# Spring 注入集合

## 注入集合

你已经看到了如何使用 **value** 属性来配置基本数据类型和在你的 bean 配置文件中使用<property>标签的 **ref** 属性来配置对象引用。这两种情况下处理奇异值传递给一个 bean。

现在如果你想传递多个值，如 Java Collection 类型 List、Set、Map 和 Properties，应该怎么做呢。为了处理这种情况，Spring 提供了四种类型的集合的配置元素，如下所示：

|  |  |
| --- | --- |
| **元素** | **描述** |
| <list> | 它有助于连线，如注入一列值，允许重复。 |
| <set> | 它有助于连线一组值，但不能重复。 |
| <map> | 它可以用来注入名称-值对的集合，其中名称和值可以是任何类型。 |
| <props> | 它可以用来注入名称-值对的集合，其中名称和值都是字符串类型。 |

你可以使用<list>或<set>来连接任何 java.util.Collection 的实现或数组。

你会遇到两种情况（a）传递集合中直接的值（b）传递一个 bean 的引用作为集合的元素。

## 例子

这里是 **JavaCollection.java** 文件的内容：

package com.tutorialspoint;

import java.util.\*;

public class JavaCollection {

List addressList;

Set addressSet;

Map addressMap;

Properties addressProp;

// a setter method to set List

public void setAddressList(List addressList) {

this.addressList = addressList;

}

// prints and returns all the elements of the list.

public List getAddressList() {

System.out.println("List Elements :" + addressList);

return addressList;

}

// a setter method to set Set

public void setAddressSet(Set addressSet) {

this.addressSet = addressSet;

}

// prints and returns all the elements of the Set.

public Set getAddressSet() {

System.out.println("Set Elements :" + addressSet);

return addressSet;

}

// a setter method to set Map

public void setAddressMap(Map addressMap) {

this.addressMap = addressMap;

}

// prints and returns all the elements of the Map.

public Map getAddressMap() {

System.out.println("Map Elements :" + addressMap);

return addressMap;

}

// a setter method to set Property

public void setAddressProp(Properties addressProp) {

this.addressProp = addressProp;

}

// prints and returns all the elements of the Property.

public Properties getAddressProp() {

System.out.println("Property Elements :" + addressProp);

return addressProp;

}

}

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");

JavaCollection jc=(JavaCollection)context.getBean("javaCollection");

jc.getAddressList();

jc.getAddressSet();

jc.getAddressMap();

jc.getAddressProp();

}

}

下面是配置所有类型的集合的配置文件 **Beans.xml** 文件：

<?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 javaCollection -->

<bean id="javaCollection" class="com.tutorialspoint.JavaCollection">

<!-- results in a setAddressList(java.util.List) call -->

<property name="addressList">

<list>

<value>INDIA</value>

<value>Pakistan</value>

<value>USA</value>

<value>USA</value>

</list>

</property>

<!-- results in a setAddressSet(java.util.Set) call -->

<property name="addressSet">

<set>

<value>INDIA</value>

<value>Pakistan</value>

<value>USA</value>

<value>USA</value>

</set>

</property>

<!-- results in a setAddressMap(java.util.Map) call -->

<property name="addressMap">

<map>

<entry key="1" value="INDIA"/>

<entry key="2" value="Pakistan"/>

<entry key="3" value="USA"/>

<entry key="4" value="USA"/>

</map>

</property>

<!-- results in a setAddressProp(java.util.Properties) call -->

<property name="addressProp">

<props>

<prop key="one">INDIA</prop>

<prop key="two">Pakistan</prop>

<prop key="three">USA</prop>

<prop key="four">USA</prop>

</props>

</property>

</bean>

</beans>

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。你应该注意这里不需要配置文件。如果你的应用程序一切都正常，将输出以下信息：

List Elements :[INDIA, Pakistan, USA, USA]

Set Elements :[INDIA, Pakistan, USA]

Map Elements :{1=INDIA, 2=Pakistan, 3=USA, 4=USA}

Property Elements :{two=Pakistan, one=INDIA, three=USA, four=USA}

## 注入 Bean 引用

下面的 Bean 定义将帮助你理解如何注入 bean 的引用作为集合的元素。甚至你可以将引用和值混合在一起，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 Definition to handle references and values -->

<bean id="..." class="...">

<!-- Passing bean reference for java.util.List -->

<property name="addressList">

<list>

<ref bean="address1"/>

<ref bean="address2"/>

<value>Pakistan</value>

</list>

</property>

<!-- Passing bean reference for java.util.Set -->

<property name="addressSet">

<set>

<ref bean="address1"/>

<ref bean="address2"/>

<value>Pakistan</value>

</set>

</property>

<!-- Passing bean reference for java.util.Map -->

<property name="addressMap">

<map>

<entry key="one" value="INDIA"/>

<entry key ="two" value-ref="address1"/>

<entry key ="three" value-ref="address2"/>

</map>

</property>

</bean>

</beans>

为了使用上面的 bean 定义，你需要定义 setter 方法，它们应该也能够是用这种方式来处理引用。

## 注入 null 和空字符串的值

如果你需要传递一个空字符串作为值，那么你可以传递它，如下所示：

<bean id="..." class="exampleBean">

<property name="email" value=""/>

</bean>

前面的例子相当于 Java 代码：exampleBean.setEmail("")。

如果你需要传递一个 NULL 值，那么你可以传递它，如下所示：

<bean id="..." class="exampleBean">

<property name="email"><null/></property>

</bean>

前面的例子相当于 Java 代码：exampleBean.setEmail(null)。