# B简历：

# -------------------------------------

基本信息 **面试常见问题：**

|  |  |  |  |
| --- | --- | --- | --- |
| 基本情况 | | | |
| 姓名 | 高文通 | 性别 | 男 |
| 出生日期 | 1995-06-15 | 属相 | 猪 |
| 现住地 | 升龙又一城 |  |  |
| 大学情况 | | | |
| 学校 | 四川工业科技学院 | 专业 | 电子信息与记算机 |
| 入学时间 | 2013-09 | 毕业时间 | 2017-07 |
| 专业课 | **计算机硬件实训、C语言、数据库原理、操作系统、Java程序设计** | | |
| 学历 | **本科** | 学位 | **学士学位** |
| 学校地址 | 四川省德阳市 | 校长 | **学校领导都没见过，所以也不知道校长的名字** |
| 去学校的方式： | 坐地铁城郊线 华南城站下（5元左右） 打小黑车（15元） | | |

|  |  |
| --- | --- |
| 第一家公司 ： | 成都天易视 |
| **地址：** |  |
| **入职时间：** | **2017年9月** |
| **离职时间：** | **2019年05月** |
| **公司性质：** | **民营** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | 李旺 |
| **薪资方面：** | **刚刚毕业转正以后工资是6K左右，过完年大概三月份涨到7K左右** |
| **路线：** | 成都市 双流区龙岗附近 |
|  |  |

|  |  |
| --- | --- |
| **第二家公司情况：** | 北京杰达昌盛科技有限公司 |
| **公司地址**： | 北京市 海淀区 中国人民革命军事博物馆 附近 |
| **公司规模：** | **80人左右，我们项目组8人，一个项目经理，1名前端，1个测试，1个运维，其他都是Java工程师** |
| **公司性质：** | **民营** |
| **公司行业：** | 技术开发、技术咨询、技术转让、技术服务；软件开发；计算机系统集成；企业管理咨询；企业策划；设计、制作、代理、发布广告 |
| **去公司路线：** | 通州北苑上地铁（八通线） 四惠站下车（D出口出） 骑小车到远洋国际 |
| **入职时间：** | **2019-06** |
| **离职时间：** | **2021-03** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **于辉** |
| **税前：** | **14K** |
| **税后：** | **13K** |
| **缴税：** | **1K左右(具体原因不清楚当时公司就是这样扣的税)** |
| **期望薪资：** | **14K** |
| **薪资的构成:** | **基本工资(5K)+绩效工资(9K)** |

|  |
| --- |
| 四年涨薪： **2017年9月工作时月薪为 6K左右**  **2018年7月工作一年涨了2K左右**  **2021年3月离职时工资为14K** |

|  |  |
| --- | --- |
| **你 为什么从上家公司离职？** |  |
| **上上家公司的福利待遇也算不错，和同事相处的也蛮开心的，但是在技术提升方面不太符合自己发展的预期。就出来历练一下** |  |
| **你交社保了吗？为啥没交？** |  |
| **交了** |  |
| |  | | --- | | **三个词形容自己**： 交流表达能力强，团队协作能力强，责任心强 | | **你的优缺点是啥？** |   **优点:**  **善于倾听，团队协作能力强，责任心强。**  **缺点: 这几年做程序，因为要不断的对代码进行验证确认，所有感觉自己现在有点强迫症。** |  |
| **你知道五险一金都是啥吗？** |  |
| **养老保险、医疗保险、失业保险、工伤保险、生育保险，和住房公积金。** |  |
| **你能接受加班吗** |  |
| **能** |  |
| **你能接受出差吗** |  |
| **能** |  |
| 你期望薪资多少？你上家工资多少？税后拿到手的有多少？扣了多少钱的税？ |  |
| 期望薪资 15K 上家工资14K 税后13K左右 扣了1K左右的税 |  |
| 你的五年规划是啥？ |  |
| **近期的话我会先让自己的技术能力得到提高，当然，尽快的掌握公司的业务，投入到工作中，然后朝着项目经理发展** |  |
| 是统招吗 是是本科吗 是能在学信网查到吗 这个因为我当时上的是民办大学，所以在民教网查的到！ |  |
| 你们公司还有什么福利？多少薪： 13薪  **车补**  300左右  **项目奖金**  7k-10k，做完项目，收完尾款的时候。  **年终奖**  这个就是1个月的工资，  过年的时候和当月工资一块发  **电话补助**  300  **饭/餐补**  **15、**30/天  **电脑补**  300 |  |

