Spring Core and Maven

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.

1. **Set Up a Spring Project:**

Create a Maven project named **LibraryManagement**

Add Spring Core dependencies in the **pom.xml** file.

<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>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

</project>

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.

<?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>">

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

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

</bean>

</beans>

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

Create a package **com.library.service** and add a class **BookService**.

package com.library.service;

public class BookService {

public void testService() {

System.***out***.println("BookService bean loaded successfully.");

}

}

Create a package **com.library.repository** and add a class **BookRepository**.

package com.library.repository;

public class BookRepository {

public void testRepository() {

System.***out***.println("BookRepository bean loaded successfully.");

}

}

1. **Run the Application:**

package com.library.main;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

import com.library.repository.BookRepository;

public class LibraryApp {

public static void main(String[] args) {

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

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

BookRepository bookRepository = (BookRepository) context.getBean ("bookRepository");

bookService.testService();

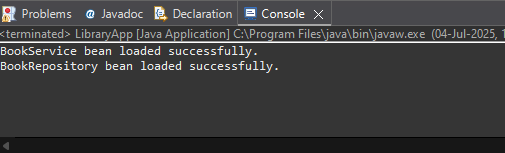
bookRepository.testRepository();

((ClassPathXmlApplicationContext) context).close();

}

}

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.

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>">

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

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

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

</bean>

</beans>

1. **Update the BookService Class:**

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

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(String bookName) {

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

bookRepository.saveBook(bookName);

}

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Book '" + bookName + "' saved to repository.");

}

}

1. **Test the Configuration:**

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

package com.library.main;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.context.ApplicationContext;

import com.library.service.BookService;

public class LibraryApp {

public static void main(String[] args) {

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

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

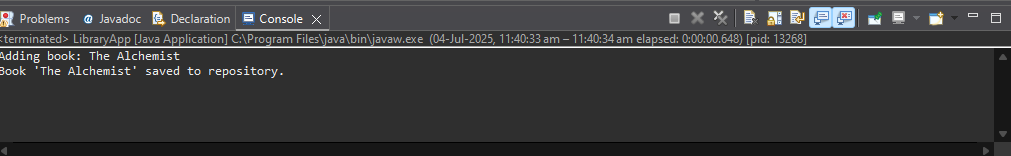
bookService.addBook("The Alchemist");

((ClassPathXmlApplicationContext) context).close();

}

}

OUTPUT:



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.

<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>

<dependencies>

<!-- Spring Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.29</version>

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.29</version>

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.29</version>

</dependency>

</dependencies>

<build>

<plugins>

<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>

Exercise 5: Configuring the Spring IoC Container

**Scenario:**

The library management application requires a central configuration for beans and dependencies.

1. **Create Spring Configuration File:**

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

Define beans for **BookService** and **BookRepository** in the XML file.

<?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>">

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

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

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

</bean>

</beans>

1. **Update the BookService Class:**

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

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 addBook(String bookName) {

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

bookRepository.saveBook(bookName);

}

}

package com.library.repository;

public class BookRepository {

public void saveBook(String bookName) {

System.***out***.println("Book '" + bookName + "' saved to repository.");

}

}

1. **Run the Application:**

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

package com.library.main;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import org.springframework.context.ApplicationContext;

import com.library.service.BookService;

public class LibraryApp {

public static void main(String[] args) {

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

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

bookService.addBook("Akbar and Birbal");

((ClassPathXmlApplicationContext) context).close();

}

}

OUTPUT:

