# B简历：

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

1. **基本信息：**

|  |  |  |  |
| --- | --- | --- | --- |
| 基本情况 | | | |
| 姓名 | 王宇杰 | 性别 | 男 |
| 出生日期 | 1996-05-10 | 属相 | 鼠 |
| 现住地 |  |  |  |
| 大学情况 | | | |
| 学校 | 潍坊科技学院 | 专业 | 软件工程 |
| 入学时间 | **2014-09-01** | 毕业时间 | 2018-07-01 |
| 专业课 | [java语言](https://www.baidu.com/s?wd=Java%E8%AF%AD%E8%A8%80&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)程序设计、Java[企业级开发](https://www.baidu.com/s?wd=%E4%BC%81%E4%B8%9A%E7%BA%A7%E5%BC%80%E5%8F%91&tn=SE_PcZhidaonwhc_ngpagmjz&rsv_dl=gh_pc_zhidao)、数据库技术与应用、大型数据库处理技术、网页设计与开发、计算机网络与通讯、软件工程 | | |
| 学历 | **本科** | 学位 | **学士学位** |
| 学校地址 | 山东省寿光市金光街1299号 | 校长 | 李昌武 |
|  |  |  |  |
| 去学校的方式： | 潍坊科技学院在寿光，寿光火车站，然后坐5路公交车经过3站即可到达。 | | |
| 上小学日期： | **2002年9月 6岁六年** | | |
| 上初中日期： | **2008年9月 12岁三年** | | |
| 上高中日期： | **2011年9月 15岁三年** | | |
| 上大学日期： | **2014年9月 18岁四年** | | |
| 毕业时间： | **2018年 7月 22岁** | | |
| 入学时间： | **2014年9月** | | |
| 毕业时间： | **2018年7月** | | |
| 在校学生： | **5000左右** | | |
| 有教师： | **不太清楚** | | |

|  |  |
| --- | --- |
| **第一家公司情况：** | **河南中原金科信息技术有限公司(只说公司简称)** |
| **公司地址**： | 郑州市 金水区 河南自贸大厦 20层 |
| **公司规模：** | **20人左右，我们项目组6人，一个项目经理，1名前端，1个测试，1个运维，两个java开发** |
| **公司性质：** | **民营** |
| **公司行业：** | 计算机硬件、IT服务、计算机软件、互联网 |
| **去公司路线：** | 5号线大概5站就到了河南自贸大厦了 |
| **入职时间：** | **2017年8月** |
| **离职时间：** | **2019年6月初** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **李顺** |
| **税前：** | **7K** |
| **税后：** | **6.5K** |
| **缴税：** | **500左右**(具体原因不清楚当时公司就是这样扣的税) |
| **期望薪资：** | **10K** |
| **薪资的构成:** | **岗位工资(5K)+绩效工资(9K)** |
| 四年涨薪  **2017年工作的时候7K，税后是6K左右，**  **转正税后7K左右**  **2018上半年涨了1K左右 ，总8K左右**  **2018年下半年月涨了1k,总的9k左右**  **2019年6月跳槽涨了2K左右 总11K**  **2020年7月涨2K总 13左右**   |  |  | | --- | --- | | **第二家公司情况：** | **北京优易捷科技有限公司(只说公司简称)** | | **公司地址**： | 北京房山区长阳天地9号院10号楼716 | | **公司规模：** | **50人左右，我们项目组8人，一个项目经理，1名前端，1个测试，2个运维，3个java开发** | | **公司性质：** | **民营** | | **公司行业：** | 计算机硬件、IT服务、计算机软件、互联网 | | **去公司路线：** | **嘉瑞通小区 房山线 良乡南关站A2口五站 在巴黎房站下车步行约10分钟** | | **入职时间：** | **2019年6月** | | **离职时间：** | **2021年4月** | | **职位：** | **Java工程师** | | **项目经理名字：** | **于扬** | | **税前：** | **10K** | | **税后：** | **9K** | | **缴税：** | **1000左右**(具体原因不清楚当时公司就是这样扣的税) | | **期望薪资：** | **15K** | | **薪资的构成:** | **岗位工资(5K)+绩效工资(9K)** |   **2021年4月离职时工资为13K** | | | |

