

Spring注解驱动开发第22讲——如何实现方法、构造器位置的自动装配？我这样回答让面试官很满意！

写在前面

在前面两讲中，我们介绍了如何使用注解来自动装配Spring组件。之前将的都是在类的字段上添加注解，那有没有什么方法可以实现方法、构造器位置上的自动装配呢？今天我们就一起来探讨下如何实现方法、构造器位置上的自动装配。

再谈@Autowired注解

在之前《Spring注解驱动开发第20讲——使用@Autowired、@Qualifier、@Primary这三大注解自动装配组件，你会了吗？》这一讲中，我简单介绍了下@Autowired注解的使用方法。下面，我们再来看下@Autowired注解的源码。

```
Autowired.class
2+ * Copyright 2002-2016 the original author or authors.
16
17 package org.springframework.beans.factory.annotation;
18
19 import java.lang.annotation.Documented;
20 import java.lang.annotation.ElementType;
21 import java.lang.annotation.Retention;
22 import java.lang.annotation.RetentionPolicy;
23 import java.lang.annotation.Target;
24
26+ * Marks a constructor, field, setter method or config method as to be
67 @Target({ElementType.CONSTRUCTOR, ElementType.METHOD, ElementType.PARAMETER, ElementType.FIELD, ElementType.ANNOTATION_TYPE})
68 @Retention(RetentionPolicy.RUNTIME)
69 @Documented
70 public @interface Autowired {
71
72     /**
73      * Declares whether the annotated dependency is required.
74      * <p>Defaults to {@code true}</p>.
75      */
76     boolean required() default true;
77
78 }
```

我们通过@Autowired注解的源码可以看出，在@Autowired注解上标注有如下的注解信息。

```
1 | @Target({ElementType.CONSTRUCTOR, ElementType.METHOD, ElementType.PARAMETER, ElementType.FIELD, ElementType.ANNOTATION_TYPE})
   | AI写代码java运行
```

可以看出@Autowired注解不仅可以标注在字段上，而且还可以标注在构造方法、实例方法以及参数上。

实战案例

案例准备

首先，我们在项目中新建一个Boss类，在Boss类中有一个Car类的引用，并且我们使用@Component注解将Dog类加载到IOC容器中，如下所示。

```
1 | package com.meimeixia.bean;
2 |
3 | import org.springframework.stereotype.Component;
4 |
5 | // 默认加在IOC容器中的组件，容器启动会调用无参构造器创建对象，然后再进行初始化、赋值等操作
6 | @Component
7 | public class Boss {
8 |
9 |     private Car car;
10 |
11 |     public Car getCar() {
12 |         return car;
13 |     }
14 |
15 |     public void setCar(Car car) {
16 |         this.car = car;
17 |     }
18 |
19 |     @Override
20 |     public String toString() {
21 |         return "Boss [car=" + car + "]";
22 |     }
23 |
24 | }
```

AI写代码java运行



注意，Car类上也要标注@Component注解，即它也要被加载到IOC容器中。

新建好以上Boss类之后，我们还需要在MainConfigOfAutowired配置类的@ComponentScan注解中进行配置，使其能够扫描com.meimeixia.bean包下的类，如下所示。

```
1 package com.meimeixia.config;
2
3 import org.springframework.context.annotation.Bean;
4 import org.springframework.context.annotation.ComponentScan;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.context.annotation.Primary;
7
8 import com.meimeixia.dao.BookDao;
9
10 /**
11  *
12  * @author liayun
13  *
14  */
15 @Configuration
16 @ComponentScan({"com.meimeixia.service", "com.meimeixia.dao", "com.meimeixia.controller", "com.meimeixia.bean"})
17 public class MainConfigOfAutowired {
18
19     @Primary
20     @Bean("bookDao2")
21     public BookDao bookDao() {
22         BookDao bookDao = new BookDao();
23         bookDao.setLable("2");
24         return bookDao;
25     }
26 }
27 }
```

AI写代码java运行



此时，我们可以直接在Boss类中的car字段上添加@Autowired注解，使其自动装配。这是我们在《Spring注解驱动开发第20讲——使用@Autowired、@Qualifier、@Primary这三大注解自动装配组件，你会了吗？》这一讲中得出的结论。那今天我们就使用其他的方式来实现car的自动装配。

