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

**Code :**

**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>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.31</version>

</dependency>

</dependencies>

</project>

**src/main/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 id="bookRepository" class="com.library.repository.BookRepository"/>

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

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

</bean>

</beans>

**src/main/java/com/library/repository/BookRepository.java**

package com.library.repository;

public class BookRepository {

public String getBookDetails() {

return "Book: Clean Code by Robert C. Martin";

}

}

**src/main/java/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 displayBook() {

System.out.println(bookRepository.getBookDetails());

}

}

**src/main/java/com/library/MainApp.java**

**output :**

Book: Clean Code by Robert C. Martin

**Exercise 2: Implementing Dependency Injection**

**Code :**

1. **BookRepository.java**

package com.example.library;

public class BookRepository {

public String getBookTitle() {

return "Effective Java";

}

}

1. **BookService.java**

package com.example.library;

public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBookTitle() {

System.out.println("Book Title: " + bookRepository.getBookTitle());

}

}

1. **LibraryManagementApplication.java**

package com.example.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

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

BookService bookService = (BookService) context.getBean("bookService");

bookService.displayBookTitle();

}

}

1. **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="com.example.library.BookRepository" />

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

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

</bean>

</beans>

**Output :**

Book Title: Effective Java

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

<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.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.36</version>

</dependency>

<!-- Servlet API (needed for Spring MVC) -->

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<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>

**Output :**

mvn clean install

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

| **Feature** | **JPA (Java Persistence API)** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification (Interface) | Implementation (Framework) | Abstraction layer on top of JPA |
| **Purpose** | Defines a standard for ORM in Java | Implements the JPA specification + extra features | Simplifies JPA usage in Spring applications |
| **Provider** | Defined by Oracle (Java EE spec) | Developed by Red Hat | Part of Spring ecosystem |
| **Dependency** | Needs an implementation like Hibernate or EclipseLink | Can be used directly or with JPA | Uses JPA (commonly with Hibernate) underneath |
| **Usage** | API to define ORM mapping (e.g., @Entity, @Id) | Provides APIs and configuration beyond JPA | Repository pattern with CRUD methods auto-implemented |
| **Boilerplate Code** | Requires writing EntityManager and queries manually | Reduces some boilerplate but still needs DAOs | Removes most boilerplate using interfaces |
| **Example Annotation** | @Entity, @Table, @Id | Same as JPA + @GenericGenerator, etc. | @Repository, extends JpaRepository, etc. |
| **Query Language** | JPQL | JPQL + Hibernate Query Language (HQL) | JPQL + Derived queries + @Query annotation |
| **Learning Curve** | Medium | Higher than JPA due to more options | Easiest due to Spring abstractions |