# **WEEK 3 Assignment**

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

**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>org.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

**applicationContent.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">

<!-- Define Repository Bean -->

<bean id="bookRepository" class="org.example.BookRepository" />

<!-- Define Service Bean with dependency injection -->

<bean id="bookService" class="org.example.BookService">

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

</bean>

</beans>

**BookRepository.java**

package org.example;

public class BookRepository {

public void saveBook(String bookName) {

System.out.println("Saving book: " + bookName);

}

}

**BookService.java**

package org.example;

import org.example.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);

}

}

**App.java**

package org.example;

import org.example.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App

{

public static void main(String[] args) {

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

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

bookService.addBook("Spring in Action");

}

}

Output :

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 2: Implementing Dependency Injection**

**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>org.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

**applicationContent.xml**

<!-- applicationContext.xml -->

<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">

<!-- BookRepository Bean -->

<bean id="bookRepository" class="org.example.BookRepository" />

<!-- BookService Bean -->

<bean id="bookService" class="org.example.BookService">

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

</bean>

</beans>

**BookRepository.java**

package org.example;

public class BookRepository {

public void listBooks() {

System.out.println("Listing all books...");

}

}

**BookService.java**

package org.example;

import org.example.BookRepository;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBooks() {

bookRepository.listBooks();

}

}

**App.java**

package org.example;

import org.example.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App

{

public static void main(String[] args) {

ApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.displayBooks();

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.

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

**pom.xml(LibraryManagement):**

<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>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.13.2</version>

<scope>test</scope>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Spring Web MVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.32</version>

</dependency>

<!-- Optional: Servlet API (for Web MVC) -->

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- 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>

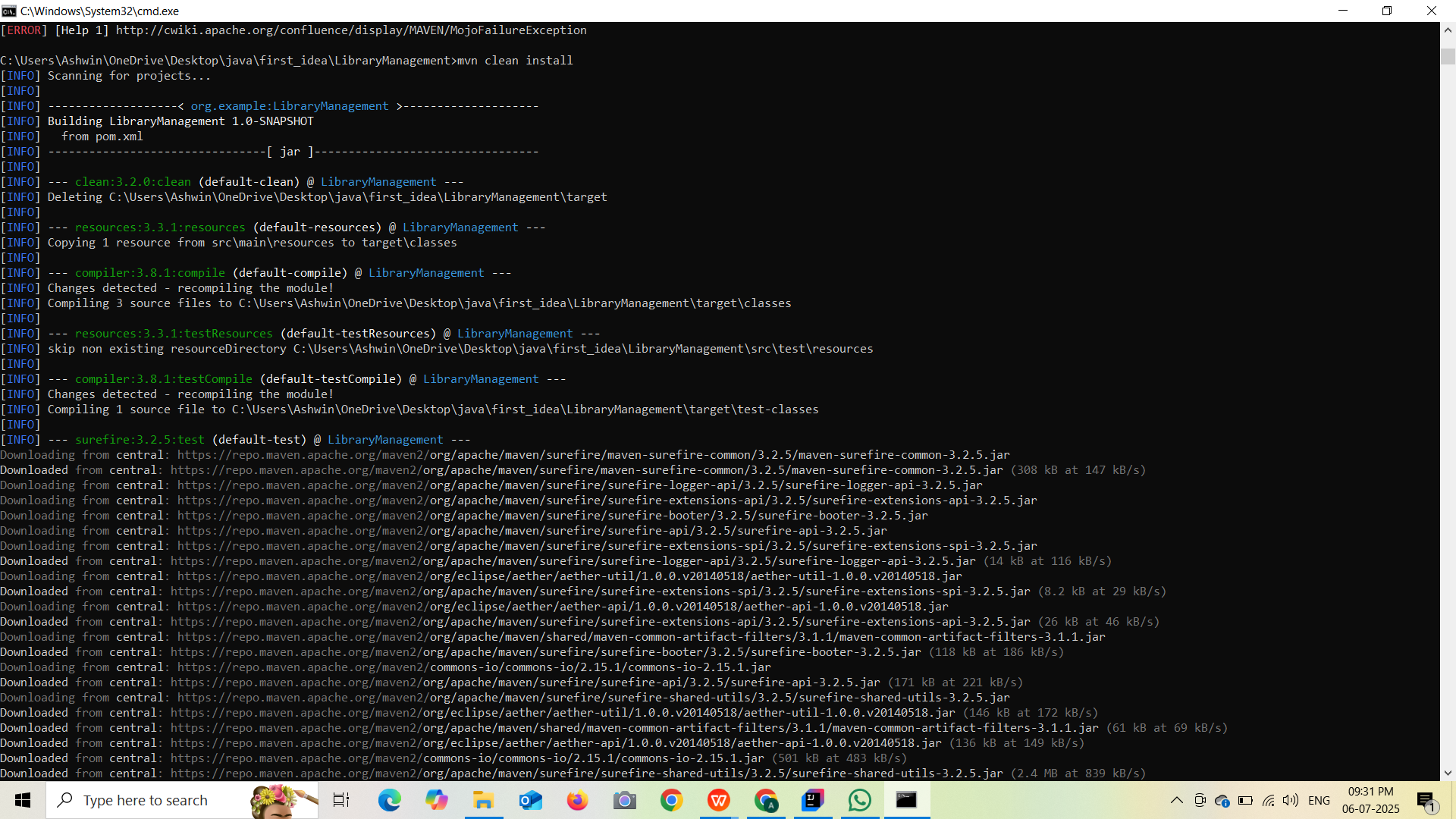
</plugin>

</plugins>

</build>

</project>

Output:



