B简历：

### 1基本信息

|  |  |  |  |
| --- | --- | --- | --- |
| **基本情况** | | | |
| **姓名** | 刘伟华 | 性别 | 男 |
| **出生日期** | 1996-03-09 | 属相 | 鼠 |
| **现住地** |  |  |  |
| **大学情况** | | | |
| **学校** | 安徽信息工程学院 | 专业 | 计算机科学与技术 |
| **入学时间** | 2014-9 | 毕业时间 | 2018-7 |
| **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" \t "https://zhidao.baidu.com/question/_blank)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" \t "https://zhidao.baidu.com/question/_blank)、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** | **安徽省芜湖市文津路8号** | 校长 | **吴敏** |
| **去学校的方式：** | **焦作西站坐车到芜湖站下车 （焦作西站乘坐G3135次列车在芜湖站下车）** | | |
| **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 ： | 安徽共生物流科技有限公司 |
| **地址：** | 芜湖镜湖区绿都新会C座（15楼）（只说大厦名称） |
| **入职时间：** | **2017年4月** |
| **离职时间：** | **2019年6月** |
| **公司性质：** | **民营** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **石宇文** |
| **薪资方面：** | **刚刚毕业转正以后工资是6K，过完年涨到7K** |
| **路线：** | **从校门口的公交车做到报业集团站下车在做到黄山东路2号站 步行一会就到了** |

|  |  |
| --- | --- |
| **第二家公司情况：** | 北京悦视科技有限责任公司(只说公司简称) |
| **公司地址**： | 北京海淀区绿地中央广场 （北京海淀区绿地中央广场12号楼8005） |
| **公司规模：** | **60人左右，我们项目组10人，一个项目经理，2名前端，1个测试，2个运维，其他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：** | 从我住的地方（昌平区风光雅苑东区） 步行 平西府地铁站 8号线 朱辛庄下车 换乘昌平线 生命科学园下车 做公交（543路）做到三星庄南口 下车 步行个800米左右就到了绿地中央广场 |
| **入职时间：** | **2019年6月** |
| **离职时间：** | **2021年3月初** |
| **职位：** | **Java工程师** |
| **项目经理名字：** | **于风** |
| **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：** | **15K** |
| **薪资的构成:** | **岗位工资(5K)+绩效工资(9K)** |

|  |
| --- |
| 四年涨薪：  **2017年工作的时候是5K，税后还是5K，**  **转正6K左右**  **2018上半年涨了1K左右 ，总7K左右**  **2018年11月涨了2k,总的9k左右**  **2019年6月跳槽涨了1K左右 总10K**  **2020年12月涨2K总 12K左右**  **2021年3月离职时工资为14K** |

**面试常见问题：**

|  |
| --- |
| **你为什么从上家公司离职？** |
| **上家公司待的时间长了，公司比较照顾老员工，让后把我分配去维护之前的项目，让后工作太轻松，环境太安逸，**  **然后就换个公司，想换个新的环境挑战一下自己！** |
| **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，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:**  **抗压自学能力强（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。），**  **喜欢钻研技术，**  **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: 研究技术时候，想把这个技术搞的很清楚，结果才开始的时候影响了项目开发的进度，我也意识到这个**  **问题，所以后来对于我比较感兴趣的技术，我会利用自己的业余时间去钻研它，在工作时以项目任务为主。** |
| **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** |
| **加班在这个行业里面比较普遍，这个没问题但我平时也会去提高自己的工作效率，避免一些不必要的加班** |
| **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？ |
| **近期的话我就是想先让自己的技术在得到以进一步的提升，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： 13薪  **车补**  300左右  **项目奖金**  7k-12k，做完项目，收完尾款的时候。  **年终奖**  这个就是1个月的工资，  一般是过完年之后发  **电话补助**  200  **饭/餐补**  30/天  **电脑补**  400 |

**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！**

### 其它问题

### 谈谈你对软件行业发展前景的理解：

### 现在的生活越来越离不开网络和便捷的生活方式，软件的行业就是顺应了这一特点，通过开发的软件简化了人们一些生活杂度，所以在不久的将来必将在以后成为科技的前端和人们生活的密切相关。

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，也可以折现，当时感觉没啥必要也就没上。折现后也就1000多块钱

