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.

Steps:

1. Set Up a Spring Project:

o Create a Maven project named LibraryManagement.

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

2. Configure the Application Context:

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

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

3. Define Service and Repository Classes:

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

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

4. Run the Application:

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

Code:

BookRepository.java

```
package com.library.repository;

public class BookRepository {

    public void saveBook(String bookName) {

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

    }

}
```

BookService.java

```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for Spring to inject BookRepository

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String bookName) {

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

        bookRepository.saveBook(bookName);

    }

}
```

App.java

```
package com.library;

import com.library.service.BookService;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

    public static void main(String[] args) {

        try (ClassPathXmlApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml")) {

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

            bookService.addBook("Spring in Action");

        }

    }

}
```

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:

o Update applicationContext.xml to wire BookRepository into BookService.

2. Update the BookService Class:

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

3. Test the Configuration:

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

Code:

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 and inject BookRepository -->
  <bean id="bookService" class="com.library.service.BookService">
    <property name="bookRepository" ref="bookRepository" />
  </bean>
</beans>
```

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:

o Create a new Maven project named LibraryManagement.

2. Add Spring Dependencies in pom.xml:

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

3. Configure Maven Plugins:

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

Code:

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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.library</groupId>
  <artifactId>LibraryManagement</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>LibraryManagement</name>
  <url>http://maven.apache.org</url>

  <dependencies>
    <!-- Spring Context -->
    <dependency>
      <groupId>org.springframework</groupId>
```

```
    <artifactId>spring-context</artifactId>
    <version>5.3.30</version>
  </dependency>

  <!-- Spring AOP -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>5.3.30</version>
  </dependency>

  <!-- Spring WebMVC -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.3.30</version>
  </dependency>
</dependencies>

<build>
  <plugins>
    <!-- Maven Compiler Plugin to set Java version -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.10.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
```

```
      </configuration>
    </plugin>

    <!-- Optional: Plugin to run the main class -->
    <plugin>
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>3.1.0</version>
      <configuration>
        <mainClass>com.library.App</mainClass>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Final Output:

```
Adding book: Spring in Action
Saving book: Spring in Action
[INFO] ------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------
[INFO] Total time:  2.957 s
```

Java: Ready