**02、Spring进价**

# 自动装配概述

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

## 在xml中实现自动装配

5种自动装配模式

|  |  |
| --- | --- |
| **模式** | **描述** |
| no | 这是默认的设置，它意味着没有自动装配，你应该使用显式的bean引用来连线。你不用为了连线做特殊的事。在依赖注入章节你已经看到这个了。 |
| [**byName**](https://www.w3cschool.cn/wkspring/fwdz1mmb.html) | 由属性名自动装配。Spring 容器看到在 XML 配置文件中 bean 的自动装配的属性设置为 byName。然后尝试匹配，并且将它的属性与在配置文件中被定义为相同名称的 beans 的属性进行连接。 |
| [byType](https://www.w3cschool.cn/wkspring/8dhy1mmd.html) | 由属性数据类型自动装配。Spring 容器看到在 XML 配置文件中 bean 的自动装配的属性设置为 byType。然后如果它的类型匹配配置文件中的一个确切的 bean 名称，它将尝试匹配和连接属性的类型。如果存在不止一个这样的 bean，则一个致命的异常将会被抛出。 |
| [constructor](https://www.w3cschool.cn/wkspring/jtlb1mmf.html) | 类似于 byType，但该类型适用于构造函数参数类型。如果在容器中没有一个构造函数参数类型的 bean，则一个致命错误将会发生。 |
| autodetect | Spring首先尝试通过 constructor 使用自动装配来连接，如果它不执行，Spring 尝试通过 byType 来自动装配。 |

### 局部配置

在有依赖的bean标签中使用autowire="byName"，只对当前bean自动装配

### 全局配置

在xml标签加入default-autowire="byName"，所有有依赖的bean实现自动装配

### 案例分析

之前的做法

<!-- 配置Human对象 -->

<bean id=*"human"* class=*"spring1.Human"*>

<property name=*"name"* value=*"tom"* />

<property name=*"age"* value=*"18"* />

<property name=*"student"* ref=*"student"*/>

</bean>

<!-- 配置Student对象 -->

<bean class=*"spring1.Student"* id=*"student"*>

<property name=*"sname"* value=*"小张同学"* />

<property name=*"score"* value=*"80"* />

</bean>

没有采用自动装配的情况下，需要使用红色部分来引用依赖对象

下面使用“自动装配”优化程序

<!-- 配置Human对象 -->

<bean id=*"human"* class=*"spring1.Human"* autowire=*"byName"*>

<property name=*"name"* value=*"tom"* />

<property name=*"age"* value=*"18"* />

</bean>

<!-- 配置Student对象 -->

<bean class=*"spring1.Student"* id=*"student"*>

<property name=*"sname"* value=*"小张同学"* />

<property name=*"score"* value=*"80"* />

</bean>

不需要在Human的bean中添加student依赖对象，程序会自动检索装配进去

# 使用注解的方式配置bean

## 准备工作

### 导入jar包



### 开启context空间的支持

在spring配置文件的xml头声明中增加context空间支持，注意顺序不能乱了

<?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"*

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

*http://www.springframework.org/schema/beans/spring-beans-4.2.xsd*

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

*http://www.springframework.org/schema/context/spring-context.xsd"*>

</beans>

### 开启自动扫描功能，并指定自动扫描的包路劲

<context:component-scan base-package=*"com.lanqiao"*>

</context:component-scan>

PS：自动扫描包可以包含多个，用英文逗号分隔，自动扫描包含包本身以及其所有的子包。

Spring从2.5版本开始引入了基于注解驱动的方式来进行bean的配置，但在使用过程中，需要先配置自动扫描的目录

## 配置Bean使用@Component注解

如需为Bean定义名称，在参数中添加Bean名称@Component("beanName")

示例如下：

@Component("human")

**public** **class** Human {

**public** **void** say() {

System.***out***.println("这个人正在与人交谈...");

}

}

测试类代码：

ApplicationContext ac = **new**

ClassPathXmlApplicationContext("applicationContext.xml");

Human human = (Human) ac.getBean("human");

human.say();

PS:使用注解的方式来配置bean，要求类必须要有不带参数的构造方法。

## 为Bean注入简单类型属性

* 1. 在属性名上方声明该属性自动装配@Autowired
  2. 在属性名上方声明该属性注入的值@Value(value)

@Autowired

@Value("张三")

**private** String name;

@Autowired

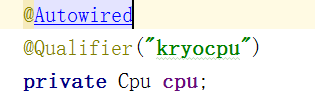
@Value("18")

