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

Objective:

Create a basic Spring application that loads a bean from an XML configuration and calls its method.

**Project Structure:**

spring-basic-app/

├── src/

│ ├── App.java

│ ├── HelloService.java

│ └── beans.xml

**Step 1: Create the HelloService Bean**

**HelloService.java**

public class HelloService {

public void sayHello() {

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

}

}

**Step 2: Configure the Bean in beans.xml**

**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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="helloService" class="HelloService"/>

</beans>

**Step 3: Load the Bean in Main Class**

**App.java**

java

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

public static void main(String[] args) {

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

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

helloService.sayHello();

}

}

**Sample Output**

Hello from Spring!

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

**Project Structure**

maven-hello-world/

├── src/

│ └── main/

│ └── java/

│ └── com/

│ └── example/

│ └── App.java

├── pom.xml

**Step 1: Create 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>maven-hello-world</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

</project>

**Step 2: Create Main Class**

**App.java**

package com.example;

public class App {

public static void main(String[] args) {

System.out.println("Maven Project Working!");

}

}

**Build and Run Commands**

mvn compile

mvn exec:java -Dexec.mainClass="com.example.App"

If exec-maven-plugin is not set, add this to pom.xml under <build>:

xml

CopyEdit

<build>

<plugins>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>exec-maven-plugin</artifactId>

<version>3.1.0</version>

<executions>

<execution>

<goals>

<goal>java</goal>

</goals>

</execution>

</executions>

</plugin>

</plugins>

</build>

**Output**

Nginx

Maven Project Working!

**Exercise 2: Implementing Dependency Injection**

**Project Structure**

spring-di-example/

├── src/

│ ├── MainApp.java

│ ├── TextEditor.java

│ ├── SpellChecker.java

│ └── beans.xml

**Step 1: SpellChecker.java**

java

CopyEdit

public class SpellChecker {

public SpellChecker() {

System.out.println("SpellChecker constructor called.");

}

public void checkSpelling() {

System.out.println("Checking spelling...");

}

}

**Step 2: TextEditor.java**

public class TextEditor {

private SpellChecker spellChecker;

// Constructor-based injection

public TextEditor(SpellChecker spellChecker) {

this.spellChecker = spellChecker;

}

public void spellCheck() {

spellChecker.checkSpelling();

}

}

**Step 3: 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

https://www.springframework.org/schema/beans/spring-beans.xsd">

<bean id="spellChecker" class="SpellChecker" />

<bean id="textEditor" class="TextEditor">

<constructor-arg ref="spellChecker" />

</bean>

</beans>

**Step 4: MainApp.java**

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");

TextEditor editor = (TextEditor) context.getBean("textEditor");

editor.spellCheck();

}

}

**Sample Output**

SpellChecker constructor called.

Checking spelling...