https://mp.weixin.qq.com/s/mVdARYO5N-ZgU4e0QdAgRg

# @ComponentScan注解

## 灵魂拷问

1. @ComponentScan注解是做什么的？
2. basePackages的方式和basePackageClasses的方式有什么区别？你建议用哪个？为什么？
3. useDefaultFilters有什么用？
4. 常见的过滤器有哪些类型？说说你知道的几个
5. @ComponentScan是在哪个类中处理的？说一下大概的解析过程？

这些问题如果都ok，恭喜你，太优秀了，不知道没关系，一起来看看。

## **背景介绍**

到目前为止，介绍了2种注册bean的方式：

1、xml中bean元素的方式

2、@Bean注解标注方法的方式

通常情况下，项目中大部分类都需要交给spring去管理，按照上面这2种方式，代码量还是挺大的。

为了更方便bean的注册，Spring提供了批量的方式注册bean，方便大量bean批量注册，spring中的@ComponentScan就是干这个事情的。

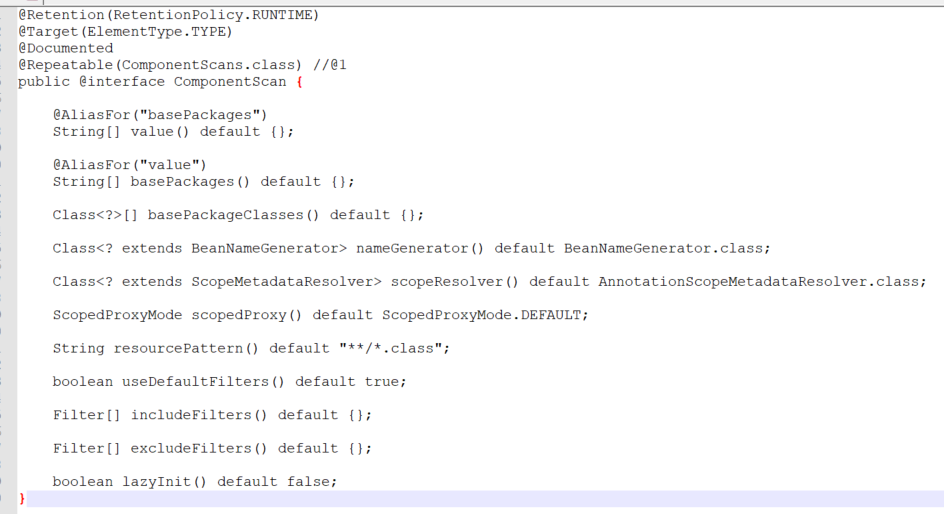
## **@ComponentScan**

@ComponentScan用于批量注册bean。

这个注解会让spring去扫描某些包及其子包中所有的类，然后将满足一定条件的类作为bean注册到spring容器容器中。

具体需要扫描哪些包？以及这些包中的类满足什么条件时被注册到容器中，这些都可以通过这个注解中的参数动态配置。

先来看一下这个注解的定义：



定义上可以看出此注解可以用在任何类型上面，不过我们通常将其用在类上面。

常用参数：

value：指定需要扫描的包，如：com.javacode2018

basePackages：作用同value；value和basePackages不能同时存在设置，可二选一

basePackageClasses：指定一些类，spring容器会扫描这些类所在的包及其子包中的类

nameGenerator：自定义bean名称生成器

resourcePattern：需要扫描包中的那些资源，默认是：\*\*/\*.class，即会扫描指定包中所有的class文件

useDefaultFilters：对扫描的类是否启用默认过滤器，默认为true

includeFilters：过滤器：用来配置被扫描出来的那些类会被作为组件注册到容器中

excludeFilters：过滤器，和includeFilters作用刚好相反，用来对扫描的类进行排除的，被排除的类不会被注册到容器中

lazyInit：是否延迟初始化被注册的bean

@1：@Repeatable(ComponentScans.class)，这个注解可以同时使用多个。

**@ComponentScan工作的过程：**

1. **Spring会扫描指定的包，且会递归下面子包，得到一批类的数组**
2. **然后这些类会经过上面的各种过滤器，最后剩下的类会被注册到容器中**

**所以玩这个注解，主要关注2个问题：**

**第一个：需要扫描哪些包？通过**value、backPackages、basePackageClasses**这3个参数来控制**

**第二：过滤器有哪些？通过**useDefaultFilters、includeFilters、excludeFilters**这3个参数来控制过滤器**

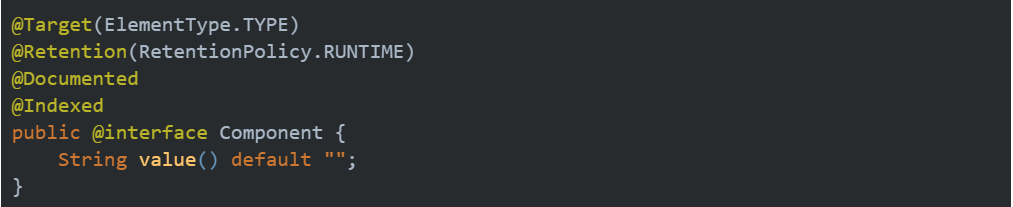
**这两个问题搞清楚了，就可以确定哪些类会被注册到容器中。**

**默认情况下，任何参数都不设置的情况下，此时，会将@ComponentScan修饰的类所在的包作为扫描包；默认情况下useDefaultFilters为true，这个为true的时候，spring容器内部会使用默认过滤器，规则是：凡是类上有**@Repository、@Service、@Controller、@Component**这几个注解中的任何一个的，那么这个类就会被作为bean注册到spring容器中，所以默认情况下，只需在类上加上这几个注解中的任何一个，这些类就会自动交给spring容器来管理了。**