**公司内其他部门：**

**技术部（开发部/研发部），行政部，销售部，财务部**

**---------------------------------------------------------------------**

**其他问题：**

**笔试问题：我工作这么多长的时间了，理论上的知识忘得差不多了，你和你们经理联系一下直接面试吧，不行的话那么我就去直接去下一家！**

**----------------------------------------------------------------------------------------------------------------------**

？？？？？？？？？？？？？？？？？？？？？

# ========（业务功能模块/核心技术点）【主动说】最近做的是基于springboot微服务的电商项目，再之前一点做的是ssm项目，负责了其中的几个模块，我先给你说下框架方面的东西吧

**框架篇**

## 首先讲下对springboot的理解

springBoot 主要是接口开发 restful风格的接口开发

**主要是微服务**

**快速搭建 spring 框架 2 约定大于配置 3 开箱即用 4 嵌入了web容器**

@SpringBootApplication

@SpringBootApplication是一个启动类注解，包括@ComponentScan，和@SpringBootConfiguration，@EnableAutoConfiguration。

@SpringBootConfiguration继承自@Configuration，二者功能也一致，标注当前类是配置类，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的作用启动自动的配置，@EnableAutoConfiguration注解的意思就是Springboot根据你添加的jar包来配置你项目的默认配置，比如根据spring-boot-starter-web ，来判断你的项目是否需要添加了webmvc和tomcat，就会自动的帮你配置web项目中所需要的默认配置。在下面博客会具体分析这个注解，快速入门的demo实际没有用到该注解。

@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。是以前的<context:component-scan>（以前使用在xml中使用的标签，用来扫描包配置的平行支持）。所以本demo中的User为何会被spring容器管理。

## Spring cloud

[Spring Cloud](http://c.biancheng.net/spring_cloud/) 是一系列框架的有序集合。它利用 [Spring](http://c.biancheng.net/spring/) Boot 的开发便利性，巧妙地简化了分布式系统基础设施的开发，如服务注册、服务发现、配置中心、消息总线、负载均衡、断路器、数据监控等，这些都可以用 Spring Boot 的开发风格做到一键启动和部署。

**优点：**

**1、服务拆分粒度更细，有利于资源重复利用，有利于提高开发效率**

**2、可以更精准的制定优化服务方案，提高系统的可维护性**

**3、微服务架构采用去中心化思想，服务之间采用Restful等轻量级通讯，比ESB更轻量**

**4、适于互联网时代，产品迭代周期更短**

**缺点：**

**1、微服务过多，治理成本高，不利于维护系统**

**2、分布式系统开发的成本高（容错，分布式事务等）对团队挑战大**

**总的来说优点大过于缺点，目前看来SpringCloud是一套非常完善的分布式框架，目前很多企业开始用微服务**

* **服务发现——Netflix Eureka**

一个RESTful风格的服务，用来定位运行在业务流程管理开发平台(AWS Enterprise BPM Platform)层的中间层服务。 1.Eureka服务器用作服务的注册中心。(对比dubbo的注册中心：zk redis等); 2.Eureka客户端其实就是一个java客户端，负责与服务器的交互，用作轮询负责均衡，故障切换支持等。 Netflix在其生产环境中使用的是另外的客户端，它提供基于流量、资源利用率以及出错状态的加权负载均衡。

依赖

@EnableEurekaServer:这个注解需要在springboot工程的启动application类上加,表示启动一个服务注册中心;

@EnableEurekaClient:表明自己是一个eurekaclient;

* Ribbon：基于客户端的负载均衡组件。

Spring Cloud Ribbon 是基于 Netflix Ribbon 实现的一套客户端负载均衡的工具。

简单的说，Ribbon 是 Netflix 发布的开源项目，主要功能是提供客户端的软件负载均衡算法，将 Netflix 的中间层服务连接在一起。Ribbon 客户端组件提供一系列完善的配置项如连接超时，重试等。简单的说，就是在配置文件中列出Load Balancer（简称LB）后面所有的机器，Ribbon 会自动的帮助你基于某种规则（如简单轮询，随机连接等）去连接这些机器。我们也很容易使用 Ribbon 实现自定义的负载均衡算法。