|  |
| --- |
| **你为什么从上家公司离职？** |
| **之前在外包公司，最近公司接的项目比较少，没什么事情，太安逸了，想换一个新的平台去充实提升自己!** |
| |  | | --- | |  | | **你的优缺点是啥？** |   **面试常见问题：**  **优点:** 团队协作能力强**抗压自学能力强（有一次项目时间很紧，经常工作到12点半之后，连续攻坚一个月终于把项目完美的完成了。），**自学能力强  **自学能力强** (**1.从官网上下载需要的文件。2.导入相关的文件到项目中。3.在需要用到的文件中引入具体的js，css。具体在做例子的时候先做静态例子，再和后台结合做出来动态数据的例子。特别在做动态数据例子的时候，我们可以通过结合官网上提供的demo,通过f12观察这个几个问题:**  **<1>js插件怎么向后台发送请求；**  **<2>传递了哪些参数以及参数的含义;**  **<3>后台怎么接受参数;**  **<4>后台接受到参数后怎么处理,才是js组件所需的响应格式;**  **<5>如何将json数据响应给前台;**  **<6>前台如何接受到该json数据；**  **<7>接受到数据之后如何处理，才能将数据展现出来。按照这个思路和流程，我们在学习第三方的js组件时，就能够快速的掌握并融入到项目中。)缺点: 这几年做程序，因为要不断的对代码进行验证确认，整体达到一定效果才可以，所以感觉自己现在有点强迫症** |
| **你交社保了吗？为啥没交？** |
| **没交，在上家公司的时候，人事说咱们公司的五险一金如果要上的话都是从自己工资里面扣的，当时感觉没啥必要也就没上。** |
| **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？** |
| **近期以提升技术为主，然后对公司的业务也要尽快掌握，然后向项目经理方面发展。** |
| **是统招吗 是**  **是本科吗 是**  **能在学信网查到吗 我当时上的是民办大学，能在民教网查到！** |
| **你们公司还有什么福利？**  **多少薪：**  14薪  **车补**  200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2个月的工资，  一般是过年的时候和当月工资一块发  **电话补助**  300  **饭/餐补**  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**

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

**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！**

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

坐稳再开口！！ 主动性！！ 开门见山！！大白话！！三秒钟原则！！学会反问！！

**Hr人事，leader领导，tean团队，pm项目/产品经理，cto首席技术执行官，coo首席运营官，CEO董事长、**

**介绍流程：**

**springboot的运行原理à微服务细讲à基于token登陆à订单**

**à支付àsql 优化àjava反射导出excelà线程池**

## 主动说：

**你好！**我叫王宇杰，今天来应聘咱们公司 java 开发！

### springboot

微服务 进行接口开发的

Spring Boot是由 Pivotal 团队提供的全新的框架，spring boot 整合了所有的框架让spring应用程序尽快跑起来并且尽可能减少你的配置文件

（1）优点：

1）为所有 Spring 开发提供一个更快更广泛的入门体验

2）零配置。无冗余代码生成和XML 强制配置，遵循“约定大于配置”（我们可以直接用它里面默认的配置，不需要去修改）

3）集成了大量常用的第三方库的配置， Spring Boot 应用为这些第三方库提供了几乎可以零配置的开箱即用的能力

4）提供一系列大型项目常用的非功能性特征，如嵌入式服务器web、安全性、度量、运行状况检查、外部化配置等

5）Spring Boot 不是Spring 的替代者，Spring 框架是通过 IOC 机制来管理 Bean 的。Spring Boot 依赖 Spring 框架来管理对象的依赖。Spring Boot 并不是Spring 的精简版本，而是为使用 Spring 做好各种产品级准备

（2）常用注解：

@restcontroller @getmapping @PathVariable @MapperScan

@springbootapplication（他是由三个注解组成的分别是@springbootConfiguration读取配置文件，配置文件的路径是当前根目录@EnableAutoConfiguration开启自动配置，扫描当前的所有依赖的jar包，发现新的依赖出现将会根据依赖完成各种自动配置@ComponentScan属于spring 框架(@Component,@Service,@Controller,@Repository,@Entity)，扫描范围默认情况下是启动类所在的同名包及其子孙包）

