## Spring的自动注入

### 名词解释

Spring 1.0 XML时代

2.5 Annotation注解时代

3.0 java config

Spring bean：Spring管理的bean

对象：java管理的对象

AnnotationConifigApplicationContext：注解启动Spring

ClassPathXmlApplicationContext：XML启动Spring

Springboot自动配置 != Spring自动装配（Spring自动注入）

注入模型是基于自动注入（自动注入只针对于XML），注入方式是基于手动注入

### IOC（Inversion of control 控制反转）和DI（Dependecy Injection 依赖注入）的区别

IOC只是DI的另外一种说法，他们两个原则上是一个概念，一个叫控制反转，一个叫依赖注入

### Spring有2种注入方式（注入方式只有手动）

1、提供一个构造方法

<**bean id="familyService" class="com.xiaobi.service.FamilyService"**>  
 *<!-- 构造器注入 -->  
 <!-- 构造器注入和Set注入不能在同时存在 -->* <**constructor-arg ref="xbBean"**></**constructor-arg**>  
</**bean**>

<**bean id="xbBean" class="com.xiaobi.service.XBService"**></**bean**>

**public** FamilyService(XBService xbService){  
 **this**.**xbService** = xbService;  
 System.***out***.println(**"print Family2"**);  
}

2、提供对象的set方法(XML配置里面的<property name="asd">这里的asd是注入对象的set方法（setasd）)

<**bean id="familyService" class="com.xiaobi.service.FamilyService"**>  
 *<!-- Set注入 -->* <**property name="XBService"**>  
 <**ref bean="xbBean"**></**ref**>  
 </**property**>

</**bean**>

<**bean id="xbBean" class="com.xiaobi.service.XBService"**></**bean**>

**public void** setXBService(XBService xbService) {  
 **this**.**xbService** = xbService;  
}

3、这种意义上不能完全是注入方式。他先实现ApplicationContextAware接口，重写createCommand后，在将对象放到Spring中(applicationContext.getBean(“name”,Class对象))

### Spring有4种自动注入模型，只针对于XML（default-autowire=”bytype”）

1、No（默认）

2、byname（要提供set方法，和setXXX这里面的XXX有关）

3、bytype（要提供set方法，和setXXX这里面的XXX无关）

**default-autowire="byType"**

4、constructor（推断构造方法，解析构造方法，通过构造方法找bean，找参数Spring在容器中最多的构造方法）

@Autowired这种不能说是自动注入，可以理解为他是手动注入，因为他已经指定了要注入的java对象。他先根据“类型”找，在根据“名称”找到要注入的java对象，这里的“类型”不等于bytype，“名称”不等于byname。这种注入java对象只针对于注解方式，并且只作用在java对象上面。

### @Atuowired和@Resource区别

@AutowiredSpring包提供的注解，他是由AutowireAnnotationBeanPostProcessor这个后置处理器解析的，他处理的逻辑是先根据“类型”找，再根据“名称”找。

@ResourceJDK提供的注解，他是有CommonAnnotationBeanPostProcessor这个后置处理器解析的，他处理的逻辑是先根据“名称”找，再根据“类型”找。

## Spring循环依赖

### 流程

循环依赖开始

1、启动容器

2、第一次getBean(indexService)

3、此时Spring会判断单例缓存池里面是否存在这个indexService Bean

4、如果存在直接返回Bean，如果不存在则会判断IndexService对象是否正在创建过程中，如果IndexService对象在创建过程中，则会创建Bean，否则执行下一步

5、单例缓存池没有这个indexService Bean也不在创建过程中

6、Spring就会创建一个IndexService对象，并把IndexService对象放到一个集合中（这个集合是为了后面再次需要getBean(indexService)的时候，给IndexService对象做一个正在创建标志）

7、同时把IndexService这个类的BeanDefinitionFactory（Spring扫描所有类都会分派到不同的BeanDefinition里面）放到另外一个集合中（这个集合是为了后面获取到BeanDefinition，Spring就可以通过解析BeanDefinition后获取Bean）

8、创建完IndexService对象之后Spring会进行依赖注入

9、发现需要注入userService

10、第一次getBean(userService)

11、Spring会执行UserService从3-9的过程

12、单例缓存池没有这个userService Bean也不在创建过程中

13、Spring创建一个UserService对象，给一个正在创建标志，BeanDefinitionFactory放到另一个集合中

14、创建完之后Spring会依赖注入，发现需要注入indexService

15、第二次getBean(indexService)

16、此时indexService不在单例缓存池，但是他在创建过程中

17、此时Spring就会执行创建Bean程序，创建indexService Bean之后，返回给第二次getBean(indexService)

18、UserService对象的依赖注入完成，但此时UserService还只是一个对象，他不是Bean，IndexService已经是对象也是Bean

19、第二次getBean(userService)

20、此时userService不在单例缓存池，但是他在创建过程中

21、Spring就会执行创建Bean程序，创建userService Bean之后，返回给第二次getBean(userService)

22、完成userService Bean的注入，此时UserService已经是对象也是Bean

循环依赖完成