1. 负载均衡

LB，即负载均衡(Load Balance)，在微服务或分布式集群中经常用的一种应用。

负载均衡简单的说就是将用户的请求平摊的分配到多个服务上，从而达到系统的 HA。

常见的负载均衡有软件Nginx，LVS，硬件 F5等。

相应的在中间件，例如：dubbo和SpringCloud中均给我们提供了负载均衡，SpringCloud的负载均衡算法可以自定义。

2. 集中式 LB

即在服务的消费方和提供方之间使用独立的LB设施(可以是硬件，如F5, 也可以是软件，如nginx), 由该设施负责把访问请求通过某种策略转发至服务的提供方

3. 进程内 LB

将LB逻辑集成到消费方，消费方从服务注册中心获知有哪些地址可用，然后自己再从这些地址中选择出一个合适的服务器。

Ribbon就属于进程内LB，它只是一个类库，集成于消费方进程，消费方通过它来获取到服务提供方的地址。

* Ribbon 在工作时分成两步

1. 第一步先选择 EurekaServer ,它优先选择在同一个区域内负载较少的server。
2. 第二步再根据用户指定的策略，在从server取到的服务注册列表中选择一个地址。
3. 其中Ribbon提供了多种策略：比如轮询、随机和根据响应时间加权。

配置

* Hystrix：容错框架，能够防止服务的雪崩效应。

 在一个分布式系统中，必然会有部分系统的调用会失败。Hystrix是一个通过添加超时容错和失败容错逻辑来帮助你控制这些分布式系统的交互。Hystrix通过隔离服务之间的访问，阻止他们之间的级联故障以及提供后背选项来实现这些，所有新而这些都用来提高系统的整体弹性。

Hystrix用来做什么

    Hystrix被设计用来解决一下几个方面

通过第三方（一般来源网络）的调用，给与保护和控制延迟和失败。

在复杂的分布式系统中复制级联失败。

在可能的情况下，回滚挥着优雅的失败。

实现几乎实时监控，警报和操作控制。

* Feign：Web 服务客户端，能够简化 HTTP 接口的调用。
* （接口调用 面向接口 声明式调用就像调用本地方法一样调用远程方法）

Feign是一个声明式WebService客户端。**使用Feign能让编写Web Service客户端更加简单**, 它的使用方法是**定义一个接口**，然后**在上面添加注解**，同时也支持JAX-RS标准的注解。Feign也支持可拔插式的编码器和解码器。Spring Cloud对Feign进行了封装，使其支持了Spring MVC标准注解和HttpMessageConverters。**Feign可以与Eureka和Ribbon组合使用以支持负载均衡**。

Feign是一个声明式的Web服务客户端，使得**编写Web服务客户端**变得非常容易，

**只需要创建一个接口，然后在上面添加注解即可**。

**Zuul(如欧)是什么**

ZUUL是Netflix开源的微服务网关，Spring Cloud Zuul对Zuul进行了整合和增强，它可以和Eureka、Ribbon、Hystrix等组件配合使用，Zuul组件包含了对**请求的路由**和**过滤**两个最主要的功能：其中路由功能负责将外部请求转发到具体的微服务实例上，是实现外部访问统一入口的基础。而过滤器功能则负责对请求的处理过程进行干预，是实现请求校验、服务聚合等功能的基础。通过将Zuul和Eureka进行整合，将Zuul自身注册为Eureka服务治理下的应用，同时从Eureka中获得其他微服务的消息，也即**以后通过客户端(浏览器，APP)访问微服务都是通过Zuul跳转后获得**。

注意：Zuul服务最终还是会注册进Eureka Server

## mysql调优

sql语句优化 查询尽量不要查询\* 需要哪些字段查询哪些字段

建索引

**什么情况下应该建索引**

l  **主键自动建立唯一索引；**

l  **频繁作为查询条件的字段应该创建索引**

l  **查询中与其它表关联的字段，外键关系建立索引**

l  **单键/组合索引的选择问题， 组合索引性价比更高**

l  **查询中排序的字段，排序字段若通过索引去访问将大大提高排序速度**

l  **查询中统计或者分组字段**

**什么情况下不应该建索引**

l  **表记录太少**

l  **经常增删改的表或者字段**

l  **Where 条件里用不到的字段不创建索引**

l  **过滤性不好的不适合建立索引(字段重复的值比较多，超过50%)**

**MySQL索引分类**

单值索引：

即一个索引只包含单个列，一个表可以有多个单列索引。

唯一索引

索引列的值必须唯一，但允许有空值

主键索引

设定为主键后数据库会自动建立索引

复合/组合索引

即一个索引包含多个列

**MySQL索引的基本语法**

|  |  |
| --- | --- |
| **操作** | **命令** |
| 创建 | create   [unique ] index indexname on table\_name(column)); |
| 删除 | drop index indexname on mytable; |
| 查看 | show index from table\_name\g; |
| 使用alter命令 | alter table tbl\_name add primary key (column\_list) : 该语句添加一个主键，这意味着索引值必须是唯一的，且不能为 null。 |
| alter table tbl\_name add primary key (column\_list) |
| alter table tbl\_name add index index\_name (column\_list): 添加普通索引，索引值可出现多次。 |
| alter table tbl\_name add fulltext index\_name (column\_list):该语句指定了索引为 fulltext ，用于全文索引。 |

