

课程目标

内容定位

1. 配置组件

- 1.1. @Configuration
- 1.2. @ComponentScan
- 1.3. @Scope
- 1.4. @Lazy
- 1.5. @Conditional
- 1.6. @Import
- 1.7. 生命周期

2. 赋值(自动装配)组件

- 2.1. @Component, @Controller, @Service, @Repository
- 2.2. @Value
- 2.3. @Autowired
- 2.4. @PropertySource
- 2.5. @Qualifier
- 2.6. @Primary
- 2.7. @Resource

3. 织入组件

4. 切面组件

课程目标

- 1、掌握XML配置和Annotation编程的区别。
- 2、掌握常用的Annotation配置功能。

内容定位

完全掌握Spring的Annotation API使用。

1. 配置组件

注解名称	说明
@Configuration	把一个类作为一个IoC容器，它的某个方法头上如果注册了@Bean，就会作为这个Spring容器中的Bean。
@ComponentScan	在配置类上添加 @ComponentScan 注解。该注解默认会扫描该类所在的包下所有的配置类，相当于之前的 <context:component-scan>
@Scope	用于指定scope作用域的（用在类上）
@Lazy	表示延迟初始化
@Conditional	Spring4开始提供，它的作用是按照一定的条件进行判断，满足条件给容器注册Bean。
@Import	导入外部资源
生命周期控制	@PostConstruct用于指定初始化方法（用在方法上） @PreDestory用于指定销毁方法（用在方法上） @DependsOn：定义Bean初始化及销毁时的顺序

1.1. @Configuration

```

1 package cn.sitedev.project.entity;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class Person {
11     private String name;
12     private int age;
13 }
14 //////////////////////////////////////
15 package cn.sitedev.project.entity;
16
17 import lombok.AllArgsConstructor;
18 import lombok.Data;
19 import lombok.NoArgsConstructor;
20
21
22 @Data
23 @AllArgsConstructor
24 @NoArgsConstructor
25 public class Man {
26     private String name;

```

```

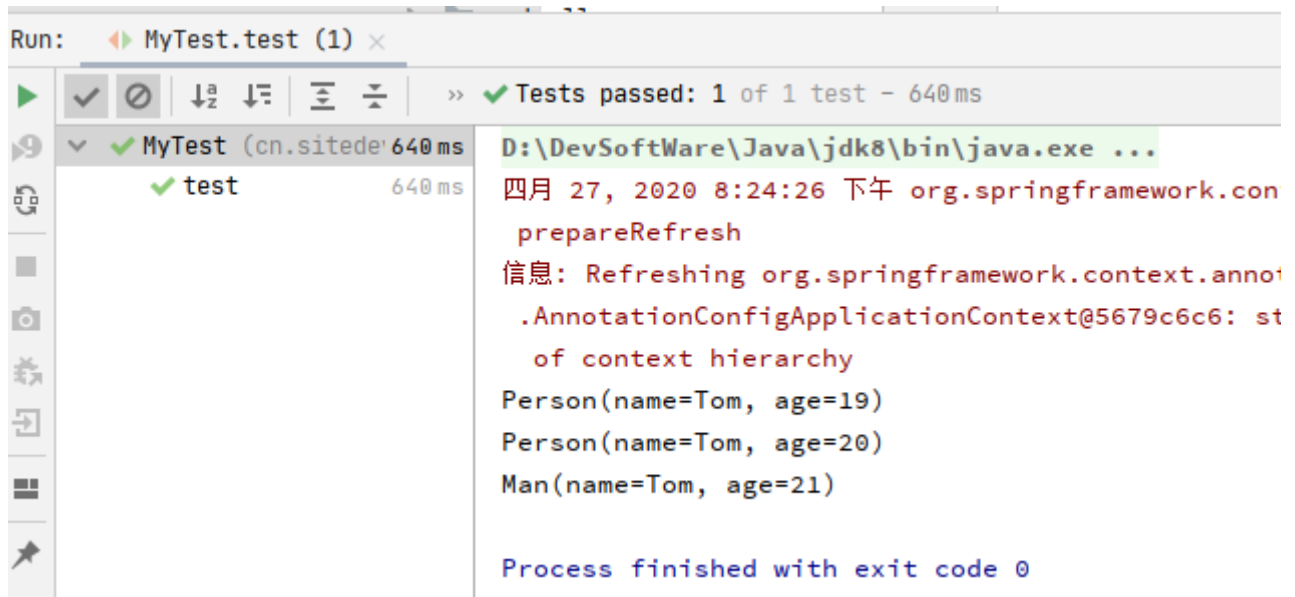
27     private int age;
28 }
29 ///////////////////////////////////////////////////
30 package cn.sitedev.demo.annotation.configure.configuration;
31
32 import cn.sitedev.project.entity.Man;
33 import cn.sitedev.project.entity.Person;
34 import org.springframework.context.annotation.Bean;
35 import org.springframework.context.annotation.Configuration;
36
37 @Configuration
38 public class MyConfig {
39
40     // 默认是取类名首字母小写
41     // 其次取方法的名称
42     // 最后取bean注解的value
43     @Bean
44     public Person person2() {
45         return new Person("Tom", 19);
46     }
47
48     @Bean("person3")
49     public Person person1() {
50         return new Person("Tom", 20);
51     }
52
53     @Bean
54     public Man man() {
55         return new Man("Tom", 21);
56     }
57 }
58 ///////////////////////////////////////////////////
59 package cn.sitedev.demo.annotation.configure.configuration;
60
61 import cn.sitedev.project.entity.Man;
62 import cn.sitedev.project.entity.Person;
63 import org.junit.Test;
64 import org.springframework.context.ApplicationContext;
65 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
66
67 public class MyTest {
68     @Test
69     public void test() {

```

```

70     ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
71     Person person2 = (Person) context.getBean("person2");
72     Person person3 = (Person) context.getBean("person3");
73
74     System.out.println(person2);
75     System.out.println(person3);
76
77     Man man = (Man) context.getBean(Man.class);
78     System.out.println(man);
79 }
80 }

```



Run: MyTest.test (1) x

✓ Tests passed: 1 of 1 test - 640ms

✓ MyTest (cn.sitedev.project.controller) 640ms

✓ test 640ms

D:\DevSoftWare\Java\jdk8\bin\java.exe ...

