**What is Dependency Injection (DI)?**

**Dependency Injection (DI)** is a design pattern used in **Spring Framework** where **objects (dependencies)** are provided to a class **by the framework**, instead of the class creating them itself.

**1.Modify the XML Configuration:**

**Step 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

https://www.springframework.org/schema/beans/spring-beans.xsd">

**<!-- BookRepository Bean -->**

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

**<!-- BookService Bean with DI via setter -->**

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

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

</bean>

</beans>

**2. Update the BookService Class:**

**Step 2: BookService Class**

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() {

System.out.println("BookService: calling BookRepository...");

bookRepository.save();

}

}

**Step 3: Ensure BookRepository Class Exists**

package com.library.repository;

public class BookRepository {

public void save() {

System.out.println("BookRepository: Saving book to the database.");

}

}

**Step 4: Test with Main Class**

**package com.library.main;**

import com.library.service.BookService;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

ClassPathXmlApplicationContext context =

new ClassPathXmlApplicationContext("applicationContext.xml");

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

bookService.addBook();

context.close();

}}

**Focus: Demonstrating IoC and DI in action**

**Code: Same classes, but now:**

* **BookRepository is actually injected into BookService**
* **BookService calls a method on BookRepository**

**Print statements show interaction, proving DI is working**