# B简历:

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

## （1）基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| 基本情况 | | | |
| 姓名 | **李欣强** | 性别 | 男 |
| 出生日期 | 1995-04-16 | 属相 | 猪 |
| 现住地 | 丰台区 东营里 2号院 |  |  |
| 大学情况 | | | |
| 学校 | 商丘工学院 | 专业 | 计算机科学与技术 |
| 入学时间 | 2013-9 | 毕业时间 | 2017-7 |
| 专业课 | 高级语言程序设计、计算机导论、汇编语言程序设计、数理与数字逻辑设计、电子技术基础、数据结构与算法、计算机组成原理、软件工程、形式语言与自动机、数据库系统、操作系统、计算机网络、算法设计与分析、编译原理、JAVA语言程序设计等 | | |
| 学历 | **本科** | 学位 | **学士学位** |
| 学校地址 | 河南省商丘市梁园区睢阳大道中段236号 | 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： | **在火车站坐69路到商丘工学院站下车** | | |
| 上小学日期： | **2001年9月 6岁 6年** | | |
| 上初中日期： | **2007年9月12岁 3年** | | |
| 上高中日期： | **2010 9月15岁 3年** | | |
| 上大学日期： | **2013年 9月 18岁** | | |
| 毕业时间： | **2017年 7月 22岁** | | |
| 在校学生： | **5000-6000左右** | | |
| 有教师： | **不太清楚** | | |

|  |  |
| --- | --- |
| 第一家公司 ： | 北京华宇思创科技有限公司 简称（北京华宇思创） |
| **地址：** | 北京昌平区国风美唐1302 （公司地址自己记详细，给人事说说简单点） |
| **入职时间：** | **2016年9月** |
| **离职时间：** | **2017年9月** |
| **公司性质：** | **民营** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **郑洋** |
| **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：** | 天通苑地铁站坐5号线到立水桥换十三号到霍营下 然后在骑个自行车一会就到了 |
|  |  |

|  |  |
| --- | --- |
| **第二家公司情况：** | 北京北鑫辰科技发展有限公司 简称（北鑫辰科技） |
| **公司地址**： | 北京市 海淀区 长远天地大厦  A1座801室 （给人事说简短的就行） |
| **公司规模：** | **60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、咨询、服务、转让；信息咨询（不含中介服务）；销售自行开发后的产品 |
| **去公司路线：** | 天通苑地铁站坐5号线到惠新西街站换乘10号线 到苏州街出站 不行300来米 |
| **入职时间：** | **2017年9月** |
| **离职时间：** | **2021年4月** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **刘曌** |
| **税前：** | **15K** |
| **税后：** | **14K** |
| **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：** | **16K** |
| **薪资的构成:** | **基本工资(6K)+绩效工资(9K)** |

|  |
| --- |
| 四年涨薪： **2016工作的时候是5.5K，税后大概5.4K左右，**  **转正6K左右**  **2017上半年涨了1K左右 ，总7K左右。**  **2017年9月跳槽涨了3K左右 总10K**  **2018年9月涨1K总 11K左右**  **2019年9月工作一年涨了2K左右**  **2020年9月份左右涨了2K左右**  **2021年4月离职时工资为15K** |

**面试常见问题：**

|  |
| --- |
| **你为什么从上家公司离职？** |
| **想换个工作环境，不断的提高充实自己** |
| **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。** |
| |  | | --- | | **三个词形容自己**： 交流表达能力强，团队协作能力强，抗压自学能力强 (**可以综合自己自学的知识表达**) | | **你的优缺点是啥？** |   **优点:**  **抗压能力强（有一次项目就要上线了，连着加班了一周，每天都到夜里很晚，但我最后也是成功完成任务。），**  **乐于分享，**  **团队协作能力强**  **缺点: 我这个人说话比较直，你像在和团队成员探讨问题的时候，容易得罪人,**  **不过现在已经改进的差不多了** |
| **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 16K 上家工资15K 税后14K左右 扣了1K左右的税 |
| 你的五年规划是啥？ |
| **前2,3年 继续加强自己的技术功底，**  **然后朝着项目经理(技术经理,产品经理)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： 13薪  **车补**  220左右  **项目奖金**  8k-10k，做完项目，收完尾款的时候发。  **年终奖**  就是1个月的工资，  最后一个月和工资一块儿发/年后发  **电话补助**  260  **饭/餐补**  **15、**30/天  **电脑补**  310 |