## Redis

非关系型数据库 nosql 主要是开源免费的

redis 是一个高性能的key-value数据库 由c++语言编写

数据类型丰富 ：string(字符串) ,list(链表),set(无序集合),zset(有序集合),hash(哈希类型)

这些数据类型都支持push/pop，add/remove及取交集并集和差集及更丰富的操作， 而且这些操作都是原子性的, Redis 是单线程模型，指的是执行 Redis 命令的核心模块是单线程的，而不是整个 Redis 实例就一个线程,redis4.0就开始由多线程的概念了,比如redis需要通过多线程方式在后台删除对象

Redis 6.0 其实也是多线程的 但是默认是关闭的 需要在redis.conf 中开启

数据类型：

String :存进去的数据 直接可以取出来用

List : 链表 可以用来做消息队列 先进先出

Set ：无序集合

Zset ：有序集合

Hash : key filed value

持久化方案 RDB AOF

Rdb : 基于快照的方式将某个时间段的数据进行存储 适用于大数据的场合使用 数据量过小的话可能会丢失某一时间段的数据

AOF : 将对数据库数据造成改变的操作已文本的方式记录到aof文件中

Rdb和Aof的区别

RDB持久化是指在指定的时间间隔内将内存中的数据集快照写入磁盘，实际操作过程是fork一个子进程，先将数据集写入临时文件，写入成功后，再替换之前的文件，用二进制压缩存储。  
  
 RDB持久化保存键空间的所有键值对(包括过期字典中的数据),并以二进制形式保存，符合rdb文件规范，根据不同数据类型会有不同处理。  
  
 AOF持久化以日志的形式记录服务器所处理的每一个写、删除操作，查询操作不会记录，以文本的方式记录，可以打开文件看到详细的操作记录。  
  
 AOF持久化保存redis服务器所执行的所有写命令来记录数据库状态,在写入之前命令存储在aof\_buf缓冲区

RDB 在恢复大数据集时的速度比 AOF 的恢复速度要快，AOF在运行效率上往往会慢于RDB。

Redis的三大问题

击穿 （恶意访问一个不存在的数据）

指的是**单个key**在缓存中查不到，去数据库查询，这样如果数据量不大或者并发不大的话是没有什么问题的。

   如果数据库数据量大并且是**高并发**的情况下那么就可能会造成数据库压力过大而崩溃

解决方案 ：

1**缓存空对象**  当查询的时候，如果缓存和数据库中都没有，我们就将这个数据以空的形式存放在缓存中，(或者是给一个false的标示)这样就不用去数据库就可以知道不存在，减少对数据库查询的次数，当我们这个值发生改变的时候，我们在重新进行赋值；

2使用布隆过滤器 BloomFilter

业务系统查询数据时，先从布隆过滤器判断该key是否存在，不存在则说明数据库也不存在，直接返回空。若存在则走正常流程。

这种方法适用于数据命中不高，数据相对固定实时性低(通常是数据集较大)的应用场景，代码维护较为复杂，但是缓存空间占用少

对于恶意攻击，往往key只会使用一次，即使缓存了也没有用，浪费缓存空间，这种场景适合使用布隆过滤器

雪崩 ：

1、大量key同一时间过期造成雪崩  
 2、缓存服务器宕机造成雪崩

解决方案 ：

1 对缓存做集群高可用部署，防止单节点宕机造成雪崩  
 2 对key过期时间增加一个随机数，防止同时大面积key过期，或者对热点数据设置为永不过期

穿透：

是指一个key非常热点，在不停的扛着大并发，大并发集中对这一个点进行访问，当这个key在失效的瞬间，持续的大并发就穿破缓存，直接请求数据库，就像在一个屏障上凿开了一个洞。

**解决方案 ：**

对热门访问key早早的做好了准备，让缓存永不过期；

Redis调优

## redis主从复制

概念

    主从复制模型中，有多个redis节点。

    其中，有且仅有一个为主节点Master。从节点Slave可以有多个。

只要网络连接正常，Master会一直将自己的数据更新同步给Slaves，保持主从同步。

特点

（1）主节点Master可读、可写.

