**Spring Bean 定义继承**

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

**Bean 定义继承**

bean 定义可以包含很多的配置信息，包括构造函数的参数，属性值，容器的具体信息例如初始化方法，静态工厂方法名，等等。

子 bean 的定义继承父定义的配置数据。子定义可以根据需要重写一些值，或者添加其他值。

Spring Bean 定义的继承与 Java 类的继承无关，但是继承的概念是一样的。你可以定义一个父 bean 的定义作为模板和其他子 bean 就可以从父 bean 中继承所需的配置。

当你使用基于 XML 的配置元数据时，通过使用父属性，指定父 bean 作为该属性的值来表明子 bean 的定义。

**例子**

下面是配置文件 **Beans.xml**，在该配置文件中我们定义有两个属性 *message1* 和 *message2* 的 “helloWorld” bean。然后，使用 **parent** 属性把 “helloIndia” bean 定义为 “helloWorld” bean 的孩子。这个子 bean 继承 *message2* 的属性，重写 *message1* 的属性，并且引入一个属性 *message3*。

<?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="helloWorld" class="com.tutorialspoint.HelloWorld">

<property name="message1" value="Hello World!"/>

<property name="message2" value="Hello Second World!"/>

</bean>

<bean id="helloIndia" class="com.tutorialspoint.HelloIndia" parent="helloWorld">

<property name="message1" value="Hello India!"/>

<property name="message3" value="Namaste India!"/>

</bean>

</beans>

这里是 **HelloWorld.java** 文件的内容：

package com.tutorialspoint;

public class HelloWorld {

private String message1;

private String message2;

public void setMessage1(String message){

this.message1 = message;

}

public void setMessage2(String message){

this.message2 = message;

}

public void getMessage1(){

System.out.println("World Message1 : " + message1);

}

public void getMessage2(){

System.out.println("World Message2 : " + message2);

}

}

这里是 **HelloIndia.java** 文件的内容：

package com.tutorialspoint;

public class HelloIndia {

private String message1;

private String message2;

private String message3;

public void setMessage1(String message){

this.message1 = message;

}

public void setMessage2(String message){

this.message2 = message;

}

public void setMessage3(String message){

this.message3 = message;

}

public void getMessage1(){

System.out.println("India Message1 : " + message1);

}

public void getMessage2(){

System.out.println("India Message2 : " + message2);

}

public void getMessage3(){

System.out.println("India Message3 : " + message3);

}

}

下面是 **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");

HelloWorld objA = (HelloWorld) context.getBean("helloWorld");

objA.getMessage1();

objA.getMessage2();

HelloIndia objB = (HelloIndia) context.getBean("helloIndia");

objB.getMessage1();

objB.getMessage2();

objB.getMessage3();

}

}

一旦你创建源代码和 bean 配置文件完成后，我们就可以运行该应用程序。如果你的应用程序一切都正常，将输出以下信息：

World Message1 : Hello World!

World Message2 : Hello Second World!

India Message1 : Hello India!

India Message2 : Hello Second World!

India Message3 : Namaste India!

在这里你可以观察到，我们创建 “helloIndia” bean 的同时并没有传递 message2，但是由于 Bean 定义的继承，所以它传递了 message2。

**Bean 定义模板**

你可以创建一个 Bean 定义模板，不需要花太多功夫它就可以被其他子 bean 定义使用。在定义一个 Bean 定义模板时，你不应该指定**类**的属性，而应该指定带 **true** 值的**抽象**属性，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"

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

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

<bean id="beanTeamplate" abstract="true">

<property name="message1" value="Hello World!"/>

<property name="message2" value="Hello Second World!"/>

<property name="message3" value="Namaste India!"/>

</bean>

<bean id="helloIndia" class="com.tutorialspoint.HelloIndia" parent="beanTeamplate">

<property name="message1" value="Hello India!"/>

<property name="message3" value="Namaste India!"/>

</bean>

</beans>

父 bean 自身不能被实例化，因为它是不完整的，而且它也被明确地标记为抽象的。当一个定义是抽象的，它仅仅作为一个纯粹的模板 bean 定义来使用的，充当子定义的父定义使用。