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、**

**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！**

### 主动说

自我介绍

你好，我叫李欣强，今天来应聘咱们公司的java 工程师

介绍一下springboot：

我们在项目中最常使用的框架就是springboot，它的主要特点是抛弃了复杂的xml文件配置，大量的使用注解的方式，提升了项目的启动速度，因为xml文件在启动的时候需要进行解析再生效，同时它可以通过maven下载依赖版本兼容的jar包。它的另一个特性就是开箱即用，比如我们引用了springboot-web 这个jar，自带springmvc和tomcat容器，如果导入了mysql的数据库连接包，就必须要配置数据连接。【这个可以通过注解参数忽略掉】

Springboot框架主要是通过一个类里面的main方法启动的，方法上需要加上@SpringbootApplication的注解

然后@SpringbootApplication这个注解是一个组合的注解，它是由三个注解组成的

@EnableAutoConfiguratio

@ComponentScan

@Configuration

然后

@EnableAutoConfiguration :启用Spring的自动配置机制也就是自动扫描其他注解让其生效的

@ComponentScan 扫描被@Component (@Service,@Controller)注解的 bean，注解默认会扫描该类所在的包下所有的类

@Configuration：允许在 Spring 上下文中注册额外的 bean 或导入其他配置类。

然后springboot+springcloud框架，我想首先说一下我使用springboot的感受，搭建项目快，配置简单只需要一个properties或者Yml完成配置，它跟我们之前使用xml的那种框架搭建方式会简化很多。另外它通过maven进行配置一些开箱即用的资源比如Mybatis,redis,mysql,web,start。同时它还内嵌了tomcat,通过一个mian和@SpringBootApplaction注解进行启动，部署的话只需要将项目打成jar,使用java -jar 包名就能启动了。

基于这些优点，我们拿它来做微服务框架的接口开发特别适合，首先将我们的项目按照功能模块的划分拆分成一个个的微服务，比如我在做电商项目是，将商品管理服务，订单服务，库存服务，用户服务，资源统一管理服务，购物车服务等块拆分成一个个的子模块，使用springboot框架快速搭建起来。那么就会出现一个问题，这么多的服务到底如何进行统一管理，相互之间如何调用的问题就凸显出来了？

其实springboot框架在设计之初就考虑到这些问题，那就是springcloud，它是基于springboot的微服务框架的实现，它提供了丰富的组件化开发

下面我就给你介绍一下我常用到的组件。

首先是eurake注册中心，主要是来注册保存我们的服务的地址列表，为消费者提供订阅服务的功能，它在搭建时需要引用eureka-server，然后在配置文件配上server端口号，然后配置register-with-eureka不注册自己，fetch-registry=false不去检索服务，然后在启动类上加上@EnableEurekaServer注解开启服务，这样注册中心就搭建完成了。注册中心主要功能是服务的注册与发现，生产者和消费者都需要将服务注册到注册中心上，注册中心会保存这些地址。并且服务会主动向注册中心发送续约请求，在一段时间不续约后就认为该服务已经挂掉了，就需要从注册中的地址列表中剔除。这个检查剔除服务的时间间隔可以在注册中心上配置。

# Eureka工作原理

服务启动向Eureka注册中心注册服务，注册中心将服务注册信息同步给其他注册中心，当服务消费者需要调用服务提供者，则向服务注册中心获取服务提供者地址，然后会将获取的地址缓存在本地，下次调用的则直接从缓存中取，当服务注册中心检测到服务提供者宕机、网络不通等服务不可用时，则在注册中心将服务置为down状态，并向订阅者发布当前服务提供者状态，订阅过的服务消费者更新本地缓存。

服务提供者启动后周期性（30秒）向服务注册中心发送心跳，以证明当前服务可用，服务中心在一定时间（默认90秒）未收到心跳，则认为该服务宕机，注销该实例。

我们这么多的微服务之间还存在问题，比如a服务需要调b服务，为了解决服务与服务之间调用的问题，我们使用到了fegin

使用fegin,它解决了让我们调用远程接口就跟调用本地方法一样，

使用流程是在pom.xml文件中引入 openfeign。因为fegin调用是基于接口的服务调用，同时它集成了ribbon,还能完成负载均衡和重试，因此我们在使用是要将我们发布的restful风格接口抽象化成一个方法，然后在该接口类上加上@FeignClient，并配置服务名，同时在启动类上加上@EnableFeignClients让feign注解生效，在调用接口是需要注意的是get请求传参时必须要加上@RequestParam注解，如果post请求传递的参数是一个实体对象还要加上@RequestBody，使用feign我觉他的好处就是像调用方法一样调用接口用起来比较简单。