## **@Component、@Repository、@Service、@Controller**

这几个注解都是spring提供的。

先说一下@Component这个注解，看一下其定义：

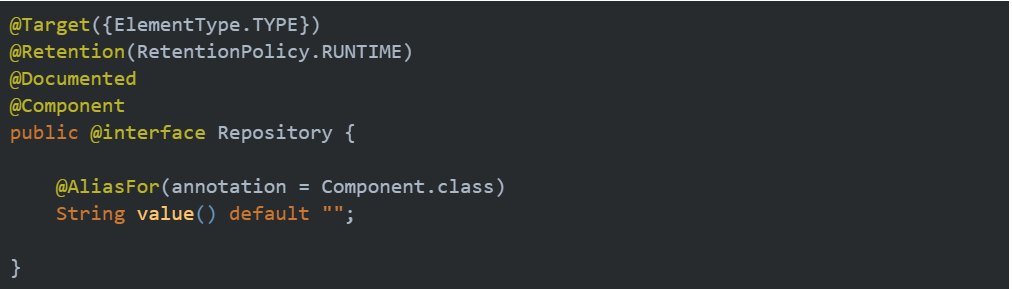


从定义中可以看出，这个注解可以用在任何类型上面。

通常情况下将这个注解用在类上面，标注这个类为一个组件，默认情况下，被扫描的时候会被作为bean注册到容器中。

value参数：被注册为bean的时候，用来指定bean的名称，如果不指定，默认为类名首字母小写。如：类UserService对应的beanname为userService

再来看看@Repository源码如下：



Repository上面有@Component注解。

value参数上面有@AliasFor(annotation = Component.class)，设置value参数的时候，也相当于给@Component注解中的value设置值。

其他两个注解@Service、@Controller源码和@Repository源码类似。

这4个注解本质上是没有任何差别，都可以用在类上面，表示这个类被spring容器扫描的时候，可以作为一个bean组件注册到spring容器中。

spring容器中对这4个注解的解析并没有进行区分，统一采用@Component注解的方式进行解析，所以这几个注解之间可以相互替换。

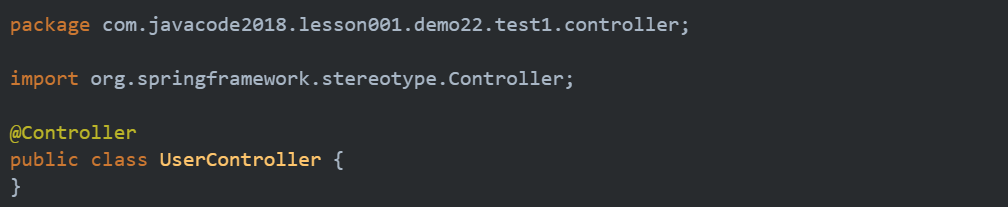
spring提供这4个注解，是为了让系统更清晰，通常情况下，系统是分层结构的，多数系统一般分为controller层、service层、dao层。

@controller通常用来标注controller层组件，@service注解标注service层的组件，@Repository标注dao层的组件，这样可以让整个系统的结构更清晰，当看到这些注解的时候，会和清晰的知道属于哪个层，对于spring来说，将这3个注解替换成@Component注解，对系统没有任何影响，产生的效果是一样的。

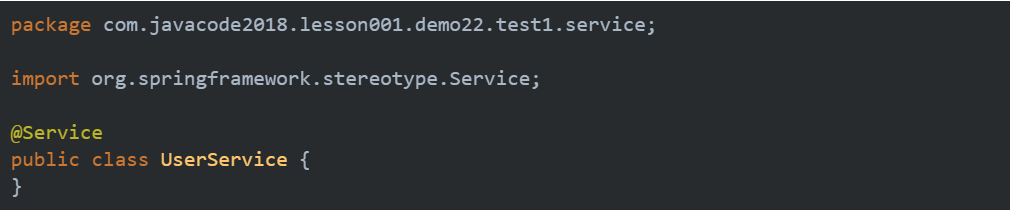
下面通过案例来感受@ComponentScan各种用法。

## **案例1：任何参数未设置**

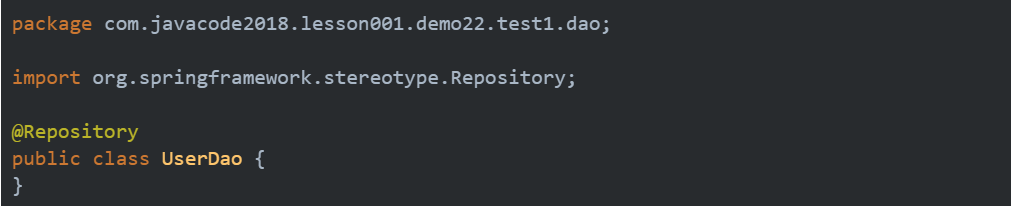
#### **UserController**



#### **UserService**



#### **UserDao**

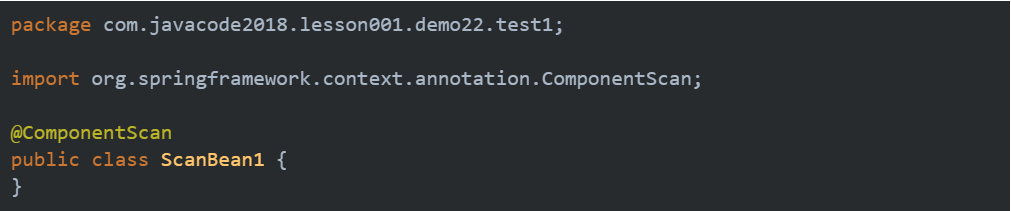


#### **UserModel**

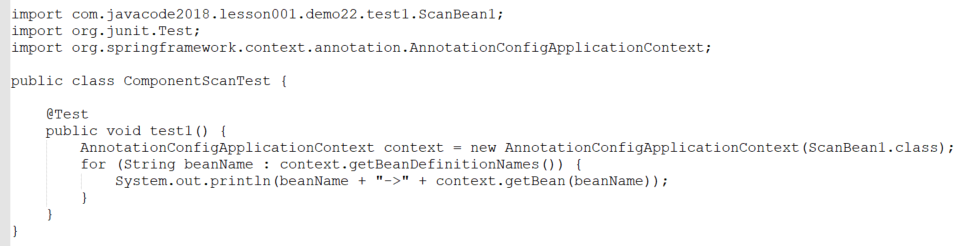


上面几个类中，分别使用了4种注解。

#### **@CompontentScan修饰的类**



#### **测试用例**

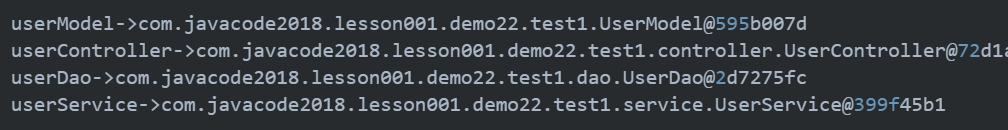


@1：使用AnnotationConfigApplicationContext作为ioc容器，将ScanBean作为参数传入。

默认会扫描ScanBean类所在的包中的所有类，类上有@Component、@Repository、@Service、@Controller任何一个注解的都会被注册到容器中

