

# Spring07、Bean的自动装配

 狂神说 - SUIP 分类: 学习笔记 创建时间: 2021/04/13 10:56 ☒ 字体 ☐ 皮肤

最后修改于: 2021/04/13 10:56

## 🏆 7、Bean的自动装配

- 自动装配是使用spring满足bean依赖的一种方法
- spring会在应用上下文中为某个bean寻找其依赖的bean。

Spring中bean有三种装配机制，分别是：

在xml中显式配置；  
在java中显式配置；  
隐式的bean发现机制和自动装配。

这里我们主要讲第三种：自动化的装配bean。

Spring的自动装配需要从两个角度来实现，或者说是两个操作：

组件扫描(component scanning)：spring会自动发现应用上下文中所创建的bean；  
自动装配(autowiring)：spring自动满足bean之间的依赖，也就是我们说的IoC/DI；

组件扫描和自动装配组合发挥巨大威力，使的显示的配置降低到最少。

**推荐不使用自动装配xml配置，而使用注解。**

### 🔪 7.1、测试环境搭建

新建一个项目

新建两个实体类，Cat Dog 都有一个叫的方法

```
1. public class Cat {  
2.     public void shout() {  
3.         System.out.println("miao~");  
4.     }  
5. }
```

```
1. public class Dog {  
2.     public void shout() {  
3.         System.out.println("wang~");  
4.     }  
5. }
```

新建一个用户类 User

```
1. public class User {  
2.     private Cat cat;  
3.     private Dog dog;  
4.     private String str;  
5. }
```

编写Spring配置文件

```

1. <?xml version="1.0" encoding="UTF-8"?>
2. <beans xmlns="http://www.springframework.org/schema/beans"
3.     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4.     xsi:schemaLocation="http://www.springframework.org/schema/beans
5.         http://www.springframework.org/schema/beans/spring-beans.xsd">
6.
7.     <bean id="dog" class="com.kuang.pojo.Dog"/>
8.     <bean id="cat" class="com.kuang.pojo.Cat"/>
9.
10.    <bean id="user" class="com.kuang.pojo.User">
11.        <property name="cat" ref="cat"/>
12.        <property name="dog" ref="dog"/>
13.        <property name="str" value="qinjiang"/>
14.    </bean>
15. </beans>

```

测试

```

1. public class MyTest {
2.     @Test
3.     public void testMethodAutowire() {
4.         ApplicationContext context = new ClassPathXmlApplicationContext("beans.xml");
5.         User user = (User) context.getBean("user");
6.         user.getCat().shout();
7.         user.getDog().shout();
8.     }
9. }

```

结果正常输出，环境OK

## 7.2、byName

### autowire byName (按名称自动装配)

由于在手动配置xml过程中，常常发生字母缺漏和大小写等错误，而无法对其进行检查，使得开发效率降低。

采用自动装配将避免这些错误，并且使配置简单化。

测试：

修改bean配置，增加一个属性 autowire=" byName"

```

1. <bean id="user" class="com.kuang.pojo.User" autowire="byName">
2.     <property name="str" value="qinjiang"/>
3. </bean>

```

再次测试，结果依旧成功输出！

我们将 cat 的bean id修改为 catXXX

再次测试，执行时报空指针java.lang.NullPointerException。因为按byName规则找不对应set方法，真正的setCat就没执行，对象就没有初始化，所以调用时就会报空指针错误。

### 小结：

当一个bean节点带有 autowire byName的属性时。

将查找其类中所有的set方法名，例如setCat，获得将set去掉并且首字母小写的字符串，即cat。  
去spring容器中寻找是否有此字符串名称id的对象。  
如果有，就取出注入；如果没有，就报空指针异常。

## 7.3、byType

## autowire byType (按类型自动装配)

使用autowire byType首先需要保证：同一类型的对象，在spring容器中唯一。如果不唯一，会报不唯一的异常。

### 1. NoUniqueBeanDefinitionException

测试：

将user的bean配置修改一下：`autowire="byType"`

测试，正常输出

在注册一个cat 的bean对象！

```
1. <bean id="dog" class="com.kuang.pojo.Dog"/>
2. <bean id="cat" class="com.kuang.pojo.Cat"/>
3. <bean id="cat2" class="com.kuang.pojo.Cat"/>
4.
5. <bean id="user" class="com.kuang.pojo.User" autowire="byType">
6.     <property name="str" value="qinjiang"/>
7. </bean>
```