四月 27, 2020 8:24:26 下午 org.springframework.context.annotation.AnnotationConfigApplicationContext@5679c6c6: start of context hierarchy

Person(name=Tom, age=19)

Person(name=Tom, age=20)

Man(name=Tom, age=21)

Process finished with exit code 0

1.2. @ComponentScan

```

1 package cn.sitedev.project.controller;
2
3 import org.springframework.stereotype.Controller;
4
5 @Controller
6 public class MyController {
7
8 }
9 //////////////////////////////////////////////////
10 package cn.sitedev.project.controller;
11
12 public class MyController2 {
13 }

```

```
14 //////////////////////////////////////////////////
15 package cn.sitedev.project.entity;
16
17 public class Member {
18 }
19 //////////////////////////////////////////////////
20 package cn.sitedev.project.entity;
21
22 import lombok.AllArgsConstructor;
23 import lombok.Data;
24 import lombok.NoArgsConstructor;
25
26 @Data
27 @AllArgsConstructor
28 @NoArgsConstructor
29 public class Person {
30     private String name;
31     private int age;
32 }
33 //////////////////////////////////////////////////
34 package cn.sitedev.project.entity;
35
36 public class User {
37 }
38 //////////////////////////////////////////////////
39 package cn.sitedev.project.service;
40
41 import org.springframework.stereotype.Service;
42
43 @Service
44 public class MyService {
45 }
46 //////////////////////////////////////////////////
47 package cn.sitedev.project.service;
48
49 import cn.sitedev.project.dao.MyDao2;
50
51 public class MyService2 {
52
53 }
54 //////////////////////////////////////////////////
55 package cn.sitedev.demo.annotation.configure.componentscan;
56
```

```

57 import org.springframework.core.io.Resource;
58 import org.springframework.core.type.AnnotatedTypeMetadata;
59 import org.springframework.core.type.ClassMetadata;
60 import org.springframework.core.type.classreading.MetadataReader;
61 import org.springframework.core.type.classreading.MetadataReaderFactory;
62 import org.springframework.core.type.filter.TypeFilter;
63
64 import java.io.IOException;
65
66 public class MyTypeFilter implements TypeFilter {
67
68     /**
69      * @param metadataReader      获取当前正在操作的类的信息
70      * @param metadataReaderFactory 获取上下文中所有的数据
71      * @return
72      * @throws IOException
73      */
74     @Override
75     public boolean match(MetadataReader metadataReader,
76                         MetadataReaderFactory metadataReaderFactory) throws IOException {
77         // 获取当前类的所有的注解信息
78         AnnotatedTypeMetadata annotatedTypeMetadata = metadataReader.getAnnotationMetadata();
79         // 获取当前扫描到的类的信息
80         ClassMetadata classMetadata = metadataReader.getClassMetadata();
81         // 获取当前类的所有的资源
82         Resource resource = metadataReader.getResource();
83         String className = classMetadata.getClassName();
84
85         if (className.contains("er")) {
86             System.out.println("=====" + className + "=====");
87             return true;
88         }
89         return false;
90     }
91 }
92 ///////////////////////////////////////////////////
93 package cn.sitedev.demo.annotation.configure.componentscan;
94
95 import cn.sitedev.project.controller.MyController2;
96 import org.springframework.context.annotation.ComponentScan;
97 import org.springframework.context.annotation.Configuration;
98 import org.springframework.context.annotation.FilterType;
99 import org.springframework.stereotype.Controller;

```

```
100
101 @Configuration
102 // FilterType.ANNOTATION => 注解
103 // FilterType.ASSIGNABLE_TYPE => 类
104 // FilterType.CUSTOM => 自定义
105 @ComponentScan(value = "cn.sitedev.project", includeFilters = {@ComponentScan.Filter(type
106     FilterType.ANNOTATION, value = {Controller.class}), @ComponentScan.Filter(type
107     FilterType.ASSIGNABLE_TYPE, value = MyController2.class), @ComponentScan.Filter
108     FilterType.CUSTOM, value = MyTypeFilter.class}), useDefaultFilters = false)
109 public class MyConfig {
110 }
111 //////////////////////////////////////
112 package cn.sitedev.demo.annotation.configure.componentscan;
113
114 import org.junit.Test;
115 import org.springframework.context.ApplicationContext;
116 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
117
118 public class MyTest {
119     @Test
120     public void test() {
121         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
122         String[] beanNames = context.getBeanDefinitionNames();
123         for (String beanName : beanNames) {
124             System.out.println("beanName = " + beanName);
125         }
126     }
127 }
```

```
MyTest (1) x
Tests passed: 1 of 1 test - 689ms
MyTest (cn.sitedev.project.entity) 689ms
test 689ms
D:\DevSoftWare\Java\jdk8\bin\java.exe ...
四月 27, 2020 8:36:57 下午 org.springframework.context.support.AbstractApplicationContext
prepareRefresh
信息: Refreshing org.springframework.context.annotation
.AnnotationConfigApplicationContext@68f7aae2: startup date [Mon Apr 27 20:36:57 CST 2020]; root
of context hierarchy
=====cn.sitedev.project.entity.Member=====
=====cn.sitedev.project.entity.Person=====
=====cn.sitedev.project.entity.User=====
=====cn.sitedev.project.service.MyService=====
=====cn.sitedev.project.service.MyService2=====
beanName = org.springframework.context.annotation.internalConfigurationAnnotationProcessor
beanName = org.springframework.context.annotation.internalAutowiredAnnotationProcessor
beanName = org.springframework.context.annotation.internalRequiredAnnotationProcessor
beanName = org.springframework.context.annotation.internalCommonAnnotationProcessor
beanName = org.springframework.context.event.internalEventListenerProcessor
beanName = org.springframework.context.event.internalEventListenerFactory
beanName = myConfig
beanName = myController
beanName = myController2
beanName = member
beanName = person
beanName = user
beanName = myService
beanName = myService2

Process finished with exit code 0
|
```