（2）从节点Slave只读。（read-only）

因此，主从模型可以提高读的能力，在一定程度上缓解了写的能力。因为能写仍然只有Master节点一个，可以将读的操作全部移交到从节点上，变相提高了写能力。

主从模式的一个缺陷 就是突然主挂掉了 那么就没有一个可往数据库中写数据的节点了 那么这个就是主从的缺陷 但是下面的东西可以解决

## Sentinel哨兵模式

监控主数据库和从数据库是否正常运行。

主数据库出现故障时自动将从数据库转换为主数据库。

Redis 的 Sentinel 系统用于管理多个 Redis 服务器（instance）， 该系统执行以下三个任务：

**监控（Monitoring）**： Sentinel 会不断地检查你的主服务器和从服务器是否运作正常。

**提醒（Notification）**： 当被监控的某个 Redis 服务器出现问题时， Sentinel 可以通过 API 向管理员或者其他应用程序发送通知。

**自动故障迁移**（Automatic failover）： 当一个主服务器不能正常工作时， Sentinel 会开始一次自动故障迁移操作， 它会进行选举，将其中一个从服务器升级为新的主服务器， 并让失效主服务器的其他从服务器改为复制新的主服务器； 当客户端试图连接失效的主服务器时， 集群也会向客户端返回新主服务器的地址， 使得集群可以使用新主服务器代替失效服务器。

## Cluster 集群模式

哨兵 Sentinel模式基本可以满足一般生产的需求，具备高可用性。**但是当遇到单机内存，并发和流量等瓶颈的时候，主从模式或哨兵Sentinel模式就不能满足需求了**，这个时候需要**对存储的数据进行分片**，**将数据存储到多个Redis实例中**。

什么是redis集群

集群由**多个节点(Node)组成**，Redis的数据分布在这些节点中。每个节点都保存着数据和整个集群的状态，每个节点都和其它所有节点连接。

Redis集群的作用

数据分区：数据分区(或称数据分片)是Redis集群最核心的功能。

Redis集群将数据分散到多个节点，一方面突破了Redis单机内存大小的限制，存储容量大大增加；另一方面每个主节点都可以对外提供读服务和写服务，大大提高了集群的响应能力。

高可用：集群支持主从复制和主节点的自动故障转移（与哨兵类似）；当任一节点发生故障时，集群仍然可以对外提供服务。

Redis集群特点

1.        所有的Redis节点彼此互联(PING-PONG机制)，内部使用二进制协议优化传输速度和带宽。

2.        节点的fail是通过集群中超过半数的节点检测失效时才生效。

3.        客户端与Redis节点直连，不需要中间proxy层。客户端不需要连接集群所有节点，连接集群中任何一个可用节点即可。

4.        Redis集群把所有的物理节点映射到[0-16383]slot上（不一定是平均分配）,集群负责维护node<->slot<->value。

5.        Redis集群预分将所有的存储空间划分为好16384个哈希槽，集群中的每个节点负责处理一部分哈希槽。当需要在 Redis 集群中放置一个 key-value 时，根据 CRC16(key) mod 16384的值，决定将一个key放到哪个哈希槽中。然后将key放到哈希槽对应的节点上。

业务

## 日志记录

涉及到 java注解 和 aop 横切切面编程

**自定义注解**

1. 注解是一种元数据形式。即注解是属于java的一种数据类型，和类、接口、数组、枚举类似。
2. 注解用来修饰，类、方法、变量、参数、包。
3. 注解不会对所修饰的代码产生直接的影响。

**自定义注解的配置使用**

* **第一步，定义注解——相当于定义标记；**
* **第二步，配置注解——把标记打在需要用到的程序代码中；**
* **第三步，解析注解——在编译期或运行时检测到标记，并进行特殊操作。**

