**Spring 基于设值函数的依赖注入**

项目地址：https://github.com/xuegang20/mystudyrecode/tree/main/code/SpringExample

**Spring 基于设值函数的依赖注入**

当容器调用一个无参的构造函数或一个无参的静态 factory 方法来初始化你的 bean 后，通过容器在你的 bean 上调用设值函数，基于设值函数的 DI 就完成了。

**示例：**

下述例子显示了一个类 *TextEditor*，它只能使用纯粹的基于设值函数的注入来实现依赖注入。

下面是 **TextEditor.java** 文件的内容：

package com.tutorialspoint;

public class TextEditor {

private SpellChecker spellChecker;

// a setter method to inject the dependency.

public void setSpellChecker(SpellChecker spellChecker) {

System.out.println("Inside setSpellChecker." );

this.spellChecker = spellChecker;

}

// a getter method to return spellChecker

public SpellChecker getSpellChecker() {

return spellChecker;

}

public void spellCheck() {

spellChecker.checkSpelling();

}

}

在这里，你需要检查设值函数方法的名称转换。要设置一个变量 **spellChecker**，我们使用 **setSpellChecker()** 方法，该方法与 Java POJO 类非常相似。让我们创建另一个依赖类文件 **SpellChecker.java** 的内容：

package com.tutorialspoint;

public class SpellChecker {

public SpellChecker(){

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

}

public void checkSpelling() {

System.out.println("Inside checkSpelling." );

}

}

以下是 **MainApp.java** 文件的内容：

package com.tutorialspoint;

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();

}

}

下面是配置文件 **Beans.xml** 的内容，该文件有基于设值函数注入的配置：

<?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-3.0.xsd">

<!-- Definition for textEditor bean -->

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

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

</bean>

<!-- Definition for spellChecker bean -->

<bean id="spellChecker" class="com.tutorialspoint.SpellChecker">

</bean>

</beans>

你应该注意定义在基于构造函数注入和基于设值函数注入中的 Beans.xml 文件的区别。唯一的区别就是在基于构造函数注入中，我们使用的是〈bean〉标签中的〈constructor-arg〉元素，而在基于设值函数的注入中，我们使用的是〈bean〉标签中的〈property〉元素。

第二个你需要注意的点是，如果你要把一个引用传递给一个对象，那么你需要使用 标签的 **ref** 属性，而如果你要直接传递一个值，那么你应该使用 **value** 属性。

当你完成了创建源和 bean 配置文件后，让我们开始运行应用程序。如果你的应用程序运行顺利的话，那么将会输出下述所示消息：

Inside SpellChecker constructor.

Inside setSpellChecker.

Inside checkSpelling.

**使用 p-namespace 实现 XML 配置：**

如果你有许多的设值函数方法，那么在 XML 配置文件中使用 **p-namespace** 是非常方便的。让我们查看一下区别：

以带有 标签的标准 XML 配置文件为例：

<?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-3.0.xsd">

<bean id="john-classic" class="com.example.Person">

<property name="name" value="John Doe"/>

<property name="spouse" ref="jane"/>

</bean>

<bean name="jane" class="com.example.Person">

<property name="name" value="John Doe"/>

</bean>

</beans>

上述 XML 配置文件可以使用 **p-namespace** 以一种更简洁的方式重写，如下所示：

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

<bean id="john-classic" class="com.example.Person"

p:name="John Doe"

p:spouse-ref="jane"/>

</bean>

<bean name="jane" class="com.example.Person"

p:name="John Doe"/>

</bean>

</beans>

在这里，你不应该区别指定原始值和带有 p-namespace 的对象引用。**-ref** 部分表明这不是一个直接的值，而是对另一个 bean 的引用。