# Feign的工作原理

* 主程序入口添加了@EnableFeignClients注解开启对FeignClient扫描加载处理。根据Feign Client的开发规范，定义接口并加@FeignClient注解。
* 当程序启动时，会进行包扫描，扫描所有@FeignClients的注解的类，并且将这些信息注入Spring IOC容器中，当定义的的Feign接口中的方法被调用时，通过JDK的代理方式，来生成具体的RequestTemplate.当生成代理时，Feign会为每个接口方法创建一个RequestTemplate。当生成代理时，Feign会为每个接口方法创建一个RequestTemplate对象，该对象封装HTTP请求需要的全部信息，如请求参数名，请求方法等信息都是在这个过程中确定的。
* 然后RequestTemplate生成Request,然后把Request交给Client去处理，这里指的是Client可以是JDK原生的URLConnection,Apache的HttpClient,也可以是OKhttp，最后Client被封装到LoadBalanceClient类，这个类结合Ribbon负载均衡发送服务器之间的调用。

ribbon

Ribbon说白了就是一个客户端负载均衡的工具，当然我们在之前的项目里没有单独使用过Ribbon，因为我们使用的Feign已经内置了Ribbon。Ribbon内置了很多负载均衡策略，比如说轮询啊、随机啊、过滤掉不可用的服务啊、根据响应时间加权啊、轮询重试啊等，大概我就记得这些。Ribbon默认使用的负载均衡策略是轮询。当然如果说Ribbon自带的负载均衡策略满足不了你的需求的话，你也可以自定义Ribbon的轮询策略。其实就是新建一个类去继承Ribbon提供的一个什么Abstract...Rule什么的[故意说的不清楚]，然后重写抽象方法，在里面实现你的负载均衡策略就可以了。

Zuul在整个微服务系统中充当网关的角色，实现路由转发和过滤的功能能，是客户端与服务器之间的中间层。因为这个外部客户端的请求都是通过Zuul网关路由到具体的微服务，所以为了保证微服务的安全，我们就在Zuul中自定义了过滤器，对所有微服务的安全进行统一的处理，还有，因为涉及到前后端分离，前端项目访问后端微服务涉及到跨域问题，所以我们在Zuul中 也自定义了关于跨域的过滤器，进行统一处理。

还有就是每个微服务都会需要鉴权、限流、权限校验等逻辑，如果每个业务都各自的处理自己的，自己造轮子实现一遍，会很头疼，完全可以抽出来，放到一个统一的地方去处理。

具体是这么做的，首先我先定义一个继承于ZuulFilter的类，重写里面的filterType()方法filterOrder()方法，shouldFilter()方法，还有一个最核心的用来写具体业务逻辑的run()方法。其中filterType方法的作用是用来返回一个字符串，指明该过滤器的类型，经常用到的有pre类型，说白了就是在请求被发送到微服务之前调用；我们的微服务安全认证以及跨域这块都是用的pre类型的过滤器，这样对非法请求，就可以在发送到具体的微服务之前拒绝它；

还有post类型，说白了就是微服务执行完后再执行该过滤器，filterOrder()方法返回一个int类型的值，用来指明该过滤器的执行顺序，数字越小表示优先级越高，就越先执行，shouldFilter()方法返回一个boolean值，用来指明该过滤器是否执行，true表示执行,false表示不执行

我们上面说了run方法主要是来写业务逻辑的

我们在run方法里面主要进行了验证token，

如果想要将zuul过滤器中的数据传递给后端微服务中使用，则需要通过currentContext.setRequestQuery Params方法来进行，而后端微服务中就可以通过request.getParameter来获取值，由此可以通过这个来判断用户是否进行了登录

我们的登录是使用jwt做的登录，等下可以具体在说一下

Hystrix总结

微服务框架搭建好了在使用的过程中并不是一定没有问题的，有可能会出现服务的雪崩问题，说白了，服务的雪崩就是，客户端访问a服务，a服务需要调用b服务，b服务需要调用c服务，如果b服务或者c服务不能及时响应，a服务在堵塞着，容器的线程资源会消耗完毕，这个时候就会导致服务的雪崩。就跟我们看的丧尸片一样，本来是一个人感染了，没有及时控制处理，导致一传十，十传百 ，最后导致大面积的感染

我们可以使用springCloud 中的hystrix来解决这个问题

具体的原理是这样的，默认情况下在10s内，每当20个请求中，有50%失败时，就会触发熔断，导致hystrix从关闭状态切换到打开状态，这样就不需要每次都去调用远程微服务而是直接走调用者本地的降级方法。5秒后hystrix会进入到半打开状态，这时如果有新的请求再过来hystrix就会尝试再次进行远程微服务调用，如果调用成功就会进入到关闭状态，如果不成功就会再次进入到打开状态，继续进行熔断。这个就是hystrix的自动检测并修复的能力。

