# Spring 最常用的 7 大类注解，史上最强整理！

点击关注  [Java派](https://mp.weixin.qq.com/s/javascript:void(0);) 2022-09-28 10:00 发表于湖南

收录于合集

#Java103个

#Spring34个



**Java派**

专注Java相关技术栈：Spring全家筒、Docker、k8s、Mysql、集群、微服务、中间件等知识。

17篇原创内容

公众号

随着技术的更新迭代，Java5.0开始支持注解。而作为java中的领军框架spring，自从更新了2.5版本之后也开始慢慢舍弃xml配置，更多使用注解来控制spring框架。

而spring的的注解那么多，可能做java很多年，都用不上。这里按照类型总结了这7种最常用的注解。

1

核心注解

**@Required**

此注解用于bean的setter方法上。表示此属性是必须的，必须在配置阶段注入，否则会抛出BeanInitializationExcepion。

**@Autowired**

此注解用于bean的field、setter方法以及构造方法上，显式地声明依赖。根据type来autowiring。

当在field上使用此注解，并且使用属性来传递值时，Spring会自动把值赋给此field。也可以将此注解用于私有属性(不推荐)，如下。

@Component  
public class User {  
    @Autowired                                 
    private Address address;  
}

最经常的用法是将此注解用于settter上，这样可以在setter方法中添加自定义代码。如下：

@Component  
public class User {  
  
   private Address address;  
  
   @AutoWired  
   public setAddress(Address address) {  
      // custom code  
      this.address=address;  
   }  
}

当在构造方法上使用此注解的时候，需要注意的一点就是一个类中只允许有一个构造方法使用此注解。此外，在Spring4.3后，如果一个类仅仅只有一个构造方法，那么即使不使用此注解，那么Spring也会自动注入相关的bean。如下：

@Component  
public class User {  
  
    private Address address;  
  
    public User(Address address) {  
       this.address=address;  
    }  
}  
  
<bean id="user" class="xx.User"/>

**@Qualifier**

此注解是和@Autowired一起使用的。使用此注解可以让你对注入的过程有更多的控制。

@Qualifier可以被用在单个构造器或者方法的参数上。当上下文有几个相同类型的bean, 使用@Autowired则无法区分要绑定的bean，此时可以使用@Qualifier来指定名称。

@Component  
public class User {  
  
    @Autowired  
    @Qualifier("address1")  
    private Address address;  
  
    ...  
  
}

**@Configuration**

此注解用在class上来定义bean。其作用和xml配置文件相同，表示此bean是一个Spring配置。此外，此类可以使用@Bean注解来初始化定义bean。

@Configuartion  
public class SpringCoreConfig {  
  
    @Bean  
    public AdminUser adminUser() {  
        AdminUser adminUser = new AdminUser();  
        return adminUser;  
    }  
}

**@ComponentScan**

此注解一般和@Configuration注解一起使用，指定Spring扫描注解的package。如果没有指定包，那么默认会扫描此配置类所在的package。

**@Lazy**

此注解使用在Spring的组件类上。默认的，Spring中Bean的依赖一开始就被创建和配置。如果想要延迟初始化一个bean，那么可以在此类上使用Lazy注解，表示此bean只有在第一次被使用的时候才会被创建和初始化。此注解也可以使用在被@Configuration注解的类上，表示其中所有被@Bean注解的方法都会延迟初始化。

**@Value**

此注解使用在字段、构造器参数和方法参数上。@Value可以指定属性取值的表达式，支持通过#{}使用SpringEL来取值，也支持使用${}来将属性来源中(Properties文件、本地环境变量、系统属性等)的值注入到bean的属性中。此注解值的注入发生在AutowiredAnnotationBeanPostProcessor类中。

2

Spring MVC和REST注解

**@Controller**

此注解使用在class上声明此类是一个Spring controller，是@Component注解的一种具体形式。

**@RequestMapping**

此注解可以用在class和method上，用来映射web请求到某一个handler类或者handler方法上。当此注解用在Class上时，就创造了一个基础url，其所有的方法上的@RequestMapping都是在此url之上的。

可以使用其method属性来限制请求匹配的http method。

@Controller  
@RequestMapping("/users")  
public class UserController {  
  
    @RequestMapping(method = RequestMethod.GET)  
    public String getUserList() {  
  
        return "users";  
    }  
}

此外，Spring4.3之后引入了一系列@RequestMapping的变种。如下：

@GetMapping  
@PostMapping  
@PutMapping  
@PatchMapping  
@DeleteMapping

分别对应了相应method的RequestMapping配置。

**@CookieValue**

此注解用在@RequestMapping声明的方法的参数上，可以把HTTP cookie中相应名称的cookie绑定上去。

@ReuestMapping("/cookieValue")  
public void getCookieValue(@CookieValue("JSESSIONID") String cookie){  
  
}

cookie即http请求中name为JSESSIONID的cookie值。

**@CrossOrigin**

此注解用在class和method上用来支持跨域请求，是Spring 4.2后引入的。

@CrossOrigin(maxAge = 3600)  
@RestController  
@RequestMapping("/users")  
public class AccountController {  
  
    @CrossOrigin(origins = "http://xx.com")  
    @RequestMapping("/login")  
    public Result userLogin() {  
  
        // ...  
  
