

}

**📄 BookRepository.java**

java

CopyEdit

package com.library.repository;

import com.library.entity.Book;

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

public interface BookRepository extends JpaRepository<Book, Long> {

Book findByTitle(String title);

}

**📄 BookService.java**

java

CopyEdit

package com.library.service;

import com.library.entity.Book;

import com.library.repository.BookRepository;

import org.springframework.stereotype.Service;

@Service

public class BookService {

private final BookRepository repository;

public BookService(BookRepository repository) {

this.repository = repository;

}

public void addBook(String title, String author) {

Book book = new Book();

book.setTitle(title);

book.setAuthor(author);

repository.save(book);

}

}

**📄 LibraryApp.java (Main Class)**

java

CopyEdit

package com.library;

import com.library.service.BookService;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibraryApp implements CommandLineRunner {

private final BookService bookService;

public LibraryApp(BookService bookService) {

this.bookService = bookService;

}

public static void main(String[] args) {

SpringApplication.run(LibraryApp.class, args);

}

@Override

public void run(String... args) {

bookService.addBook("The Alchemist", "Paulo Coelho");

}

}

**📄 application.properties**

properties

CopyEdit

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.show-sql=true

spring.jpa.hibernate.ddl-auto=update

+---------------------+----------------------+------------------------+--------------------------+

| Feature | JPA | Hibernate | Spring Data JPA |

+---------------------+----------------------+------------------------+--------------------------+

| Type | Specification | Implementation | Spring abstraction |

| Provided By | Java (Jakarta EE) | Hibernate ORM | Spring Framework |

| Purpose | Standard API | Actual ORM provider | Simplifies JPA usage |

| CRUD Operations | Manual (EntityManager)| Yes (Session/API) | Auto via JpaRepository |

| Extra Features | No | Yes (caching, SQL) | Yes (query derivation) |

| Configuration | Verbose | Medium | Minimal (Boot-friendly) |

| Learning Curve | Medium | High | Low |

| Use in Projects | Base for ORM | Common JPA provider | Widely used in Spring |

+---------------------+----------------------+------------------------+--------------------------+