# springcloud分布式框架

### spring-cloud-eureka：

spring-cloud-eureka是spring-cloud-nettfix微服务套件中的一部分， 它基于nettfix-

eureka做了二次封装，主要负责微服务架构中的服务治理功能。

服务治理一般都会有两个功能：服务注册、服务发现。通常会有一个注册中心，每个服务单

元向注册中心登记自己信息，比如提供的服务，ip, 端口以及一些附近加信息等。注册中心

会将新的服务实例发送给其它依赖此服务的实例。

核心注解@EnableEurekaClient

### springcloud-Feign:

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也是运行在消费者端的，使用 Ribbon 进行负载均衡，所以 Feign 直接内置了 Ribbon。而与Ribbon不同的是，通过feign只需要定义服务绑定接口且以声明式的方法，优雅而简单的实现了服务调用。

核心注解：@EnableFeignClients//声明fegin

### Spring Cloud Ribbon:

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。核心：

启动类中注入bean

@Bean

@LoadBalanced //声明 ribbon （并对resttemmplate进行负载均衡 调用的时候直接写服务名就行 不用写ip和端口）

RestTemplate restTemplate() {

return new RestTemplate();// RestTemplate对httpclient发送请求 进行封装了

}

### Spring Cloud Hystrix：

Hystrix中文是豪猪的意思，在Spring Cloud中防雪崩的利器就是Spring Cloud Hystrix，Spring Cloud Hystrix是基于Netflix Hystrix实现的。在分布式系统里，许多依赖不可避免的会调用失败，比如超时、异常等，Hystrix能够保证在一个依赖出问题的情况下，不会导致整体服务失败，避免级联故

Hystrix能干啥

l 服务熔断

当Hystrix Command请求后端服务失败数量超过一定比例(默认50%), 断路器会切换到开路状态(Open). 这时所有请求会直接失败而不会发送到后端服务. 断路器保持在开路状态一段时间后(默认5秒), 自动切换到半开路状态(HALF-OPEN).

l 服务降级

Fallback相当于是降级操作, 我们可以实现一个fallback方法, 当请求后端服务出现异常或者服务熔断后的时候, 直接调用本地fallback方法，告知后面的请求服务不可用了，不要再来了。

l 服务容错

l 线程和信号隔离

在Hystrix中, 主要通过线程池来实现资源隔离. 通常在使用的时候我们会根据调用的远程服务划分出多个线程池.比如说，一个服务调用两外两个服务，你如果调用两个服务都用一个线程池，那么如果一个服务卡在哪里，资源没被释放。后面的请求又来了，导致后面的请求都卡在哪里等待，导致你依赖的A服务把你卡在哪里，耗尽了资源，也导致了你另外一个B服务也不可用了。这时如果依赖隔离，某一个服务调用A B两个服务，如果这时我有100个线程可用，我给A服务分配50个，给B服务分配50个，这样就算A服务挂了，我的B服务依然可以用。

l 请求缓存

比如一个请求过来请求我userId=1的数据，你后面的请求也过来请求同样的数据，这时我不会继续走原来的那条请求链路了，而是把第一次请求缓存过了，把第一次的请求结果返回给后面的请求。

l 请求合并

我依赖于某一个服务，我要调用N次，我发了N条请求发然后拿到一堆结果，这时候我们可以把多个请求合并成一个请求，这样我们只需调用一次远程服务，提升了效率。

l 近乎实时的服务监控

### Hystrix 客户端核心：

@EnableHystrix// 熔断 启动类中

对你要熔断的方法

// 此注解标识是要熔断 fallbackMethod 熔断后进行执行的方法

@HystrixCommand(fallbackMethod = "errorMethod",commandProperties = {

@HystrixProperty(name = "execution.isolation.thread.timeoutInMilliseconds",value = "3000")

}) //对此方法进行熔断

Hystrix 服务端：

启动类

@EnableHystrixDashboard

启动类中

/\* 提供监控平台的路径\*/

@Bean

public ServletRegistrationBean getServlet() {

HystrixMetricsStreamServlet streamServlet = new HystrixMetricsStreamServlet();

ServletRegistrationBean registrationBean = new ServletRegistrationBean(streamServlet);

registrationBean.setLoadOnStartup(1);

registrationBean.addUrlMappings("/hystrix.stream");

registrationBean.setName("HystrixMetricsStreamServlet");

return registrationBean;

