# Spring Web MVC

## @RequestMapping

pring MVC和Spring WebFlux都通过RquestMappingHandlerMapping和RequestMappingHndlerAdapter两个类来提供对@RequestMapping注解的支持。@RequestMapping注解对请求处理类中的请求处理方法进行标注；

注解的属性

* value:映射的请求URL或者其别名
* method:兼容HTTP的方法名
* params:根据HTTP参数的存在、缺省或值对请求进行过滤
* header:根据HTTP Header的存在、缺省或值对请求进行过滤
* consume:设定在HTTP请求正文中允许使用的媒体类型
* product:在HTTP响应体中允许使用的媒体类型

## @ControllerAdvice

@Component注解的一个延伸注解，Spring会自动扫描并检测被@ControllerAdvice所标注的类。

@ControllerAdvice需要和@ExceptionHandler、@InitBinder以及@ModelAttribute注解搭配使用，主要是用来处理控制器所抛出的异常信息。

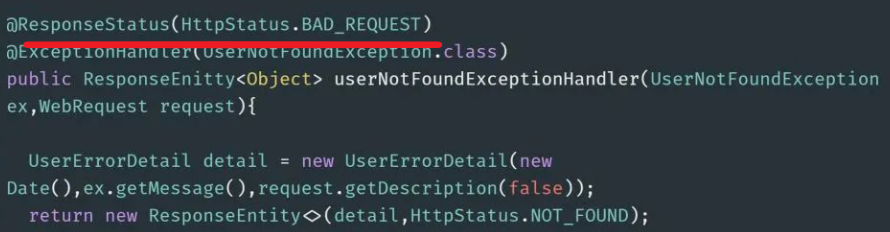
首先，我们需要定义一个被@ControllerAdvice所标注的类，在该类中，定义一个用于处理具体异常的方法，并使用

* @ExceptionHandler注解进行标记
* @InitBinder在类中进行全局的配置
* @ModelAttribute配置与视图相关的参数。



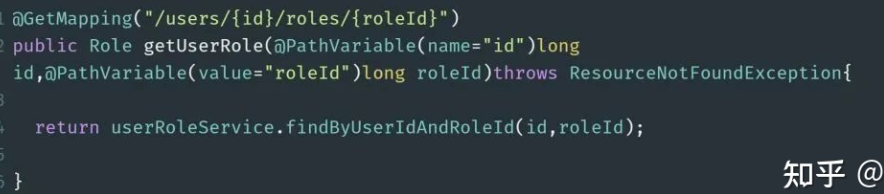
## @ResponseBody

## @ResponseStatus



## @PathVariable

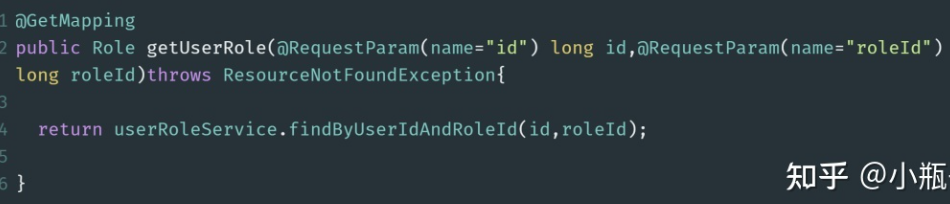
@PathVariable注解是将方法中的参数绑定到请求URI中的模板变量上。可以通过@RequestMapping注解来指定URI的模板变量，然后使用@PathVariable注解将方法中的参数绑定到模板变量上。特别地，@PathVariable注解允许我们使用value或name属性来给参数取一个别名。