#### **运行输出**

部分输出如下：

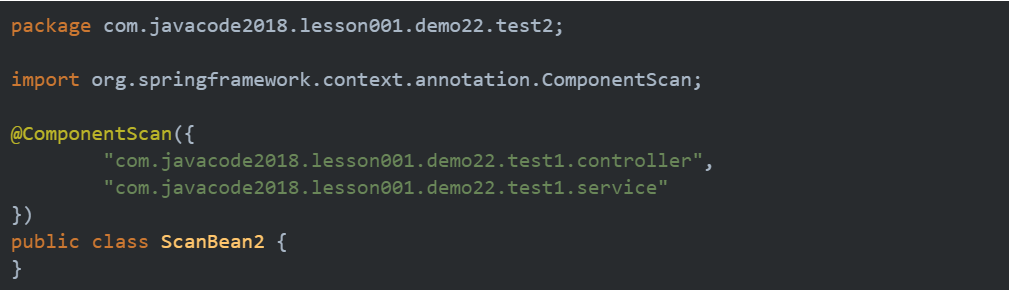


注意最后4行这几个bean，都被注册成功了。

## **案例2：指定需要扫描的包**

指定需要扫毛哪些包，可以通过value或者basePackage来配置，二者选其一，都配置运行会报错，下面我们通过value来配置。

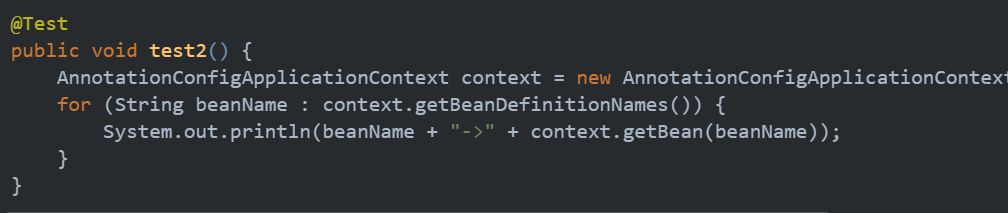
#### **ScanBean2**



上面指定了2需要扫描的包，这两个包中有2个类。

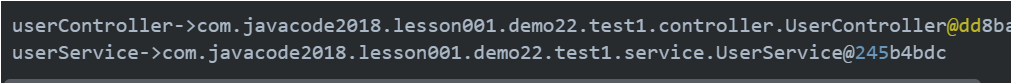
#### **测试用例**

ComponentScanTest中新增个方法



#### **运行输出**

截取了关键几行如下：



可以看出只有controller包和service包中的2个类被注册为bean了。

#### **注意**

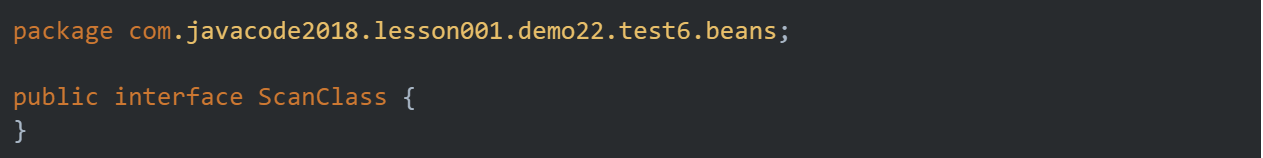
**指定包名的方式扫描存在的一个隐患，若包被重名了，会导致扫描会失效，一般情况下面我们使用basePackageClasses的方式来指定需要扫描的包，这个参数可以指定一些类型，默认会扫描这些类所在的包及其子包中所有的类，这种方式可以有效避免这种问题。**

