# WEEK 3 ASSIGNMENTS:

## Exercise 1: Configuring a Basic Spring Application CODE:

**Hello.java**

package com.example.springcore; public class Hello {

public void sayHello() {

System.out.println("Hello from Spring!");

}

}

## AppConfig.java

package com.example.springcore;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration; @Configuration

public class AppConfig { @Bean

public Hello helloWorld() { return new Hello();

}

}

## App.java

package com.example.springcore;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext; public class App {

public static void main(String[] args) {

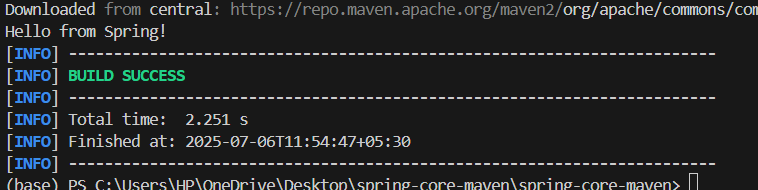
ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

Hello helloWorld = context.getBean(Hello.class); helloWorld.sayHello();

}

}

# OUTPUT:

****

## Exercise 2: Implementing Dependency Injection GreetingService.java

package com.example;

public interface GreetingService { void sayHello();

}

## GreetingServiceImpl.java

package com.example;

import org.springframework.stereotype.Service; @Service

public class GreetingServiceImpl implements GreetingService { @Override

public void sayHello() {

System.out.println("Hello from GreetingService!");

}

}

package com.example;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component;

@Component

public class GreetingComponent { @Autowired

private GreetingService greetingService; public void greet() {

greetingService.sayHello();

}

}

## AppConfig.java

package com.example;

import org.springframework.context.annotation.ComponentScan; import org.springframework.context.annotation.Configuration; @Configuration

@ComponentScan(basePackages = "com.example") public class AppConfig {

}

## App.java

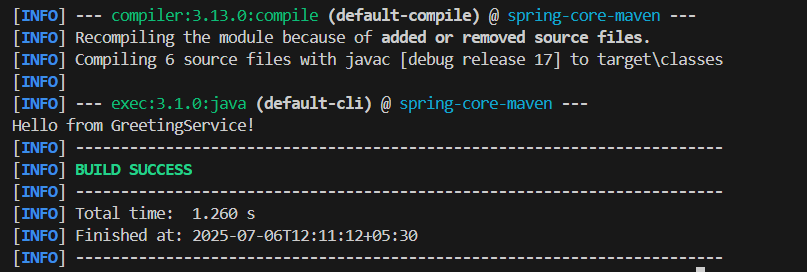
package com.example;

import org.springframework.context.ApplicationContext;

importorg.springframework.context.annotation.AnnotationConfigApplicationContext; public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);



## EXERCISE 4: Creating and Configuring a Maven Project GreetingService.java

package com.example;

public interface GreetingService { void sayHello();

}

## GreetingServiceImpl.java

package com.example;

import org.springframework.stereotype.Service; @Service

public class GreetingServiceImpl implements GreetingService { @Override

public void sayHello() {

System.out.println("Hello from GreetingService!");

}

}

package com.example;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component;

@Component

public class GreetingComponent { @Autowired

private GreetingService greetingService; public void greet() {

greetingService.sayHello();

}

}

## AppConfig.java

package com.example;

import org.springframework.context.annotation.ComponentScan; import org.springframework.context.annotation.Configuration; @Configuration