### Spring Cloud zull：

Zuul是Netflix开源的微服务网关，他可以和Eureka,Ribbon,Hystrix等组件配合使用。Zuul组件的核心是一系列的过滤器，这些过滤器可以完成以下功能：

身份认证和安全:

识别每一个资源的验证要求，并拒绝那些不符的请求审查与监控：

***动态路由：***

动态将请求路由到不同后端集群

***压力测试：***

逐渐增加指向集群的流量，以了解性能

***负载分配：***

为每一种负载类型分配对应容量，并弃用超出限定值的请求

***静态响应处理：***

边缘位置进行响应，避免转发到内部集群

# Docker

## 什么是docker（虚拟机VMware）

Docker是一个开源的**应用容器引擎，**Docker可以让开发者可以**打包他们的应用以及依赖包到一个轻量级、可移植的容器中**，然后发布到任何流行的Linux机器上，也可以实 现虚拟化，容器是完全使用**沙箱机制**，相互之间不会有任何接口，**重要的是容器性能开销极低**。

Docker实现了一次安装到处运行；方便做持续集成并有助于整体发布的容器虚拟化技术。

## docker的优点：

* **更快速的启动时间**
* **更快速的应用交付和部署**
* **更便捷的升级和扩缩容**
* **更简单的系统运维**
* **更高效的计算资源利用**

## Docker环境搭建

centOS6.9安装docker：

1. yum install -y epel-release

Docker使用EPEL发布，RHEL系的OS首先要确保已经持有EPEL仓库，否则先检 查OS的版本，然后安装相应的EPEL包。

1. yum install -y docker-io
2. 安装后的配置文件：/etc/sysconfig/docker
3. 启动Docker后台服务：service docker start

docker version验证

* **CentOS7安装Docker**

1. 安装需要的软件包

yum install -y yum-utils device-mapper-persistent-data lvm2

1. 设置Docker的yum源

yum-config-manager --add-repo <http://mirrors.aliyun.com/docker-ce/linux/centos/docker-ce.repo>

1. 更新yum软件包索引

yum makecache fast

1. 安装Docker CE

yum -y install docker-ce

1. 启动Docker

systemctl start docker

1. 测试

docker version

1. 卸载

systemctl stop docker

yum -y remove docker-ce

rm -rf /var/lib/docker

## 自定义镜像

有两种方式

commit 常用的是build 命令时：docker build -f Dockerfile的路径 -t 镜像名称:版本号 .

