**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.**
2. **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.**
3. **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.**
4. **Run the Application:**

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

**BookRepository.java**

package com.library.repository;

public class BookRepository {

public void save() {

System.out.println("Book saved to database.");

}

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for Spring injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook() {

System.out.println("Inside BookService - addBook()");

bookRepository.save();

}

}

**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">

<!-- Define Repository Bean -->

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

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

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

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

</bean>

</beans>

**LibraryApp.java**

package com.library;

import com.library.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {

public static void main(String[] args) {

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

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

service.addBook();

}

}

**A screenshot of a computer

AI-generated content may be incorrect.**

**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**.
2. **Update the BookService Class:**
   * Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**

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

**Book.java**

public class Book {

    private int id;

    private String title;

    public Book() {}

    public Book(int id, String title) {

        this.id = id;

        this.title = title;

    }

    public int getId() { return id; }

    public String getTitle() { return title; }

    @Override

    public String toString() {

        return "Book [id=" + id + ", title=" + title + "]";

    }

}

**BookRepository.java**

import java.util.\*;

public class BookRepository {

public List<Book> findAllBooks() {

List<Book> books = new ArrayList<>();

books.add(new Book(1, "Java Fundamentals"));

books.add(new Book(2, "Spring in Action"));

return books;

}

}

**BookService.java**

import java.util.List;

public class BookService {

private BookRepository bookRepository;

// Setter for Dependency Injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void displayAllBooks() {

List<Book> books = bookRepository.findAllBooks();

for (Book book : books) {

System.out.println(book);

}

}

}

**LibraryManagementApplication.java**

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.displayAllBooks();

}

}

**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">

<!-- BookRepository Bean -->

<bean id="bookRepository" class="BookRepository" />

<!-- BookService Bean with DI -->

<bean id="bookService" class="BookService">

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

</bean>

</beans>

A screenshot of a computer program

AI-generated content may be incorrect.

* + messages indicating method execution times.

**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**.
2. **Add Spring Dependencies in pom.xml:**
   * Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
   * Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml 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

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.library</groupId>

<artifactId>LibraryManagement</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>5.3.33</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.3.33</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.11.0</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

</configuration>

</plugin>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<configuration>

<mainClass>com.library.MainApp</mainClass>

</configuration>

</plugin>

</plugins>

</build>

</project>

**LibraryService.java**

package com.library;

public class LibraryService {

public void displayBooks() {

System.out.println("Listing all available books in the library...");

}

}

**LibraryConfig.java**

package com.library;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class LibraryConfig {

@Bean

public LibraryService libraryService() {

return new LibraryService();

}

}

**MainApp.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(LibraryConfig.class);

LibraryService libraryService = context.getBean(LibraryService.class);

libraryService.displayBooks();

}

}

A screenshot of a computer program

AI-generated content may be incorrect.