Dependency Injection with Factory Method in Spring

Spring framework provides facility to inject bean using factory method. To do so, we can use two attributes of bean element.

1. **factory-method:** represents the factory method that will be invoked to inject the bean.
2. **factory-bean:** represents the reference of the bean by which factory method will be invoked. It is used if factory method is non-static.

A method that returns instance of a class is called **factory method**.

1. **public** **class** A {
2. **public** **static** A getA(){//factory method
3. **return** **new** A();
4. }
5. }

Factory Method Types

There can be three types of factory method:

1) A **static factory method** that returns instance of **its own** class. It is used in singleton design pattern.

1. <bean id="a" **class**="com.javatpoint.A" factory-method="getA"></bean>

2) A **static factory method** that returns instance of **another** class. It is used instance is not known and decided at runtime.

1. <bean id="b" **class**="com.javatpoint.A" factory-method="getB"></bean>

3) A **non-static factory** method that returns instance of **another** class. It is used instance is not known and decided at runtime.

1. <bean id="a" **class**="com.javatpoint.A"></bean>
2. <bean id="b" **class**="com.javatpoint.A" factory-method="getB" factory-bean="a"></bean>

Type 1

Let's see the simple code to inject the dependency by static factory method.

1. <bean id="a" **class**="com.javatpoint.A" factory-method="getA"></bean>

Let's see the full example to inject dependency using factory method in spring. To create this example, we have created 3 files.

1. **A.java**
2. **applicationContext.xml**
3. **Test.java**

**A.java**

This class is a singleton class.

1. **package** com.javatpoint;
2. **public** **class** A {
3. **private** **static** **final** A obj=**new** A();
4. **private** A(){System.out.println("private constructor");}
5. **public** **static** A getA(){
6. System.out.println("factory method ");
7. **return** obj;
8. }
9. **public** **void** msg(){
10. System.out.println("hello user");
11. }
12. }

**applicationContext.xml**

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="a" **class**="com.javatpoint.A" factory-method="getA"></bean>
11. </beans>

**Test.java**

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

1. **package** org.sssit;
2. **import** org.springframework.context.ApplicationContext;
3. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
4. **public** **class** Test {
5. **public** **static** **void** main(String[] args) {
6. ApplicationContext context=**new** ClassPathXmlApplicationContext("applicationContext.xml");
7. A a=(A)context.getBean("a");
8. a.msg();
9. }
10. }

Output:

private constructor

factory method

hello user

Type 2

Let's see the simple code to inject the dependency by static factory method that returns the instance of another class.

To create this example, we have created 6 files.

1. **Printable.java**
2. **A.java**
3. **B.java**
4. **PrintableFactory.java**
5. **applicationContext.xml**
6. **Test.java**

**Printable.java**

1. **package** com.javatpoint;
2. **public** **interface** Printable {
3. **void** print();
4. }

**A.java**

1. **package** com.javatpoint;
2. **public** **class** A **implements** Printable{
3. @Override
4. **public** **void** print() {
5. System.out.println("hello a");
6. }
8. }

**B.java**

1. **package** com.javatpoint;
2. **public** **class** B **implements** Printable{
3. @Override
4. **public** **void** print() {
5. System.out.println("hello b");
6. }
7. }

**PrintableFactory.java**

1. **package** com.javatpoint;
2. **public** **class** PrintableFactory {
3. **public** **static** Printable getPrintable(){
4. //return new B();
5. **return** **new** A();//return any one instance, either A or B
6. }
7. }

**applicationContext.xml**

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="p" **class**="com.javatpoint.PrintableFactory" factory-method="getPrintable"></bean>
11. </beans>

**Test.java**

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

1. **package** org.sssit;
2. **import** org.springframework.context.ApplicationContext;
3. **import** org.springframework.context.support.ClassPathXmlApplicationContext;
4. **public** **class** Test {
5. **public** **static** **void** main(String[] args) {
6. ApplicationContext context=**new** ClassPathXmlApplicationContext("applicationContext.xml");
7. Printable p=(Printable)context.getBean("p");
8. p.print();
9. }
10. }

Output:

hello a

Type 3

Let's see the example to inject the dependency by non-static factory method that returns the instance of another class.

To create this example, we have created 6 files.

1. **Printable.java**
2. **A.java**
3. **B.java**
4. **PrintableFactory.java**
5. **applicationContext.xml**
6. **Test.java**

All files are same as previous, you need to change only 2 files: PrintableFactory and applicationContext.xml.

**PrintableFactory.java**

1. **package** com.javatpoint;
2. **public** **class** PrintableFactory {
3. //non-static factory method
4. **public** Printable getPrintable(){
5. **return** **new** A();//return any one instance, either A or B
6. }
7. }

**applicationContext.xml**

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="pfactory" **class**="com.javatpoint.PrintableFactory"></bean>
10. <bean id="p" **class**="com.javatpoint.PrintableFactory" factory-method="getPrintable"
11. factory-bean="pfactory"></bean>
13. </beans>

Output:

hello a

<https://www.javatpoint.com/dependency-injection-with-factory-method>