**WEEK 3- Spring core and Moven**

**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.**

1. **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.**

1. **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.**

1. **Run the Application:**

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

**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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>SpringBasicApp</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.30</version>

</dependency>

</dependencies>

</project>

## beans.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="helloWorld" class="com.example.HelloWorld">

<property name="message" value="Hello Spring!" />

</bean>

</beans>

## HelloWorld.java

package spring;

public class HelloWorld {

private String message;

public void setMessage(String message) {

this.message = message;

}

public void getMessage() {

System.out.println("Your Message: " + message);

}

}

// File: src/main/java/com/example/MainApp.java

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainApp {

public static void main(String[] args) {

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

HelloWorld obj = (HelloWorld) context.getBean("helloWorld");

obj.getMessage();

}

}

## MainApp.java

<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</groupId>

<artifactId>SpringBasicApp</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

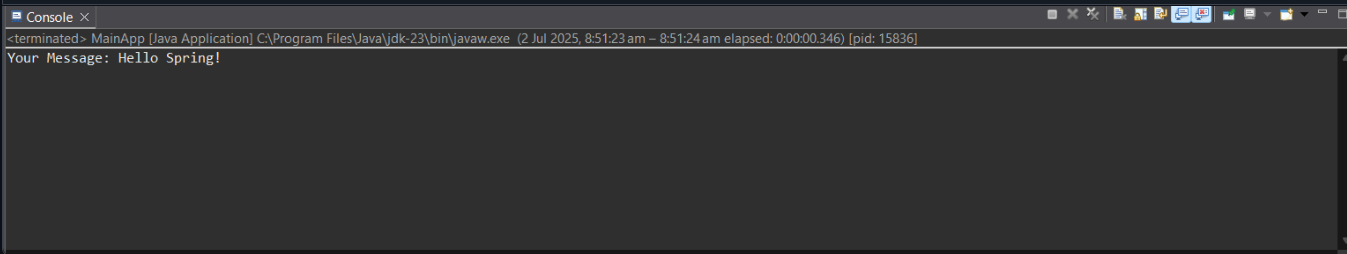
<version>5.3.30</version>

</dependency>

</dependencies>

</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. **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.**

**Code:**

### 1. EmailService.java

java

CopyEdit

package com.example;

public class EmailService {

public void sendEmail(String recipient, String message) {

System.out.println("Email sent to " + recipient + " with message: " + message);

}

}

### 2. NotificationService.java

java

CopyEdit

package com.example;

public class NotificationService {

private EmailService emailService;

// Setter injection

public void setEmailService(EmailService emailService) {

this.emailService = emailService;

}

public void notifyUser(String recipient, String message) {

emailService.sendEmail(recipient, message);

}

}

### 3. applicationContext.xml

xml

CopyEdit

<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="emailService" class="com.example.EmailService"/>

<bean id="notificationService" class="com.example.NotificationService">

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

</bean>

</beans>

**4. MainApp.java**

java

CopyEdit

package com.example;

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");

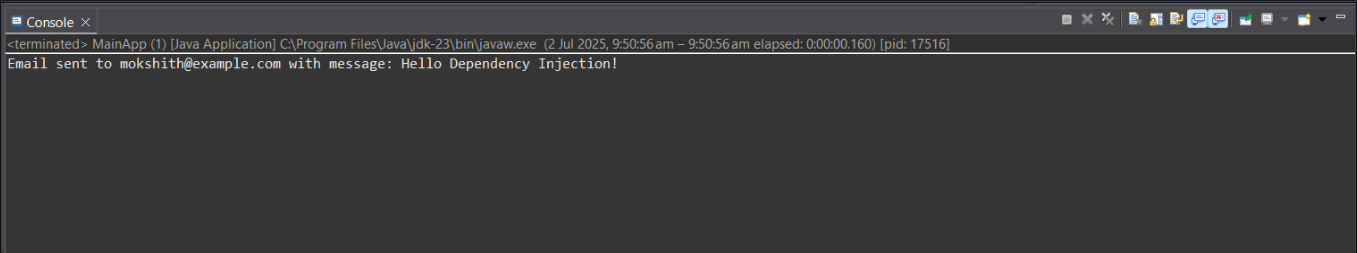
NotificationService notificationService = (NotificationService) context.getBean("notificationService");

notificationService.notifyUser("mokshith@example.com", "Hello Dependency Injection!");

}

}

**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.**

1. **Add Spring Dependencies in pom.xml:**

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

1. **Configure Maven Plugins:**

* **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/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>MavenHelloWorld</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

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

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

</properties>

</project>

**App.java**

package com.example.maven;

public class App {

public static void main(String[] args) {

System.out.println("Hello from Maven Project!");

}

}

**Output:**