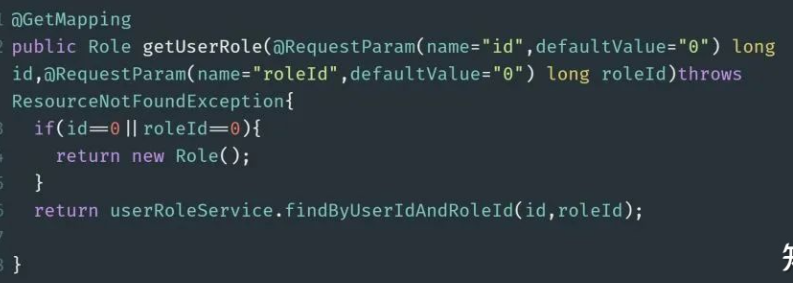


## @RequestParam

@RequestParam注解用于将方法的参数与Web请求的传递的参数进行绑定。使用@RequestParam可以轻松的访问HTTP请求参数的值



该注解的其他属性配置与@PathVariable的配置相同，特别的，如果传递的参数为空，还可以通过defaultValue设置一个默认值。示例代码如下



## @Controller

Spring会自动扫描并配置被该注解标注的类。此注解用于标注Spring MVC的控制器。下面是使用此注解的示例代码

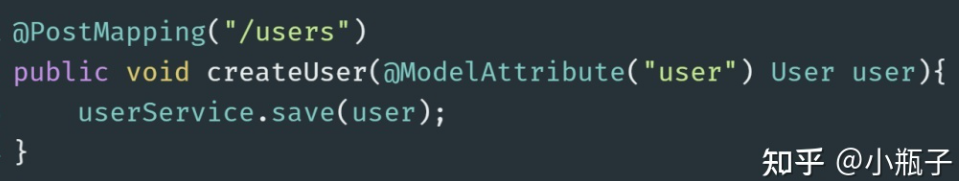


## @RestController

@RestController是在Spring 4.0开始引入的，这是一个特定的控制器注解。此注解相当于@Controller和@ResponseBody的快捷方式。当使用此注解时，不需要再在方法上使用@ResponseBody注解。

## @ModelAttribute

通过此注解，可以通过模型索引名称来访问已经存在于控制器中的model。

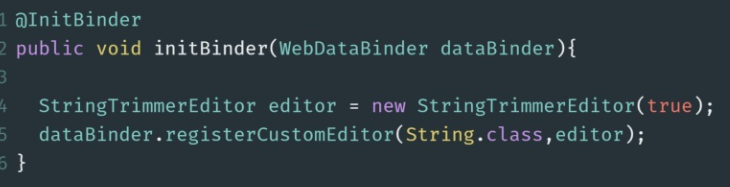


## @CrossOrigin

@CrossOrigin注解将为请求处理类或请求处理方法提供跨域调用支持。如果我们将此注解标注类，那么类中的所有方法都将获得支持跨域的能力。使用此注解的好处是可以微调跨域行为。

## @InitBinder

@InitBinder注解用于标注初始化WebDataBinider的方法，该方法用于对Http请求传递的表单数据进行处理，如时间格式化、字符串处理等



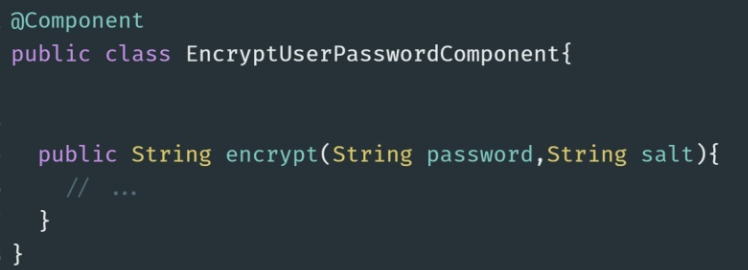
# Spring Bean

## @ComponentScan

用于配置Spring需要扫描的被组件注解注释的类所在的包。可以通过配置其basePackages属性或者value属性来配置需要扫描的包路径。value属性是basePackages的别名。

## @Component

@Component注解用于标注一个普通的组件类，它没有明确的业务范围，只是通知Spring被此注解的类需要被纳入到Spring Bean容器中并进行管理。此注解的使用示例如下：



## @Service

标注业务逻辑类。与@Component注解一样，被此注解标注的类，会自动被Spring所管理。



## @Repository

@Repository注解也是@Component注解的延伸，与@Component注解一样，被此注解标注的类会被Spring自动管理起来，@Repository注解用于标注DAO层的数据持久化类。