标注在实例方法上

我们可以将@Autowired注解标注在setter方法上，如下所示。

```
1 @Autowired
2 public void setCar(Car car) {
3     this.car = car;
4 }
```

AI写代码java运行

当@Autowired注解标注在方法上时，Spring容器在创建当前对象的时候，就会调用相应的方法为对象赋值。如果标注的方法存在参数时，那么方法使用的参数和自定义类型的值，需要从IOC容器中获取。

然后，我们将IOCTest_Autowired类的test01()方法中有关获取和打印BookService信息的代码注释掉，新增获取和打印Boss信息的代码，如下所示。

```
1 package com.meimeixia.test;
2
3 import org.junit.Test;
4 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
5
6 import com.meimeixia.bean.Boss;
7 import com.meimeixia.config.MainConfigOfAutowired;
8 import com.meimeixia.service.BookService;
9
10 public class IOCTest_Autowired {
11
12     @Test
13     public void test01() {
14         AnnotationConfigApplicationContext applicationContext = new AnnotationConfigApplicationContext(MainConfigOfAutowired.class);
15
16         // BookService bookService = applicationContext.getBean(BookService.class);
17         // System.out.println(bookService);
18     }
19 }
```

```

19 //      BookDao bookDao = applicationContext.getBean(BookDao.class);
20 //      System.out.println(bookDao);
21
22      Boss boss = applicationContext.getBean(Boss.class);
23      System.out.println(boss);
24
25      applicationContext.close();
26  }
27 }
28 }

```

AI写代码java运行



运行以上test01()方法进行测试，可以看到，结果信息中输出了如下信息。

```

<terminated> IOCTest_Autowired.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午3:40:14)
postProcessAfterInitialization...boss=>Boss [car=com.meimeixia.bean.Car@61862a7f]
cat constructor...
postProcessBeforeInitialization...cat=>com.meimeixia.bean.Cat@4d9e68d0
cat afterPropertiesSet...
postProcessAfterInitialization...cat=>com.meimeixia.bean.Cat@4d9e68d0
dog constructor...
postProcessBeforeInitialization...dog=>com.meimeixia.bean.Dog@62150f9e
dog...@PostConstruct...
postProcessAfterInitialization...dog=>com.meimeixia.bean.Dog@62150f9e
Boss [car=com.meimeixia.bean.Car@61862a7f]
十二月 03, 2020 3:40:15 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date [Thu Dec 03 15:40:14 CST 2020]; root of context hierarchy
dog...@PreDestroy...
cat destroy...

```

说明已经获取到了car的信息，也就是说可以将@Autowired注解标注在方法上。

为了验证最终的输出结果是否是从IOC容器中获取的，我们可以在IOCTest_Autowired类的test01()方法中直接获取Car对象的信息，如下所示。

```

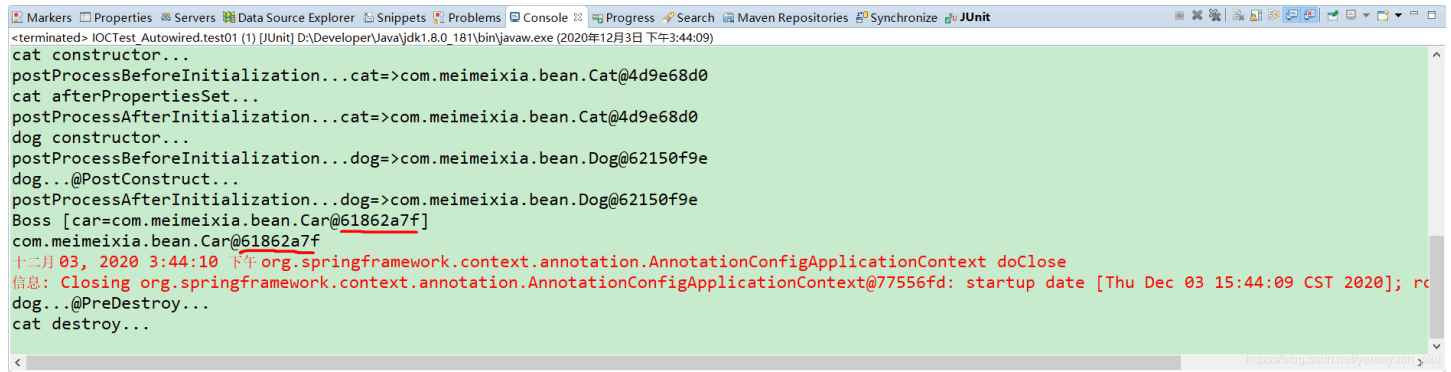
1 package com.meimeixia.test;
2
3 import org.junit.Test;
4 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
5
6 import com.meimeixia.bean.Boss;
7 import com.meimeixia.bean.Car;
8 import com.meimeixia.config.MainConfigOfAutowired;
9 import com.meimeixia.service.BookService;
10
11 public class IOCTest_Autowired {
12
13     @Test
14     public void test01() {
15         AnnotationConfigApplicationContext applicationContext = new AnnotationConfigApplicationContext(MainConfigOfAutowired.class);
16
17         //      BookService bookService = applicationContext.getBean(BookService.class);
18         //      System.out.println(bookService);
19
20         //      BookDao bookDao = applicationContext.getBean(BookDao.class);
21         //      System.out.println(bookDao);
22
23         Boss boss = applicationContext.getBean(Boss.class);
24         System.out.println(boss);
25
26         Car car = applicationContext.getBean(Car.class);
27         System.out.println(car);
28
29         applicationContext.close();
30     }
31 }
32 }