下面来看一下basePackageClasses的方式。

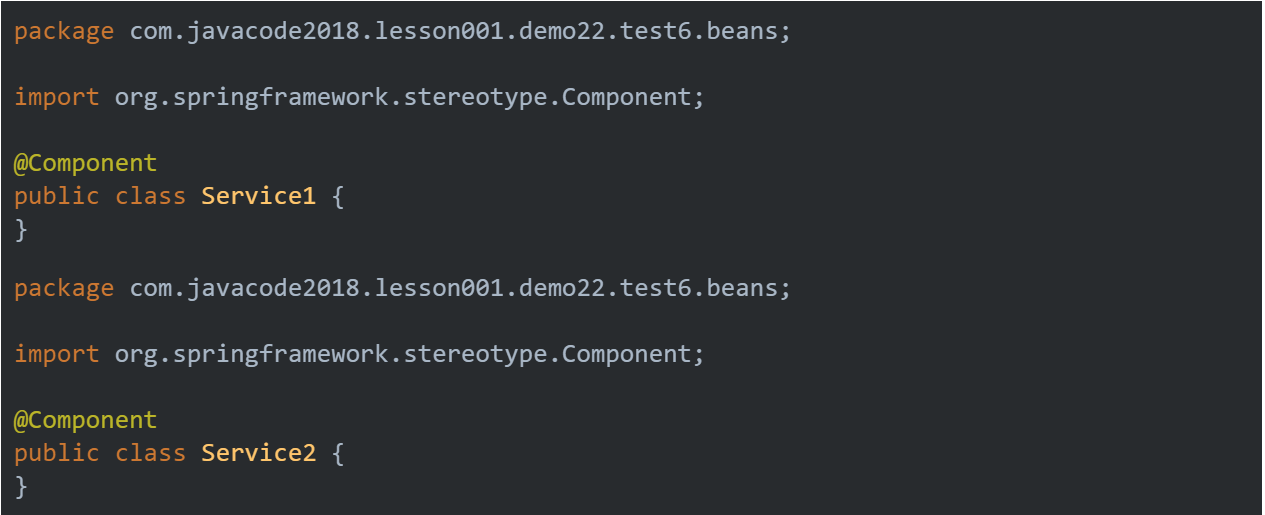
## **案例3：basePackageClasses指定扫描范围**

我们可以在需要扫描的包中定义一个标记的接口或者类，他们的唯一的作用是作为basePackageClasses的值，其他没有任何用途。

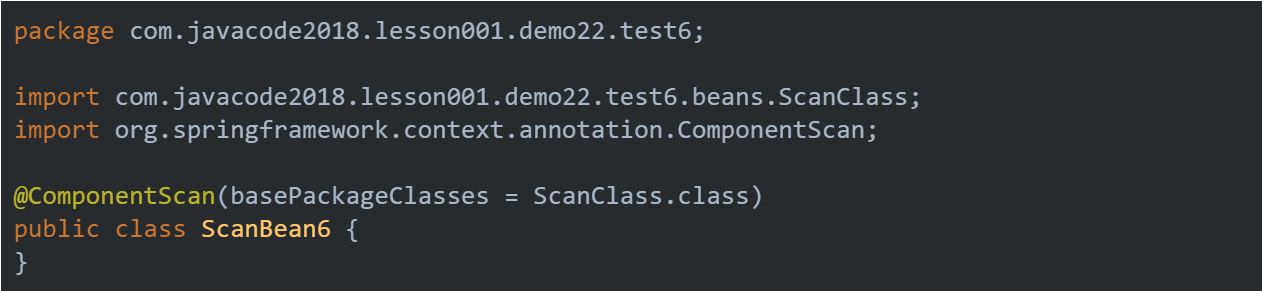
#### **下面我们定义这样一个接口**



#### **再来定义2个类，用@Component注解标记**

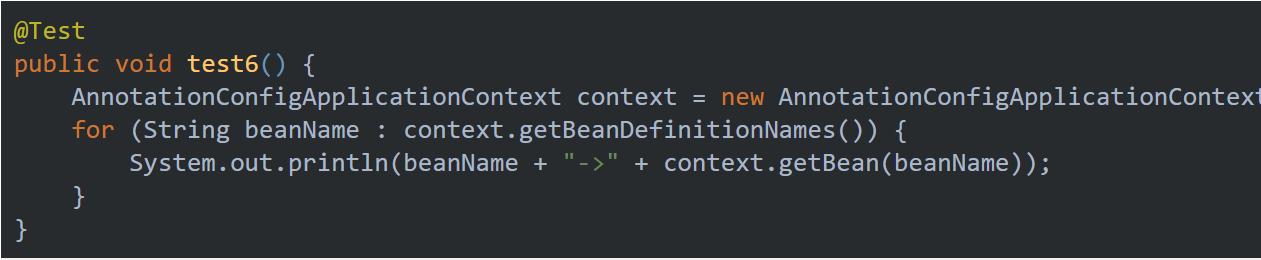


#### **来一个@CompontentScan标记的类**

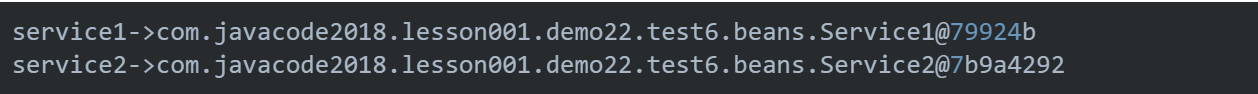


#### **测试用例**

ComponentScanTest中新增个方法



#### **运行输出**



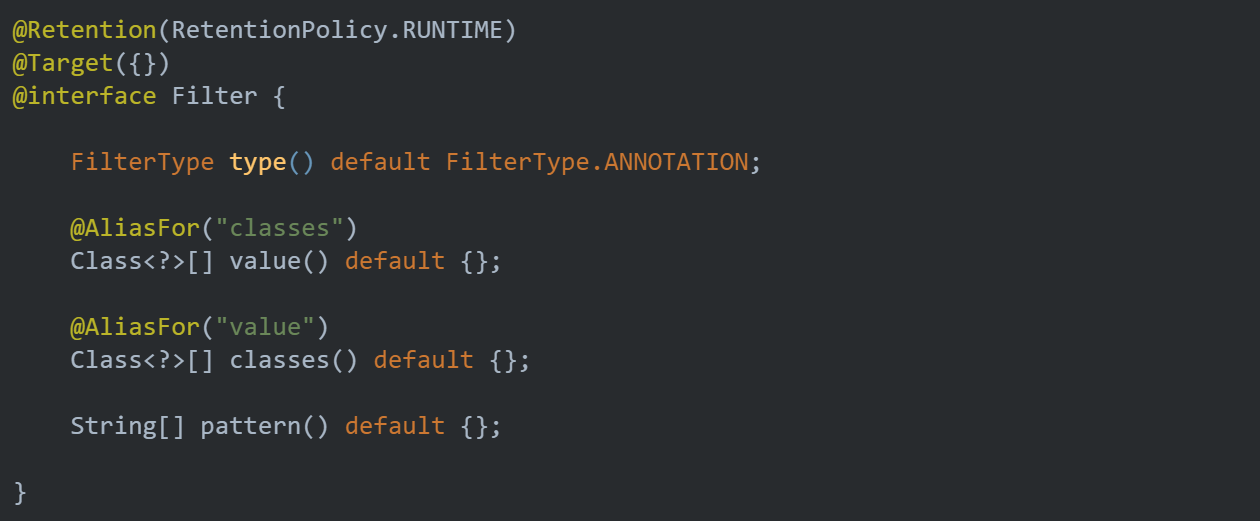
## **includeFilters的使用**

### **用法**

再来看一下includeFilters这个参数的定义：



是一个Filter类型的数组，**多个Filter之间为或者关系，即满足任意一个就可以了**，看一下Filter的代码：



可以看出Filter也是一个注解，参数：

**type：过滤器的类型，是个枚举类型，5种类型**

ANNOTATION：通过注解的方式来筛选候选者，即判断候选者是否有指定的注解

ASSIGNABLE\_TYPE：通过指定的类型来筛选候选者，即判断候选者是否是指定的类型

ASPECTJ：ASPECTJ表达式方式，即判断候选者是否匹配ASPECTJ表达式

REGEX：正则表达式方式，即判断候选者的完整名称是否和正则表达式匹配

CUSTOM：用户自定义过滤器来筛选候选者，对候选者的筛选交给用户自己来判断

**value：和参数classes效果一样，二选一**

**classes：3种情况如下**

当type=FilterType.ANNOTATION时，通过classes参数可以指定一些注解，用来判断被扫描的类上是否有classes参数指定的注解

当type=FilterType.ASSIGNABLE\_TYPE时，通过classes参数可以指定一些类型，用来判断被扫描的类是否是classes参数指定的类型

当type=FilterType.CUSTOM时，表示这个过滤器是用户自定义的，classes参数就是用来指定用户自定义的过滤器，自定义的过滤器需要实现org.springframework.core.type.filter.TypeFilter接口

