**Mandatory hands-on**

**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.

1. **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.

1. **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.**

1. **Run the Application:**

* Create a main class to load the Spring context and test the configuration.

**Steps:**

* + 1. **Set up a Spring Project:**

Create a Maven project named LibraryManagement: project name is LibraryManagement.

Add spring core dependencies in the pom.xml file: Go to pom.xml and add the following dependencies in that file.

**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

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>

<dependencies>

<!-- Spring Core -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

</project>

* + 1. **Configure the Application Context:**

Create an XML configuration file named applicationContext.xml in the **src/main/resources**

directory.

Creating a file in src/main/resources as applicationContext.xml

**application.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">

<!-- Repository Bean -->

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

<!-- Service Bean -->

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

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

</bean>

</beans>

* + 1. **Define Service and Repository Classes:**

In src/main/java create a new package named as com.library.service, in that create a class named

as Bookservice:

package com.library.service;

import com.library.repository.Bookrepository;

public class Bookservice {

private Bookrepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(Bookrepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBook(int id) {

String book = bookRepository.findBookById(id);

System.out.println(book);

}

}

In src/main/java create another package named as com.library,repository, in that create one class

name it as Bookreository:

package com.library.repository;

public class Bookrepository {

public String findBookById(int id) {

return "Book with ID: " + id;

}

}

* + 1. **Run the Application:**

Create a main class to load the Spring context and test the configuration.

Creating a class in the package name com.library, class name Mainjava

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");

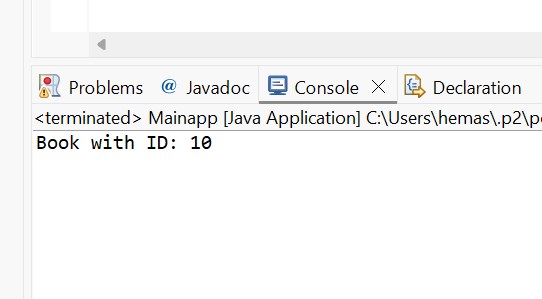
bookService.displayBook(10);

}

}

Right click Mainjava the Run as Java application.

**Output:**



**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**.

1. **Update the BookService Class:**

* Ensure that **BookService** class has a setter method for **BookRepository**.

1. **Test the Configuration:**

* Run the **LibraryManagementApplication** main class to verify the dependency injection.

**Step:**

1. **Modify the XML Configuration:**

Update applicationContext.xml to wire BookRepository into Bookservice:

<?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 Dependency Injection -->

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

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

</bean>

</beans>

1. **Update the BookService Class:**

Ensure there's a settermethod named setBookRepository(...) for Spring to inject the dependency.

package com.library.service;

import com.library.repository.Bookrepository;

public class Bookservice {

private Bookrepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(Bookrepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayBook(int id) {

String book = bookRepository.findBookById(id);

System.out.println(book);

}

}

1. **Test the Configuration:**

Run a simple main class to test whether Spring's IoC container performs the injection.

package com.library;

import com.library.service.Bookservice;

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.displayBook(42);

}

}

**Output:**

A screenshot of a computer

AI-generated content may be incorrect.

* This confirms that Bookrepository is successfully injected into Bookservice.
* That means, Spring’s IoC and DI are working properly.

**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**.

1. **Add Spring Dependencies in pom.xml:**

* Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.

1. **Configure Maven Plugins:**

* Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

**Steps:**

* + 1. **Create a new maven project:**

File > New > Maven Project

Check "Create a simple project (skip archetype selection)" → Click Next

Fill in:

* Group Id: com.library
* Artifact Id: LibraryManagement
* Version: leave as default 0.0.1-SNAPSHOT

Click Finish

* + 1. **Add Spring Dependencies in 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>0.0.1-SNAPSHOT</version>

<properties>

<maven.compiler.source>1.8</maven.compiler.source>

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

</properties>

<dependencies>

<!-- Spring Context (Core DI) -->

<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>

</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>

**3. Configure Maven Plugins:**

After saving pom.xml:

1. Right-click your project → Maven > Update Project
2. Check "Force Update of Snapshots/Releases" → Click OK

This will download all Spring dependencies to your local .m2 Maven repository.