**Spring Core\_Maven**

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

**Problem statement**

Develop a basic Spring Core application using Maven that demonstrates the configuration of a simple Spring Bean using Java-based annotation configuration (@Configuration and @Bean). The application should display a simple message to the console using a managed Spring Bean.

**Requirements:**

1. Create a Maven project in Eclipse.
2. Add the required Spring Core dependency using Maven (spring-context).
3. Define a simple Java class MessageService with a method to print a message.
4. Create a Java-based configuration class AppConfig using @Configuration and @Bean to define the Spring bean.
5. Write a MainApp class to:
   * Load the Spring ApplicationContext
   * Retrieve the MessageService bean
   * Call its method to display the message

**CODING:**

**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>

<!-- Spring Core and Context -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

</dependencies>

</project>

**MessageService.java**

package com.example;

public class MessageService {

public void displayMessage() {

System.out.println("Hello, this is a basic Spring Application!");

}

}

**AppConfig.java**

package com.example;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public MessageService messageService() {

return new MessageService();

}

}

**MainApp.java**

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

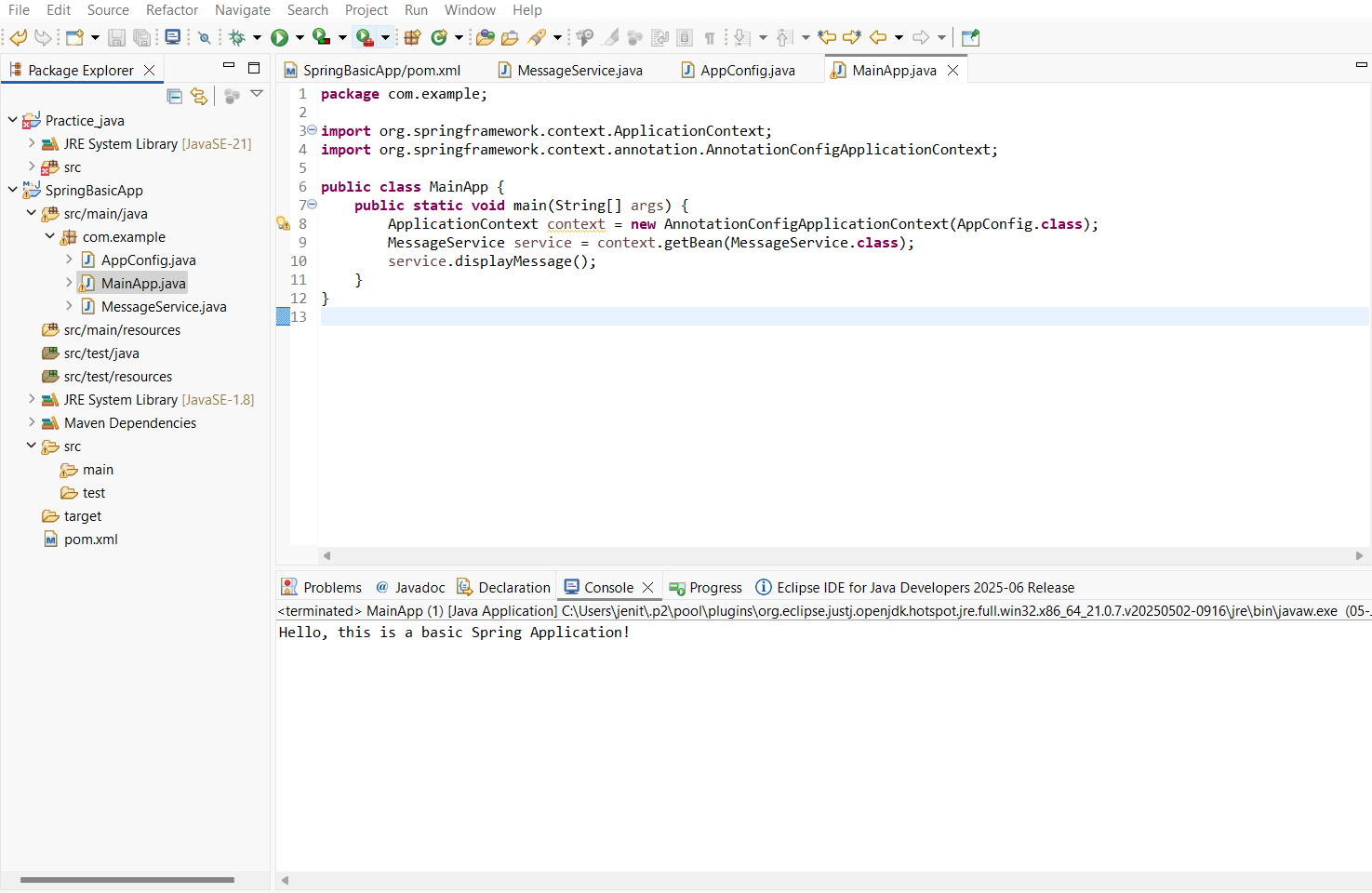
MessageService service = context.getBean(MessageService.class);

service.displayMessage();