编写Dockerfile文件

语法

FROM 指定基础镜像

MAINTAINER 指定镜像维护者信息 maintainer

RUN 用于执行指定脚本命令

CMD 指定启动容器时执行的命令

EXPOSE 指定容器暴露的端口

ENV 指定环境变量

ADD 将文件从宿主机复制到容器指定位置，同时对压缩文件有自动解压功能

COPY 将文件从宿主机复制到容器指定位置

ENTRPOINT 设置容器启动时需要运行的命令

WORKDIR 为后续的如RUN、CMD、ENTRYPOINT、COPY、ADD指定工作目录

## 创建私有仓库

1、拉取私有仓库镜像

docker pull registry

2、启动私有仓库容器

docker run -id --name=registry -p 5000:5000 registry

3、打开浏览器 输入地址

http://私有仓库服务器ip:5000/v2/\_catalog，看到{"repositories":[]} 表示私有仓 库搭建成功

4、修改daemon.json

vim /etc/docker/daemon.json

# 在上述文件中添加一个key，保存退出。此步用于让 docker 信任私有仓库地 址；注意将私有仓库服务器ip修改为自己私有仓库服务器真实ip

{"insecure-registries":["私有仓库服务器ip:5000"]}

5、重启docker 服务

systemctl restart docker

## Docker总结 (面试就是linux的常用命令，docker的相关命令)

帮助命令：docker info

查看镜像：docker images

删除镜像：docker rmi 镜像

搜索镜像：docker search 镜像名

拉取镜像：docker pull 镜像名

按照镜像名称搜索镜像docker search

创建容器：docker run -id –name 容器名 -p 8080:8080 镜像名:版本号

删除容器：docker rm -f 容器ID

查容器：docker ps 查询当前运行 -a 查询运行和不运行的容器

# *事务*

## 什么是事务？

事务就是一组有逻辑的操作单元，即一组sql语句，事务是访问数据库的一个操作序列，数据库应用系统通过事务集来完成对数据库的存取

## 事务的四大特性

ACID

原子性：指事务是不可分割的工作单位，事务的操作要么都发生要么都不发生

隔离性：指并发执行的事务之间不能互相干扰，事务与事务之间是相互隔离的

一致性：指事务必须从一个一致性状态到另一个一致性状态

持久性：指事务操作的结果不轻易发生改变，事务的操作是永久性的即使后来的故障或异常也不会改变结果。

## 事务的传播特性

1 .PROPAGATION\_REQUIRED（required）: 如果存在一个事务，则支持当前事务。 如果没有事务则开启

2. PROPAGATION\_SUPPORTS(supports): 如果存在一个事务，支持当前事务。如果 没有事 务，则非事务的执行

3. PROPAGATION\_MANDATORY(mandatory): 如果已经存在一个事务，支持当前 事务。如果没有一个活动的事务，则抛出异常。

4. PROPAGATION\_REQUIRES\_NEW(requires\_new): 总是开启一个新的事务。如果 一个事务已经存在，则将这个存在的事务挂起。

5. PROPAGATION\_NOT\_SUPPORTED(supported): 总是非事务地执行，并挂起任何 存在的事务。

6. PROPAGATION\_NEVER(never): 总是非事务地执行，如果存在一个活动事务，则 抛出异常

7. PROPAGATION\_NESTED(nested)：如果一个活动的事务存在，则运行在一个嵌 套的事务中. 如果没有活动事务,则按 TransactionDefinition.PROPAGATION\_REQUIRED 属性执行

## 事务的隔离级别

并发事务导致的问题：

在许多事务处理同一个数据时，由于没有采取有效的隔离机制，并发处理数据时，会带来一些问题。

①、第一类丢失更新：撤销一个事务时，把其他事务已提交的更新数据覆盖。

②、脏读：一个事务读取到另一个事务未提交的更新数据。

③、幻读也叫虚读：一个事务执行两次查询，第二次结果集包含第一次中没有或某些行已经被删除的数据，造成两次结果不一致，只是另一个事务在这两次查询中间插入或删除了数据造成的。

④、不可重复读：一个事务两次读取同一行的数据，结果得到不同状态的结果，中间正好另一个事务更新了该数据，两次结果相异，不可被信任。

⑤、第二类丢失更新：是不可重复读的特殊情况。如果两个事物都读取同一行，然后两个都进行写操作，并提交，第一个事物所做的改变就会丢失。

1、DEFAULT（default）

默认隔离级别，每种数据库支持的事务隔离级别不一样，如果Spring配置事 务时将isolation设置为这个值的话，那么将使用底层数据库的默认事务隔离 级别。顺便说一句，如果使用的MySQL，可以使用"select @@tx\_isolation" 来查看默认的事务隔离级别

2、READ\_UNCOMMITTED(read\_uncommitted)

读未提交，即能够读取到没有被提交的数据，所以很明显这个级别的隔离机 制无法解决脏读、不可重复读、幻读中的任何一种，因此很少使用

3、READ\_COMMITED(read\_commited)

读已提交，即能够读到那些已经提交的数据，自然能够防止脏读，但是无法 限制不可重复读和幻读

4、REPEATABLE\_READ(repeatable\_read)

重复读取，即在数据读出来之后加锁，类似"select \* from XXX for update"， 明确数据读取出来就是为了更新用的，所以要加一把锁，防止别人修改它。 REPEATABLE\_READ的意思也类似，读取了一条数据，这个事务不结束，别 的事务就不可以改这条记录，这样就解决了脏读、不可重复读的问题，但是 幻读的问题还是无法解决

5、SERLALIZABLE(serlalizable)

串行化，最高的事务隔离级别，不管多少事务，挨个运行完一个事务的所有 子事务之后才可以执行另外一个事务里面的所有子事务，这样就解决了脏读、 不可重复读和幻读的问题了

