**Exercise 1: Configuring a Basic Spring Application**

**Scenario**: Create a Spring-based library management application using Maven, configure the application context with XML, define service and repository classes, and test the setup.

1 **Set Up a Spring Project**

* **Create a Maven Project**:
  + Project Name: LibraryManagement

2.  **Configure** pom.xml:

* Add the Spring Context dependency for Spring Core functionality.

<?xml version="1.0" encoding="UTF-8"?>

<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>

<properties>

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

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

<spring.version>3.2.18.RELEASE</spring.version>

</properties>

<dependencies>

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

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

</dependencies>

</project>

3.  **Configure the Application Context**

* **Create** applicationContext.xml:
  + Location: src/main/resources/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">

<!-- Define BookRepository bean -->

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

<!-- Define BookService bean -->

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

</beans>

4.  **Define Service and Repository Classes**

* **Create** BookRepository **Class**:
  + Location: src/main/java/com/library/repository/BookRepository.java

package com.library.repository;

public class BookRepository {

public void saveBook(String bookTitle) {

System.out.println("Saving book: " + bookTitle);

}

}

5.  **Create** BookService **Class**:

* Location: src/main/java/com/library/service/BookService.java

package com.library.service;

public class BookService {

public void addBook(String bookTitle) {

System.out.println("Adding book to service: " + bookTitle);

}

}

6.  **Run the Application**

* **Create Main Class**:
  + Location: src/main/java/com/library/LibraryManagementApplication.java

package com.library;

import com.linear.service.BookService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagementApplication {

public static void main(String[] args) {

// Load the Spring context from applicationContext.xml

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

// Retrieve the BookService bean

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

// Test the configuration

bookService.addBook("Spring in Action");

}

}

**Execution**:

mvn clean install

java -cp target/LibraryManagement-1.0-SNAPSHOT.jar com.library.LibraryManagementApplication

**Output**:

Adding book to service: Spring in Action

**Exercise 2: Implementing Dependency Injection**

**Scenario**: Modify the library management application to use Spring’s IoC and dependency injection to wire BookRepository into BookService.

1  **Modify the XML Configuration**

* **Update** applicationContext.xml:
  + Location: src/main/resources/applicationContext.xml
  + Content: Add dependency injection to wire BookRepository into BookService using setter injection.

<?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">

<!-- Define BookRepository bean -->

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

<!-- Define BookService bean with dependency injection -->

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

<!-- Inject bookRepository using setter injection -->

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

</bean>

</beans>

2  **Update the** BookService **Class**

* **Modify** BookService.java:
  + Location: src/main/java/com/library/service/BookService.java
  + Content: Add a BookRepository field, a setter method, and update addBook to use the repository.

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

private BookRepository bookRepository;

// Setter method for dependency injection

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void addBook(String bookTitle) {

System.out.println("Adding book to service: " + bookTitle);

// Delegate to repository

bookRepository.saveBook(bookTitle);

}

}

3 **Test the Configuration**

* **Use the Existing Main Class**:
  + Location: src/main/java/com/library/LibraryManagementApplication.java

4 **Execution**:

mvn clean install

java -cp target/LibraryManagement-1.0-SNAPSHOT.jar com.library.LibraryManagementApplication

5 **Output**:

Adding book to service: Spring in Action

Saving book: Spring in Action

**Exercise 4: Creating and Configuring a Maven Project**

**Scenario**: Set up a Maven project for the library management application, add Spring dependencies, and configure the Maven Compiler Plugin for Java 1.8.

1 **Create a New Maven Project**

* **Project Setup**:
  + The project (LibraryManagement) is already created in Exercise 1. If starting fresh, use:

mvn archetype:generate -DgroupId=com.library -DartifactId=LibraryManagement -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

2 **Add Spring Dependencies in** pom.xml

* **Update** pom.xml:
  + Location: pom.xml
  + Content: Include dependencies for spring-context, spring-aop, and spring-webmvc.

<?xml version="1.0" encoding="UTF-8"?>

<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>

<properties>

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

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

<spring.version>3.2.18.RELEASE</spring.version>

</properties>

<dependencies>

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

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

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

</dependency>

<!-- Spring AOP -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

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

</dependency>

<!-- Spring WebMVC -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

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

</dependency>

</dependencies>

<build>

<plugins>

<!-- Maven Compiler Plugin -->

<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>

4 **Configure Maven Plugins**

* The pom.xml above includes the Maven Compiler Plugin configured for Java 1.8.
* **Verification**:

mvn clean install