```

AI写代码java运行



我们再次运行以上test01()方法进行测试，可以在输出的结果信息中看到如下两行内容。



```

<terminated> IOCTest Autowired.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午3:44:09)
cat constructor...
postProcessBeforeInitialization...cat=>com.meimeixia.bean.Cat@4d9e68d0
cat afterPropertiesSet...
postProcessAfterInitialization...cat=>com.meimeixia.bean.Cat@4d9e68d0
dog constructor...
postProcessBeforeInitialization...dog=>com.meimeixia.bean.Dog@62150f9e
dog...@PostConstruct...
postProcessAfterInitialization...dog=>com.meimeixia.bean.Dog@62150f9e
Boss [car=com.meimeixia.bean.Car@61862a7f]
com.meimeixia.bean.Car@61862a7f
十二月 03, 2020 3:44:10 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date [Thu Dec 03 15:44:09 CST 2020]; root of context hierarchy
dog...@PreDestroy...
cat destroy...

```

这已然说明在Boss类中通过@Autowired注解获取到的Car对象和直接从IOC容器中获取到Car对象是同一个对象。

标注在构造方法上

在上面的案例中，我们在Boss类上使用了@Component注解，如下所示。

```

1 package com.meimeixia.bean;
2
3 import org.springframework.stereotype.Component;
4
5 // 默认加在IOC容器中的组件，容器启动会调用无参构造器创建对象，然后再进行初始化、赋值等操作
6 @Component
7 public class Boss {
8
9     private Car car;
10
11     public Car getCar() {
12         return car;
13     }
14
15     @Autowired
16     public void setCar(Car car) {
17         this.car = car;
18     }
19
20     @Override
21     public String toString() {
22         return "Boss [car=" + car + "]";
23     }
24 }
25

```

AI写代码java运行



此时，Spring会默认将该类加载进IOC容器中，IOC容器启动的时候默认会调用bean的无参构造器创建对象，然后再进行初始化、赋值等操作。

接下来，我们为Boss类添加一个有参构造方法，然后去除setCar()方法上的@Autowired注解，将@Autowired注解标注在有参构造方法上，并在构造方法中打印一条信息，如下所示。

```

1 package com.meimeixia.bean;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Component;
5
6 // 默认加在IOC容器中的组件，容器启动会调用无参构造器创建对象，然后再进行初始化、赋值等操作
7 @Component
8 public class Boss {
9
10     private Car car;
11
12     @Autowired
13     public Boss(Car car) {
14         this.car = car;
15         System.out.println("Boss...有参构造器");
16     }
17
18     public Car getCar() {
19         return car;
20     }
21
22 }
23

```

```

23     public void setCar(Car car) {
24         this.car = car;
25     }
26
27     @Override
28     public String toString() {
29         return "Boss [car=" + car + "]\n";
30     }
31 }

