**SPRING CORE\_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 new Maven Project named LibraryManagement.
* File -> new -> Maven Project
* Add Spring Core dependencies in the pom.xml file.

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

1. Configure the Application Context:

* Right click on src/main/resources -> file -> name it as 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>

1. Define Service and Repository Classes:

* Src/main//java -> com.library.service (package) -> BookService (class name).
* 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("Book: " + bookRepository.getBook());

}

}

* Src/main/java -> com.library.repository (package) -> BookRepository (class name).
* BookRepository.java

package com.library.repository;

public class BookRepository {

public String getBook() {

return "Programming with Java by Balaguruswamy.";

}

}

1. Run the Application

* Src/main/java -> com.library (package) -> LibraryManagementApplication (class name)
* LibraryManagementApplication.java

package com.library;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryManagementApplication {

public static void main(String[] args) {

try (ClassPathXmlApplicationContext context = new

ClassPathXmlApplicationContext("applicationContext.xml")) {

BookService bookService = context.getBean("bookService",

BookService.class);

bookService.displayBook();

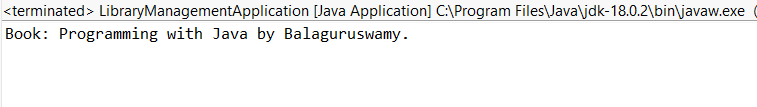
}

}

}

* LibraryManagementApplication.java -> 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.

1. Modify the xml configuration:

* 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">

<!-- Repository Bean -->

<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 BookService Class:

* 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("Book: " + bookRepository.getBook());

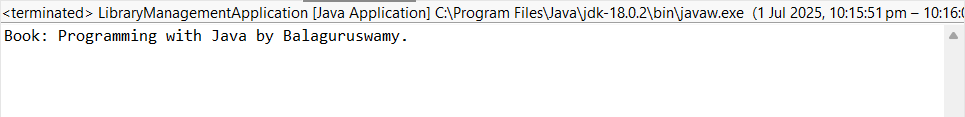
}

}

1. Test the configuration:

* Right click LibraryManagementApplication -> Run As -> Java Application.

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.

1. Create a new maven project:

* File > New -> Maven Project

1. Pom.xml (Spring Dependencies)

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

</dependencies>

1. Configure:

* Src/main/java/com/example/LibraryApp
* LibraryApp.java

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {

public static void main(String[] args) {

ApplicationContext context = new

ClassPathXmlApplicationContext("beans.xml");

LibraryService libraryService = context.getBean("libraryService",

LibraryService.class);

libraryService.displayMessage(); } }

* Src/main/java/com/example/LibraryService
* LibraryService.java

package com.example;

public class LibraryService {

public void displayMessage() {

System.out.println("LibraryService is working via Spring Context!");

}

}

* Src/main/resources/beans.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

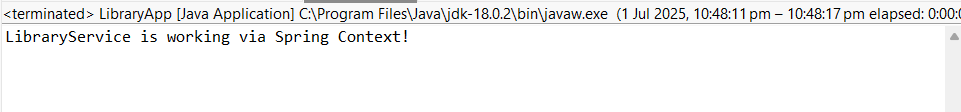
http://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="libraryService" class="com.example.LibraryService" />

</beans>

* Right click -> LibraryApp.java -> Run As -> Java Application

OUTPUT:



**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

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

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

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

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

</bean>

</beans>

1. Update classes:

* Src/main/java -> com.example -> BookService class.
* BookService.java

package com.example;

public class BookService {

private BookRepository bookRepository;

//setter method

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayService() {

System.out.println("BookService: Serving book-related operations.");

bookRepository.displayRepo();

}

}

* Src/main/java -> BookRepository

package com.example;

public class BookRepository {

public void displayRepo() {

System.out.println("BookRepository: Fetching books from database....");

}

}

1. Run Main Application

package com.example;

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");

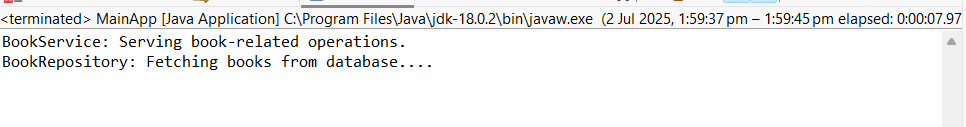
BookService bookService = context.getBean("bookService", BookService.class);

bookService.displayService();

}

}

OUTPUT:



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

**Scenario:** The library management application requires both constructor and setter injection for better control over bean initialization.

1. Update 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.BookRepository" />

<bean id="libraryConfig" class="com.example.LibraryConfig">

<constructor-arg value="City Central Library"/>

</bean>

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

<constructor-arg ref="libraryConfig" />

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

</bean>

</beans>

1. Ensure that the BookService class has a setter method for BookRepository and configure it in applicationContext.xml.

* BookService.java

package com.example;

public class BookService {

private BookRepository bookRepository;

private LibraryConfig libraryConfig;

// Constructor

public BookService(LibraryConfig libraryConfig) {

this.libraryConfig = libraryConfig;

}

// Setter

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayInfo() {

System.out.println("BookService: Initialized with both dependencies.");

libraryConfig.printLibraryName();

bookRepository.displayRepo();

}

}

* BookRepository.java

package com.example;

public class BookRepository {

public void displayRepo() {

System.out.println("BookRepository: Providing data from repository.");

}

}

* LibraryConfig.java

package com.example;

public class LibraryConfig {

private String libraryName;

public LibraryConfig(String libraryName) {

this.libraryName = libraryName;

}

public void printLibraryName() {

System.out.println("LibraryConfig: Library Name is " + libraryName);

}

}

1. Test the Injection:
   * Run the LibraryManagementApplication main class to verify both constructor and setter injection.
   * LibraryManagementApplication.java

package com.example;

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

bookService.displayInfo();

}

}

* + LibraryManagementApplication.java -> Run As -> Java Application.

OUTPUT:

