Dependency Injection by setter method

We can inject the dependency by setter method also. The **<property>**subelement of **<bean>** is used for setter injection. Here we are going to inject

1. primitive and String-based values
2. Dependent object (contained object)
3. Collection values etc.

### Injecting primitive and string-based values by setter method

Let’s see the simple example to inject primitive and string-based values by setter method. We have created three files here:

Employee.java

applicationContext.xml

Test.java

* **package** com.pioneer;
* **public** **class** Employee {
* **private** **int** id;
* **private** String name;
* **private** String city;
* **public** **int** getId() {
* **return** id;
* }
* **public** **void** setId(**int** id) {
* **this**.id = id;
* }
* **public** String getName() {
* **return** name;
* }
* **public** **void** setName(String name) {
* **this**.name = name;
* }
* **public** String getCity() {
* **return** city;
* }
* **public** **void** setCity(String city) {
* **this**.city = city;
* }
* **void** display(){
* System.out.println(id+" "+name+" "+city);
* }
* }

**applicationContext.xml**

We are providing the information into the bean by this file. The property element invokes the setter method. The value subelement of property will assign the specified value.

<?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="obj" **class**="com.pioneer.Employee">

<property name="id">

<value>10</value>

</property>

<property name="name">

<value>pioneer</value>

</property>

<property name="city">

<value>coders</value>

</property>

</bean>

</beans>

**Test.java**

This class gets the bean from the applicationContext.xml file and calls the display method.

**package** com.pioneer;

**import** org.springframework.beans.factory.BeanFactory;

**import** org.springframework.beans.factory.xml.XmlBeanFactory;

**import** org.springframework.core.io.\*;

**public** **class** Test {

**public** **static** **void** main(String[] args) {

        Resource r=**new** ClassPathResource("applicationContext.xml");

        BeanFactory factory=**new** XmlBeanFactory(r);

        Employee e=(Employee)factory.getBean("obj");

        s.display();

    }

}

**Output:**10 pioneercoders

# Setter Injection with Dependent Object Example

# Like Constructor Injection, we can inject the dependency of another bean using setters. In such case, we use property element. Here, our scenario is Employee HAS-A Address. The Address class object will be termed as the dependent object. Let's see the Address class first:

**Address.java**

This class contains four properties, setters and getters and toString() method.

**package** com.pioneer;

**public** **class** Address {

**private** String addressLine1,city,state,country;

//getters and setters

**public** String toString(){

**return** addressLine1+" "+city+" "+state+" "+country;

}

**Employee.java**

It contains three properties id, name and address(dependent object) , setters and getters with displayInfo() method.

**package** com.pioneer;

**public** **class** Employee {

**private** **int** id;

**private** String name;

**private** Address address;

//setters and getters

**void** displayInfo(){

    System.out.println(id+" "+name);

    System.out.println(address);

}

}

**applicationContext.xml**

The **ref** attribute of **property** elements is used to define the reference of another bean.

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans
3. xmlns="http://www.springframework.org/schema/beans"
4. xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
5. xmlns:p="http://www.springframework.org/schema/p"
6. xsi:schemaLocation="http://www.springframework.org/schema/beans
7. http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
9. <bean id="address1" **class**="com.javatpoint.Address">
10. <property name="addressLine1" value="51,Lohianagar"></property>
11. <property name="city" value="Ghaziabad"></property>
12. <property name="state" value="UP"></property>
13. <property name="country" value="India"></property>
14. </bean>
16. <bean id="obj" **class**="com.javatpoint.Employee">
17. <property name="id" value="1"></property>
18. <property name="name" value="Sachin Yadav"></property>
19. <property name="address" ref="address1"></property>
20. </bean>
22. </beans>

**Test.java**

This class gets the bean from the applicationContext.xml file and calls the displayInfo() method.

1. **package** com.javatpoint;
3. **import** org.springframework.beans.factory.BeanFactory;
4. **import** org.springframework.beans.factory.xml.XmlBeanFactory;
5. **import** org.springframework.context.ApplicationContext;
6. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
7. **import** org.springframework.core.io.ClassPathResource;
8. **import** org.springframework.core.io.Resource;
10. **public** **class** Test {
11. **public** **static** **void** main(String[] args) {
12. Resource r=**new** ClassPathResource("applicationContext.xml");
13. BeanFactory factory=**new** XmlBeanFactory(r);
15. Employee e=(Employee)factory.getBean("obj");
16. e.displayInfo();
18. }
19. }