# Mysql （优化）

## 创建索引

什么情况下创建索引：

1、**主键自动建立唯一索引；**

**2、频繁作为查询条件的字段应该创建索引**

**3、查询中与其它表关联的字段，外键关系建立索引**

**4、单键/组合索引的选择问题， 组合索引性价比更高**

**5、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**

**6、查询中统计或者分组字段**

**什么情况下不应该创建索引：**

**1、表记录太少**

**2、经常增删改的表或者字段**

**3、Where 条件里用不到的字段不创建索引**

**4、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%)**

## 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

## 创建索引命令：

|  |  |
| --- | --- |
| 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。 |

分表

# Redis：

## 什么是redis:

Redis本质上是一个Key-Value类型的内存数据库, 整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据flush(刷新)到硬盘上进行保存。因为是纯内存操作，Redis的性能非常出色，每秒可以处理超过 10万次读写操作，是已知性能最快的Key-Value DB。

## Redis支持的数据类型：

String:字符串类型；

Hash：键值对类型

Set：无序不可重复集合

Zset:带有分数的集合

List：有序可重复集合

## 怎么理解Redis事务？

Redis无法做到像关系型数据库事务那样严格的ACID属性，特别是Redis官网明确指出了Redis为什么不支持回滚。

为了内部结构简单、运行效率更高，Redis舍弃了事务控制过程中的回滚支持。一个队列中的多个命令除非是在加入队列时发现错误会做到整个事务都不执行，否则所有命令都会执行，哪怕是队列中有的命令执行失败——显然Redis并没有在这里对队列中的多条命令进行回滚处理。Redis认为这些错误都应该在开发过程中被发现，而不是产品上线之后。

配合WATCH命令之后Redis的事务可以实现乐观锁效果：一个队列中的命令在执行时如果检测到碰撞，则放弃自己的操作。

## 高性能:

## 主从复制：

**主从复制是指将一台Redis服务器的数据，复制到其它的Redis服务器。前者称为主节点(master)，后者称为从节点(slave)；数据的复制是单向的，只能由主节点到从节点。默认情况下，每台Redis服务器都是主节点，且一个主节点可以有多个从节点（或没有从节点），但一个从节点只能有一个主节点。**

### 主从复制的作用：

**1、数据冗余**：主从复制实现了数据的热备份，是持久化之外的一种数据冗余方式。

**2、故障恢复**：当主节点出现问题时，可以由从节点提供服务，实现快速的故障恢复；实际上是一种服务的冗余。

**3、负载均衡**：在主从复制的基础上，配合读写分离，可以由主节点提供写服务，由从节点提供读服务（即写Redis数据时应用连接主节点，读Redis数据时应用连接从节点），分担服务器负载；尤其是在写少读多的场景下，通过多个从节点分担读负载，可以大大提高Redis服务器的并发量。

**4、读写分离**：可以用于实现读写分离，主库写、从库读，读写分离不仅可以提高服务器的负载能力，同时可根据需求的变化，改变从库的数量。

**5、高可用基石**：除了上述作用以外，主从复制还是哨兵和集群能够实施的基础，因此**说主从复制是Redis高可用的基础**。

### 主从复制模式存在的问题：

如果现在主节点挂掉了怎么办？**一旦主节点出现故障无法提供服务，这时候需要人工介入手工将从节点调整为主节点，同时应用端还需要修改新的主节点地址，这种故障转移的方式对于很多应用场景是不能容忍的。**正是由于这个问题，Redis提供了Sentinel(哨兵)架构来解决这个问题。