1.3. @Scope

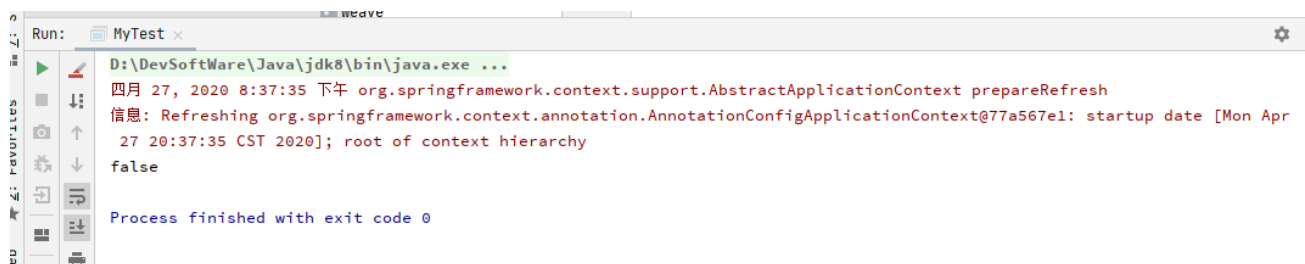
```
1 package cn.sitedev.project.entity;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class Person {
11     private String name;
12     private int age;
13 }
14 //////////////////////////////////////
15 package cn.sitedev.demo.annotation.configure.scope;
16
17 import cn.sitedev.project.entity.Person;
18 import org.springframework.context.annotation.Bean;
19 import org.springframework.context.annotation.Configuration;
20 import org.springframework.context.annotation.Scope;
21
```



```

22 @Configuration
23 public class MyConfig {
24
25     /**
26      * prototype : 原型，多例
27      * singleton : 单例，默认
28      * request : 主要应用于web模块，同一次请求只创建一个实例
29      * session : 主要应用于web模块，同一个会话只创建一个实例
30      *
31      * @return
32      */
33     @Scope("prototype")
34     @Bean("person5")
35     public Person person() {
36         return new Person("Tom", 21);
37     }
38 }
39 ///////////////////////////////////////////////////
40 package cn.sitedev.demo.annotation.configure.scope;
41
42 import cn.sitedev.project.entity.Person;
43 import org.springframework.context.ApplicationContext;
44 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
45
46 public class MyTest {
47     public static void main(String[] args) {
48         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
49         Person person = (Person) context.getBean("person5");
50         Person person2 = (Person) context.getBean("person5");
51         System.out.println(person == person2);
52     }
53 }

```



```

Run: MyTest
D:\DevSoftWare\Java\jdk8\bin\java.exe ...
四月 27, 2020 8:37:35 下午 org.springframework.context.support.AbstractApplicationContext prepareRefresh
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@77a567e1: startup date [Mon Apr
27 20:37:35 CST 2020]; root of context hierarchy
false
Process finished with exit code 0

```

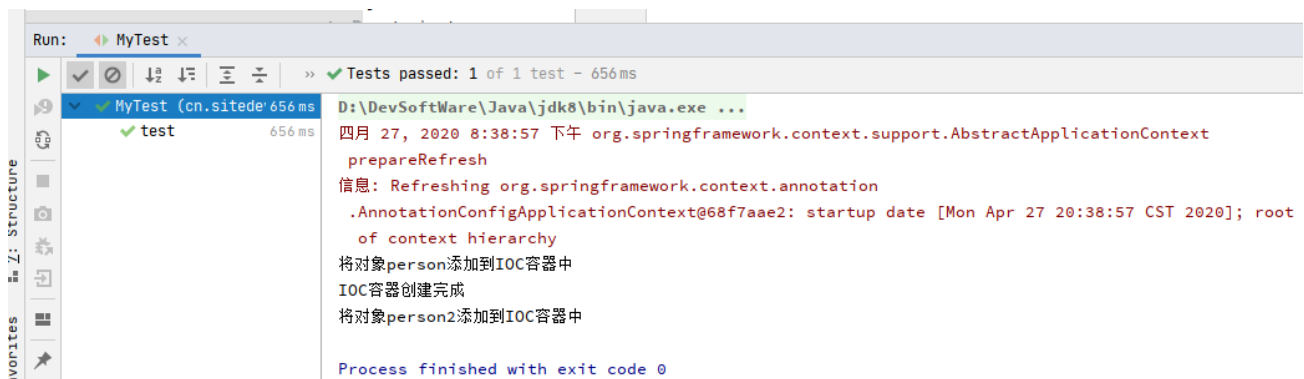
1.4. @Lazy

```
1 package cn.sitedev.project.entity;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class Person {
11     private String name;
12     private int age;
13 }
14 //////////////////////////////////////
15 package cn.sitedev.demo.annotation.configure.lazy;
16
17 import cn.sitedev.project.entity.Person;
18 import org.springframework.context.annotation.Bean;
19 import org.springframework.context.annotation.Configuration;
20 import org.springframework.context.annotation.Lazy;
21
22 @Configuration
23 public class MyConfig {
24     @Bean
25     public Person person() {
26         System.out.println("将对象person添加到IOC容器中");
27         return new Person("Tom", 26);
28     }
29
30     /**
31      * 默认是非延时加载的
32      * 延时加载,懒加载, 只针对单例Bean起作用
33      * 默认容器启动时不创建对象, 调用对象的功能时才创建
34      *
35      * @return
36      */
37     @Lazy
38     @Bean
39     public Person person2() {
40         System.out.println("将对象person2添加到IOC容器中");
41         return new Person("Tom", 27);
42     }
43 }
```

```

44 ///////////////////////////////////////////////////
45 package cn.sitedev.demo.annotation.configure.lazy;
46
47 import org.junit.Test;
48 import org.springframework.context.ApplicationContext;
49 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
50
51 public class MyTest {
52     @Test
53     public void test() {
54         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
55         System.out.println("IOC容器创建完成");
56         context.getBean("person");
57         context.getBean("person2");
58     }
59 }
60 }

```



1.5. @Conditional

```

1 package cn.sitedev.project.entity;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Data
8 @AllArgsConstructor
9 @NoArgsConstructor
10 public class Person {
11     private String name;
12     private int age;