@ComponentScan(basePackages = "com.example") public class AppConfig {

}

## App.java

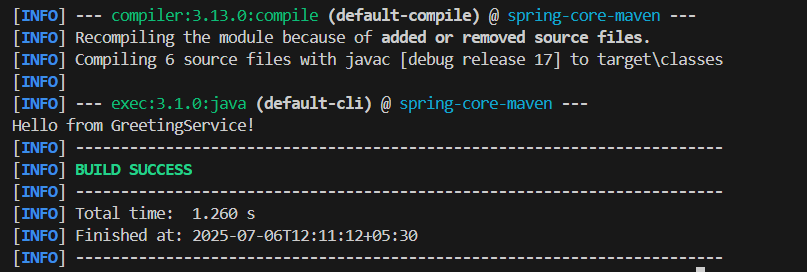
package com.example;

import org.springframework.context.ApplicationContext;

importorg.springframework.context.annotation.AnnotationConfigApplicationContext; public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);



## EXERCISE 5: Configuring the Spring IoC Container AppConfig.java

package com.example;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration; @Configuration

public class AppConfig {

@Bean

public GreetingService greetingService() { return new GreetingServiceImpl();

}

@Bean

public GreetingComponent greetingComponent() { return new GreetingComponent(greetingService());

}

}

## Modify GreetingComponent.java

package com.example;

public class GreetingComponent {

private GreetingService greetingService;

public GreetingComponent(GreetingService greetingService) { this.greetingService = greetingService;

}

public void greet() { greetingService.sayHello();

}

}

## MainApp.java

package com.example;

import org.springframework.context.ApplicationContext;

importorg.springframework.context.annotation.AnnotationConfigApplicationContext; public class MainApp {

public static void main(String[] args) {

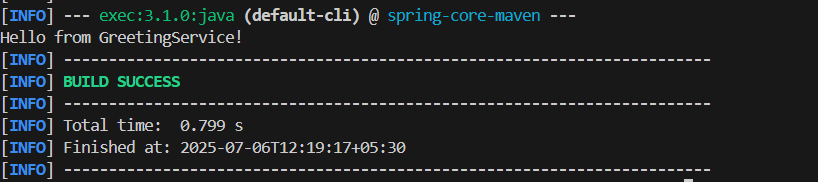
ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class);

GreetingComponent component = context.getBean(GreetingComponent.class); component.greet();

}

}

# OUTPUT:



## EXERCISE 7: Implementing Constructor and Setter Injection GreetingComponent.java

package com.example;

import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Component;

@Component

public class GreetingComponent {

private GreetingService greetingService;

// Constructor injection @Autowired

public GreetingComponent(GreetingService greetingService) { this.greetingService = greetingService;

}

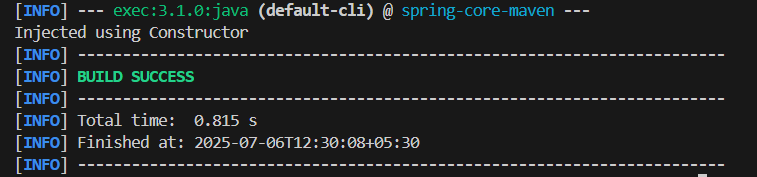
}

public void greet() { greetingService.sayHello();

}

}

# OUTPUT:



## EXERCISE G: Creating a Spring Boot Application DemoApplication.java

package com.example.springcore;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class DemoApplication {

public static void main(String[] args) {

SpringApplication.run(DemoApplication.class, args);

}

}

## GreetingController.java

package com.example.springcore;

import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RestController;

@RestController

public class GreetingController {

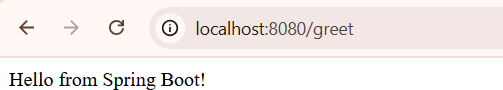
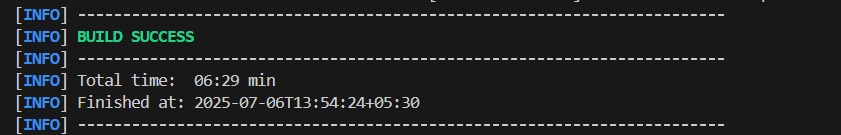
@GetMapping("/greet") public String greet() {

return "Hello from Spring Boot!";

}

}

**OUTPUT:**

****