## 哨兵模式：

什么是哨兵模式：

**Redis Sentinel是一个分布式的架构，它本身也是一个独立的 Redis 节点**，**只不过它不存储数据**，**只支持部分命令**，**它能够自动完成故障发现和故障转移，并通知应用方，从而实现高可用**。

哨兵的功能：

**1、监控**（Monitoring）：哨兵会不断地检查主节点和从节点是否运作正常。

**2、自动故障转移**（Automatic failover）：当主节点不能正常工作时，哨兵会开始自动故障转移操作，它会将失效主节点的其中一个从节点升级为新的主节点，并让其他从节点改为复制新的主节点。

**3**、**配置提供者**（Configuration provider）：客户端在初始化时，通过连接哨兵来获得当前Redis服务的主节点地址。

**4**、**通知**（Notification）：哨兵可以将故障转移的结果发送给客户端。

## 集群模式：

### 什么是Redis集群？

Redis Cluster是Redis官方提供的分布式存储解决方案，是在3.0版本后推出的。

Redis集群采用**无中心结构**，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。

集群中的节点分为主节点和从节点：只有主节点负责读写请求和集群信息的维护；从节点只进行主节点数据和状态信息的复制。

### 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安全问题

缓存穿透：

#### 缓存穿透定义：

按照KEY去查询VALUE，当KEY对应的VALUE一定不存在的时候并对KEY并发请求量很大的时候，就会对后端造成很大的压力。（查询一个必然不存在的数据。比如文章表，查询一个不存在的id，每次都会访问DB，如果有人恶意破坏，很可能直接对DB造成影响。）

由于缓存不命中，每次都要查询持久层。从而失去缓存的意义。

**解决方法：**

（1）缓存层缓 存空值。

缓存太多空值，占用更多空间。（优化：给个空值过 期时间）

存储层更新代码了，缓存层还是空值。（优化：后台设置时主动删除空值，并缓存空值进去）

（2）将数据库中所有的查询条件，放布隆过滤器中。当一个查询请求来临的时候，先经过布隆过滤器进行查，如果请求存在这个条件中，那么继续执行，如果不在，直接丢弃。

缓存击穿：

指的是redis中的热点数据热点key失效时，并发大量请求去访问这个失效的key造成了一个透过redis请求到数据库的情况，

### 雪崩：

指的是大部分的key同时失效导致大量请求访问到数据库造成数据库崩塌。

## redis常用的有两种存储结构 rdb和aof :

rdb是基于快照的方式来保存数据的 ，当一定时间内 规定数量的key发生变化 rdb就会进行一次快照来保存数据 每次redis重启的时候都是从快照中读取数据 性能很高 但是数据保存能力很差 如果在某一瞬间 rdb 没有拍快照 但是我保存了很多数据 这个时候服务宕机了 那么在重启以后 redis会丢失宕机时大量的没有拍快照的数据

aof 是基于日志的方式来保存数据的，每次新增一个key 他都会将数据以新增的方式保存在日志中，这样即使服务宕机最多丢失一条数据 保存数据性能很高 ，但是保存大量数据后 redis 重启后 会从aof的日志中 将这些保存的数据从新给一个一个的新增出来 这样的话性能会非常的慢

# Aop做日志记录

当时我在公司负责的日志记录用户方面的一系列操作，经理当时就把这个业务给我负责了，当时考虑到如果用log4j做的话，一些非开发人员就看不懂了，所以当时我们这个小组就想到了使用aop+自定义注解+反射的技术

aop面向切面编程，他是一种思想，就是在不修改原来 的代码的基础上增加一些新的功能

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。

切面 就是要切入的哪个地方

切点 连接点

通知，前置通知，后置通知，环绕通知（joinPoint.proceed()），异常通知，

注解，就是java标注，

自定义注解 ，@inteface

java 注解包含四个元注解，

target 注解用在什么地方，mthod 方法 ,field属性 constroct用于描述构造器 paramater 用于描述参数

documented 注解是否包含在Java文档中

inherted 是否允许子类继承该注解

retation 声明注解的声明周期 runtime项目运行的时候就执行 resources 在编译阶段丢弃 class 在类加载的时候丢弃

这样就可以通过反射去获取注解的信息

在注解中声明了一个sting 类型的value 手工设置日志的信息，之后就可以在controller层的方法上加上注解并对value进行赋值，最后在日志记录的切面类中加上一个注解@component 交给ioc去管理

@aspect声明这是一个切面类，

对于类来说我能知道你的所有属性和方法，对于对象来说我能和调用你的方法

然后在类中加入通知，当时我们使用的是前置通知before

使用joinpoint.signature 获取方法的签名

signature .getMethod 获取方法对象

method.getAnnotion 获取方法上面有没有这个注解

join point.getArgs来获取参数

如果有则使用annotion.value 来获取操作的内容,最后将记录添加到数据库中即可

反射

对于类来说，我能知道你的所有属性和方法，对于对象来说我能得到你的所有属性值和调用你的方法

①、**得到 Class 的三种方式**

//1、通过类对象调用 getClass() 方法来获取

Person p1 = new Person();

Class c1 = p1.getClass();

