类的注解

# 四大注解类

@Component：通用标注

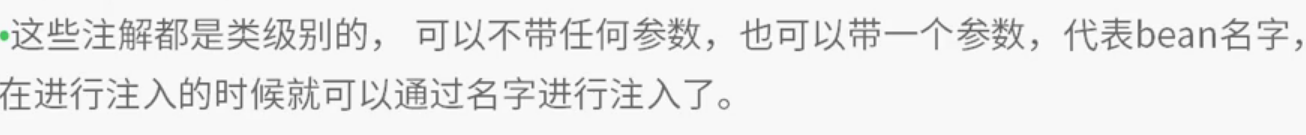
**@Controller：标注Web控制器**

**@Service：标注Service层的服务**

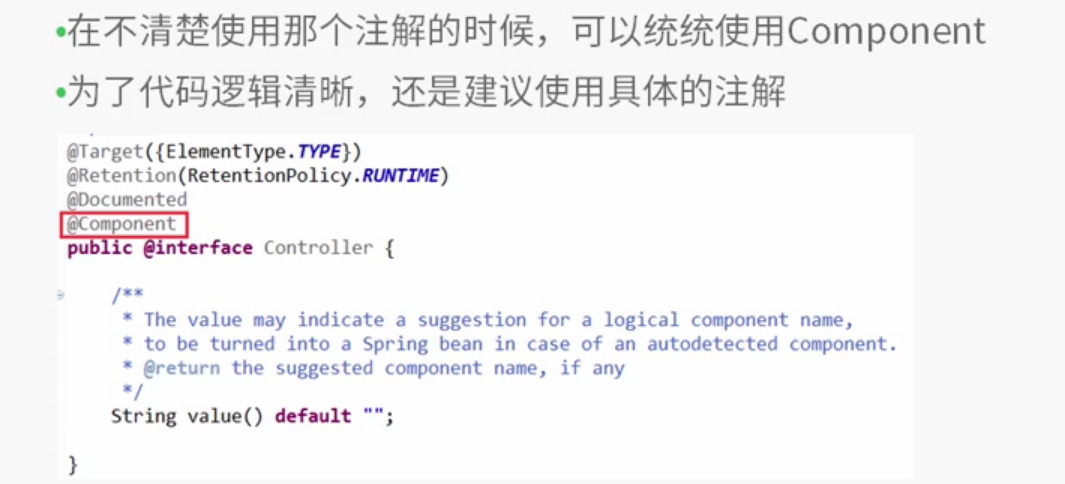
**@Repository：标注DAO层的数据访问**

后三者都是继承@Component。

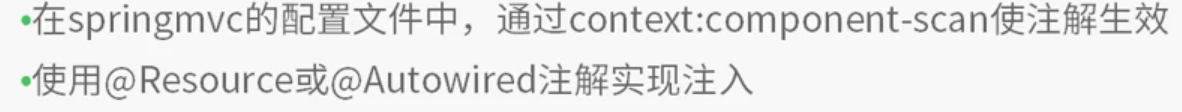
注意：@Resource与@Autowired的区别。



# @Component







# @Resource注解学习

## Spring的依赖注入几种方式

### 使用**构造器**注入；

### 使用属性**setter方法**注入。

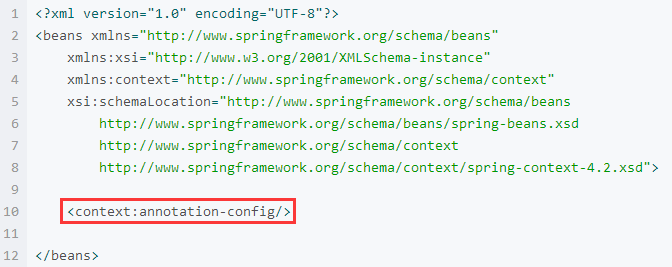
但其实还有一种注入方式，那就是**使用Field注入**（用于注解方式）。

**注入依赖对象**可以采用手工装配或自动装配，在实际应用中建议使用手工装配，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。 手工装配依赖对象，在这种方式中又有两种编程方式：

### 在xml配置文件中，通过在bean节点下配置，如：



### 在java代码中使用**@Autowired**或**@Resource**注解方式进行装配。但我们需要在xml配置文件中配置以下信息：



利用注解方式注入依赖对象，必须在配置文件中配置注解解析处理的处理器，添加

**<context:annotation-config**/> 。



**这个配置隐式注册了多个对注解进行解析处理的处理器**：

AutowiredAnnotationBeanPostProcessor

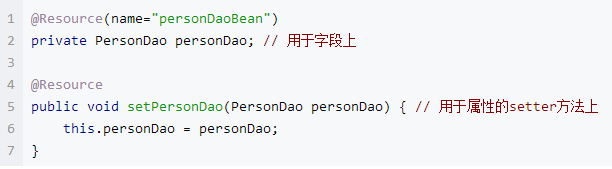
CommonAnnotationBeanPostProcessor

PersistenceAnnotationBeanPostProcessor

RequiredAnnotationBeanPostProcessor

## @Resource注解

@Resource注解和@Autowired一样，同样也可以标注在字段或属性的setter方法上，但它默认按**名称**装配。名称可以通过@Resource的**name属性**指定，如果没有指定name属性，当注解标注在**字段**上时，即默认取**字段的名称**作为bean名称寻找依赖对象，当注解标注在属性的setter方法上时，即默认取**属性名**作为bean名称寻找依赖对象。当找不到与名称相匹配的bean时，才按照类型进行装配。但是需要注意的是，如果name属性一旦指定，就只会按照名称进行装配。如：



