springboot注解：

@Service: 注解在类上，表示这是一个业务层bean

@Controller：注解在类上，表示这是一个控制层bean

@Repository: 注解在类上，表示这是一个数据访问层bean

@Component： 注解在类上，表示通用bean ，value不写默认就是类名首字母小写

@Autowired：按类型注入.默认属性required= true;当不能确定 Spring 容器中一定拥有某个类的Bean 时， 可以在需要自动注入该类 Bean 的地方可以使用 @Autowired(required = false)， 这等于告诉Spring：在找不到匹配Bean时也不抛出BeanCreationException 异常。@Autowired 和 @Qualifier 结合使用时，自动注入的策略就从 byType 转变byName 了。@Autowired可以对成员变量、方法以及构造函数进行注释，而 @Qualifier 的标注对象是成员变量、方法入参、构造函数入参。正是由于注释对象的不同，所以 Spring 不将 @Autowired 和 @Qualifier 统一成一个注释类。

@Resource： 按名称装配

区别：

@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

@Resource(importjavax.annotation.Resource;)是J2EE的注解，@Autowired(importorg.springframework.beans.factory.annotation.Autowired;)是Spring的注解

@Configuration：注解在类上，表示这是一个IOC容器，相当于spring的配置文件，java配置的方式。 IOC容器的配置类一般与 @Bean 注解配合使用，用 @Configuration 注解类等价与 XML 中配置 beans，用@Bean 注解方法等价于 XML 中配置 bean。

@Bean： 注解在方法上，声明当前方法返回一个Bean

@Scope：注解在类上，描述spring容器如何创建Bean实例。

（1）singleton： 表示在spring容器中的单例，通过spring容器获得该bean时总是返回唯一的实例

（2）prototype：表示每次获得bean都会生成一个新的对象

（3）request：表示在一次http请求内有效（只适用于web应用）

（4）session：表示在一个用户会话内有效（只适用于web应用）

（5）globalSession：表示在全局会话内有效（只适用于web应用）

在多数情况，我们只会使用singleton和prototype两种scope，如果未指定scope属性，默认为singleton

@Value：注解在变量上，从配置文件中读取。

例如：@Value(value = “#{message}”)

@ConfigurationProperties   赋值，将注解转换成对象。给对象赋值。车险项目：HttpClientSetting类

@Profile：注解在方法类上在不同情况下选择实例化不同的Bean特定环境下生效！！！！！！！！！！！！！！！！！

@SpringBootApplication：@SpringBootApplication=@ComponentScan+@Configuration+@EnableAutoConfiguration：约定优于配置

@EnableAutoConfiguration启用 Spring 应用程序上下文的自动配置，试图猜测和配置您可能需要的bean。自动配置类通常采用基于你的classpath 和已经定义的 beans 对象进行应用。被 @EnableAutoConfiguration 注解的类所在的包有特定的意义，并且作为默认配置使用。通常推荐将 @EnableAutoConfiguration 配置在 root 包下，这样所有的子包、类都可以被查找到。

@ComponentScan：注解在类上，扫描标注了@Controller等注解的类，注册为bean 。@ComponentScan 为 @Configuration注解的类配置组件扫描指令。@ComponentScan 注解会自动扫描指定包下的全部标有 @Component注解的类，并注册成bean，当然包括 @Component下的子注解@Service、@Repository、@Controller。

@RestController @RestController 是一个结合了 @ResponseBody 和 @Controller 的注解

@Responsebody 注解表示该方法的返回的结果直接写入 HTTP 响应正文（ResponseBody）中，一般在异步获取数据时使用，通常是在使用 @RequestMapping 后，返回值通常解析为跳转路径，加上@Responsebody 后返回结果不会被解析为跳转路径，而是直接写入HTTP 响应正文中。

@RequestBody

@PathVariable

@RequestParam

两者的作用都是将request里的参数的值绑定到contorl里的方法参数里的，区别在于，URL写法不同。

当请求参数username不存在时会有异常发生,可以通过设置属性required=false解决,例如:

@RequestParam(value="username",required=false)