}

}

**OUTPUT:**

****

**Exercise 2: Implementing Dependency Injection**

**Problem Statement**

Create a Spring Core application using Java-based configuration to implement Dependency Injection via constructor injection.  
Inject a Course object into a Student class and display the enrolled course.

**CODING:**

**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>SpringDIApp</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Context for Dependency Injection -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

</dependencies>

</project>

**Course.java**

package com.example;

public class Course {

public String getCourseName() {

return "Core Spring Framework";

}

}

**Student.java**

package com.example;

public class Student {

private Course course;

// Constructor Injection

public Student(Course course) {

this.course = course;

}

public void showCourse() {

System.out.println("Student is enrolled in: " + course.getCourseName());

}

}

**AppConfig.java**

package com.example;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

@Configuration

public class AppConfig {

@Bean

public Course course() {

return new Course();

}

@Bean

public Student student() {

return new Student(course()); // Constructor injection

}

}

**MainApp.java**

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

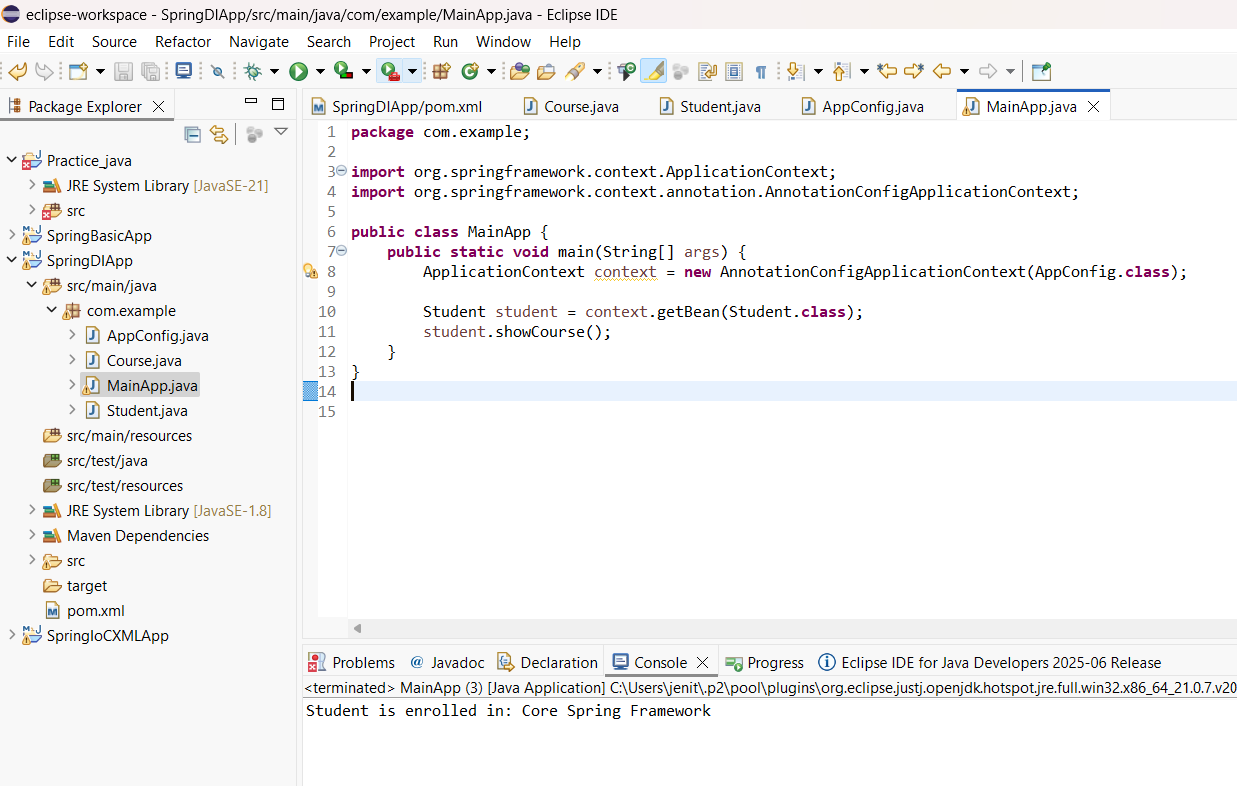
Student student = context.getBean(Student.class);

student.showCourse();

}

}

**OUTPUT:**

****

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

**Problem Statement**

Create a basic Maven Project in Eclipse IDE using the spring-context dependency.