//2、直接通过 类名.class 的方式得到,该方法最为安全可靠，程序性能更高

Class c2 = Person.class;

//3、**通过 Class 对象的 forName() 静态方法来获取，用的最多，**

// 但可能抛出 ClassNotFoundException 异常

Class c3 = Class.forName("com.ys.reflex.Person");

其中有很多方法，比如说

getModifiers();获取类上的修饰符

getName()：获得类的完整名字。

getPackage(): 获得该类所在的包路径

getDeclaredAnnotations(); 获取该类上所有注解

getFields()：获得类的public类型的属性。

getDeclaredFields()：获得类的所有属性。包括private 声明的和继承类

getMethods()：获得类的public类型的方法。

getDeclaredMethods()：获得类的所有方法。包括private 声明的和继承类

getConstructors()：获得类的public类型的构造方法。

# mybatis $和#

#将传入的数据根据类型进行相应的转换，如果类型不匹配则报错。如果传入的是字符串则自动加上双引号。

$将传入的数据直接显示在sql中。

#方式能够很大程度上防止sql注入，$方式无法防止Sql注入，所以一般能用#就别用$.

# 基于token的登录注册：

在坐登录注册的业务时，我决定用redis作为注册时发送验证码的缓存，在业务中首先是验证手机号格式

从redis缓存获取用户输入的手机号是否已经发送过验证码，验证手机号是不是已经注册过，防止手机号重复，然后用邮箱的方式给用户发送验证码，在redis的缓存中设置验证码的存储时间为一分钟，

当用户输入验证码传到后台后，从redis获取验证码是否一致，如果一致则为注册成功，当用户点击注册按钮时，发送到后台的用户名则判断是否唯一或者不可以是空值，密码使用MD5加密方式可以保证一定的安全度，在用户注册成功之后会发送注册成功的邮件到用户的邮箱使用了多线程可以，提高了用户体验度，

用户登录时后台根据用户名获取用户数据，判断用户的密码是否正确，用JWT令牌生成token代表用户的身份信息存储到sessionStore中，因为JWT是基于json的一种公开规范，非常方便解在用户退出登录时删除存储的token信息的一套登录注册业务；

# 权限管理:

简单来说就是每个人根据每个人的权限看到的内容是不同的

具体实现是 创建五张表 表结构 表的关系 表如何查询的 然后怎么展示 来设置的

五张表有用户表 角色表 权限表 角色权限表 用户角色表

表结构就是

用户表 账号 密码

角色表 id 和角色名称

权限表 id 和权限名称 和地址

角色权限表 角色id 和权限id

用户角色表 用户id 和角色id

查询是 用户登录以后 发送请求时把 用户的id传到后台 然后根据id查找用户的角色 在根据角色查找用户的权限 然后返回查询到的数据 权限信息以动态拼接的方式拼接到导航栏上面

# POI导出步骤

做poi导出，我们同样需要使用反射和自定义注解，但不同的是，这时候自定义注解添加的位置成了属性，作用是为了得到需要导出的字段以及相应的数据。

最初，我只是做了一个简单的Excel表，使用XSSFWorkbook，生成sheet页、行、列，随之放入数据，最后写入Excel中，关流即可。这样写并没有什么问题，开发和测试都一切顺利。但是到了上线阶段数据量上去之后就不适用了。

后来我想到了生成多个excel文件多个sheet页并将其做成一个压缩包，这样再多的数据量也不会有事了。我先是尝试了使用流的方式进行操作。首先创建了ServletOutputStrea流和ZipOutputStream流，在此，我们需要确定每个sheet页数据的条数和sheet页的个数，以此来确定excel文件的个数，然后使用HSSFWorkbook通过循环，生成sheet页、行、列，随后放入数据，生成excel，依次写入zip压缩流下载完成。

再之后，需求发生了变化，需要做成多线程的poi导出，（此时先介绍多线程）这时，我之前做的那种思路的zip压缩流就不能再用了，因为它无法加到多线程中去。我换了一个思路，先在服务器上创建了一个临时文件夹，通过多线程去生成多个的excel文件，将其放到临时文件夹中。这些都完成之后，通过FileInputStream流依次读取临时文件夹中的数据，再通过ZipEntry来进行写入，最后下载zip压缩包，关流。至此，多线程poi导出便写完了。