使用@RequestParam时，URL是这样的：http://host:port/path?参数名=参数值

使用@PathVariable时，URL是这样的：http://host:port/path/参数值

不写的时候也可以获取到参数值，但是必须名称对应。参数可以省略不写

@RequestMapping　　和请求报文是做对应的

　　a：value，指定请求的地址

　　b：method 请求方法类型 这个不写的话，自适应：get或者post

　　c：consumes 请求的提交内容类型

　　d：produces 指定返回的内容类型 仅当request请求头中的(Accept)类型中包含该指定类型才返回

　　e: params 指定request中必须包含某些参数值

　　f：headers 指定request中必须包含指定的header值

g: name  指定映射的名称

 @RequestMapping(method = RequestMethod.GET)

 @RequestMapping(method = RequestMethod.POST)

 @RequestMapping(method = RequestMethod.PUT)

 @RequestMapping(method = RequestMethod.DELETE)

 当然也可以使用

 @GetMapping

 @PostMapping

 @PutMapping

 @DeleteMapping 这与上面的是一样的效果

@EnablCaching @EnableCaching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。如果你使用了这个注解，那么你就不需要在XML文件中配置cache manager了。

@suppresswarnings 抑制警告

@Modifying 如果是增，改，删加上此注解

1：方法的返回值应该是int，表示更新语句所影响的行数。

2：在调用的地方必须加事务，没有事务不能正常执行。@Transactional  事务注解

@Query 自定义查询语句 JPQL

JPA注解

@Entity：

@Table(name=“”)：注解在类上表明这是一个实体类。一般用于jpa这两个注解一般一块使用，但是如果表名和实体类名相同的话，@Table可以省略

@Column：通过@Column注解设置，包含的设置如下

name：数据库表字段名

unique：是否唯一

nullable：是否可以为空

Length:长度

inserttable：是否可以插入

updateable：是否可以更新

columnDefinition: 定义建表时创建此列的DDL

secondaryTable: 从表名。如果此列不建在主表上（默认建在主表），该属性定义该列所在从表的名字。

@Column(name = "user\_code", nullable = false, length=32)//设置属性userCode对应的字段为user\_code，长度为32，非空

private String userCode;

@Column(name = "user\_wages", nullable = true, precision=12,scale=2)//设置属性wages对应的字段为user\_wages，12位数字可保留两位小数，可以为空

private double wages;

@Id：表示该属性为主键。

@Temporal(TemporalType.DATE)//设置为时间类型

private Date joinDate;

@Transient：表示该属性并非一个到数据库表的字段的映射,ORM框架将忽略该属性。如果一个属性并非数据库表的字段映射,就务必将其标示为@Transient,否则,ORM框架默认其注解为@Basic。@Basic(fetch=FetchType.LAZY)：标记可以指定实体属性的加载方式

@JsonIgnore：作用是json序列化时将Java bean中的一些属性忽略掉,序列化和反序列化都受影响。

@JoinColumn（name=”loginId”）:一对一：本表中指向另一个表的外键。一对多：另一个表指向本表的外键。

@OneToOne、@OneToMany、@ManyToOne：对应hibernate配置文件中的一对一，一对多，多对一。

@GeneratedValue 用于标注主键的生成策略，通过 strategy 属性指定。默认情况下，JPA 自动选择一个最适合底层数据库的主键生成策略：SqlServer 对应 identity，MySQL 对应 auto increment。 在 javax.persistence.GenerationType 中定义了以下几种可供选择的策略：

IDENTITY：采用数据库 ID自增长的方式来自增主键字段，Oracle 不支持这种方式；

AUTO： JPA自动选择合适的策略，是默认选项；

SEQUENCE：通过序列产生主键，通过 @SequenceGenerator 注解指定序列名，MySql 不支持这种方式

TABLE：通过表产生主键，框架借由表模拟序列产生主键，使用该策略可以使应用更易于数据库移植。

---------------------

作者：java-小学生

来源：CSDN

原文：https://blog.csdn.net/yitian\_66/article/details/80866571

版权声明：本文为博主原创文章，转载请附上博文链接！