[Spring依赖注入（DI）的三种方式](https://www.cnblogs.com/ycq-qiang/p/11161941.html)

**Spring依赖注入（DI）的三种方式**Spring依赖注入（DI）的三种方式，分别为：

1． 接口注入

2． Setter方法注入

3． 构造方法注入

下面介绍一下这三种依赖注入在Spring中是怎么样实现的。

首先我们需要以下几个类：

接口 Login.java

接口实现类 LoginImpl.java

三个处理类 SetterAction.java, ConstructorAction.java, InterfaceAction.java

还有一个测试类 TestMain.java

LoginImpl.java如下：

public class LoginImpl implements Login{  
 public String getName(){  
 return "Weslie";  
 }  
}

先看**Setter方法注入**，下面按照注入的方法来看SetterAction.java类

public class SetterAction {  
  
 private Login login;  
  
 public void execute() {  
 String name = login.getName();  
 System.***out***.println("My Name Is " + name);  
 }  
  
 public Login getLogin() {  
 return login;  
 }  
  
 public void setLogin(Login login) {  
 this.login = login;  
 }  
}

定义了一个Login 类型的变量 login, 在LoginAction并没有对login 进行实例化，而只有他对应的

setter/getter方法，因为我们这里使用的是Spring的依赖注入的方式

applicationContext.xml配置文件如下:

<bean id="login" class="com.example.ioc.di.LoginImpl"/>  
<bean id="setterAction" class="com.example.ioc.di.SetterAction">  
 <property name="login" ref="login"></property>  
</bean>

现在可以运行testMain.java了，代码如下：

public class TestMain {public static void main(String[] args) {  
 *// 得到ApplicationContext对象* ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 *//Setter注入  
 //得到Bean* SetterAction setterAction = (SetterAction) ctx.getBean("setterAction");  
 setterAction.execute();  
 }  
}

输出：

My Name Is Weslie

下面我们来看看**构造方法注入**

顾名思义，构造方法注入，就是我们依靠LoginAction的构造方法来达到DI的目的，如下所示：

public class ConstructorAction {  
  
 private Login login;  
  
 public ConstructorAction(Login login) {  
 this.login = login;  
 }  
   
 public void execute() {  
 String name = login.getName();  
 System.***out***.println("My Name Is " + name);  
 }  
}

这里我们添加了一个LoginAction的构造方法

applicationContext.xml配置文件如下:

<bean id="login" class="com.example.ioc.di.LoginImpl"/>  
<bean id="constructorAction" class="com.example.ioc.di.ConstructorAction">  
 <constructor-arg index="0" ref="login"></constructor-arg>  
</bean>

我们使用constructor­arg来进行配置, index属性是用来表示构造方法中参数的顺序的，如果有多

个参数，则按照顺序，从 0,1...来配置

我们现在可以运行testMain.java了,结果跟使用Setter方法注入完全一样.其中需要注意一点有：构造函数有多个参数的话，如：参数1，参数2，而参数2依赖于参数1，这

中情况则要注意构造函数的顺序，必须将参数1放在参数2之前。

下面继续说说我们不常用到的**接口注入**，如下所示：

public class InterfaceAction {  
  
 private Login login;  
  
 public void execute() {  
 try {  
 Object obj = Class.forName("com.example.ioc.di.LoginImpl").newInstance();  
 login = (Login) obj;  
 String name = login.getName();  
 System.***out***.println("My Name Is " + name);  
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 }  
}

配置文件：

<bean id="login" class="com.example.ioc.di.LoginImpl"/>  
<bean id="interfaceAction" class="com.example.ioc.di.InterfaceAction"/>

我们最常用就是Setter和构造函数这两种注入方法

对于Spring的依赖注入，最重要的就是理解他的，一旦理解了，将会觉得非常的简单。无非就是让

容器来给我们实例化那些类，我们要做的就是给容器提供这个接口，这个接口就我们的set方法或

者构造函数了。

三个案例完整的TestMain.java和配置文件如下：

import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class TestMain {  
public static void main(String[] args) {  
  
 *// 得到ApplicationContext对象* ApplicationContext ctx = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 *//Setter注入* SetterAction setterAction = (SetterAction) ctx.getBean("setterAction");  
 setterAction.execute();  
  
 *//构造方法注入* ConstructorAction constructorAction = (ConstructorAction) ctx.getBean("constructorAction");  
 constructorAction.execute();  
  
 *//接口注入* InterfaceAction interfaceAction = (InterfaceAction) ctx.getBean("interfaceAction");  
 interfaceAction.execute();  
 }  
}

<?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.xsd">  
  
 <bean id="login" class="com.example.ioc.di.LoginImpl"/>  
  
 *<!--Setter注入-->* <bean id="setterAction" class="com.example.ioc.di.SetterAction">  
 <property name="login" ref="login"></property>  
 </bean>  
 *<!--构造方法注入-->* <bean id="constructorAction" class="com.example.ioc.di.ConstructorAction">  
 <constructor-arg index="0" ref="login"></constructor-arg>  
 </bean>  
 *<!--接口注入-->* <bean id="interfaceAction" class="com.example.ioc.di.InterfaceAction">  
 </bean>  
</beans>

三个案例运行结果如下：

My Name Is Weslie

My Name Is Weslie

My Name Is Weslie