你的优缺点是啥？

优点: 抗压能力强 [ 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 ]

缺点: 研究技术时候，想把这个技术搞的很清楚，结果才开始的时候影响了项目开发的进度，我也意识到这个问题，所以后来对于我比较感兴趣的技术，我会利用自己的业余时间去钻研它，在工作时以项目任务为主。

你的五年规划是啥？

近期继续加强自己的技术功底，尽快熟悉公司的业务，然后朝着技术经理方面发展

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。

你们公司有多少人？你们公司有哪几个部门？

八十人左右

技术部, 运维部, 财务部, 人事部, 销售部

你们团队几个人?都分别干什么?

10个人

1个项目经理, 1个测试, 2个前端, 4个开发，2个运维

你能接受加班吗

加班在这个行业里面比较普遍，这个没问题但我平时也会去提高自己的工作效率，避免一些不必要的加班

你对我们呢公司还有什么要了解的

1.我想问下咱们公司在技术这块会定期举行技术交流会吗

2.我想问下咱们公司的关于福利这块都有哪些

3.我想问下咱们公司的晋升制度

你能接受出差吗

能

你们公司还有什么福利

多少薪： 13薪

交通补 200

电话补助 100

项目奖金 8000-10000，做完项目，收完尾款的时候。

年终奖 这个就是1的月的工资，

一般是过完年之后发/等到过年的时候和当月工资一块发

最后一个月发四个月工资

你最快什么时候能来上班

3-4天的缓冲期

能在学信网查到吗

不能

我当时上的是民办大学, 能在我们学校查到/能在民教网查到

二 、Spring的理解：

首先他给我们编程提升了很大的一个效率，它整合了我们市场上绝大部分优良的框架，咱们都知道Spring的核心就是IOC和AOP这两大特性，IOC是基于工厂设计模式，我的理

解就是说原来咱们想要创建一个类的对象，得自己通过new的方式进行，而现在可以通过在Spring的配置文件中写上一段<ben id=”” class=””>的配置，从而让创建对象实例的

过程交给Spring容器来完成，而我们在程序中可以直接获取创建好的对象实例进行使用。这就是IOC的控制反转。

AOP是基于代理设计模式，代理分为动态代理和静态代理，在项目中默认使用的是基于

jdk的动态代理，它需要接口的支持，这就是在写service业务逻辑层的时候通常先写接口，

再写实现类的原因，如果没有接口只有类，这个时候可以使用Cglib这个动态代理来实现。

Aop说白了就是面向切面编程，就是把一些非核心的业务逻辑提取出来形成一个切面，从而

让咱们程序员在编写代码时候，只用关注核心业务逻辑的处理，至于这些非核心业务逻辑统

一在切面里面处理。像日志记录，性能统计，事务处理，安全控制，这些都可以通过AOP的方式进行统一的处理。

切面说的简单点就是在 指定类的 指定方法的 前后 执行特定的 横切业务逻辑。

切点就是为了定位指定类的指定方法，方位包括 前置通知，环绕通知，抛出异常通

知，返回后通知这些； 横切业务逻辑就是一些公共的非核心业务代码。

Aop的切面在项目中用的还是比较多的，咱们都知道事务管理就是通过Aop切面的方式

来实现的，切面的好处就是把分散在代码中的重复代码提取出来进行统一的维护和控制，我

在项目中就使用aop完成日志的统一处理。原来是把日志记录的相关代码分散到各个控制层

的相关方法中，但这就会导致程序员在开发时候不能将经理集中到业务逻辑的处理上，还得

考虑记录日志，工作效率就大打折扣，我在项目中负责过日志管理模块。

日志管理模块：

在项目中我做过后台管理系统的日志管理模块，日志管理模块的作用说白了就是记录用户的操作，这样就知道谁在什么时候干了什么事情。之前做其他项目的时候，日志处理模块通常都是在控制层结合LOG4J进行日志的控制台打印以及日志文件的存储，除此之外还会把日志信息插入到mysql数据库中存储起来，方便查看。但这样做会有个问题，就是需要在每个Controller类的每个方法中都写上相关的日志记录代码，这样就会出现大量的代码重复，以后维护起来也特别麻烦。