**pattern：2种情况如下**

当type=FilterType.ASPECTJ时，通过pattern来指定需要匹配的ASPECTJ表达式的值

当type=FilterType.REGEX时，通过pattern来自正则表达式的值

### **案例：扫描包含注解的类**

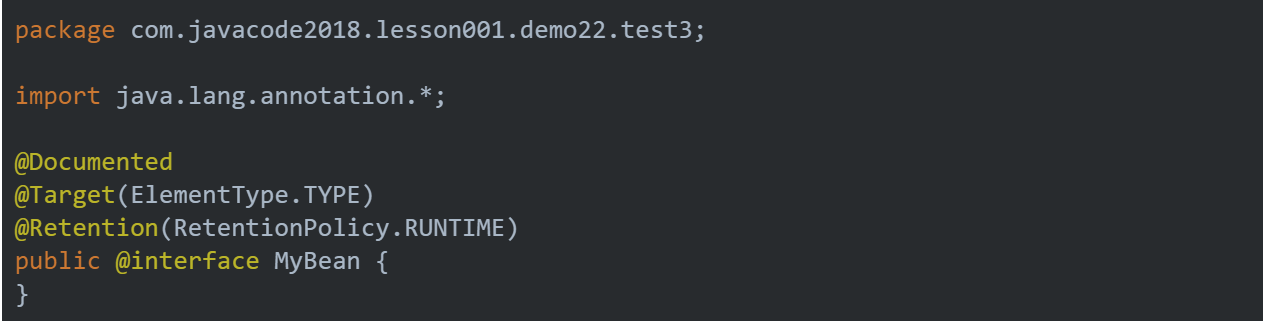
#### **需求**

我们自定义一个注解，让标注有这些注解的类自动注册到容器中

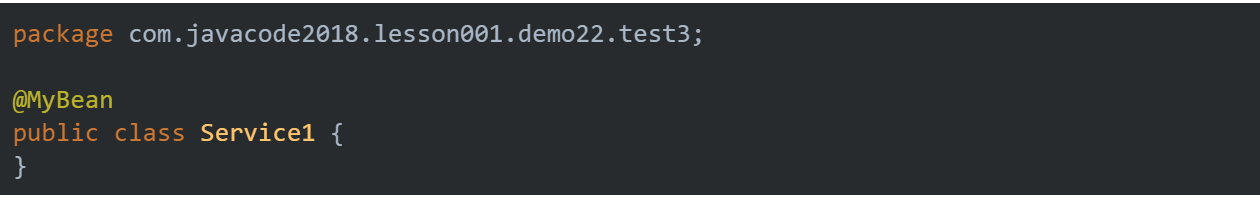
#### **代码实现**

下面的代码都在com.javacode2018.lesson001.demo22.test3包中。

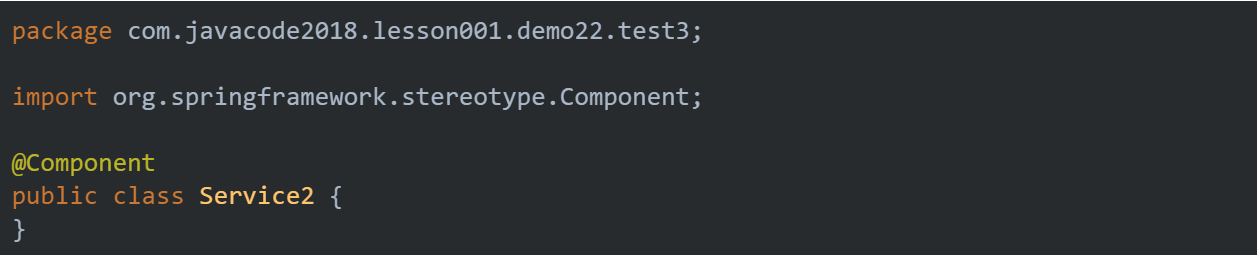
##### **定义一个注解**



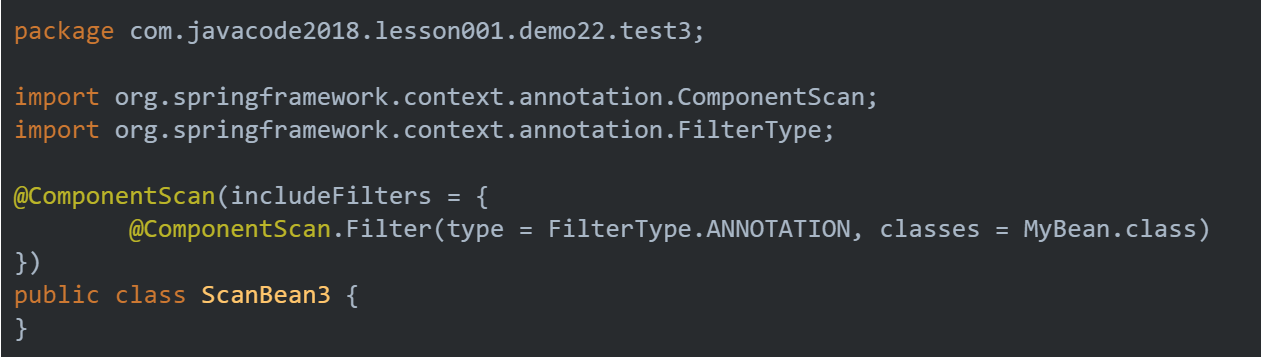
##### **创建一个类，使用这个注解标注**



##### **再来一个类，使用spring中的`@Compontent`标注**



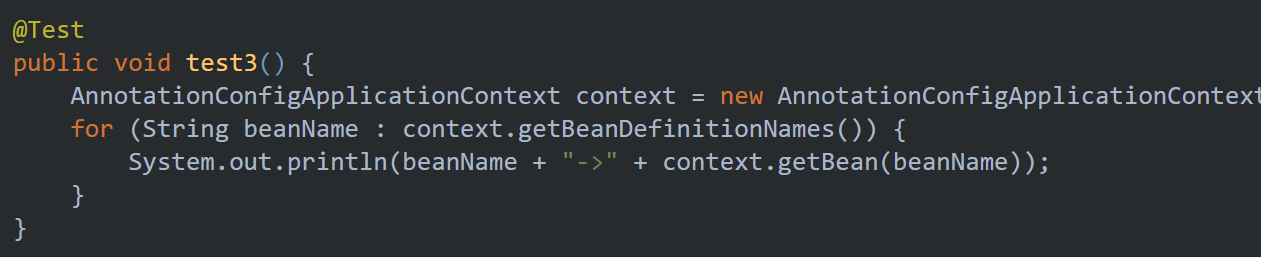
##### **再来一个类，使用@CompontentScan标注**



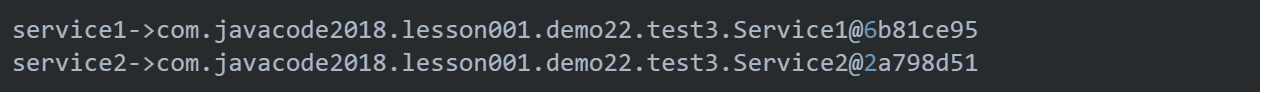
上面指定了Filter的type为注解的类型，只要类上面有@MyBean注解的，都会被作为bean注册到容器中。