注意：如果没有指定name属性，并且按照默认的名称仍然找不到依赖对象时， @Resource注解会回退到按类型装配。但一旦指定了name属性，就只能按名称装配了。

注意：当我们使用注解方式注入属性时，一定要往项目中导入spring-aop-4.2.5.RELEASE.jar包，不然就会报异常。

# @Resource注解

@Resource注解的类路径是：**javax.annotation.Resource**，是annotation-api.jar包中的类。如果你用的是tomcat6话，tomcat已经包含了这个jar包，不需要自己导；如果不是的话，那就手动导入这个包。

使用@Resource注解注入依赖的bean

注解的意义：@Resouce告诉Spring要给这个对象属性注入数据， 括号里面要注入的bean的ID， 如果要注入的bean是单例的，则括号里可以省略。

**问题**：在Spring项目中引入**@Resource**注解的时候，有红色下划线错误，而且输入注解的时候不能出现自动代码补全。加入javax.annotation包后，出现警告。

**解决办法**：Spring项目中缺少**javax.annotation**包的依赖。在maven配置文件pom.xml中加入依赖。

<!-- https://mvnrepository.com/artifact/javax.annotation/javax.annotation-api -->

<dependency>

<groupId>javax.annotation</groupId>

<artifactId>**javax.annotation-api**</artifactId>

<version>1.3.2</version>

</dependency>

# @Autowired与@Resource的区别

在java代码中使用**@Autowired或@Resource**注解方式进行装配。

## 相同点

@Autowired与@Resource都可以用来装配bean；都可以写在字段上,或写在setter方法上。

## 不同点

### @Autowired默认按**类型(type)**装配，@Resource默认按**名称**(name)装配，当找不到与名称匹配的bean才会按类型装配；

### @Autowired属于Spring的，@Resource属于J2EE的；(建议使用**@Resource**)；

建议使用Resource注解，是J2EE的注解，减小与Spring的耦合。

### @Autowired默认按**类型**装配（这个注解是属业spring的），默认情况下必须要求依赖对象必须存在，如果要允许null 值，可以设置它的required属性为false，如：@Autowired(required=false) ，如果我们想使用名称装配可以结合**@Qualifier注解**进行使用，如下：

Java代码：

@Autowired() @Qualifier("baseDao")

private BaseDao baseDao;

### @Resource（这个注解属于J2EE的），默认安照名称进行装配，**名称可以通过name属性进行指定**， 如果没有指定name属性，当注解写在字段上时，默认取**字段名**进行按照名称查找，如果注解写在setter方法上默认取**属性名**进行装配。 当找不到与名称匹配的bean时才按照**类型**进行装配。但是需要注意的是，如果name属性一旦指定，就只会直接按照名称进行装配。

Java代码：

@Resource(name="baseDao")

private BaseDao baseDao;

推荐使用：**@Resource注解**在字段上，且这个注解是属于**J2EE**的，减少了与spring的耦合。最重要的这样代码看起就比较优雅。

## 示例：

@Resource(name = **"stud1"**)//根据名寻找  
**public** Student **liuran**;  
@Autowired() @Qualifier(**"stud2"**)//根据名寻找  
**public** Student **zhaohong**;  
@ResponseBody  
@RequestMapping(value = **"/getLiuRan"**,method = RequestMethod.***GET***)  
**public** String getLiuRan(){  
 **return liuran**.toString();  
}  
@ResponseBody  
@RequestMapping(value = **"/getZhaoHong"**,method = RequestMethod.***GET***)  
**public** String getZhaoHong(){  
 **return zhaohong**.toString();  
}

<**bean class="xin.zhaohong.www.Student" name="stud1"**>  
 <**property name="id" value="520"**/>  
 <**property name="name" value="liuran"**/>  
 <**property name="address" value="CaoXianDong"**/>  
</**bean**>  
<**bean class="xin.zhaohong.www.Student" name="stud2"**>  
 <**property name="id" value="521"**/>  
 <**property name="name" value="zhaohong"**/>  
 <**property name="address" value="CaoXianNan"**/>  
</**bean**>

<http://localhost:8080/SpringMVCDemo0711/test/getLiuRan>

Student{id=520, name='liuran', address='CaoXianDong'}

<http://localhost:8080/SpringMVCDemo0711/test/getZhaoHong>

Student{id=521, name='zhaohong', address='CaoXianNan'}