# 彻底弄懂@Controller 、@Service、@Component、@Repository

今天主要聊聊@Controller 、@Service和@Component这三个注解的关系和区别。网上很多人对这三个注解进行了详细的解释，但是仅仅局限于理论，个人对于没有经过自己验证的结果总是持怀疑态度，所有花时间研究了一下，也对这三个注解理解的更加透彻。

附上三个注解的源代码：

@Target(ElementType.TYPE)

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Indexed

public @interface Component {

String value() default "";

}

@Target({ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Component // 关键注解

public @interface Controller {

@AliasFor(annotation = Component.class)

String value() default "";

}

@Target({ElementType.TYPE})

@Retention(RetentionPolicy.RUNTIME)

@Documented

@Component // 关键注解

public @interface Service {

@AliasFor(annotation = Component.class)

String value() default "";

}

@Target(ElementType.*TYPE*)  
@Retention(RetentionPolicy.*RUNTIME*)  
@Documented  
@Component  
public @interface Repository {  
  
 */\*\*  
 \* Alias for {@link Component#value}.  
 \*/* @AliasFor(annotation = Component.class)  
 String value() default "";  
}

# 注解扫描

首先说说这三个注解的关系，从源码中可以看出，@Controller和@Service都派生于@Component，所以三者的使用方式基本没什么差别。

在平时的开发中，我们通常在控制层采用注解@Controller，在业务层采用注解@Service。spring在启动时，有一个非常核心的类**ConfigurationClassPostProcessor**会对类路径下的所以类进行扫描，将符合条件的bean扫描出来添加到beanDefinitionMap集合中，方便接下来的实例化。具体的扫描过程比较复杂，仅仅贴出核心判断逻辑代码。

**org.springframework.core.type.filter.AnnotationTypeFilter**

protected boolean matchSelf(MetadataReader metadataReader) {

AnnotationMetadata metadata = metadataReader.getAnnotationMetadata();

return metadata.hasAnnotation(this.annotationType.getName()) ||

(this.considerMetaAnnotations && metadata.hasMetaAnnotation(this.annotationType.getName()));

}

代码解释：

1、this.annotationType.getName():获取的是注解@Component的全路径名org.springframework.stereotype.Component。

2、[metadata](https://zhida.zhihu.com/search?content_id=189161544&content_type=Article&match_order=6&q=metadata&zhida_source=entity" \t "_blank).hasAnnotation(this.annotationType.getName())：判断当前的类是否直接采用注解@Component。

3、metadata.hasMetaAnnotation(this.annotationType.getName())：如果当前的类没有直接采用@Component，而是采用了类组合注解@Controller，判断组合注解@Controller中是否包含@Component。

至此，所有添加了注解@Controller、@Service和@Component都被spring扫描出来了。（ps：这就说明了其实在扫描的时候spring其实将这三个注解都按照@Component进行扫描的）

# @Controller分析

如果不使用springMVC时，三者使用其实是没有什么差别的，但如果使用了springMVC，@Controller就被赋予了特殊的含义。

spring会遍历上面扫描出来的所有bean，过滤出那些添加了注解@Controller的bean，将Controller中所有添加了注解@RequestMapping的方法解析出来封装成**RequestMappingInfo**存储到**RequestMappingHandlerMapping**中的**mappingRegistry**。后续请求到达时，会从mappingRegistry中查找能够处理该请求的方法。

部分核心代码如下：

**org.springframework.web.reactive.result.method.annotation.RequestMappingHandlerMapping**

protected boolean isHandler(Class<?> beanType) {

// 判断扫描出来的bean是否包含注解@Controller，

// 如果包含，springMVC会将其封装为RequestMappingInfo

return (AnnotatedElementUtils.hasAnnotation(beanType, Controller.class) ||

AnnotatedElementUtils.hasAnnotation(beanType, RequestMapping.class));

}

private RequestMappingInfo createRequestMappingInfo(AnnotatedElement element) {

// 判断传递进来的方法是否包含@RequestMapping，

// 如果包含，就将其封装成RequestMappingInfo

RequestMapping requestMapping = AnnotatedElementUtils.findMergedAnnotation(element, RequestMapping.class);

RequestCondition<?> condition = (element instanceof Class ?

getCustomTypeCondition((Class<?>) element) : getCustomMethodCondition((Method) element));

return (requestMapping != null ? createRequestMappingInfo(requestMapping, condition) : null);

}

# @Service分析

目前@Service本人没有找到其特殊之处，可能spring官方后续会添加特殊的操作吧。

# @Component分析

该注解是万能的注解，通常加在配置类上。

# @Repository分析