# Spring Inheritance

Spring supports Inheritance, So as you guess by the inheritance we can inherit the properties from parent bean. By Spring inheritance any child bean can inherit parent properties but don’t mix this inheritance with java inheritance. Spring inheritance is totally different than Java inheritance only thing is common between them is both inherits property from Parent bean.

**The main difference between java and Spring inheritance,**

Let try to understand by a problem,

Suppose I have a Java class Basic Watch which just shows time and date. Now If I want to create a SmartWatch which will show same date and time and on top of it, we can access the internet in it.

So in java, we just extend Basic Watch and add the internet Access mechanism to Smartwatch in order to satisfy the requirement. Very easy by java inheritance.

**Things to notice here in Smartwatch we don’t redefine the property date, it inherits from the parent.**

**Solution:**

**BasicWatch.java**

package com.example.inheritence;

import java.util.Date;

public class BasicWatch {

Date date=new Date();

public void display()

{

System.out.println(this.getClass().getName());

System.out.println(date);

System.out.println(date.getHours()+":"+ date.getMinutes());

}

public static void main(String[] args) {

BasicWatch watch = new BasicWatch();

BasicWatch swatch = new SmartWatch();

watch.display();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

swatch.display();

}

}

**SmartWatch.java**

package com.example.inheritence;

public class SmartWatch extends BasicWatch{

private String internetAccess = "Support internet!!!!";

@Override

public void display()

{

super.display();

System.out.println(internetAccess);

}

}

**Output:**

com.example.inheritence.BasicWatch

Mon Oct 24 20:22:03 IST 2016

20:22

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

com.example.inheritence.SmartWatch

Mon Oct 24 20:22:03 IST 2016

20:22

Support internet!!!!

Now Coming to Spring inheritance,

To achieve the solution by Spring bean we need to define two beans in Spring config file and SmartWatch has to declare a property ,

**parent=”<id of BasicWatch bean> *to indicate BasicWatch is the parent of SmartWatch.***

Now try to modify the above java solution to fits in Spring.

1. Remove extends keyword as now Spring takes care of inheritance.
2. Define two beans definition in Spring config file.
3. As we remove the extends keyword compiler complain about super.display() method as there is no such method in Object. So I comment it out.
4. In Basic watch class, in main method, compiler complain about polymorphic assignment

BasicWatch sw = new SmartWatch(); as now BasicWatch is not java parent of Smart watch rather by Spring, it is the parent of Smartwatch, so comment it now and try to run the new solution.

**Solution:**

**BasicWatch.java**

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.spring.ApplicationContext.HelloWorldUsingContext;

public class BasicWatch {

Date date=new Date();

public void display()

{

System.out.println(this.getClass().getName());

System.out.printlpackage com.example.inheritence;n(date);

System.out.println(date.getHours()+":"+ date.getMinutes());

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public static void main(String[] args) {

ApplicationContext ctx =

new ClassPathXmlApplicationContext("configFiles/SpringInheritance.xml");

SmartWatch aWatch =(SmartWatch) ctx.getBean("advWatch");

aWatch.display();

BasicWatch bWatch =(BasicWatch) ctx.getBean("basicWatch");

bWatch.display();

}

}

**SmartWatch.java**

package com.example.inheritence;

public class SmartWatch{ //extends BasicWatch{

private String internetAccess = "Support internet!!!!";

//@Override

public void display()

{

//super.display();

System.out.println(internetAccess);

}

public String getInternetAccess() {

return internetAccess;

}

public void setInternetAccess(String internetAccess) {

this.internetAccess = internetAccess;

}

}

**SpringInheritance.xml**

<?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="basicWatch" class="com.example.inheritence.BasicWatch">

<property name="date" >

<bean class="java.util.Date" />

</property>

</bean>

<bean id="advWatch" class="com.example.inheritence.SmartWatch" parent="basicWatch">