测试，报错：NoUniqueBeanDefinitionException

删掉cat2，将cat的bean名称改掉！测试！因为是按类型装配，所以并不会报异常，也不影响最后的结果。甚至将id属性去掉，也不影响结果。

这就是按照类型自动装配！

## 7.4 使用注解

jdk1.5开始支持注解，spring2.5开始全面支持注解。

准备工作：利用注解的方式注入属性。

在spring配置文件中引入context文件头

```
1. xmlns:context="http://www.springframework.org/schema/context"
2.
3. http://www.springframework.org/schema/context
4. http://www.springframework.org/schema/context/spring-context.xsd
```

开启属性注解支持！

```
1. <context:annotation-config/>
```

### 7.4.1、@Autowired

- @Autowired是按类型自动转配的，不支持id匹配。
- 需要导入 spring-aop的包！

测试：

将User类中的set方法去掉，使用@Autowired注解

```

1. public class User {
2.     @Autowired
3.     private Cat cat;
4.     @Autowired
5.     private Dog dog;
6.     private String str;
7.
8.     public Cat getCat() {
9.         return cat;
10.    }
11.    public Dog getDog() {
12.        return dog;
13.    }
14.    public String getStr() {
15.        return str;
16.    }
17. }

```

此时配置文件内容

```

1. <context:annotation-config/>
2.
3. <bean id="dog" class="com.kuang.pojo.Dog"/>
4. <bean id="cat" class="com.kuang.pojo.Cat"/>
5. <bean id="user" class="com.kuang.pojo.User"/>

```

测试，成功输出结果！

【小狂神科普时间】

@Autowired(required=false) 说明： false，对象可以为null； true，对象必须存对象，不能为null。

```

1. //如果允许对象为null，设置required = false,默认为true
2. @Autowired(required = false)
3. private Cat cat;

```

#### 7.4.2、@Qualifier

- @Autowired是根据类型自动装配的，加上@Qualifier则可以根据byName的方式自动装配
- @Qualifier不能单独使用。

测试实验步骤：

配置文件修改内容，保证类型存在对象。且名字不为类的默认名字！

```

1. <bean id="dog1" class="com.kuang.pojo.Dog"/>
2. <bean id="dog2" class="com.kuang.pojo.Dog"/>
3. <bean id="cat1" class="com.kuang.pojo.Cat"/>
4. <bean id="cat2" class="com.kuang.pojo.Cat"/>

```

没有加Qualifier测试，直接报错

在属性上添加Qualifier注解

```

1. @Autowired
2. @Qualifier(value = "cat2")
3. private Cat cat;
4. @Autowired
5. @Qualifier(value = "dog2")
6. private Dog dog;

```

测试，成功输出！

### 7.4.3、@Resource

- @Resource如有指定的name属性，先按该属性进行byName方式查找装配；
- 其次再进行默认的byName方式进行装配；
- 如果以上都不成功，则按byType的方式自动装配。
- 都不成功，则报异常。

实体类：

```
1. public class User {  
2.     //如果允许对象为null，设置required = false,默认为true  
3.     @Resource(name = "cat2")  
4.     private Cat cat;  
5.     @Resource  
6.     private Dog dog;  
7.     private String str;  
8. }
```

beans.xml

```
1. <bean id="dog" class="com.kuang.pojo.Dog"/>  
2. <bean id="cat1" class="com.kuang.pojo.Cat"/>  
3. <bean id="cat2" class="com.kuang.pojo.Cat"/>  
4.  
5. <bean id="user" class="com.kuang.pojo.User"/>
```

测试：结果OK

配置文件2：beans.xml，删掉cat2

```
1. <bean id="dog" class="com.kuang.pojo.Dog"/>  
2. <bean id="cat1" class="com.kuang.pojo.Cat"/>
```

实体类上只保留注解

```
1. @Resource  
2. private Cat cat;  
3. @Resource  
4. private Dog dog;
```

结果：OK

结论：先进行byName查找，失败；再进行byType查找，成功。

## 7.5、小结

@Autowired与@Resource异同：

@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。

@Autowired默认按类型装配（属于spring规范），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

@Resource（属于J2EE规范），默认按照名称进行装配，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，就只会按照名称进行装配。

它们的作用相同都是用注解方式注入对象，但执行顺序不同。@Autowired先byType，@Resource先byName。

[关于我们](#) | [加入我们](#) | [联系我们](#)

Copyright © 广东学相伴网络科技有限公司 [粤ICP备 - 2020109190号](#)