```

```

13 }
14 //////////////////////////////////////
15 package cn.sitedev.demo.annotation.configure.conditional;
16
17 import org.springframework.beans.factory.config.ConfigurableListableBeanFactory;
18 import org.springframework.context.annotation.Condition;
19 import org.springframework.context.annotation.ConditionContext;
20 import org.springframework.core.env.Environment;
21 import org.springframework.core.type.AnnotatedTypeMetadata;
22
23 public class WinCondition implements Condition {
24
25     @Override
26     public boolean matches(ConditionContext context, AnnotatedTypeMetadata metadata) {
27         ConfigurableListableBeanFactory beanFactory = context.getBeanFactory();
28
29         Environment environment = context.getEnvironment();
30
31         String os = environment.getProperty("os.name");
32         if (os.contains("Windows")) {
33             return true;
34         }
35         return false;
36     }
37 }
38 //////////////////////////////////////
39 package cn.sitedev.demo.annotation.configure.conditional;
40
41 import org.springframework.beans.factory.config.ConfigurableListableBeanFactory;
42 import org.springframework.context.annotation.Condition;
43 import org.springframework.context.annotation.ConditionContext;
44 import org.springframework.core.env.Environment;
45 import org.springframework.core.type.AnnotatedTypeMetadata;
46
47 public class LinuxCondition implements Condition {
48     @Override
49     public boolean matches(ConditionContext context, AnnotatedTypeMetadata metadata) {
50         ConfigurableListableBeanFactory beanFactory = context.getBeanFactory();
51
52         Environment environment = context.getEnvironment();
53
54         String os = environment.getProperty("os.name");
55         if (os.contains("Linux")) {

```

```

56         return true;
57     }
58     return false;
59 }
60 }
61 //////////////////////////////////////////////////
62 package cn.sitedev.demo.annotation.configure.conditional;
63
64 import cn.sitedev.project.entity.Person;
65 import org.springframework.context.annotation.Bean;
66 import org.springframework.context.annotation.Conditional;
67
68 public class MyConfig {
69
70     @Bean
71     @Conditional(WinCondition.class)
72     public Person person() {
73         System.out.println("向IOC容器中注入person");
74         return new Person();
75     }
76
77     @Bean
78     @Conditional(LinuxCondition.class)
79     public Person person2() {
80         System.out.println("向IOC容器中注入person2");
81         return new Person();
82     }
83
84 }
85 //////////////////////////////////////////////////
86 package cn.sitedev.demo.annotation.configure.conditional;
87
88 import org.junit.Test;
89 import org.springframework.context.ApplicationContext;
90 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
91
92 public class MyTest {
93
94     @Test
95     public void test() {
96         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
97         System.out.println("IOC容器初始化完成");
98         // 假设当前操作系统如果是windows, 则加载person, 如果是 Linux,则加载person2

```

```

99
100     }
101 }

```

```

Run: MyTest (2) x
  Tests passed: 1 of 1 test - 563ms
  MyTest (cn.sitedev) 563ms
    test 563ms
      D:\DevSoftWare\Java\jdk8\bin\java.exe ...
      四月 27, 2020 8:40:07 下午 org.springframework.context.support.AbstractApplicationContext
        prepareRefresh
      信息: Refreshing org.springframework.context.annotation
        .AnnotationConfigApplicationContext@68f7aae2: startup date [Mon Apr 27 20:40:07 CST 2020]; root
        of context hierarchy
      向IOC容器中注入person
      IOC容器初始化完成

      Process finished with exit code 0

```

1.6. @Import

```

1 package cn.sitedev.project.entity;
2
3 public class Company {
4 }
5 ///////////////////////////////////////////////////
6 package cn.sitedev.project.entity;
7
8 public class Member {
9 }
10 ///////////////////////////////////////////////////
11 package cn.sitedev.project.entity;
12
13 public class Monkey {
14 }
15 ///////////////////////////////////////////////////
16 package cn.sitedev.project.entity;
17
18 public class User {
19 }
20 ///////////////////////////////////////////////////
21 package cn.sitedev.demo.annotation.configure.imports;
22
23 import cn.sitedev.project.entity.Monkey;
24 import org.springframework.beans.factory.FactoryBean;
25
26 public class MyFactoryBean implements FactoryBean<Monkey> {

```

```

27     @Override
28     public Monkey getObject() throws Exception {
29         return new Monkey();
30     }
31
32     @Override
33     public Class<?> getObjectType() {
34         return Monkey.class;
35     }
36
37     @Override
38     public boolean isSingleton() {
39         return true;
40     }
41 }
42 //////////////////////////////////////////////////
43 package cn.sitedev.demo.annotation.configure.imports;
44
45 import cn.sitedev.project.entity.User;
46 import org.springframework.beans.factory.config.BeanDefinition;
47 import org.springframework.beans.factory.support.BeanDefinitionRegistry;
48 import org.springframework.beans.factory.support.RootBeanDefinition;
49 import org.springframework.context.annotation.ImportBeanDefinitionRegistrar;
50 import org.springframework.core.type.AnnotationMetadata;
51
52 public class MyImportBeanDefinitionRegistrar implements ImportBeanDefinitionRegistrar {
53     @Override
54     public void registerBeanDefinitions(AnnotationMetadata importingClassMetadata, BeanDefinitionRegistry registry) {
55         // 包里面如果声明了Company和Member这两个类，才把User对象注册到IOC容器中
56         boolean hasCompanyBean = registry.containsBeanDefinition("cn.sitedev.project.entity.Company");
57         boolean hasMemberBean = registry.containsBeanDefinition("cn.sitedev.project.entity.Member");
58
59         if (hasCompanyBean && hasMemberBean) {
60             BeanDefinition beanDefinition = new RootBeanDefinition(User.class);
61             registry.registerBeanDefinition("user", beanDefinition);
62         }
63     }
64 }
65 //////////////////////////////////////////////////
66 package cn.sitedev.demo.annotation.configure.imports;
67
68 import org.springframework.context.annotation.ImportSelector;
69 import org.springframework.core.type.AnnotationMetadata;