##### **测试用例**

ComponentScanTest中新增个测试用例



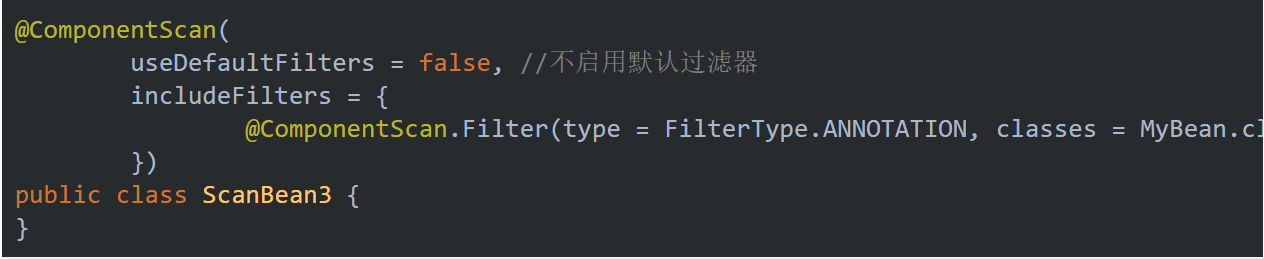
##### **运行输出，截取了主要的几行**



**Service1上标注了**@MyBean**注解，被注册到容器了，但是**Service2**上没有标注**@MyBean**啊，怎么也被注册到容器了？**

**原因：Service2上标注了**@Compontent**注解，而@CompontentScan注解中的**useDefaultFilters**默认是**true**，表示也会启用默认的过滤器，而默认的过滤器会将标注有**@Component、@Repository、@Service、@Controller**这几个注解的类也注册到容器中**

如果我们只想将标注有@MyBean注解的bean注册到容器，需要将默认过滤器关闭，即：useDefaultFilters=false，我们修改一下ScanBean3的代码如下：

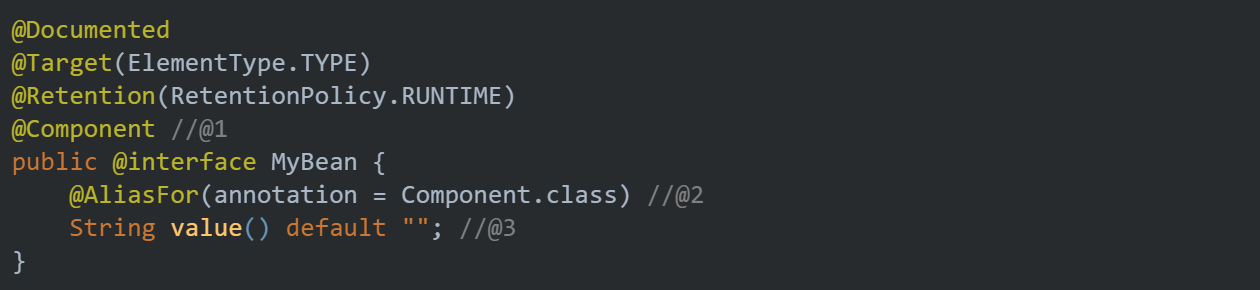


再次运行test3输出：



#### **扩展：自定义注解支持定义bean名称**

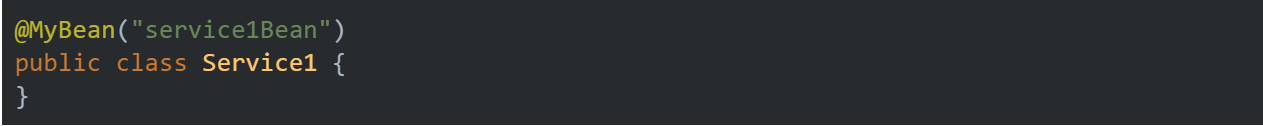
上面的自定义的@MyBean注解，是无法指定bean的名称的，可以对这个注解做一下改造，加个value参数来指定bean的名称，如下：



重点在于@1和@2这2个地方的代码，通过上面的参数可以间接给@Component注解中的value设置值。

这块用到了@AliasFor注解，对这块不了解的，可以去看一下：[java注解详解及spring对注解的增强](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648934137&idx=1&sn=3775d5d7a23c43616d1274b0b52a9f99&chksm=88621ec7bf1597d1b16d91cfb28e63bef485f10883c7ca30d09838667f65e3d214b9e1cebd47&token=1372043037&lang=zh_CN&scene=21" \l "wechat_redirect)