Validate that the project builds and runs successfully with a simple Spring Core example.

**CODING:**

**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>SpringMavenDemo</artifactId>

<version>1.0-SNAPSHOT</version>

<dependencies>

<!-- Spring Core Context Dependency -->

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.3.32</version>

</dependency>

</dependencies>

</project>

**HelloService.java**

package com.example;

public class HelloService {

public String sayHello() {

return "Hello from Spring Maven Project!";

}

}

**MainApp.java**

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

HelloService helloService = context.getBean(HelloService.class);

System.out.println(helloService.sayHello());

}

}

@Configuration

class AppConfig {

@Bean

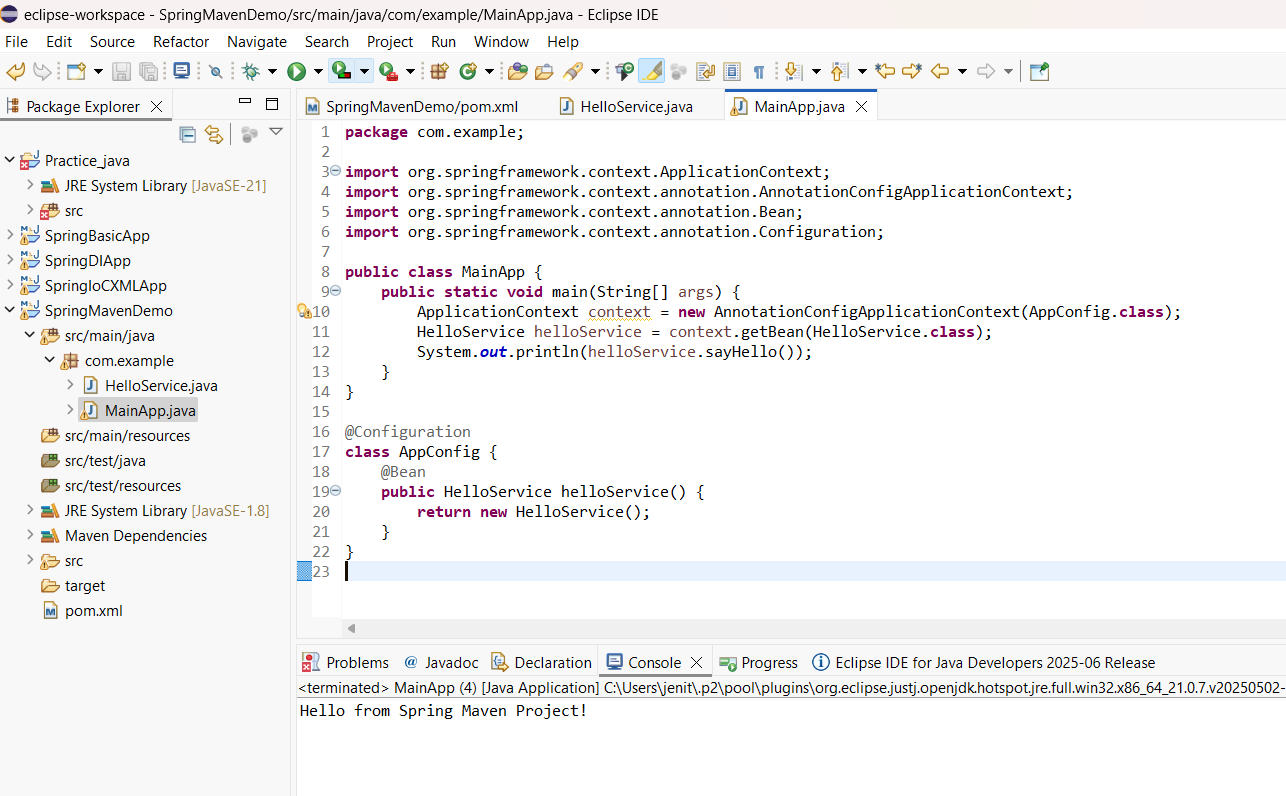
public HelloService helloService() {

return new HelloService();

}

}

**OUTPUT:**

****