说白了使用hystrix，我们可以让异常的请求，快速失败，结束。就可以避免服务雪崩

基于token的登陆

用户登录流程：

用户在前端输入用户名和密码，后台根据用户名查询数据库有没有对应的用户，有对应的用户说明已经注册过了，验证通过后，我们使用jwt给当前用户进行了加密，并且设置了超时时间，把加密后生成的token传到前端，前端通过返回的状态码来判断用户是否登录成功，登录成功，把token拿出来放入到了sessionStorage中。

我们有的页面如果用户没登陆肯定是不能让他访问的，所以说又使用了vue的路由守卫，配置了全局守卫，每次进行路由跳转页面的时候都会先进入到路由守卫，看sessionStorage里是否有数据，如果有的话说明用户已经登录正常访问，如果没有数据用户没登陆，返回到登陆页面

登录成功后，用户发送请求，服务端会先进行拦截请求，判断每次是否携带了token，没有携带token，说明用户没登陆发送的非法请求，如果有token，将token进行解密，如果不为空，放开请求

因为我们前端每次发送请求的话，每个请求都加token的话是很麻烦的一件事情，所以我们再vue里配置了请求拦截器，每次请求先进入拦截器，把token给请求携带上.

**定时任务**

我们公司之前做的项目，在注册的时候要求了用户需要填上自己的生日，然后我们就出了一个业务，找到生日的用户给他发送邮件祝福生日快乐

我们当时项目的架构是 springboot+springcloud

定时器我们用的是@ Scheduled

启动类上加注解

@EnableScheduling //启用调度器

创建一个类，类上加@Compont 交给ioc管理

创建方法，方法上加@Scheduled

当时用的是多线程进行发送邮件，考虑到用户较多，每次发送邮件需要创建一个线程，使用完毕在销毁。如果有很多用户在今天都需要发送邮件，那我们就需要频繁的创建销毁线程，这样会很消耗我们的系统资源，然后我们就使用了线程池，

线程池就类似于一个容器，事先先在线程池中创建一些可执行的线程，当我们有任务进来，直接从线程池中拿取空闲的线程使用，执行完任务在还回来，实现了线程的复用，降低了系统资源消耗

线程池工作原理，当有任务进来的话，先去看核心的线程池是不是已经满了，没满的话直接就创建线程执行任务，满了的话，会查看队列是否已经满了，队列没满把任务放入队列等待有空闲的线程来执行，队列满了的话，会去查看当前的线程是否已经达到了核心线程最大线程数，如果达到了，直接就会拒绝执行任务

我们用的是FixedThreadPool 线程池，FixedThreadPool线程池它会创建固定的工作线程数量，并且它的核心线程数和最大线程数是一样的，任务过多会在队列中阻塞，一个线程因为异常执行结束，线程池会补充一个新的线程。

然后发送邮件，先把所有的用户查询到，拿出用户注册时输入的生日，和当前的时间进行比较，相等了的话就会发送邮件。

当时这个做完测试是没问题的，然后我们服务进行集群后发现，存在重复发送邮件的问题，为了解决这个问题我们当时决定借用redis的getAndset进行解决，getAndset 的特性是，设置一个新值并且会返回之前的旧值，redis操作都是原子性的，当两个请求进来，两个请求只有一个会先去执行getandset，第一个执行返回的是null，我们给key设置一个过期时间

根据这个我们可以进行判断，如果getandset 返回不为空，说明已经发过邮件了，不需要在进行发送了，这样就可以解决重复发送的问题

Aop日志记录

当时在公司负责日志记录用户的一系列操作，经理当时就把这个业务给我负责了，

当时考虑是用log4j来实现，但是这样一些非开发的人员就看不懂

然后我们当时商量讨论了一下决定使用，aop+自定义的注解+反射的技术来实现，把日志的信息新建了一张表存入到了一张表中，

Aop是面向切面编程，它是一种思想，使用aop的这种思想，我们可以在不修改原本的代码，给我们的代码增加新的功能

Aop可以适合很多场景，比如事务管理，日志记录，性能的统计

Aop还有两种代理方式，默认是通过jdk的动态代理，还有cglib代理

Jdk的代理需要接口的支持，没有接口的话通过cgliba代理

当时我们是这样做的

导入相关的依赖包，创建一个类类上加注解@Aspect 声明这是一个切面类

在加注解@Component 将当前类叫给spring的容器去管理

在创建一个自定义注解的类，

自定义注解中包含了四个元注解

@Target target 声明注解用在什么地方，分别有 mthod 方法 ,field属性 constroct用于描述构造器 paramater 用于描述参数