4

## @DependsOn

配置Spring IoC容器在初始化一个Bean之前，先初始化其他的Bean对象。



## @Bean

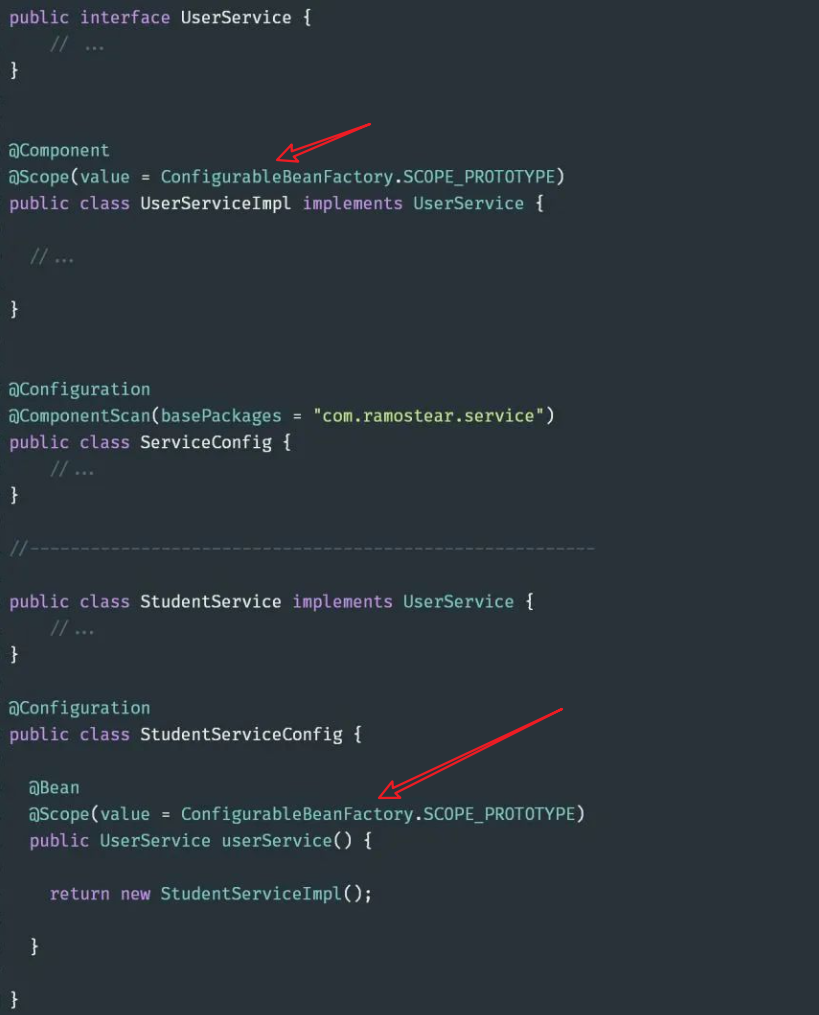
@Bean注解主要的作用是告知Spring，被此注解所标注的类将需要纳入到Bean管理工厂中。



## @Scope

用来定义@Component标注的类的作用范围以及@Bean所标记的类的作用范围。@Scope所限定的作用范围有：singleton、prototype、request、session、globalSession或者其他的自定义范围。这里以prototype为例子进行讲解。当一个Spring Bean被声明为prototype（原型模式）时，在每次需要使用到该类的时候，Spring IoC容器都会初始化一个新的改类的实例。在定义一个Bean时，可以设置Bean的scope属性为prototype：scope=“prototype”,也可以使用@Scope注解设置，如下：

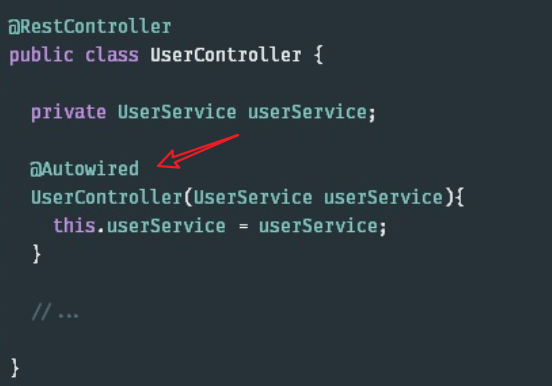
@Scope(value=ConfigurableBeanFactory.SCOPE\_PROPTOTYPE)



