**Week 3 – Hands-on : Spring Core Maven**

Exercise 1 : Configuring a basic Spring Application

First we set up Spring project by creating a new maven project and adding Spring Core dependencies to pom.xml file.

<?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>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.29</version>  
 </dependency>  
 </dependencies>  
</project>

Then in the ‘src/main/resources’ folder we create applicationContext.xml file to configure “Spring beans”.

<?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"/>  
  
</beans>

Now, We create packages like “com.library.service” and “com.library.repository” and define the classes

**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook() {  
 System.*out*.println("BookRepository: Saving book to the database...");  
 }  
}

**BookService.java**

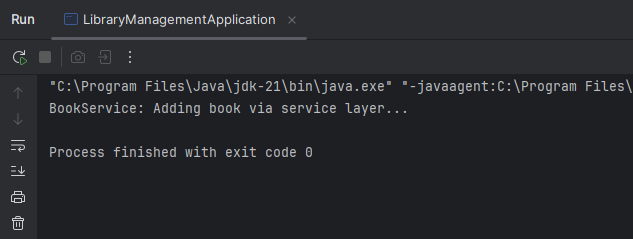
package com.library.service;  
  
public class BookService {  
 public void addBook() {  
 System.*out*.println("BookService: Adding book via service layer...");  
 }  
}

Finally we create the main class.

**LibraryManagementApplication.java**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagementApplication {  
  
 public static void main(String[] args) {  
 // first we load the spring context from the applicationContext.xml  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // get the bookService bean  
 BookService bookService = context.getBean("bookService", BookService.class);  
  
 // we use the method to see if it is working  
 bookService.addBook();  
 }  
}

Output



Exercise 2 : Implementing Dependency Injection

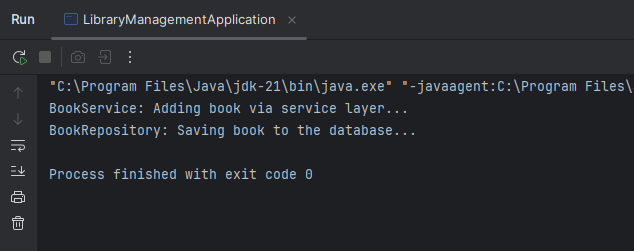
First we modify the existing “applicationContext.xml” to wire BookRepository into BookService

<?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  
 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>

Then we add a setter method for BookRepository in BookService class

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
  
 private BookRepository bookRepository;  
  
 // Added setter method  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook() {  
 System.*out*.println("BookService: Adding book via service layer...");  
 bookRepository.saveBook(); // calling the bookRepository method  
 }  
}

Output



Exercise 4 : Creating and configuring a Maven Project

**As given in step 1**, We already created a maven project named “LibraryManagementApplication”

**As for step 2**, We already have the Spring Context dependency additionally we add Spring AOP and Spring WebMVC dependency

Solution

<?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>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.29</version>  
 </dependency>  
  
 <!-- additionally we add Spring AOP dependency -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.29</version>  
 </dependency>  
  
 <!-- and Spring WebMVC dependency -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.29</version>  
 </dependency>  
 </dependencies>  
  
</project>

**Finally for step 3**, to configure maven compiler plugin for java 1.8 in pom.xml, **We have done it already** but in a different approach as :