```

AI写代码java运行



接着，我们运行IOCTest_Autowired类中的test01()方法进行测试，可以看到输出结果信息中存在如下行信息。

```

car constructor...
postProcessBeforeInitialization...car=>com.meimeixia.bean.Car@4d9e68d0
postProcessAfterInitialization...car=>com.meimeixia.bean.Car@4d9e68d0
Boss...有参构造器
postProcessBeforeInitialization...boss=>Boss [car=com.meimeixia.bean.Car@4d9e68d0]
postProcessAfterInitialization...boss=>Boss [car=com.meimeixia.bean.Car@4d9e68d0]
cat constructor...
postProcessBeforeInitialization...cat=>com.meimeixia.bean.Cat@7fa98a66
cat afterPropertiesSet...
postProcessAfterInitialization...cat=>com.meimeixia.bean.Cat@7fa98a66
dog constructor...
postProcessBeforeInitialization...dog=>com.meimeixia.bean.Dog@32eff876
dog...@PostConstruct...
postProcessAfterInitialization...dog=>com.meimeixia.bean.Dog@32eff876
Boss [car=com.meimeixia.bean.Car@4d9e68d0]
com.meimeixia.bean.Car@4d9e68d0
十二月 03, 2020 3:55:09 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date [Thu Dec 03 15:55:09 CST 2020]; root of context hierarchy
dog...@PreDestroy...
cat destroy...

```

说明IOC容器在启动的时候调用了Boss类的有参构造方法。并且还可以从输出的如下两行信息中看出，通过Boss类的toString()方法打印出的Car对象和直接从IOC容器中获取的Car对象是同一个对象。

```

1 Boss [car=com.meimeixia.bean.Car@4d9e68d0]
2 com.meimeixia.bean.Car@4d9e68d0

```

AI写代码java运行

这里，需要大家注意的是，使用@Autowired注解标注在构造方法上时，构造方法中的参数对象也是从IOC容器中获取的。

其实，还有一点我得说明一下，使用@Autowired注解标注在构造方法上时，如果组件中只有一个有参构造方法，那么这个有参构造方法上的@Autowired注解可以省略，并且参数位置的组件还是可以自动从IOC容器中获取。

标注在参数上

我们也可以将@Autowired注解标注在参数上，例如，在Boss类中我们将构造方法上的@Autowired注解标注在构造方法的参数上，如下所示。

```

1 public Boss(@Autowired Car car) {
2     this.car = car;
3     System.out.println("Boss...有参构造器");
4 }

```

AI写代码java运行

当然了，也可以将@Autowired注解标注在setter方法的参数上，如下所示。

```

1 public void setCar(@Autowired Car car) {
2     this.car = car;
3 }

```

AI写代码java运行

最终的效果与标注在字段、实例方法和构造方法上的效果都是一样的。

例如，我们将@Autowired注解标注在构造方法的参数上，运行IOCTest_Autowired类中的test01()方法进行测试，可以看到，输出结果中同样包含如下三行信息。

```

<terminated> IOCTest.Autowired.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午4:07:17)
postProcessAfterInitialization...car=>com.meimeixia.bean.Car@441772e
Boss...有参构造器
postProcessBeforeInitialization...boss=>Boss [car=com.meimeixia.bean.Car@441772e]
postProcessAfterInitialization...boss=>Boss [car=com.meimeixia.bean.Car@441772e]
cat constructor...
postProcessBeforeInitialization...cat=>com.meimeixia.bean.Cat@6c80d78a
cat afterPropertiesSet...
postProcessAfterInitialization...cat=>com.meimeixia.bean.Cat@6c80d78a
dog constructor...
postProcessBeforeInitialization...dog=>com.meimeixia.bean.Dog@5fdcaa40
dog...@PostConstruct...
postProcessAfterInitialization...dog=>com.meimeixia.bean.Dog@5fdcaa40
Boss [car=com.meimeixia.bean.Car@441772e]
com.meimeixia.bean.Car@441772e
十二月 03, 2020 4:07:17 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date [Thu Dec 03 16:07:17 CST 2020]; root of
dog...@PreDestroy...
cat destroy...