```

```

70
71 public class MyImportSelector implements ImportSelector {
72     @Override
73     public String[] selectImports(AnnotationMetadata importingClassMetadata) {
74         return new String[]{"cn.sitedev.project.entity.Company", "cn.sitedev.project.er
75     }
76 }
77 //////////////////////////////////////////////////
78 package cn.sitedev.demo.annotation.configure.imports;
79
80 import cn.sitedev.project.entity.Cat;
81 import org.springframework.context.annotation.Bean;
82 import org.springframework.context.annotation.Configuration;
83 import org.springframework.context.annotation.Import;
84
85 @Configuration
86 @Import(value = {Cat.class, MyImportSelector.class, MyImportBeanDefinitionRegistrar.class})
87 public class MyConfig {
88
89     /**
90      * 给IOC容器注册Bean的方式：
91      * 1. @Bean直接导入单个类
92      * 2. @ComponentScan 默认扫描(@Controller, @Service, @Repository, @Component)
93      * 3. @Import：快速给容器导入Bean的方式
94      * 3.1. @Import直接参数导入
95      * 3.2. 实现ImportSelector，自定义实现规则
96      * 3.3. 实现ImportBeanDefinitionRegistrar
97      * 4. FactoryBean:把需要注册的对象封装为FactoryBean
98      * 4.1. FactoryBean:负责将Bean注册到IOC容器中
99      * 4.2. BeanFactory:负责从IOC容器中获取Bean对象
100     */
101
102     @Bean
103     public MyFactoryBean monkey() {
104         return new MyFactoryBean();
105     }
106 }
107 //////////////////////////////////////////////////
108 package cn.sitedev.demo.annotation.configure.imports;
109
110 import org.junit.Test;
111 import org.springframework.context.ApplicationContext;
112 import org.springframework.context.annotation.AnnotationConfigApplicationContext;

```



```

113
114 public class MyTest {
115     @Test
116     public void test() {
117         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
118         for (String beanName : context.getBeanDefinitionNames()) {
119             System.out.println(beanName);
120         }
121
122
123         System.out.println("=====");
124         Object monkey = context.getBean("monkey");
125         System.out.println("monkey = " + monkey);
126         Object monkey2 = context.getBean("&monkey");
127         System.out.println("&monkey = " + monkey2);
128     }
129 }

```

```

in: MyTest (3) x
✓ Tests passed: 1 of 1 test - 681ms
✓ MyTest (cn.sitedev 681ms)
  ✓ test 681ms
    D:\DevSoftWare\Java\jdk8\bin\java.exe ...
    四月 27, 2020 8:44:02 下午 org.springframework.context.support.AbstractApplicationContext
      prepareRefresh
    信息: Refreshing org.springframework.context.annotation
      .AnnotationConfigApplicationContext@68f7aae2: startup date [Mon Apr 27 20:44:02 CST 2020]; root
      of context hierarchy
    org.springframework.context.annotation.internalConfigurationAnnotationProcessor
    org.springframework.context.annotation.internalAutowiredAnnotationProcessor
    org.springframework.context.annotation.internalRequiredAnnotationProcessor
    org.springframework.context.annotation.internalCommonAnnotationProcessor
    org.springframework.context.event.internalEventListenerProcessor
    org.springframework.context.event.internalEventListenerFactory
    myConfig
    cn.sitedev.project.entity.Cat
    cn.sitedev.project.entity.Company
    cn.sitedev.project.entity.Member
    monkey
    user
    =====
    monkey = cn.sitedev.project.entity.Monkey@5a955565
    &monkey = cn.sitedev.demo.annotation.config.imports.MyFactoryBean@6293abcc
    Process finished with exit code 0

```

1.7. 生命周期

```

1 package cn.sitedev.project.entity;
2
3 import org.springframework.stereotype.Component;
4
5 import javax.annotation.PostConstruct;

```

```
6 import javax.annotation.PreDestroy;
7
8 @Component
9 public class AirPlane {
10     public AirPlane() {
11         System.out.println("调用AirPlane的构造方法");
12     }
13
14     @PostConstruct
15     public void addOil() {
16         System.out.println("飞机起飞前加油");
17     }
18
19     public void run() {
20         System.out.println("飞机在空中飞行");
21     }
22
23     @PreDestroy
24     public void stop() {
25         System.out.println("飞机停止飞行");
26     }
27 }
28 ///////////////////////////////////////////////////
29 package cn.sitedev.project.entity;
30
31 public class Car {
32     public Car() {
33         System.out.println("调用Car的构造方法");
34     }
35
36     public void addOil() {
37         System.out.println("加油");
38     }
39
40     public void run() {
41         System.out.println("行驶");
42     }
43
44     public void stop() {
45         System.out.println("停止");
46     }
47 }
48 ///////////////////////////////////////////////////
```

```

49 package cn.sitedev.project.entity;
50
51 import org.springframework.beans.factory.DisposableBean;
52 import org.springframework.beans.factory.InitializingBean;
53 import org.springframework.stereotype.Component;
54
55 @Component
56 public class Train implements InitializingBean, DisposableBean {
57     @Override
58     public void destroy() throws Exception {
59         System.out.println("火车对象销毁");
60     }
61
62     @Override
63     public void afterPropertiesSet() throws Exception {
64         System.out.println("火车对象初始化");
65     }
66 }
67 ///////////////////////////////////////////////////
68 package cn.sitedev.demo.annotation.configure.lifecycle;
69
70 import org.springframework.beans.BeansException;
71 import org.springframework.beans.factory.config.BeanPostProcessor;
72 import org.springframework.stereotype.Component;
73
74 @Component
75 public class MyBeanPostProcessor implements BeanPostProcessor {
76     @Override
77     public Object postProcessBeforeInitialization(Object bean, String beanName) throws BeansException {
78         System.out.println("MyBeanPostProcessor.postProcessBeforeInitialization : " + beanName);
79         return bean;
80     }
81
82     @Override
83     public Object postProcessAfterInitialization(Object bean, String beanName) throws BeansException {
84         System.out.println("MyBeanPostProcessor.postProcessAfterInitialization : " + beanName);
85         return bean;
86     }
87 }
88 ///////////////////////////////////////////////////
89 package cn.sitedev.demo.annotation.configure.lifecycle;
90
91 import cn.sitedev.project.entity.Car;