<property name="internetAccess" value="Support internet!!!!"/>

</bean>

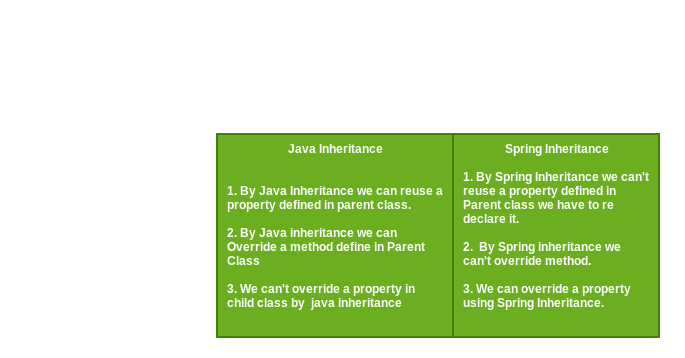
</beans>

**Output :**

org.springframework.beans.factory.BeanCreationException: Error creating bean with name 'advWatch' defined in class path resource [configFiles/SpringInheritance.xml]: Error setting property values; nested exception is org.springframework.beans.NotWritablePropertyException: Invalid property 'date' of bean class [com.example.inheritence.SmartWatch]: Bean property 'date' is not writable or has an invalid setter method. Does the parameter type of the setter match the return type of the getter?

Oops , It gives us an error saying that SmartWatch should have a property date. But we know by Java inheritance once we declare a property in Parent class we do not redefine it in Child class but in Spring Inheritance it is not working like that to fix it we have to redeclare the property.

So, From above example,we can find following difference between java Inheritance and Spring Inheritance.



**So, Why should we use Spring Inheritance?**

Actually, the power of Spring inheritance is ,we can find common attributes among the beans and make them as a template so that all bean can use the template without declaring the same property in every bean and if one bean wants to override the property value it has provision to do so.

By our example, the common propeties between BasicWatch and Smart Watch is the date property as both shows date and time, so we can create a template bean which defines the date property and BasicWatch and SmartWatch declare this template bean as a parent bean and use that property.

Template bean contains common attributes among all beans and it is abstract so Spring can’t create an instance for it. To achieve this we have to make **abstract=”true”** for that bean

**Solution** :

**SpringInheritance.xml**

<?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="templateWatch" class="com.example.inheritence.TemplateWatch" abstract="true">

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

<property name="date" >

<bean class="java.util.Date" />

</property>

</bean>

<bean id="basicWatch" class="com.example.inheritence.BasicWatch" parent="templateWatch"/>

<bean id="advWatch" class="com.example.inheritence.SmartWatch" parent="templateWatch">

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

<property name="internetAccess" value="Support internet!!!!"/>

</bean>

</beans>

**TemplateWatch.java**

package com.example.inheritence;

import java.util.Date;

public class TemplateWatch {

private Date date;

private String name;

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**BasicWatch.java**

package com.example.inheritence;

import java.util.Date;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.spring.ApplicationContext.HelloWorldUsingContext;

public class BasicWatch {

private Date date;

private String name;

public void display()

{

System.out.println(name);

System.out.println(date);

System.out.println(date.getHours()+":"+ date.getMinutes());

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public static void main(String[] args) {

ApplicationContext ctx =

new ClassPathXmlApplicationContext("configFiles/SpringInheritance.xml");

SmartWatch aWatch =(SmartWatch) ctx.getBean("advWatch");

aWatch.display();

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

BasicWatch bWatch =(BasicWatch) ctx.getBean("basicWatch");

bWatch.display();

}

}

**SmartWatch.java**

package com.example.inheritence;

import java.util.Date;

public class SmartWatch{

private Date date;

private String name;

private String internetAccess = "Support internet!!!!";

public void display()

{

System.out.println(name);

System.out.println(date);

System.out.println(date.getHours()+":"+ date.getMinutes());

System.out.println(internetAccess);

}

public String getInternetAccess() {

return internetAccess;

}

public void setInternetAccess(String internetAccess) {

this.internetAccess = internetAccess;

}

public Date getDate() {

return date;

}

public void setDate(Date date) {

this.date = date;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**Output**:

SmartWatch

Tue Oct 25 12:13:01 IST 2016

12:13

Support internet!!!!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*

BasicWatch

Tue Oct 25 12:13:01 IST 2016

12:13

In this example, TemplateWatch contains two properties name and Date which are common among Smart and Basic watch. In SmartWatch we override the name property and give a new value “SmartWatch”.