    }  
}

**@ExceptionHandler**

此注解使用在方法级别，声明对Exception的处理逻辑。可以指定目标Exception。

**@InitBinder**

此注解使用在方法上，声明对WebDataBinder的初始化(绑定请求参数到JavaBean上的DataBinder)。在controller上使用此注解可以自定义请求参数的绑定。

**@MatrixVariable**

此注解使用在请求handler方法的参数上，Spring可以注入matrix url中相关的值。这里的矩阵变量可以出现在url中的任何地方，变量之间用;分隔。如下：

// GET /pets/42;q=11;r=22  
@RequestMapping(value = "/pets/{petId}")  
public void findPet(@PathVariable String petId, @MatrixVariable int q) {  
  
    // petId == 42  
    // q == 11  
}

需要注意的是默认Spring mvc是不支持矩阵变量的，需要开启。

<mvc:annotation-driven enable-matrix-variables="true" />

注解配置则需要如下开启：

@Configuration  
public class WebConfig extends WebMvcConfigurerAdapter {  
  
    @Override  
    public void configurePathMatch(PathMatchConfigurer configurer) {  
  
        UrlPathHelper urlPathHelper = new UrlPathHelper();  
        urlPathHelper.setRemoveSemicolonContent(false);  
        configurer.setUrlPathHelper(urlPathHelper);  
    }  
}

**@PathVariable**

此注解使用在请求handler方法的参数上。@RequestMapping可以定义动态路径，如:

@RequestMapping("/users/{uid}")

可以使用@PathVariable将路径中的参数绑定到请求方法参数上。

@RequestMapping("/users/{uid}")  
public String execute(@PathVariable("uid") String uid){  
  
}

**@RequestAttribute**

此注解用在请求handler方法的参数上，用于将web请求中的属性(request attributes，是服务器放入的属性值)绑定到方法参数上。

**@RequestBody**

此注解用在请求handler方法的参数上，用于将http请求的Body映射绑定到此参数上。

HttpMessageConverter负责将对象转换为http请求。

**@RequestHeader**

此注解用在请求handler方法的参数上，用于将http请求头部的值绑定到参数上。

**@RequestParam**

此注解用在请求handler方法的参数上，用于将http请求参数的值绑定到参数上。

**@RequestPart**

此注解用在请求handler方法的参数上，用于将文件之类的multipart绑定到参数上。

**@ResponseBody**

此注解用在请求handler方法上。和@RequestBody作用类似，用于将方法的返回对象直接输出到http响应中。

**@ResponseStatus**

此注解用于方法和exception类上，声明此方法或者异常类返回的http状态码。可以在Controller上使用此注解，这样所有的@RequestMapping都会继承。

**@ControllerAdvice**

此注解用于class上。前面说过可以对每一个controller声明一个ExceptionMethod。这里可以使用@ControllerAdvice来声明一个类来统一对所有@RequestMapping方法来做@ExceptionHandler、@InitBinder以及@ModelAttribute处理。

**@RestController**

此注解用于class上，声明此controller返回的不是一个视图而是一个领域对象。其同时引入了@Controller和@ResponseBody两个注解。

**@RestControllerAdvice**

此注解用于class上，同时引入了@ControllerAdvice和@ResponseBody两个注解。

**@SessionAttribute**

此注解用于方法的参数上，用于将session中的属性绑定到参数。

**@SessionAttributes**

此注解用于type级别，用于将JavaBean对象存储到session中。一般和@ModelAttribute注解一起使用。如下：

@ModelAttribute("user")  
public PUser getUser() {}  
  
// controller和上面的代码在同一controller中  
@Controller  
@SeesionAttributes(value = "user", types = {  
  
    User.class  
})  
public class UserController {}

3

Spring Boot注解

**@EnableAutoConfiguration**

此注解通常被用在主应用class上，告诉Spring Boot自动基于当前包添加Bean、对bean的属性进行设置等。

**@SpringBootApplication**

此注解用在Spring Boot项目的应用主类上（此类需要在base package中）。使用了此注解的类首先会让Spring Boot启动对base package以及其sub-pacakage下的类进行component scan。

此注解同时添加了以下几个注解：

@Configuration  
@EnableAutoConfiguration  
@ComponentScan

4

Stereotype注解

**@Component**

此注解使用在class上来声明一个Spring组件(Bean), 将其加入到应用上下文中。

**@Controller**

前文已经提到过

**@Service**

此注解使用在class上，声明此类是一个服务类，执行业务逻辑、计算、调用内部api等。是@Component注解的一种具体形式。

**@Repository**

此类使用在class上声明此类用于访问数据库，一般作为DAO的角色。

此注解有自动翻译的特性，例如：当此种component抛出了一个异常，那么会有一个handler来处理此异常，无需使用try-catch块。

5

数据访问注解

**@Transactional**

此注解使用在接口定义、接口中的方法、类定义或者类中的public方法上。需要注意的是此注解并不激活事务行为，它仅仅是一个元数据，会被一些运行时基础设施来消费。

5

任务执行、调度注解

**@Scheduled**

此注解使用在方法上，声明此方法被定时调度。使用了此注解的方法返回类型需要是Void，并且不能接受任何参数。

@Scheduled(fixedDelay=1000)  
public void schedule() {  
  
}  
  
@Scheduled(fixedRate=1000)  
public void schedulg() {  
  
}

第二个与第一个不同之处在于其不会等待上一次的任务执行结束。

**@Async**

此注解使用在方法上，声明此方法会在一个单独的线程中执行。不同于Scheduled注解，此注解可以接受参数。

使用此注解的方法的返回类型可以是Void也可是返回值。但是返回值的类型必须是一个Future。

7

测试注解

**@ContextConfiguration**

此注解使用在Class上，声明测试使用的配置文件，此外，也可以指定加载上下文的类。

此注解一般需要搭配SpringJUnit4ClassRunner使用。

@RunWith(SpringJUnit4ClassRunner.class)  
@ContextConfiguration(classes = SpringCoreConfig.class)  
public class UserServiceTest {  
  
}