Dependency Injection (DI)

Describe the concept of Dependency Injection (DI) in Spring. Explain how to perform constructor-based and setter-based DI using XML configuration with examples.

Dependency Injection (DI) is a design pattern used in Spring to manage the dependencies between objects. It allows for loose coupling by injecting dependencies from the outside rather than creating them within the objects themselves. Spring's DI can be achieved through constructor-based injection or setter-based injection. Let's explore both methods using XML configuration.

**1. Constructor-Based Dependency Injection**

In constructor-based DI, the dependencies are provided through the class constructor. Spring's container uses the constructor to initialize the beans.

Example:

SpellChecker.java

public class SpellChecker {

public SpellChecker() {

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

}

public void checkSpelling() {

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

}

}

TextEditor.java

public class TextEditor {

private SpellChecker spellChecker;

public TextEditor(SpellChecker spellChecker) {

System.out.println("TextEditor constructor");

this.spellChecker = spellChecker;

}

public void spellCheck() {

spellChecker.checkSpelling();

}

}

beans.xml

<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">

<!-- Definition for spellChecker bean -->

<bean id="spellChecker" class="com.example.SpellChecker"/>

<!-- Definition for textEditor bean -->

<bean id="textEditor" class="com.example.TextEditor">

<constructor-arg ref="spellChecker"/>

</bean>

</beans>

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

te.spellCheck();

}

}

### 2. Setter-Based Dependency Injection

In setter-based DI, the dependencies are provided through setter methods of the class.

#### Example:

**TextEditor.java**

public class TextEditor {

private SpellChecker spellChecker;

public void setSpellChecker(SpellChecker spellChecker) {

this.spellChecker = spellChecker;

}

public void spellCheck() {

spellChecker.checkSpelling();

}

}

beans.xml

<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">

<!-- Definition for spellChecker bean -->

<bean id="spellChecker" class="com.example.SpellChecker"/>

<!-- Definition for textEditor bean -->

<bean id="textEditor" class="com.example.TextEditor">

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

</bean>

</beans>