**注解其实就是一种标记，可以在程序代码中的关键节点（类、方法、变量、参数、包）上打上这些标记，然后程序在编译时或运行时可以检测到这些标记从而执行一些特殊操作**

注解的定义

注解通过 @interface 关键字进行定义。

public @interface TestAnnotation { }

元注解

元标签有 @Retention 翻译为持久力、保持力。即用来修饰自定义注解的生命力。

注解的生命周期有三个阶段：1、Java源文件阶段；2、编译到class文件阶 段；3、运行期阶段。

@Documented 是被用来指定自定义注解是否能随着被定义的java文件生成到JavaDoc文档当中

@Target 是专门用来限定某个自定义注解能够被应用在哪些Java元素上面的。

@Inherited 是指定某个自定义注解如果写在了父类的声明部分，那么子类的声明部分也能自动拥有该注解。@Inherited注解只对那些@Target被定义为ElementType.TYPE的自定义注解起作用。

**aop 横切切面编程**

aop：面向切面编程是软件编程思想发展到一定阶段的产物，是面向对象编程(OOP)的有益补充。AOP一般适用于具有横切逻辑的场合，如安全控制、事务管理、日志记录、性能统计等。面向切面编程简单地说就是在不改变源程序的基础上为代码段增加新的功能，对代码段进行增强处理。

aop的代理通过jdk动态代理，也可以通过cglib（谁个来波）实现，默认是通过jdk动态代理实现的。JDK动态代理需要接口的支持，如果没有接口只有类，则使用cglib实现。

@Before 前置通知，在方法执行之前执行

@After 后置通知，在方法执行之后执行（无论是否发生异常）还不能访问目标方法执行的结果

@AfterRunning 返回通知，在方法正常结束后 返回结果之后执行 可以访问方法的返回值

@AfterThrowing 异常通知，在方法抛出异常之后

@Around 环绕通知，围绕着方法执行

**Java反射**

1. *//第一种方式获取Class对象*
2. Student stu1 = new Student();*//这一new 产生一个Student对象，一个Class对象。*
3. Class stuClass = stu1.getClass();*//获取Class对象*

*2.//第二种方式获取Class对象*

1. Class stuClass2 = Student.class;
2. *//第三种方式获取Class对象*

Class stuClass3 = Class.forName("fanshe.Student");//注意此字符串必须是真实路径，就是带包名的类路径，包名.类名

1. *//1.加载Class对象*

Class class = Class.forName("类的全路径");

获取所有的构造方法(包括：私有、受保护、默认、公有)

conArray = class.getDeclaredConstructors();

获取所有公有构造方法

Constructor[] conArray = class.getConstructors();

获取公有、无参的构造方法

Constructor con = class.getConstructor(null);

Method[] conarray = class.getDeclaredMethods();//获取自身所有的方法(private、public、protected，和访问权限无关)，不包括继承的

**日志记录具体实现**

声明一个普通类 加上以下注解

@Component *//把类交给ioc容器管理*

@Aspect *//声明切面类*

通过aop的

*//前置通知 @before*

@Before(**"execution(\* org.fh.controller.\*.\*(..))"**)

方法参数中加入 JoinPoint

*//获取方法签名*

MethodSignature methodSignature = (MethodSignature)joinPoint.getSignature();

*//获取方法签名对象*

Method method = methodSignature.getMethod();

*//判断此方法是否需要记录日志*

*//判断方法上是否有日志注解*

logJilv annotation = method.getAnnotation(logJilv.**class**);

if(annotation != null){

}

有注解的话做日志记录

## 定时任务

springboot项目中

布置定时任务

1

在启动类上加上 @EnableScheduling（哎bo死曾怼林）注解 开启定时任务

新建一个类 加上@component 将此类交给ioc进行管理

5秒后执行此方法

要在任务方法上加@Scheduled(fixedDelay = 5000)

//表示每天8时30分0秒执行 @Scheduled(cron = "0 0,30 0,8 ? \* ? ")

\* 第一位，表示秒，取值0-59 \* 第二位，表示分，取值0-59 \* 第三位，表示小时，取值0-23 \* 第四位，日期天/日，取值1-31 \* 第五位，日期月份，取值1-12 \* 第六位，星期，取值1-7，星期一，星期二...，注：不是第1周，第二周的意思 另外：1表示星期天，2表示星期一。 \* 第7为，年份，可以留空，取值1970-2099\*

下面列举几个例子供大家来验证：

0 0 3 \* \* ? 每天3点执行 0 5 3 \* \* ? 每天3点5分执行 0 5 3 ? \* \* 每天3点5分执行，与上面作用相同 0 5/10 3 \* \* ? 每天3点的 5分，15分，25分，35分，45分，55分这几个时间点执行 0 10 3 ? \* 1 每周星期天，3点10分 执行，注：1表示星期天 0 10 3 ? \* 1#3 每个月的第三个星期，星期天 执行，#号只能出现在星期的位置