@Retention 这个是声明我们注解的声明周期

@Documented 注解是否将包含在JavaDoc中

@Inherited 是否允许子类继承该注解

在切面类中我们需要加 aop通知，当时我们用的是前置通知，切的是controller层，到时候我们可以把自定义注解加入到controller里的方法上，看哪个方法我们需要进行日志记录

我们可以通过joinpoint参数，先获取到了方法的签名，然后接着就可以获取到方法的对像，

然后获取到annotation 自定注解，

通过joinpoint.getArgs() 获取参数

把需要记录日志的进行新增到我们的日志表中，通过annotation.value 获取自定注解中的数据

权限管理

首先我们的权限管理是RBAC模型的，意思也就是基于角色的权限控制，有别于传统模型中的直接把权限赋予账号，RBAC模型中增加了“角色”的概念，把权限赋予角色，再将角色赋予账号。提高了账号管理效率，降低了出错的概率。用户与角色是一多关系，角色与权限是一对多关系，所以说我们权限管理在数据库设计过程中涉及到了**五张表。**这五张表分别是用户表、角色表、权限表、用户角色关系表、角色权限关系表，其中用户表和角色表是一对多的关系，所以我们引入了一张用户角色关系表配置用户和角色的一对多关系。而角色表和权限表也是一对多的关系，所以我们引入了一张角色权限关系表来配置角色和权限的一对多关系。这样就实现权限的管理，给账号赋权后，每个用户看到的都不一样

意思就是，我们可以把角色分为超级管理员，普通管理员，普通用户

比如我们有增加，查看，删除，修改

我们给超级管理员所有的权限，到时候给用户权限的话我们直接就可以把当前的角色给用户，不用直接给用户赋权限，给用户赋权限的话我们需要赋四次权限，也比较繁琐

生成订单

添加完购物车后，这个时候用户进入到购物车的话，点击进行支付的话需要进行生成订单了，

在前台把需要生成订单的信息传到后台，把购物车的数据传到后台

然后在后台处理逻辑

订单这个模块我在之前的项目中也做过，这个模块相对比较复杂，考虑的事情也比较多。像表设计、订单接口幂等性的问题，商品超卖的问题，包括订单也都是用户登录后才进行的操作等。

接口幂等性问题，当时我们借助redis来进行

------------------------------------上面是业务-------------------------------

## Java的反射

Java的反射机制是在运行状态中对于任意一个类，都能知道这个类所有的属性和方法

对于任意的一个对象，都能够调用它的任意一个方法和属性

动态获取的信息和动态的调用对象的方法功能是java的反射机制

Java反射得到class对象有三种方式

类名.class 接收返回值 返回到类对象

第二种通过class.ForName(); 通过类的全路径获取到

第三种的话是 new对象 通过对象名 get获取class对象

反射中的对象

Class对象 类对象

Filed 属性 属性对象

Method 方法 方法对象

Constructor 构造方法 构造方法对象

Annotation 注解对象

## Java的自定义注解

Java的注解是java里的一种标注是jdk5.0引入的一种注释机制

有四个元注解

@Target 注解用于什么地方

@Retention 声明注解的声明周期

@Documented 注解是否将包含在javaDoc中

@Inherited 是否允许子类继承该注解

## 多线程

进程是程序的一次动态执行过程，它需要经历从代码加载，代码执行到执行完毕的一个完整的过程，这个过程也是进程本身从产生，发展到最终消亡的过程。多进程操作系统能同时达运行多个进程（程序），由于 CPU 具备分时机制，所以每个进程都能循环获得自己的CPU 时间片。由于 CPU 执行速度非常快，使得所有程序好像是在同时运行一样。

多线程是实现并发机制的一种有效手段。进程和线程一样，都是实现并发的一个基本单位。线程是比进程更小的执行单位，线程是进程的基础之上进行进一步的划分。所谓多线程是指一个进程在执行过程中可以产生多个更小的程序单元，这些更小的单元称为线程，这些线程可以同时存在，同时运行，一个进程可能包含多个同时执行的线程。进程与线程的区别如图所示：

Java中使用多线程两种方式

实现Runnable接口

继承thread类

死锁造成的条件

现在有两个线程 然后我创建了两个对象

线程1进来我们判断条件执行完obj1的对象然后线程资源耗尽没有释放锁

然后cpu调用线程2执行线程 线程2进入判断条件执行obj2 执行完之后去执行obj1的时候发现obj1正在被锁这然后没办法执行，堵塞在这，然后线程1去执行obj2的时候发现被锁了也无法执行，堵塞在这里造成了死锁