修改一下Service1的代码：



运行test3用例输出：

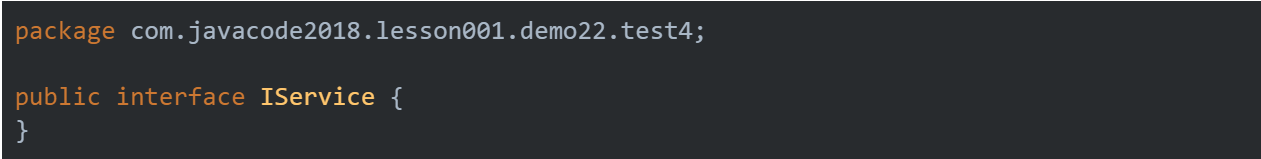


此时bean名称就变成了service1Bean。

### **案例：包含指定类型的类**

下面的代码都位于com.javacode2018.lesson001.demo22.test4包中。

#### **来个接口**



让spring来进行扫描，类型满足IService的都将其注册到容器中。

#### **来2个实现类**



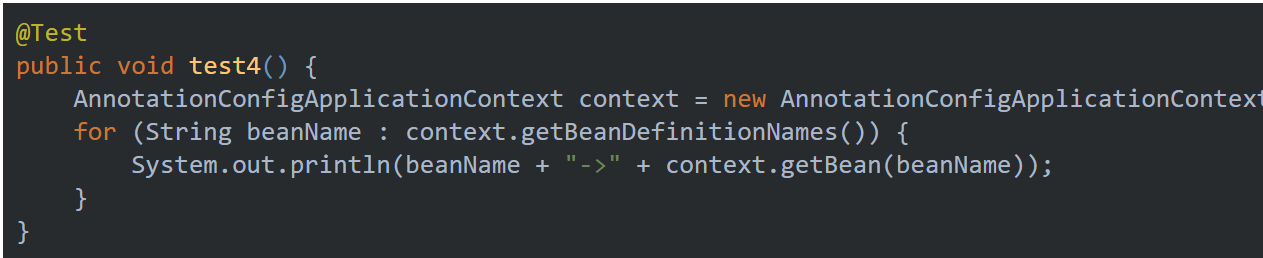
#### **来一个@CompontentScan标注的类**



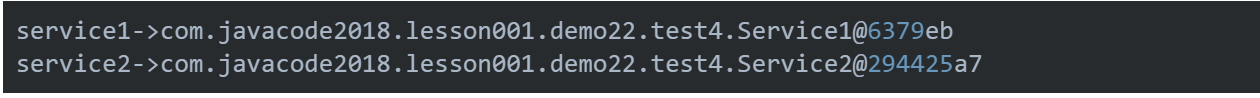
@1：被扫描的类满足IService.class.isAssignableFrom(被扫描的类)条件的都会被注册到spring容器中

#### **来个测试用例**

ComponentScanTest中新增个测试用例



#### **运行输出**



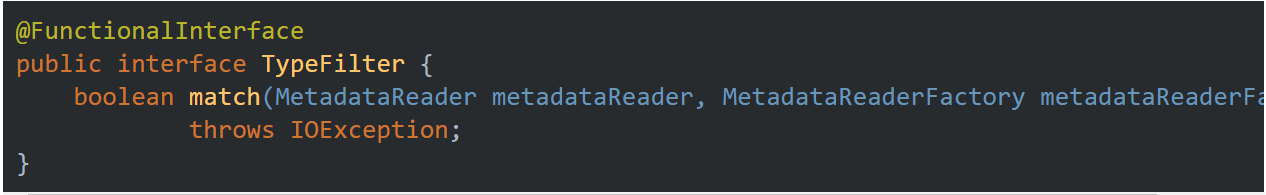
## **自定义Filter**

### **用法**

有时候我们需要用到自定义的过滤器，使用自定义过滤器的步骤：

1. 设置@Filter中type的类型为：FilterType.CUSTOM  
   2.自定义过滤器类，需要实现接口：org.springframework.core.type.filter.TypeFilter  
   3.设置@Filter中的classses为自定义的过滤器类型