**private** **int** age;

PS：注解自动装配属性值无需提供对应属性的setter方法

## 为Bean注入引用类型属性

* 1. 在属性名上方声明该属性自动装配@Autowired
  2. 在属性名上方声明该属性注入的值@Qualifier(bean引用名称【name|id】)



PS：被注入的Bean一定要存在，无需提供对应属性的setter方法；

被注入的bean若在整个应用中仅有一个实例，则可以不用@Qualitfier

（一个接口下有多个实现类，则需要使用到@Qualitfier）

@Service("service1")

public class EmployeeServiceImpl implements EmployeeService {

public EmployeeDto getEmployeeById(Long id) {

return new EmployeeDto();

}

}

@Service("service2")

public class EmployeeServiceImpl1 implements EmployeeService {

public EmployeeDto getEmployeeById(Long id) {

return new EmployeeDto();

}

}

public class EmployeeInfoControl {

@Autowired

**@Qualifier("service1")**

EmployeeService employeeService;

@RequestMapping(params = "method=showEmplayeeInfo")

public void showEmplayeeInfo(HttpServletRequest request, HttpServletResponse response, EmployeeDto dto) {

#略

}

}

## @Autowired注解属性required用于声明自动装配的数据是否是必须的

如果required=true，必须为其装配数据，如果没有值，抛出异常

如果required=false，自动装配数据失败，则注入值为null

@Autowired(required = **true**)

@Value("张三")

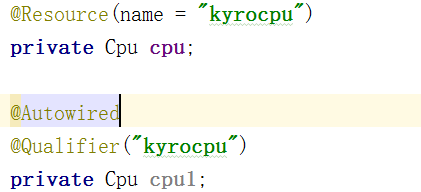
**private** String name;

@Autowired(required = **false**)

**private** **int** age;

PS:默认值为true

## Spring支持JSR-250规范提供的@Resource注解的支持



以上二者的含义一致。

## .Bean声明周期注解

@PostConstruct

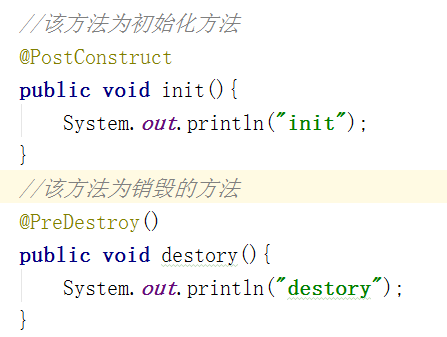
功能：为当前Bean指定init-method参数

格式：定义在成员方法的上方，兼容静态方法

@PreDestroy

功能：为当前Bean指定destory-method参数

格式：定义在成员方法的上方，兼容静态方法



PS：要求当前类被注册为Bean，否则无效果

## Bean作用域注解

在类的定义上方添加@Scope指定Bean的作用域

常用：@Scope("prototype")

默认：@Scope("singleton")

PS：要求当前类被注册为Bean，否则无效果

## Bean的兼容格式

Spring提供了3个注解开发Bean的兼容格式，用于后期分层设计扩展，可以将这3个注解理解为@Component的子注解。在spring+mybaitis集成时会应用到。

@Repository 用于数据层实现类标注（DAO）

@Service 用于业务逻辑层实现类标注（service）

@Controller 用于控制层实现类进行标注（控制层）

# Spring整合junit

## 添加jar包



## 为测试类添加相关注解

设置类运行器(注解添加到运行程序类的上方)

@RunWith(SpringJUnit4ClassRunner.class)

设置读取Spring的配置文件路径  
@ContextConfiguration(locations="classpath:/applicationContext.xml")

PS：ContextConfigration的参数是一个字符串数组，可以支持多个spring配置文件，classpath:/代表项目的编译路径根目录。

## 要测试的Bean必须称为测试类的属性注入进来，然后对其进行测试

@RunWith(SpringJUnit4ClassRunner.**class**)

@ContextConfiguration(locations="classpath:/applicationContext.xml")

**public** **class** Test1 {

@Autowired

**private** Human human;

@Test

**public** **void** testHuman() {

human.say();

}

}

PS：使用junit整合spring时，必须保障运行类中要注入要测试的Bean。整合完毕后，当前的测试类将作为Spring的Bean出现，而测试的Bean对象作为测试类的注入资源进行运行。