## @Autowired

用于标记Spring将要解析和注入的依赖项。此注解可以作用在构造函数、字段和setter方法上。

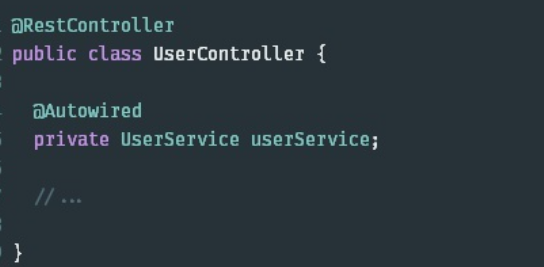
* 作用于构造函数下面是@Autowired注解标注构造函数的使用示例：



* 作用于setter方法下面是@Autowired注解标注setter方法的示例代码：



* 作用于字段@Autowired注解标注字段是最简单的，只需要在对应的字段上加入此注解即可



## @Primary

当系统中需要配置多个具有相同类型的bean时，@Primary可以定义这些Bean的优先级



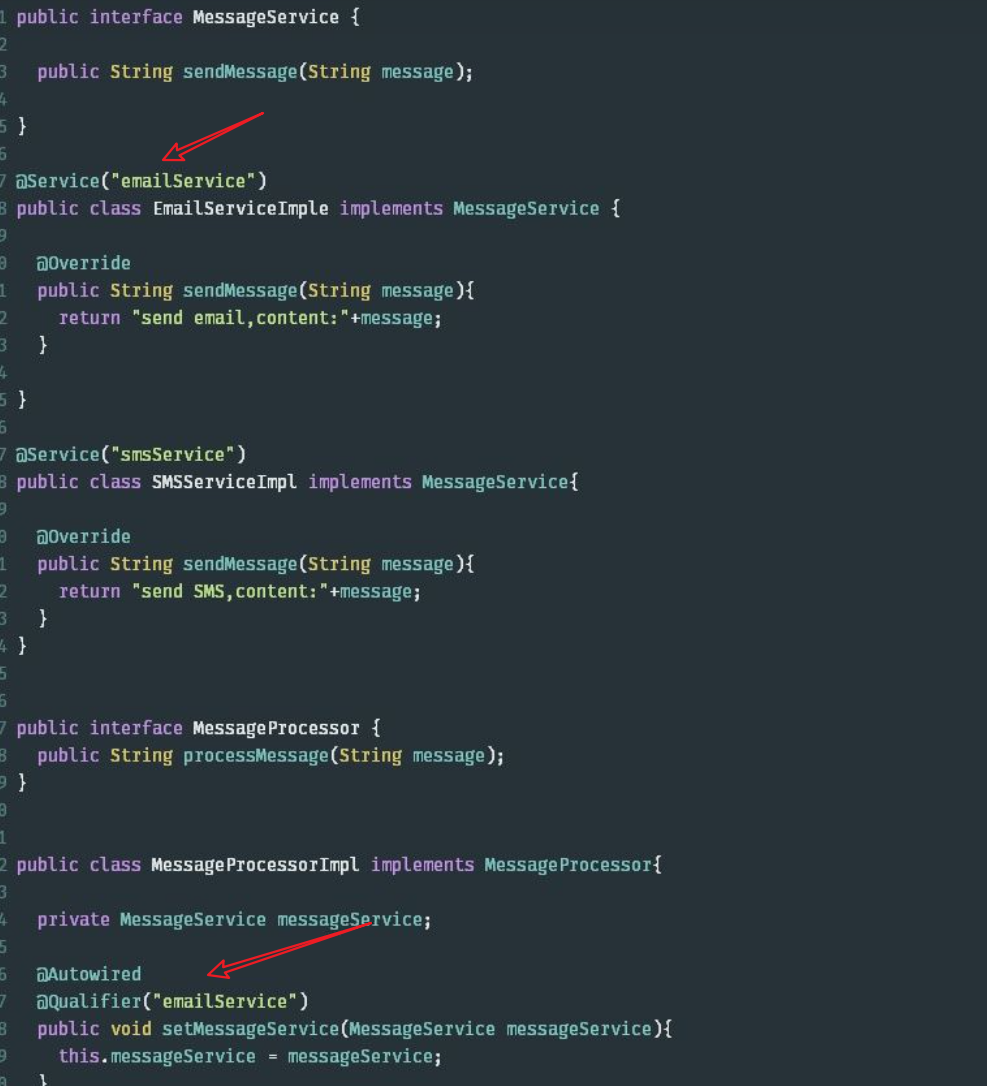
## @PostConstruct与@PreDestroy

值得注意的是，这两个注解不属于Spring,它们是源于JSR-250中的两个注解，位于common-annotations.jar中。@PostConstruct注解用于标注在Bean被Spring初始化之前需要执行的方法。@PreDestroy注解用于标注Bean被销毁前需要执行的方法。



## @Qualifier

当系统中存在同一类型的多个Bean时，@Autowired在进行依赖注入的时候就不知道该选择哪一个实现类进行注入。此时，我们可以使用@Qualifier注解来微调，帮助@Autowired选择正确的依赖项。



# 三、SpringBoot

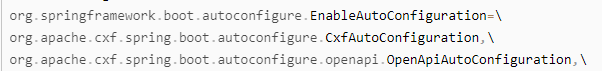
## @SpringBootApplication

@SpringBootApplication注解是一个快捷的配置注解，在被它标注的类中，可以定义一个或多个Bean，并自动触发自动配置Bean和自动扫描组件。此注解相当于@Configuration、@EnableAutoConfiguration和@ComponentScan的组合。在Spring Boot应用程序的主类中，就使用了此注解。