Spring data JPA – Quick Example:

Pom.xml:

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>com.h2database</groupId>

<artifactId>h2</artifactId>

<scope>runtime</scope>

</dependency>

</dependencies>

@SpringBootApplication

public class SpringDataJpaDemoApplication implements CommandLineRunner {

@Autowired

private UserRepository userRepository;

public static void main(String[] args) {

SpringApplication.run(SpringDataJpaDemoApplication.class, args);

}

@Override

public void run(String... args) {

userRepository.save(new User("Alice", "alice@example.com"));

userRepository.save(new User("Bob", "bob@example.com"));

userRepository.save(new User("Alice", "alice123@example.com"));

List<User> users = userRepository.findByName("Alice");

users.forEach(System.out::println);

}

@Entity

public static class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

public User() {}

public User(String name, String email) {

this.name = name;

this.email = email;

}

@Override

public String toString() {

return "User{id=" + id + ", name='" + name + "', email='" + email + "'}";

}

}

public interface UserRepository extends JpaRepository<User, Long> {

List<User> findByName(String name);

}

}

Difference between JPA,Hibernate and spring data JPA

**JPA**

JPA (Java Persistence API) is a specification provided by Oracle as part of Java EE (now Jakarta EE). It defines the standard for object-relational mapping in Java but does not provide a working implementation on its own. Developers often need to manage the EntityManager, define entities using annotations, and write queries manually. It involves a moderate amount of boilerplate code and is commonly used as the base API for persistence in Java applications.

**Hibernate**

Hibernate is a framework developed by Red Hat that implements the JPA specification and extends it with additional features. It simplifies persistence by providing flexible APIs, automatic dirty checking, and caching. Compared to JPA, Hibernate requires less boilerplate code and offers more control over the database interaction. Queries may still need to be written, but Hibernate provides multiple ways to handle them more easily than raw JPA.

**Spring Data JPA**

Spring Data JPA is a module of the Spring Framework that builds on top of JPA and typically uses Hibernate as the default provider. It abstracts much of the boilerplate involved in data access, allowing developers to define repositories with minimal code. Most queries are automatically generated from method names, making manual query implementation rare. Spring Data JPA is designed for maximum productivity and seamless integration with the rest of the Spring ecosystem

Additional Exercise

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

**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>org.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

**applicationContent.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">

<!-- Define BookRepository Bean -->

<bean id="bookRepository" class="org.example.BookRepository" />

<!-- Define BookService Bean and inject BookRepository -->

<bean id="bookService" class="org.example.BookService">

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

</bean>

</beans>

**BookRepository.java:**

package org.example;

public class BookRepository {

public void saveBook(String title) {

System.out.println("Saving book: " + title);

}

}

**BookService.java:**

package org.example;

import org.example.BookRepository;

public class BookService{

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String title) {

System.out.println("Adding book: " + title);

bookRepository.saveBook(title);

}

}

**App.java:**

package org.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App

{

public static void main(String[] args) {

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

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

bookService.addBook("Java for Beginners");

}

}

Output:

A screenshot of a computer

AI-generated content may be incorrect.

**Exercise 7: Implementing Constructor and Setter Injection**

**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>org.example</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<packaging>jar</packaging>

<name>LibraryManagement</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

**applicationContent.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">

<!-- BookRepository Bean -->

<bean id="bookRepository" class="org.example.BookRepository" />

<!-- BookService Bean using both constructor and setter injection -->

<bean id="bookService" class="org.example.BookService">

<!-- Constructor Injection -->

<constructor-arg value="Madurai Public Library"/>

<!-- Setter Injection -->

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

</bean>

</beans>

**BookRepository.java**

package org.example;

public class BookRepository {

public void saveBook(String title) {

System.out.println("Saving book: " + title);

}

}

**BookService.java**

package org.example;

public class BookService {

private final String libraryName;

private BookRepository bookRepository;

public BookService(String libraryName) {

this.libraryName = libraryName;

}

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String title) {

System.out.println("[" + libraryName + "] Adding book: " + title);

bookRepository.saveBook(title);

}

}

**App.java**

package org.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App

{

public static void main(String[] args) {

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

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

bookService.addBook("Spring in Action");

}

}

Output:

