**[SpringBoot之常用注解](https://www.cnblogs.com/nihaorz/p/10528121.html)**

**在spring boot中，摒弃了spring以往项目中大量繁琐的配置，遵循约定大于配置的原则，通过自身默认配置，极大的降低了项目搭建的复杂度。同样在spring boot中，大量注解的使用，使得代码看起来更加简洁，提高开发的效率。这些注解不光包括spring boot自有，也有一些是继承自spring的。**

**本文中将spring boot项目中常用的一些核心注解归类总结，并结合实际使用的角度来解释其作用。**

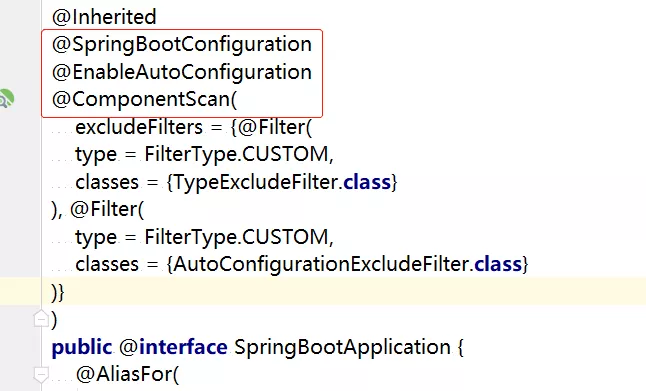
**• 项目配置注解**

**1、@SpringBootApplication 注解**

**查看源码可发现，@SpringBootApplication是一个复合注解，包含了@SpringBootConfiguration，@EnableAutoConfiguration，@ComponentScan这三个注解。**

**这三个注解的作用分别为：**

* **@SpringBootConfiguration:标注当前类是配置类，这个注解继承自@Configuration。并会将当前类内声明的一个或多个以@Bean注解标记的方法的实例纳入到srping容器中，并且实例名就是方法名。**
* **@EnableAutoConfiguration:是自动配置的注解，这个注解会根据我们添加的组件jar来完成一些默认配置，我们做微服时会添加spring-boot-starter-web这个组件jar的pom依赖，这样配置会默认配置springmvc 和tomcat。**
* **@ComponentScan:扫描当前包及其子包下被@Component，@Controller，@Service，@Repository注解标记的类并纳入到spring容器中进行管理。等价于<context:component-scan>的xml配置文件中的配置项。**



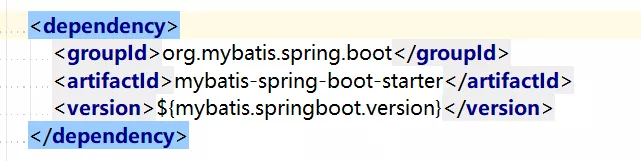
**大多数情况下，这3个注解会被同时使用，基于最佳实践，这三个注解就被做了包装，成为了@SpringBootApplication注解。**

**2、@ServletComponentScan:Servlet、Filter、Listener 可以直接通过 @WebServlet、@WebFilter、@WebListener 注解自动注册，这样通过注解servlet ，拦截器，监听器的功能而无需其他配置，所以这次相中使用到了filter的实现，用到了这个注解。**

**3、@MapperScan:spring-boot支持mybatis组件的一个注解，通过此注解指定mybatis接口类的路径，即可完成对mybatis接口的扫描。**

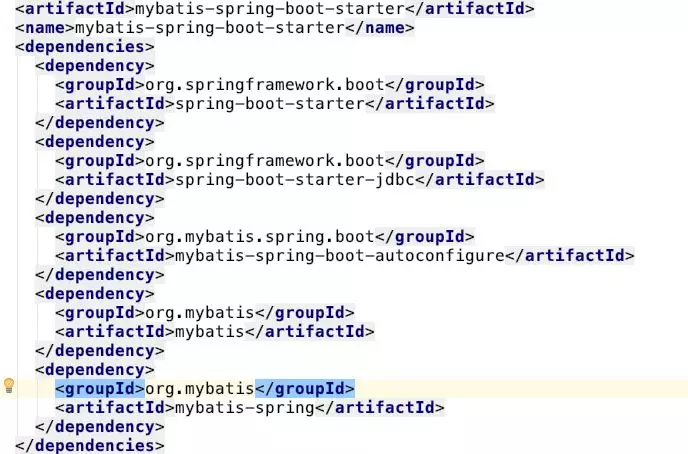
**它和@mapper注解是一样的作用，不同的地方是扫描入口不一样。@mapper需要加在每一个mapper接口类上面。所以大多数情况下，都是在规划好工程目录之后，通过@MapperScan注解配置路径完成mapper接口的注入。**

**添加mybatis相应组建依赖之后。就可以使用该注解。**



**进一步查看mybatis-spring-boot-starter包，可以找到这里已经将mybatis做了包装。**

**这也是spring的一个理念，不重复造轮子，整合优秀的资源进入spring的体系中。**



**4、资源导入注解：@ImportResource @Import @PropertySource 这三个注解都是用来导入自定义的一些配置文件。**

**@ImportResource(locations={}) 导入其他xml配置文件，需要标准在主配置类上。**

**导入property的配置文件 @PropertySource指定文件路径，这个相当于使用spring的<importresource/>标签来完成配置项的引入。**

**@import注解是一个可以将普通类导入到spring容器中做管理**

**• controller 层**

**1、@Controller 表明这个类是一个控制器类，和@RequestMapping来配合使用拦截请求，如果不在method中注明请求的方式，默认是拦截get和post请求。这样请求会完成后转向一个视图解析器。但是在大多微服务搭建的时候，前后端会做分离。所以请求后端只关注数据处理，后端返回json数据的话，需要配合@ResponseBody注解来完成。**

**这样一个只需要返回数据的接口就需要3个注解来完成，大多情况我们都是需要返回数据。也是基于最佳实践，所以将这三个注解进一步整合。**

**@RestController 是@Controller 和@ResponseBody的结合，一个类被加上@RestController 注解，数据接口中就不再需要添加@ResponseBody。更加简洁。**

**同样的情况,@RequestMapping(value="",method= RequestMethod.GET ),我们都需要明确请求方式。这样的写法又会显得比较繁琐，于是乎就有了如下的几个注解。**

| **普通风格** | **Rest风格** |
| --- | --- |
| **@RequestMapping(value=“”,method = RequestMethod.GET)** | **@GetMapping(value =“”)** |
| **@RequestMapping(value=“”,method = RequestMethod.POST)** | **@PostMapping(value =“”)** |
| **@RequestMapping(value=“”,method = RequestMethod.PUT)** | **@PutMapping(value =“”)** |
| **@RequestMapping(value=“”,method = RequestMethod.DELETE)** | **@DeleteMapping(value =“”)** |

**这几个注解是 @RequestMapping(value="",method= RequestMethod.xxx )的最佳实践。为了代码的更加简洁。**

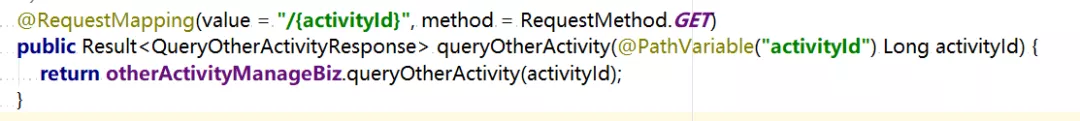
**2、@CrossOrigin:@CrossOrigin(origins = "", maxAge = 1000) 这个注解主要是为了解决跨域访问的问题。这个注解可以为整个controller配置启用跨域，也可以在方法级别启用。**

**我们在项目中使用这个注解是为了解决微服在做定时任务调度编排的时候，会访问不同的spider节点而出现跨域问题。**

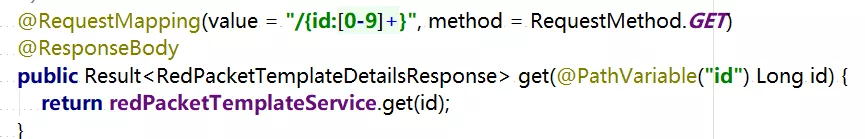
**3、@Autowired:这是个最熟悉的注解，是spring的自动装配，这个个注解可以用到构造器，变量域，方法，注解类型上。当我们需要从bean 工厂中获取一个bean时，Spring会自动为我们装配该bean中标记为@Autowired的元素。**

**4、@EnablCaching@EnableCaching: 这个注解是spring framework中的注解驱动的缓存管理功能。自spring版本3.1起加入了该注解。其作用相当于spring配置文件中的cache manager标签。**

**5、@PathVariable：路径变量注解，@RequestMapping中用{}来定义url部分的变量名，如：**



**同样可以支持变量名加正则表达式的方式，变量名:[正则表达式]。**



**• servcie层注解**

**1、@Service：这个注解用来标记业务层的组件，我们会将业务逻辑处理的类都会加上这个注解交给spring容器。事务的切面也会配置在这一层。当让 这个注解不是一定要用。有个泛指组件的注解，当我们不能确定具体作用的时候 可以用泛指组件的注解托付给spring容器。**

**2、@Resource：@Resource和@Autowired一样都可以用来装配bean，都可以标注字段上，或者方法上。 @resource注解不是spring提供的，是属于J2EE规范的注解。**

**两个之前的区别就是匹配方式上有点不同，@Resource默认按照名称方式进行bean匹配，@Autowired默认按照类型方式进行bean匹配。**

**• 持久层注解**

**1、@Repository：@Repository注解类作为DAO对象，管理操作数据库的对象。**

**总得来看，@Component, @Service, @Controller, @Repository是spring注解，注解后可以被spring框架所扫描并注入到spring容器来进行管理**

**@Component是通用注解，其他三个注解是这个注解的拓展，并且具有了特定的功能。**

**通过这些注解的分层管理，就能将请求处理，义务逻辑处理，数据库操作处理分离出来，为代码解耦，也方便了以后项目的维护和开发。**

**所以我们在正常开发中，如果能用@Service, @Controller, @Repository其中一个标注这个类的定位的时候，就不要用@Component来标注。**

**2、@Transactional： 通过这个注解可以声明事务，可以添加在类上或者方法上。**

**在spring boot中 不用再单独配置事务管理，一般情况是我们会在servcie层添加了事务注解，即可开启事务。要注意的是，事务的开启只能在public 方法上。并且主要事务切面的回滚条件。正常我们配置rollbackfor exception时 ，如果在方法里捕获了异常就会导致事务切面配置的失效。**

**• 其他相关注解**

* **@ControllerAdvice 和 @RestControllerAdvice：通常和@ExceptionHandler、@InitBinder、@ModelAttribute一起配合使用。**
* **@ControllerAdvice 和 @ExceptionHandler 配合完成统一异常拦截处理。**
* **@RestControllerAdvice 是 @ControllerAdvice 和 @ResponseBody的合集，可以将异常以json的格式返回数据。**

**如下面对数据异常返回的统一处理。**

