在Spring容器中为一个bean配置依赖注入有三种方式：

**· 使用属性的setter方法注入  这是最常用的方式；**

**· 使用构造器注入；**

**· 使用Filed注入（用于注解方式）.**

**使用属性的setter方法注入**

首先要配置被注入的bean，在该bean对应的类中，应该有要注入的对象属性或者基本数据类型的属性。例如：为UserBiz类注入UserDAO，同时为UserBiz注入基本数据类型String，那么这时，就要为UserDAO对象和String类型设置setter方法.，用于进行依赖注入。

如何配置该bean呢？

<bean id="userBiz" class="com.text.biz.impl.UserBizImpl">

<property name="userDao">

<ref>userDao</ref>

</property>

</bean>

以上就是一个使用属性的setter方法的方式进行依赖注入。

**使用构造器注入**

第一，在PersonBiz类中注入PersonDAO和一个String类型的数据；在该类中，不用为PersonDAO属性和String数据类型的属性设置setter方法，但是需要生成该类的构造方法；如下：

public class PersonBizImpl implements PersonBiz {

// 声明"依赖对象"PersonDAO

PersonDAO personDao = null;

// 声明"依赖的基本数据类型"

String str = null;

// 生成无参构造方法

public PersonBizImpl() {

super();

}

// 生成带参构造方法

public PersonBizImpl(PersonDAO personDao, String str) {

super();

this.personDao = personDao;

this.str = str;

}

public void addPerson() {

this.personDao.addPerson();

System.out.println(str);

}

}

第二，在配置文件中配置该类的bean，并配置构造器，在配置构造器中用到了<constructor-arg>节点，该节点有四个属性：

· index是索引，指定注入的属性，从0开始，如：0代表personDao，1代表str属性；

· type是指该属性所对应的类型，如Persondao对应的是com.aptech.dao.PersonDAO；

· ref 是指引用的依赖对象；

· value 当注入的不是依赖对象，而是基本数据类型时，就用value；

如下：

<!-- 利用构造器配置依赖注入 -->

<bean id="personDao" class="com.aptech.dao.impl.PersonDAOImpl"></bean>

<bean id="personBiz" class="com.aptech.biz.impl.PersonBizImpl">

<constructorarg index="0" type="com.aptech.dao.PersonDAO"ref="personDao"></constructor-arg>

<constructor-arg index="1" value="Spring学习"></constructor-arg>

</bean>

**使用字段(Filed)注入（用注解方式）**

在Spring中，注入依赖对象可以采用手工装配或自动装配，在实际应用开发中建议使用手工装配，因为自动装配会产生许多未知情况，开发人员无法预见最终的装配结果。

手工装配依赖对象又分为两种方式：

一种是在XML文件中，通过在bean节点下配置；如上面讲到的使用属性的setter方法注入依赖对象和使用构造器方法注入依赖对象都是这种方式。

另一种就是在java代码中使用注解的方式进行装配，在代码中加入@Resource或者@Autowired、

怎样使用注解的方式来为某个bena注入依赖对象呢？

首先，我们需要在Spring容器的配置文件applicationContext.Xml文件中配置以下信息,该信心是一个Spring配置文件的模板：

<?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:context="http://www.springframework.org/schema/context"

xmlns:p="http://www.springframework.org/schema/p"

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-2.5.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-2.5.xsd

">

</beans>

注意：只有配置了红色部分的代码才可以引入注解的命名空间，否则报错。  以上的配置隐式的注册了多个对注释进行解析的处理器：AutowiredAnnotationBeanPostProcessor、CommonAnnotationBeanPostProcessor、PersistenceAnnotationBeanPostProcessor等。

其次，在配置文件中打开<context:annotation-config>节点，告诉Spring容器可以用注解的方式注入依赖对象；其在配置文件中的代码如下：

<beans>

……

<context:annotation-config></context:annotation-config>

……

</beans>

第三，在配置文件中配置bean对象，如下：

<bean id="userDao" class="com.springtest.dao.impl.UserDAOImpl"></bean>

<bean id="userBiz" class="com.springtest.biz.impl.UserBizImpl"></bean>

第四，在需要依赖注入的BIZ类中，声明一个依赖对象，不用生成该依赖对象的setter方法，并且为该对象添加注解：

public class UserBizImpl implements UserBiz {

@Resource(name="userDao")

private UserDAO userDao = null;

public void addUser() {

this.userDao.addUser();

}

}

其中，在Java代码中可以使用@Autowired或@Resource注解方式进行Spring的依赖注入。两者的区别是：@Autowired默认按类型装配，@Resource默认按名称装配，当找不到与名称匹配的bean时，才会按类型装配。

比如：我们用@Autowired为上面的代码UserDAO接口的实例对象进行注解，它会到Spring容器中去寻找与UserDAO对象相匹配的类型，如果找到该类型则将该类型注入到userdao字段中；

如果用@Resource进行依赖注入，它先会根据指定的name属性去Spring容器中寻找与该名称匹配的类型，例如：@Resource(name="userDao")，如果没有找到该名称，则会按照类型去寻找，找到之后，会对字段userDao进行注入。

通常我们使用@Resource。

使用注解注入依赖对象不用再在代码中写依赖对象的setter方法或者该类的构造方法，并且不用再配置文件中配置大量的依赖对象，使代码更加简洁，清晰，易于维护。

在Spring IOC编程的实际开发中推荐使用注解的方式进行依赖注入。

    依赖注入—自动装配 Spring中提供了自动装配依赖对象的机制，但是在实际应用中并不推荐使用自动装配，因为自动装配会产生未知情况，开发人员无法预见最终的装配结果。 自动装配是在配置文件中实现的，如下： <bean id="\*\*\*" class="\*\*\*" autowire="byType"> 只需要配置一个autowire属性即可完成自动装配，不用再配置文件中写<property>,但是在类中还是要生成依赖对象的setter方法。 Autowire的属性值有如下几个： · byType 按类型装配  可以根据属性类型，在容器中寻找该类型匹配的bean，如有多个，则会抛出异常，如果没有找到，则属性值为null； · byName 按名称装配  可以根据属性的名称在容器中查询与该属性名称相同的bean，如果没有找到，则属性值为null； · constructor 与byType方式相似，不同之处在与它应用于构造器参数，如果在容器中没有找到与构造器参数类型一致的bean，那么将抛出异常； · autodetect 通过bean类的自省机制(introspection)来决定是使用constructor还是byType的方式进行自动装配。如果发现默认的构造器，那么将使用byType的方式。