**Week-2: 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:   * 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.   **Project Structure:**  **Code:**  MainApp.java (Main Class): package com.library;  import com.library.service.BookService;  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 = (BookService) context.getBean("bookService");  bookService.addBook("The Great Gatsby");  }  }  BookRepository.java: package com.library.repository;  public class BookRepository {  public void saveBook(String title) {  System.out.println("Saving book: " + title);  }  }  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 addBook(String title) {  System.out.println("Adding book: " + title);  bookRepository.saveBook(title);  }  }  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">  <bean id="bookRepository" class="com.library.repository.BookRepository"/>  <bean id="bookService" class="com.library.service.BookService">  <property name="bookRepository" ref="bookRepository"/>  </bean>  </beans>  Pom.xml (With JUnit Dependency)  <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  https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.library</groupId>  <artifactId>LibraryManagement</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>5.3.33</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.36</version>  </dependency>  <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-classic</artifactId>  <version>1.2.11</version>  </dependency>  </dependencies>  <build>  </build>  </project>  **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.  **Steps:**   1. 1. Modify the XML Configuration:  * Update applicationContext.xml to wire BookRepository into BookService.  1. Update the BookService Class:  * Ensure that BookService class has a setter method for BookRepository.  1. Test the Configuration:  * Run the LibraryManagementApplication main class to verify the dependency injection.   **Project Structure**    **Code:**  MainApp.java  package com.library;  import com.library.service.BookService;  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  public class MainApp {  public static void main(String[] args) {  // Load Spring container from XML  ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  // Get the BookService bean from container  BookService bookService = (BookService) context.getBean("bookService");  // Use the service  bookService.addBook("The Alchemist");  }  }  BookRepository.java: package com.library.repository;  public class BookRepository {  public void saveBook(String title) {  System.out.println("Saving book: " + title);  }  }  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 addBook(String title) {  System.out.println("Adding book: " + title);  bookRepository.saveBook(title);  }  }  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">  <bean id="bookRepository" class="com.library.repository.BookRepository"/>  <bean id="bookService" class="com.library.service.BookService">  <property name="bookRepository" ref="bookRepository"/>  </bean>  </beans>  Pom.xml (With JUnit Dependency)  <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  https://maven.apache.org/xsd/maven-4.0.0.xsd">  <modelVersion>4.0.0</modelVersion>  <groupId>com.library</groupId>  <artifactId>LibraryManagement</artifactId>  <version>0.0.1-SNAPSHOT</version>  <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>5.3.33</version>  </dependency>  <dependency>  <groupId>org.slf4j</groupId>  <artifactId>slf4j-api</artifactId>  <version>1.7.36</version>  </dependency>  <dependency>  <groupId>ch.qos.logback</groupId>  <artifactId>logback-classic</artifactId>  <version>1.2.11</version>  </dependency>  </dependencies>  <build>  </build>  </project>  **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.  **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.   **Project Structure:**    **Code:**  LibraryManagementApplication.java  package com.example.library.app;  import com.example.library.service.BookService; // Import BookService  import org.springframework.context.ApplicationContext;  import org.springframework.context.support.ClassPathXmlApplicationContext;  public class LibraryManagementApplication {  public static void main(String[] args) {  // Load the Spring application context from the XML file  System.out.println("LibraryManagementApplication: Loading Spring Application Context...");  ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  System.out.println("LibraryManagementApplication: Spring Application Context loaded.");  // Retrieve the BookService bean from the context  System.out.println("LibraryManagementApplication: Retrieving BookService bean...");  BookService bookService = (BookService) context.getBean("bookService");  System.out.println("LibraryManagementApplication: BookService bean retrieved.");  // Use the BookService to perform an operation (which will internally use BookRepository)  bookService.addBook("The Adventures of Tom Sawyer");  // Close the context (important for proper shutdown in real applications)  ((ClassPathXmlApplicationContext) context).close();  }  }  BookRepository.java  package com.example.library.repository;  public class BookRepository {  public void saveBook(String bookTitle) {  System.out.println("BookRepository: Saving book - " + bookTitle);  }  }  BookService.java  package com.example.library.service;  import com.example.library.repository.BookRepository; // Import statement  public class BookService {  private BookRepository bookRepository;  // Setter method for BookRepository - Spring will use this for injection  public void setBookRepository(BookRepository bookRepository) {  System.out.println("BookService: setBookRepository called. Injecting BookRepository.");  this.bookRepository = bookRepository;  }  public void addBook(String title) {  System.out.println("BookService: Adding book - " + title);  if (bookRepository != null) {  bookRepository.saveBook(title);  } else {  System.out.println("BookService: BookRepository not injected!");  }  }  }  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.library.repository.BookRepository"/>  <bean id="bookService" class="com.example.library.service.BookService">  <property name="bookRepository" ref="bookRepository"/>  </bean>  </beans>  Pom.xml  <?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.example.library</groupId>  <artifactId>LibraryManagementApp</artifactId>  <version>1.0.0-SNAPSHOT</version>  <properties>  <maven.compiler.source>17</maven.compiler.source> <maven.compiler.target>17</maven.compiler.target> <spring.version>6.1.10</spring.version> </properties>  <dependencies>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-context</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-core</artifactId>  <version>${spring.version}</version>  </dependency>  <dependency>  <groupId>org.springframework</groupId>  <artifactId>spring-beans</artifactId>  <version>${spring.version}</version>  </dependency>  </dependencies>  </project>  **Output:** |
| --- |