## @EnableAutoConfiguration

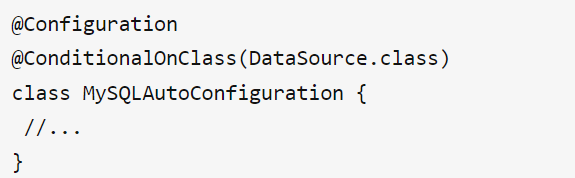
用于通知Spring，根据当前类路径下引入的依赖包，自动配置与这些依赖包相关的配置项。

通过SPI机制，在spring.factories中增加对应配置，注明自动配置的类



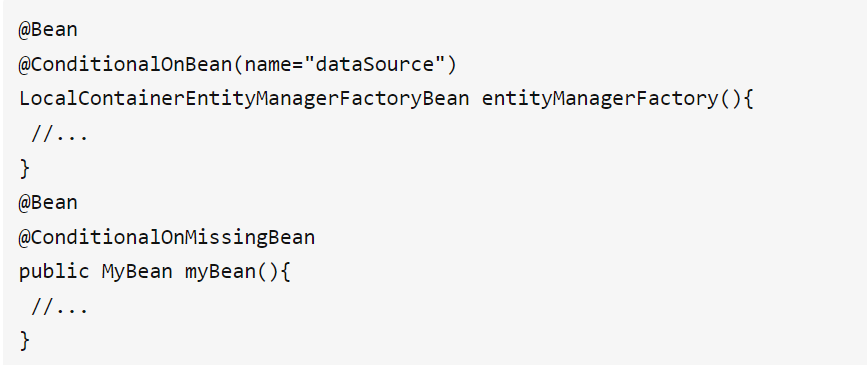
## @ConditionalOnClass与@ConditionalOnMissingClass

属于类条件注解，它们根据是否存在某个类作为判断依据来决定是否要执行某些配置。下



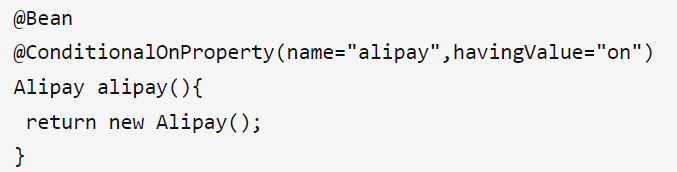
## @ConditionalOnBean与@ConditionalOnMissingBean

属于对象条件注解，根据是否存在某个对象作为依据来决定是否要执行某些配置方法。



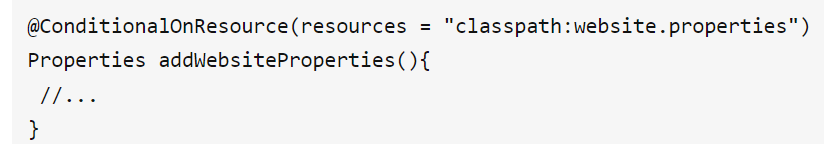
## @ConditionalOnProperty

@ConditionalOnProperty注解会根据Spring配置文件中的配置项是否满足配置要求，从而决定是否要执行被其标注的方法



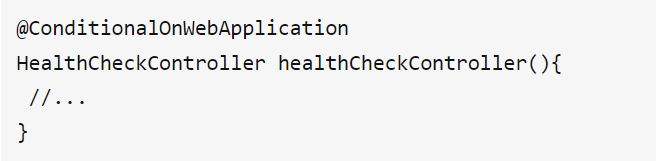
## @ConditionalOnResource

用于检测当某个配置文件存在使，则触发被其标注的方法，下面是使用此注解的代码示例



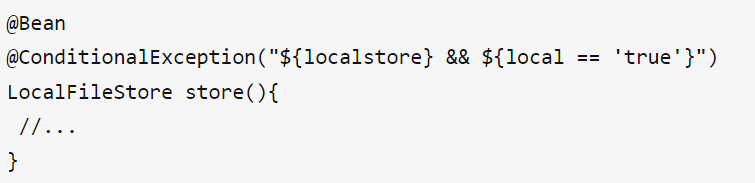
## @ConditionalOnWebApplication与@ConditionalOnNotWebApplication

用于判断当前的应用程序是否是Web应用程序。如果当前应用是Web应用程序，则使用Spring WebApplicationContext,并定义其会话的生命周期。



## @ConditionalExpression

控制更细粒度的基于表达式的配置条件限制。当表达式满足某个条件或者表达式为真的时候，将会执行被此注解标注的方法。



## @Conditional

可以控制更为复杂的配置条件。在Spring内置的条件控制注解不满足应用需求的时候，可以使用此注解定义自定义的控制条件，以达到自定义的要求。下面是使用该注解的简单示例：