```

于是，我们可以得出结论：无论@Autowired注解是标注在字段上、实例方法上、构造方法上还是参数上，参数位置的组件都是从IOC容器中获取。

如果Spring的bean中只有一个有参构造方法，并且这个有参构造方法只有一个参数，这个参数还是IOC容器中的对象，当@Autowired注解标注在这个构造方法的参数上时，那么我们可以将其省略掉，如下所示。

```

1 public Boss(*@Autowired* Car car) {
2     this.car = car;
3     System.out.println("Boss...有参构造器");
4 }

```

AI写代码java运行

此时，我们再次运行IOCTest_Autowired类中的test01()方法进行测试，从输出的结果信息中可以看出，同样输出了下面的三行信息。

```

<terminated> IOCTest.Autowired.test01 (1) [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午4:14:35)
postProcessAfterInitialization...car=>com.meimeixia.bean.Car@441772e
Boss...有参构造器
postProcessBeforeInitialization...boss=>Boss [car=com.meimeixia.bean.Car@441772e]
postProcessAfterInitialization...boss=>Boss [car=com.meimeixia.bean.Car@441772e]
cat constructor...
postProcessBeforeInitialization...cat=>com.meimeixia.bean.Cat@6c80d78a
cat afterPropertiesSet...
postProcessAfterInitialization...cat=>com.meimeixia.bean.Cat@6c80d78a
dog constructor...
postProcessBeforeInitialization...dog=>com.meimeixia.bean.Dog@5fdcaa40
dog...@PostConstruct...
postProcessAfterInitialization...dog=>com.meimeixia.bean.Dog@5fdcaa40
Boss [car=com.meimeixia.bean.Car@441772e]
com.meimeixia.bean.Car@441772e
十二月 03, 2020 4:14:36 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date [Thu Dec 03 16:14:36 CST 2020]; root of
dog...@PreDestroy...
cat destroy...

```

标注在方法位置

@Autowired注解可以标注在某个方法的位置上。这里，为了更好的演示效果，我们新建一个Color类，在Color类中有一个Car类型的成员变量，如下所示。

```

1 package com.meimeixia.bean;
2
3 public class Color {
4
5     public Car car;
6
7     public Car getCar() {
8         return car;
9     }
10
11     public void setCar(Car car) {
12         this.car = car;
13     }
14
15     @Override
16     public String toString() {
17         return "Color [car=" + car + "]";
18     }
19 }
20

```

AI写代码java运行

然后，我们在MainConfigOfAutowired配置类中实例化Color类，如下所示。

```
1 @Bean
2 public Color color() {
3     Color color = new Color();
4     return color;
5 }
```

AI写代码java运行

接着，我们在IOCTest_Autowired类中再创建一个test02()测试方法，如下所示。

```
1 @Test
2 public void test02() {
3     AnnotationConfigApplicationContext applicationContext = new AnnotationConfigApplicationContext(MainConfigOfAutowired.class);
4
5     Color color = applicationContext.getBean(Color.class);
6     System.out.println(color);
7
8     applicationContext.close();
9 }
```

AI写代码java运行



紧接着，运行以上test02()方法，发现在输出的结果信息中存在如下信息。

```
<terminated> IOCTest_Autowired.test02 [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午4:24:01)
cat constructor...
postProcessBeforeInitialization...cat=>com.meimeixia.bean.Cat@6dc17b83
cat afterPropertiesSet...
postProcessAfterInitialization...cat=>com.meimeixia.bean.Cat@6dc17b83
dog constructor...
postProcessBeforeInitialization...dog=>com.meimeixia.bean.Dog@71809907
dog...@PostConstruct...
postProcessAfterInitialization...dog=>com.meimeixia.bean.Dog@71809907
postProcessBeforeInitialization...color=>Color [car=null]
postProcessAfterInitialization...color=>Color [car=null]
Color [car=null]
十二月 03, 2020 4:24:01 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date
dog...@PreDestroy...
cat destroy...
```

说明此时的Color对象中的Car对象为空。此时，我们可以将Car对象作为一个参数传递到MainConfigOfAutowired配置类的color()方法中，并且将该Car对象设置到Color对象中，如下所示。

```
1 @Bean
2 public Color color(Car car) {
3     Color color = new Color();
4     color.setCar(car);
5     return color;
6 }
```