```

```
92 import org.springframework.context.annotation.Bean;
93 import org.springframework.context.annotation.ComponentScan;
94 import org.springframework.context.annotation.ComponentScans;
95 import org.springframework.context.annotation.Configuration;
96
97 @Configuration
98 @ComponentScans({@ComponentScan("cn.sitedev.project.entity"), @ComponentScan("cn.sitedev.project.service")})
99 public class MyConfig {
100
101     /**
102      * 对Bean生命周期的控制
103      * 1. 配置@Bean的参数
104      * 2. 分别实现InitializingBean和DisposableBean接口
105      * 3. 使用@PostConstruct和@PreDestroy注解
106      * 4. 自己编写一个类，实现BeanPostProcessor接口
107      *
108      * @return
109      */
110     @Bean(initMethod = "addOil", destroyMethod = "stop")
111     public Car car() {
112         return new Car();
113     }
114 }
115 ///////////////////////////////////////////////////
116 package cn.sitedev.demo.annotation.configure.lifecycle;
117
118 import cn.sitedev.project.entity.Car;
119 import org.junit.Test;
120 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
121
122 public class MyTest {
123     @Test
124     public void test() {
125         AnnotationConfigApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
126         System.out.println("IOC容器创建完成");
127         Car car = context.getBean(Car.class);
128         car.run();
129         context.close();
130     }
131 }
```

```
MyTest.test x
Tests passed: 1 of 1 test - 656ms
MyTest (cn.sitedev. 656ms)
test 656ms
四月 27, 2020 8:45:41 下午 org.springframework.context.support.AbstractApplicationContext prepareRefresh
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext@5679c6c6:
startup date [Mon Apr 27 20:45:41 CST 2020]; root of context hierarchy
MyBeanPostProcessor.postProcessBeforeInitialization : org.springframework.context.event
.EventListenerMethodProcessor@475e586c, org.springframework.context.event
.internalEventListenerProcessor
MyBeanPostProcessor.postProcessAfterInitialization : org.springframework.context.event
.EventListenerMethodProcessor@475e586c, org.springframework.context.event
.internalEventListenerProcessor
MyBeanPostProcessor.postProcessBeforeInitialization : org.springframework.context.event
.DefaultEventListenerFactory@436a4e4b, org.springframework.context.event.internalEventListenerFactory
MyBeanPostProcessor.postProcessAfterInitialization : org.springframework.context.event
.DefaultEventListenerFactory@436a4e4b, org.springframework.context.event.internalEventListenerFactory
MyBeanPostProcessor.postProcessBeforeInitialization : cn.sitedev.demo.annotation.configure.lifecycle
.MyConfig$$EnhancerBySpringCGLIB$$2929b55f@f2f2cc1, myConfig
MyBeanPostProcessor.postProcessAfterInitialization : cn.sitedev.demo.annotation.configure.lifecycle
.MyConfig$$EnhancerBySpringCGLIB$$2929b55f@f2f2cc1, myConfig
调用AirPlane的构造方法
MyBeanPostProcessor.postProcessBeforeInitialization : cn.sitedev.project.entity.AirPlane@5ad851c9,
airPlane
飞机起飞前加油
MyBeanPostProcessor.postProcessAfterInitialization : cn.sitedev.project.entity.AirPlane@5ad851c9,
airPlane
MyBeanPostProcessor.postProcessBeforeInitialization : cn.sitedev.project.entity.Train@b62fe6d, train
火车对象初始化
MyBeanPostProcessor.postProcessAfterInitialization : cn.sitedev.project.entity.Train@b62fe6d, train
调用Car的构造方法
MyBeanPostProcessor.postProcessBeforeInitialization : cn.sitedev.project.entity.Car@501edcf1, car
加油
MyBeanPostProcessor.postProcessAfterInitialization : cn.sitedev.project.entity.Car@501edcf1, car
IOC容器创建完成
行驶
四月 27, 2020 8:45:41 下午 org.springframework.context.support.AbstractApplicationContext doClose
信息: Closing org.springframework.context.annotation.AnnotationConfigApplicationContext@5679c6c6:
startup date [Mon Apr 27 20:45:41 CST 2020]; root of context hierarchy
停止
火车对象销毁
飞机停止飞行
Process finished with exit code 0
```

2. 赋值(自动装配)组件

注解名称	说明
@Component	泛指组件，当组件不好归类的时候，我们可以使用这个注解进行标注。
@Service	用于标注业务层组件
@Controller	用于标注控制层组件
@Repository	用于标注数据访问组件，即DAO组件。
@Value	普通数据类型赋值
@Autowired	默认按类型装配，如果我们想使用按名称装配，可以结合@Qualifier注解一起使用
@PropertySource	读取配置文件赋值
@Qualifier	如存在多个实例配合使用
@Primary	自动装配时当出现多个Bean候选者时，被注解为@Primary的Bean将作为首选者，否则将抛出异常
@Resource	默认按名称装配，当找不到与名称匹配的bean才会按类型装配。

2.1. @Component, @Controller, @Service, @Repository

2.2. @Value

```
1 package cn.sitedev.project.entity;
2
3 import lombok.Data;
4 import org.springframework.beans.factory.annotation.Value;
5
6 @Data
7 public class Bird {
8     /**
9      * 支持的类型：
10     * 1. 基本数据类型
11     * 2. 支持Spring EL表达式
12     */
13     @Value("鹦鹉")
14     private String name;
15
16     @Value("#{8-5}")
17     private int age;
18
19     @Value("${bird.color}")
20     private String color;
21 }
22 ///////////////////////////////////////////////////
23 package cn.sitedev.demo.annotation.injection.value;
24
25 import cn.sitedev.project.entity.Bird;
26 import org.springframework.context.annotation.Bean;
27 import org.springframework.context.annotation.Configuration;
28
29 @Configuration
30 public class MyConfig {
31     @Bean
32     public Bird bird() {
33         return new Bird();
34     }
35 }
36 ///////////////////////////////////////////////////
37 package cn.sitedev.demo.annotation.injection.value;
38
39 import cn.sitedev.project.entity.Bird;
40 import org.springframework.context.ApplicationContext;
```

```

41 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
42
43 public class MyTest {
44     public static void main(String[] args) {
45         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.cl
46         Bird bird = (Bird) context.getBean("bird");
47         System.out.println(bird);
48
49     }
50 }

```



```

Run: MyTest x
D:\DevSoftWare\Java\jdk8\bin\java.exe ...
四月 27, 2020 8:50:07 下午 org.springframework.context.support.AbstractApplica
信息: Refreshing org.springframework.context.annotation.AnnotationConfigAppli
20:50:07 CST 2020]; root of context hierarchy
Bird(name=鸚鵡, age=3, color=${bird.color})
Process finished with exit code 0

