BEAN INHERITANCE

SpringTutors January 10, 2016 Features 89 Views

Bean Inheritance

Reusing the features of one bean to another bean is known as bean inheritance.

Bean which is inherited is called as parent bean and bean which is inheriting is called as child bean.

When to use bean inheritance

Let's suppose there is a Car showroom of Maruti Suzuki in which only one model of car with similar features are available

```
class Car
 {
 private int id;
 private String model;
 private String manufacturer;
 private String fuelType;
 private String colour;
 public void setId(int id)
 {
 this.id=id;
 }
 public void setModel(String model)
 {
 this.model=model;
 }
 public void setManufacturer(String manufacturer)
 {
 this.manufacturer=manufacturer;
 }
 public void setFuelType(String fuelType)
 {
 this.fuelType=fuelType;
 }
 public void setColour(String colour)
 {
 this.colour=colour;
 }
 }
```

Configuring the above class

```
<bean id="car1" class="Car">
 <property name="id" value="10"/>
 <property name="model" value="swift"/>
 <property name="manufacturer" value="Maruti Suzuki"/>
 <property name="fuelType" value="Disele"/>
 <property name="colour" value="Red"/>
```

```
<bean id="car2" class="Car">
 <property name="id" value="20"/>
 <property name="model" value="swift"/><property name="manufacturer" value="Maruti Suzuki"/>
 <property name="fuelType" value="Disele"/>
 <property name="colour" value="Red"/>
```

In the above configuration we can see that whatever be the configuration properties are in car1 all are required to car2 except id.

If we are writing the same configuration properties in different-different bean then it leads to duplication of code which cause many problems like maintenance, hard to read and understand etc.

To solve this problem Bean Inheritance comes into the picture

It gives us the facility of reusing the configuration properties by inheriting the features from parent to child.

How does it Work

1. **Setter:-**

   Spring has given an attribute called "parent". It is used on < bean> tag level to those beans which requires the features of another bean.

   In the above bean configuration car2 requires the configuration

   properties of car1 then car2 has to write parent="car1" at car2 bean tag level like following-

   <bean id="car2" class="Car" parent="car1"/>

   When IOC container find the bean definition of car2 it sees the parent

   attribute then it goes to the parent bean car1 and copy all the configuration properties from there and comes back to child bean car2 and placed all that configuration properties in child. In this way it work in case of setter when all the properties are required by child from parent.

   But when the child wants the same property as of parent with different value then child has to write that property configuration explicitly like following-

In the above bean car2 wants the id property with different id value then it has to write id property explicitly-

```
<bean id="car2" class="Car" parent="car1">
 <property name="id" value="20"/>
 </bean>
```

In this case IOC will go to parent but it will not copy the id property because it is already defined in child. We can say that id property is overridden by child but actually it is not copied from parent and rest of the configuration properties are copied and placed in child as usual.

Based on property name IOC container will copy the bean from parent to child if same property name which are defined in child are found in

parent then that configuration property will not be copied.

2.Constructor:-

```
class Car
{
```

```
private int id;
private String model;
 private String manufacturer;
 private String fuelType;
 private String colour;
 public Car(int id, String model, String manufacturer, String fuelType, String colour)
{
this.id=id;
this.model=model;
this.manufacturer=manufacturer;
this.fuelType=fuelType;
this.colour=colour;
}
 }
```

Bean configuration:-

```
<bean id="car1" class="Car">
<constructor-arg value="10"/>
<constructor-arg value="swift"/>
<constructor-arg value="Maruti-Suziki"/>
<constructor-arg value="Disele"/>
<constructor-arg value="red"/>
</bean>
```

```
<bean id="car2" class="Car">
 <constructor-arg value="20"/>
 <constructor-arg value="swift"/>
 <constructor-arg value="Maruti-Suziki"/>
 <constructor-arg value="Disele"/>
 <constructor-arg value="red"/>
 </bean>
```

Now through constructor also if one bean requires all the features of another bean then we will use bean inheritance concept by using
"parent" attribute at <bean> tag level like following-
<bean id="car2" class="Car" parent="car1"/>
In this case IOC container will first see the parent attribute in bean definition and go to parent bean and go to respective bean class and find the injection type if it is constructor injection it will copy all the configuration properties from parent bean and placed it into child bean. In this case if any extra property or any existing property value is required is different then it will not work.
In simple word we can say that overriding of properties configuration in case of constructor injection is not possible by default.

```
<bean id="car2" class="Car" parent="car1">
 <constructor-arg value="20"/>
 </bean>
```

The above bean configuration will not work because we are trying to override the existing parent bean id value using constructor injection.
In this case IOC container will not decide which type of value is going to be override. So IOC container will consider it as an extra argument for constructor but in our original class we have limited argument so it will raise exception.

This problem can be solved by using two attribute-

1.”name”

2.”index”

```
<bean id=”car1” class=”Car”>
<constructor-arg value=”10” name=”id”/>
</bean>
<bean id=”car2” class=”Car” parent=”car1”>
<constructor-arg value=”20” name=”id”/>
</bean>
```

Or

```
<bean id=”car1” class=”Car”>
<constructor-arg value=”20” index=”0”/>
</bean>
<bean id=”car2” class=”Car” parent=”car1”>
<constructor-arg value=”20” index=”0”/>
</bean>
```

By using above any one configuration method in case of constructor injection of bean inheritance we can override the existing parent bean value.