来看一下TypeFilter这个接口的定义：



是一个函数式接口，包含一个match方法，方法返回boolean类型，有2个参数，都是接口类型的，下面介绍一下这2个接口。

#### **MetadataReader接口**

**类元数据读取器，可以读取一个类上的任意信息，如类上面的注解信息、类的磁盘路径信息、类的class对象的各种信息，spring进行了封装，提供了各种方便使用的方法。**

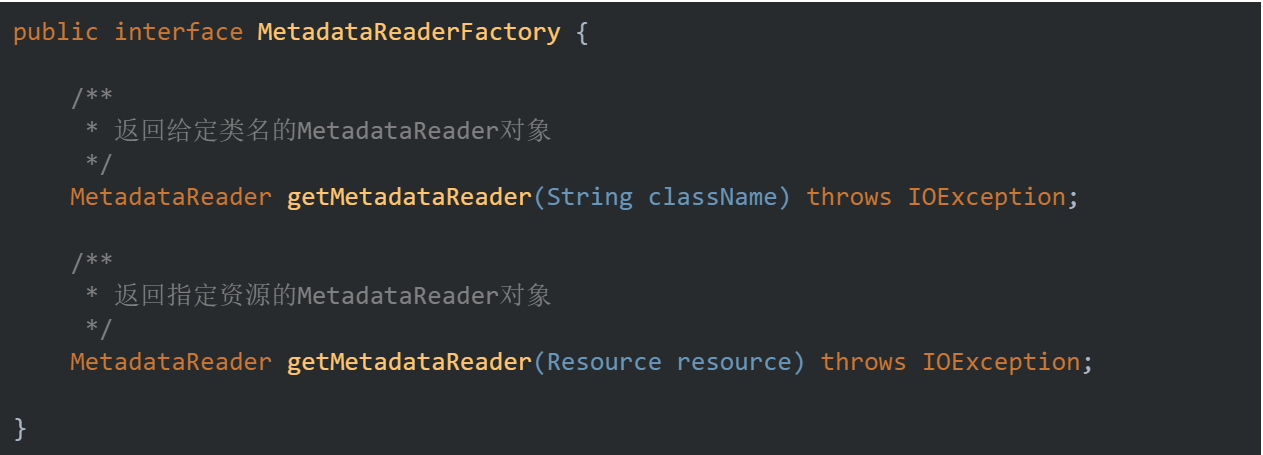
看一下这个接口的定义：



#### **MetadataReaderFactory接口**

**类元数据读取器工厂，可以通过这个类获取任意一个类的MetadataReader对象。**

源码：



### **自定义Filter案例**

#### **需求**

我们来个自定义的Filter，判断被扫描的类如果是IService接口类型的，就让其注册到容器中。

#### **代码实现**

来个自定义的TypeFilter类：



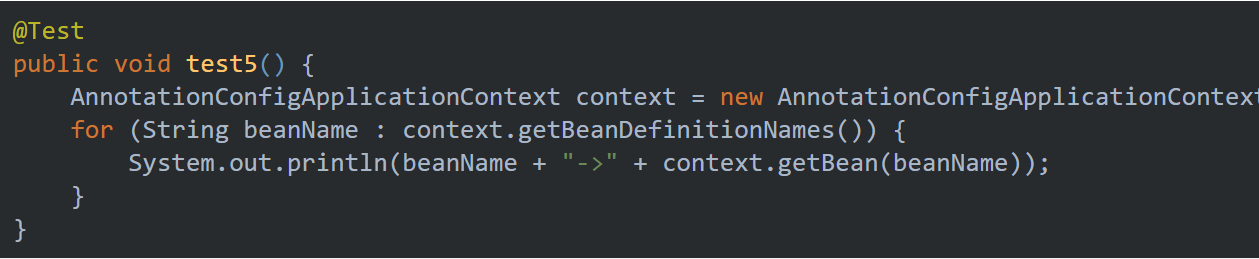
#### **来一个@CompontentScan标注的类**



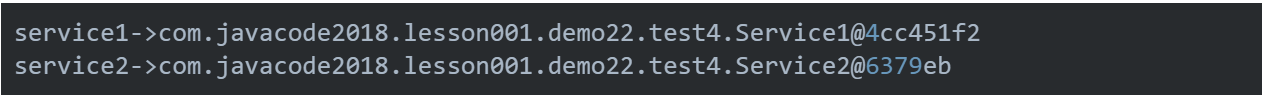
@1：type为FilterType.CUSTOM，表示Filter是用户自定义的，classes为自定义的过滤器

#### **再来个测试用例**

ComponentScanTest中新增个测试用例



#### **运行输出**

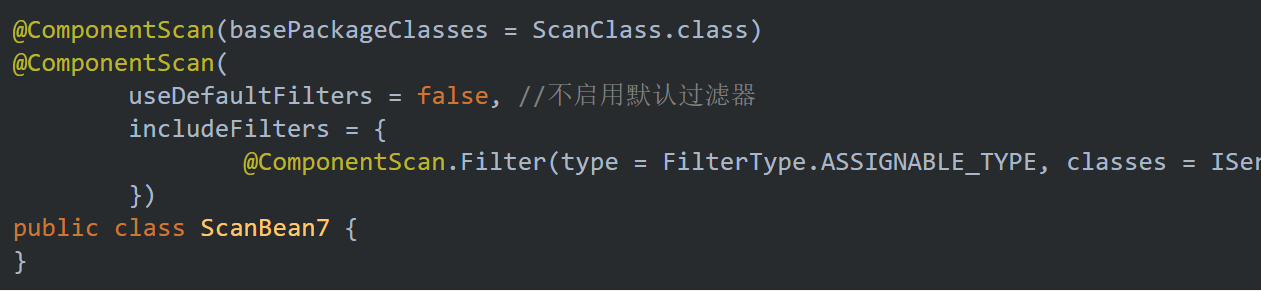


## **excludeFilters**

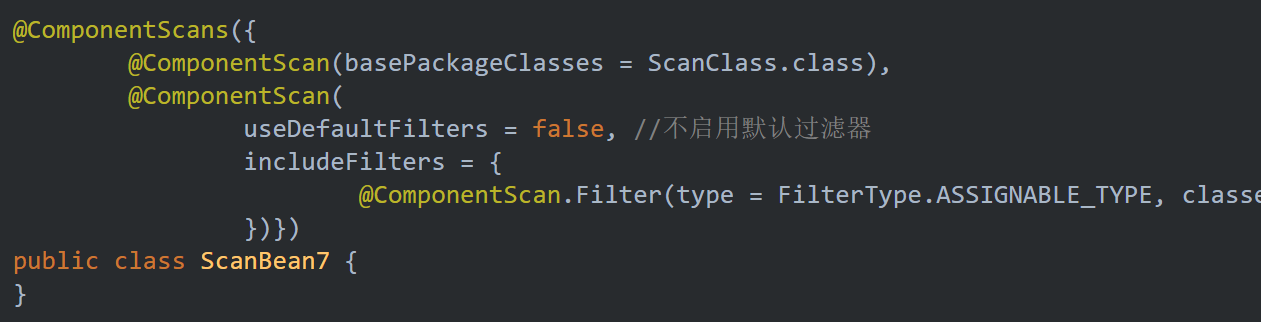
**配置排除的过滤器，满足这些过滤器的类不会被注册到容器中，用法上面和includeFilters用一样，这个我就不演示了，可以自己玩玩**

## **@ComponentScan重复使用**

从这个注解的定义上可以看出这个注解可以同时使用多个，如：



还有一种写法，使用@ComponentScans的方式：



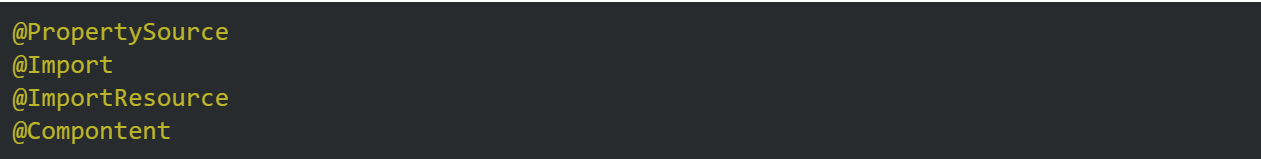
## **Spring中这块的源码**

**@CompontentScan注解是被下面这个类处理的**



**这个类非常非常关键，主要用户bean的注册，前面我们介绍的@Configuration,@Bean注解也是被这个类处理的。**

还有下面这些注解：



**以上这些注解都是被ConfigurationClassPostProcessor这个类处理的，内部会递归处理这些注解，完成bean的注册。**

**以@CompontentScan来说一下过程，第一次扫描之后会得到一批需要注册的类，然后会对这些需要注册的类进行遍历，判断是否有上面任意一个注解，如果有，会将这个类交给ConfigurationClassPostProcessor继续处理，直到递归完成所有bean的注册。**

**想成为高手，这个类是必看的。**

总结

1. @ComponentScan用于批量注册bean，spring会按照这个注解的配置，递归扫描指定包中的所有类，将满足条件的类批量注册到spring容器中
2. 可以通过value、basePackages、basePackageClasses 这几个参数来配置包的扫描范围
3. 可以通过useDefaultFilters、includeFilters、excludeFilters这几个参数来配置类的过滤器，被过滤器处理之后剩下的类会被注册到容器中
4. 指定包名的方式配置扫描范围存在隐患，包名被重命名之后，会导致扫描实现，所以一般我们在需要扫描的包中可以创建一个标记的接口或者类，作为basePackageClasses的值，通过这个来控制包的扫描范围
5. @CompontScan注解会被ConfigurationClassPostProcessor类递归处理，最终得到所有需要注册的类。