```

2.3. @Autowired

```

1 package cn.sitedev.project.service;
2
3 import cn.sitedev.project.dao.MyDao;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.stereotype.Service;
6
7 @Service
8 public class MyService {
9     @Autowired
10    private MyDao myDao;
11
12    public void printMyDao() {
13        System.out.println(myDao);
14    }
15
16 }
17 ///////////////////////////////////////////////////
18 package cn.sitedev.project.dao;
19

```

```

20 import lombok.ToString;
21 import org.springframework.stereotype.Repository;
22
23 @Repository
24 @ToString
25 public class MyDao {
26     public String flag = "1";
27 }
28 //////////////////////////////////////
29 package cn.sitedev.demo.annotation.injection.autowired;
30
31 import org.springframework.context.annotation.ComponentScan;
32 import org.springframework.context.annotation.Configuration;
33
34 @Configuration
35 @ComponentScan({"cn.sitedev.project.controller", "cn.sitedev.project.service", "cn.sitedev.project.dao"})
36 public class MyConfig {
37 }
38 //////////////////////////////////////
39 package cn.sitedev.demo.annotation.injection.autowired;
40
41 import cn.sitedev.project.dao.MyDao;
42 import cn.sitedev.project.service.MyService;
43 import org.springframework.context.ApplicationContext;
44 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
45
46 public class MyTest {
47     public static void main(String[] args) {
48         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
49         MyService service = (MyService) context.getBean("myService");
50         service.printMyDao();
51
52         MyDao dao = (MyDao) context.getBean("myDao");
53         System.out.println(dao);
54
55
56     }
57 }

```




2.4. @PropertySource

value.properties

```
1 bird.color=blue
```

```

1 package cn.sitedev.project.entity;
2
3 import lombok.Data;
4 import org.springframework.beans.factory.annotation.Value;
5
6 @Data
7 public class Bird {
8     /**
9      * 支持的类型:
10     * 1. 基本数据类型
11     * 2. 支持Spring EL表达式
12     */
13     @Value("鸚鵡")
14     private String name;
15
16     @Value("#{8-5}")
17     private int age;
18
19     @Value("${bird.color}")
20     private String color;
21 }
22 ///////////////////////////////////////////////////

```

```

23 package cn.sitedev.demo.annotation.injection.propertysource;
24
25 import cn.sitedev.project.entity.Bird;
26 import org.springframework.context.annotation.Bean;
27 import org.springframework.context.annotation.Configuration;
28 import org.springframework.context.annotation.PropertySource;
29
30 @Configuration
31 @PropertySource("classpath:value.properties")
32 public class MyConfig {
33     @Bean
34     public Bird bird() {
35         return new Bird();
36     }
37 }
38 //////////////////////////////////////
39 package cn.sitedev.demo.annotation.injection.propertysource;
40
41 import cn.sitedev.project.entity.Bird;
42 import org.springframework.context.ApplicationContext;
43 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
44 import org.springframework.core.env.Environment;
45
46 public class MyTest {
47     public static void main(String[] args) {
48         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
49         Bird bird = (Bird) context.getBean("bird");
50         System.out.println(bird);
51
52         System.out.println("=====从环境变量中取值=====");
53         Environment environment = context.getEnvironment();
54         System.out.println(environment.getProperty("bird.color"));
55
56     }
57 }

```

```
Run: MyTest (2) x
D:\DevSoftWare\Java\jdk8\bin\java.exe ...
四月 27, 2020 8:51:25 下午 org.springframework.context.support.Abstract
信息: Refreshing org.springframework.context.annotation.AnnotationConf
20:51:25 CST 2020]; root of context hierarchy
Bird(name=鸚鵡, age=3, color=blue)
=====从环境变量中取值=====
blue
Process finished with exit code 0
```

2.5. @Qualifier

```
1 package cn.sitedev.project.service;
2
3 import cn.sitedev.project.dao.MyDao;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.beans.factory.annotation.Qualifier;
6 import org.springframework.stereotype.Service;
7
8 @Service
9 public class MyService {
10
11     @Qualifier("dao")
12     @Autowired
13     private MyDao myDao;
14
15     public void printMyDao() {
16         System.out.println(myDao);
17     }
18
19 }
20 ///////////////
21 package cn.sitedev.project.dao;
22
23 import lombok.ToString;
24 import org.springframework.stereotype.Repository;
25
26 @Repository
27 @ToString
28 public class MyDao {
29     public String flag = "1";
```

```

30 }
31 //////////////////////////////////////////////////
32 package cn.sitedev.demo.annotation.injection.qualifier;
33
34 import cn.sitedev.project.dao.MyDao;
35 import org.springframework.context.annotation.Bean;
36 import org.springframework.context.annotation.ComponentScan;
37 import org.springframework.context.annotation.Configuration;
38
39 @Configuration
40 @ComponentScan({"cn.sitedev.project.service", "cn.sitedev.project.dao"})
41 public class MyConfig {
42
43     @Bean
44     public MyDao dao() {
45         MyDao myDao = new MyDao();
46         myDao.flag = "2";
47         return myDao;
48     }
49 }
50
51 //////////////////////////////////////////////////
52 package cn.sitedev.demo.annotation.injection.qualifier;
53
54 import cn.sitedev.project.service.MyService;
55 import org.springframework.context.ApplicationContext;
56 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
57
58 public class MyTest {
59     public static void main(String[] args) {
60         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
61         MyService service = (MyService) context.getBean("myService");
62         service.printMyDao();
63     }
64 }

```

```
Run: MyTest (3) x
D:\DevSoftWare\Java\jdk8\bin\java.exe ...
四月 27, 2020 8:57:28 下午 org.springframework.context.support
信息: Refreshing org.springframework.context.annotation.AnnotationConfigApplicationContext[org.springframework.context.support.AnnotationConfigApplicationContext@20:57:28 CST 2020]; root of context hierarchy
MyDao(flag=2)
|
Process finished with exit code 0
```