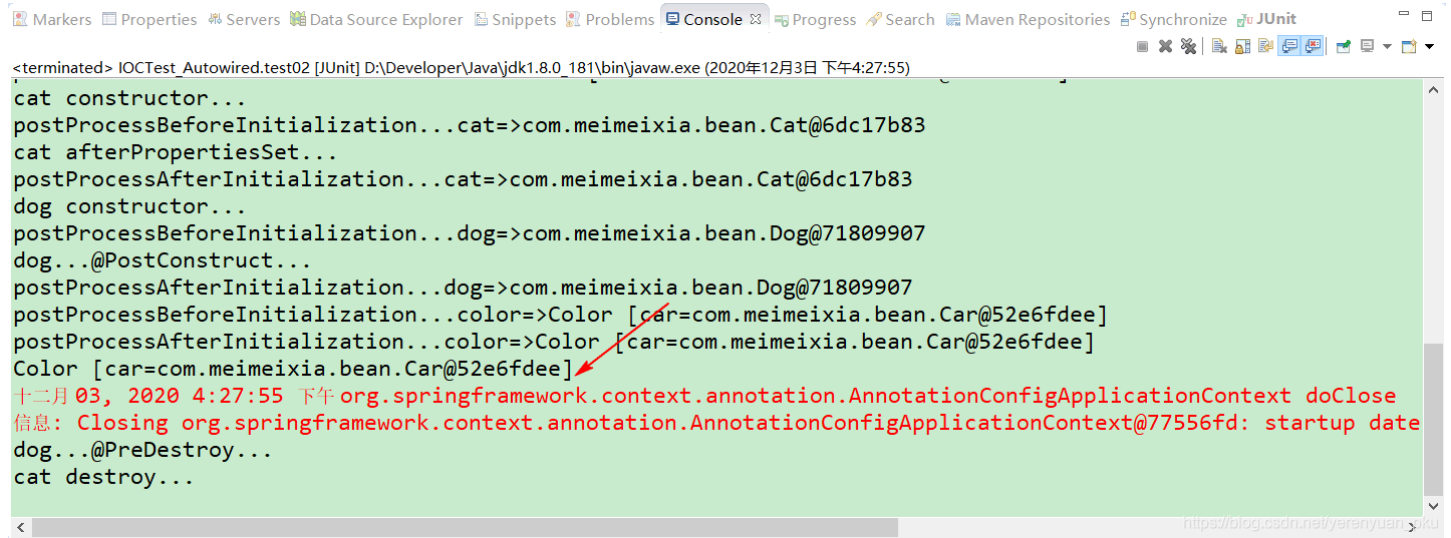
AI写代码java运行

当然了，我们也可以使用@Autowired注解来标注color()方法中的car参数，就像下面这样。

```
1 @Bean
2 public Color color(@Autowired Car car) {
3     Color color = new Color();
4     color.setCar(car);
5     return color;
6 }
```

AI写代码java运行

接下来，我们再次运行test02()方法，可以看到在输出的结果信息中存在如下信息。



```
<terminated> IOCTest_Autowired.test02 [JUnit] D:\Developer\Java\jdk1.8.0_181\bin\javaw.exe (2020年12月3日 下午4:27:55)
cat constructor...
postProcessBeforeInitialization...cat=>com.meimeixia.bean.Cat@6dc17b83
cat afterPropertiesSet...
postProcessAfterInitialization...cat=>com.meimeixia.bean.Cat@6dc17b83
dog constructor...
postProcessBeforeInitialization...dog=>com.meimeixia.bean.Dog@71809907
dog...@PostConstruct...
postProcessAfterInitialization...dog=>com.meimeixia.bean.Dog@71809907
postProcessBeforeInitialization...color=>Color [car=com.meimeixia.bean.Car@52e6fdee]
postProcessAfterInitialization...color=>Color [car=com.meimeixia.bean.Car@52e6fdee]
Color [car=com.meimeixia.bean.Car@52e6fdee]
十二月 03, 2020 4:27:55 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@77556fd: startup date
dog...@PreDestroy...
cat destroy...
```

说明Car对象被成功创建并设置到Color对象中了。

至此，我们可以得出结论：如果方法只有一个IOC容器中的对象作为参数，当@Autowired注解标注在这个方法的参数上时，我们可以将@Autowired注解省略掉。也就是说@Bean注解标注的方法在创建对象的时候，方法参数的值是从IOC容器中获取的，此外，标注在这个方法的参数上的@Autowired注解可以省略。

其实，我们用到最多的还是把@Autowired注解标注在方法位置，即使用@Bean注解+方法参数这种形式，此时，该方法参数的值从IOC容器中获取，并且还可以默认不写@Autowired注解，因为效果都是一样的，都能实现自动装配！