2.6. @Primary

```
1 package cn.sitedev.project.dao;
2
3 import lombok.Data;
4 import org.springframework.stereotype.Repository;
5
6 @Repository
7 @Data
8 public class MyDao2 {
9     private String flag = "1";
10 }
11 ///////////////////////////////////////////////////
12 package cn.sitedev.demo.annotation.injection.primary;
13
14 import cn.sitedev.project.dao.MyDao2;
15 import org.springframework.context.annotation.Bean;
16 import org.springframework.context.annotation.Configuration;
17 import org.springframework.context.annotation.Primary;
18
19 @Configuration
20 public class MyConfig {
21     @Bean
22     public MyDao2 dao1() {
23         MyDao2 dao = new MyDao2();
24         dao.setFlag("2");
25         return dao;
26     }
27
28     @Bean
29     @Primary
```

```

30     public MyDao2 dao2() {
31         MyDao2 dao = new MyDao2();
32         dao.setFlag("4");
33         return dao;
34     }
35 }
36 //////////////////////////////////////////////////
37 package cn.sitedev.demo.annotation.injection.primary;
38
39 import cn.sitedev.project.dao.MyDao2;
40 import org.junit.Test;
41 import org.springframework.context.ApplicationContext;
42 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
43
44 public class MyTest {
45     @Test
46     public void test() {
47         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
48         Object myDao2 = context.getBean(MyDao2.class);
49         System.out.println(myDao2);
50     }
51 }

```

```

Run: MyTest x
✓ Tests passed: 1 of 1 test - 638ms
✓ MyTest (cn.sitedev.c 638ms
  ✓ test 638ms
D:\DevSoftWare\Java\jdk8\bin\java.exe ...
四月 27, 2020 8:59:29 下午 org.springframework.context:
信息: Refreshing org.springframework.context.annotat
startup date [Mon Apr 27 20:59:29 CST 2020]; root o
MyDao2(flag=4)

Process finished with exit code 0
|

```

2.7. @Resource

```

1 package cn.sitedev.project.dao;
2
3 import lombok.ToString;
4 import org.springframework.stereotype.Repository;
5

```

```

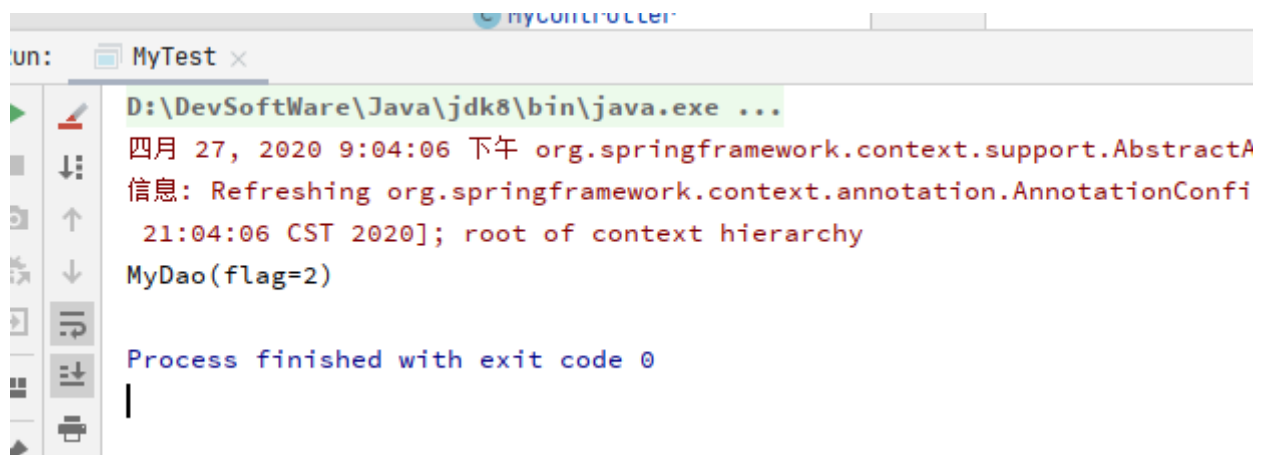
6 @Repository
7 @ToString
8 public class MyDao {
9     public String flag = "1";
10 }
11 //////////////////////////////////////////////////
12 package cn.sitedev.project.service;
13
14 import cn.sitedev.project.dao.MyDao;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.beans.factory.annotation.Qualifier;
17 import org.springframework.stereotype.Service;
18
19 import javax.annotation.Resource;
20
21 @Service
22 public class MyService {
23     // 优先级: Resource > Qualifier > Autowired
24     @Resource(name = "dao2")
25     @Qualifier("dao")
26     @Autowired
27     private MyDao myDao;
28
29     public void printMyDao() {
30         System.out.println(myDao);
31     }
32
33 }
34 //////////////////////////////////////////////////
35 package cn.sitedev.demo.annotation.injection.resource;
36
37 import cn.sitedev.project.dao.MyDao;
38 import org.springframework.context.annotation.Bean;
39 import org.springframework.context.annotation.ComponentScan;
40 import org.springframework.context.annotation.Configuration;
41
42 @Configuration
43 @ComponentScan({"cn.sitedev.project.service", "cn.sitedev.project.dao"})
44 public class MyConfig {
45
46     @Bean
47     public MyDao dao() {
48         MyDao myDao = new MyDao();

```

```

49     myDao.flag = "4";
50     return myDao;
51
52 }
53
54 @Bean
55 public MyDao dao2() {
56     MyDao myDao = new MyDao();
57     myDao.flag = "2";
58     return myDao;
59 }
60 }
61 ///////////////////////////////////////////////////
62 package cn.sitedev.demo.annotation.injection.resource;
63
64 import cn.sitedev.project.service.MyService;
65 import org.springframework.context.ApplicationContext;
66 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
67
68 public class MyTest {
69     public static void main(String[] args) {
70         ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);
71         MyService service = (MyService) context.getBean("myService");
72         service.printMyDao();
73     }
74 }

```



```

run:  D:\DevSoftWare\Java\jdk8\bin\java.exe ...
四月 27, 2020 9:04:06 下午 org.springframework.context.support.AbstractA
信息: Refreshing org.springframework.context.annotation.AnnotationConf
21:04:06 CST 2020]; root of context hierarchy
MyDao(flag=2)
Process finished with exit code 0

```

3. 织入组件

注解名称	说明
ApplicationContextAware	可以通过这个上下文环境对象得到Spring容器中的Bean
BeanDefinitionRegistryPostProcessor	BeanDefinitionRegistryPostProcessor实现了BeanFactoryPostProcessor接口，是Spring框架的BeanDefinitionRegistry的后处理器，用来注册额外的BeanDefinition

4. 切面组件

注解名称	说明
@EnableTransactionManagement	添加对事务管理的支持
@Transactional	配置声明式事务信息