所以说后面在做日志管理模块的时候，我就考虑到可以使用AOP做统一日志的处理，这样就可以让我们的程序员在工作的时候把精力花在核心业务代码的处理上。

具体在做的时候我是这样写的，首先写一个日志的切面类，这个切面类说白了就是一个 普通的java类，后面会通过配置文件的配置让他具备切面类的功能。在这个普通的java类中 我会自定义一个横切逻辑，就是一个普通方法，但这个方法中需要特别注意几点，因为当时 在项目中我使用的是环绕通知，所以proceedingJoinPrint类型的参数。

具体在写方法的时候有几个特别关键的地方，首先因为要获取当前登录的用户，用户的信息是存储在session中的，我当时采用的解决方案是使用ThreadLocal+Filter来完成在一个普通的java类中获取当前请求的request对象，进而获取存储在session中的用户信息。

ThreadLocal可以把它理解成一个Map，但它特殊地方就是它用当前线程充当key，所以在使用的时候，存储信息用set（value）就行了，之所以没有写key，就是因为当前访问的新城就是默认的key，同理取数据用get（）；当时我封装了一个工具类，工具类中有个setRequest方法，就是将request对象存储到ThreadLocal中，同样还有个getRequest方法，就是获取当前线程对应的request对象。

之后我会在自定义的Filter中的doFilter方法中，调用工具类的setRequest方法，将当前请求存储到ThreadLocal中，当然还得再web.xml中配置fifilter使其生效。接下来我会在日志切面类中通过调用工具类的getRequest方法来获取request进而通过fetSession来获取session，这样就可以取到存在session中的相关用户信息了。

获取到用户信息之后就要记录用户做了什么事情，在这块我们当时的项目是这么规定格式的，就是要记录用户执行了哪个类的哪个方法，并且要把执行这个方法时候对应的参数信息也给获取到，比如用户添加了一个商品，那 记录的信息就应该调用了ProductController 类的addProduct方法，并且也要获取到添加的商品信息参数，这样才能看的更明白。

获取类名和方法名这块是通过反射机制，调用ProceedingJoinPoint的相关方法获取的，获取参数信息是通过request.getParameterMap（）之后对其进行循环遍历，这样就获取到了提交时候的参数详情在这个切面类中还有一个特别重要的方法就是proceedingJoinProceed（）；它代表的就是实际要执行的 核心业务逻辑，它的返回值就是实际执行方法的返回值，比如刚才说的ProductController类中的addProduct方法，这个proceed（）代表的就是addProduct方法，而他的返回值就是控制层中addProduct方法的返回值。之所以 他的返回值类型为Object就是因为不同方法的返回值不一样，但他们都属于Object对象。

接着我会把proceed方法进行try..cach..捕获，然后在这个方法执行完后记录操作成功的日志并且信息插入数据库，在catch捕获异常的时候记录错误日志，同时会把异常信息记录数据库。

最后需要在spring的配置文件中配置aop：confifig以及配置切点表达式来对控制层中的增删改方法进行拦截，这里就用到了切点表达式中特殊符号的双竖杠||。

在上交完任务后，我们经理给我说这个东西，做的整体上还不错，但是有个问题，就是日志虽然记录了操作哪个 类的哪个方法，程序员可以读懂，但业务员根本看不明白，不够人性化，让我的日志记录再改进下。

我通过和我们团队的人讨论，最终决定通过自定义注解来完成这个改进。

在写自定义日志注解时候，通过@Tarfet设置为Method指明该注解只能用在方法上面，通过将@Restention设置为RunƟme指明将注解保留至运行时。这样就可以通过反射去获取注解信息。

在注解中声明了一个String类型的value来让程序员手工设置日志的信息，之所以采用value，是因为value这个字段有特殊的含义，它可以在使用自定义注解给日志信息赋值的时候省略不写，用起来更加方便。之后就可以在Controller中的方法上加入自定义注解并且对value进行日志信息的赋值，如@Log（“增加商品”）。

最后在AOP的日志切面类中通过获取方法签名得到Method，通过Method的

jsAnntationPresent判断该方法上面是否加入了自定义日志注解，如果是则再通过Method的getAnnotaƟon来获取自定义的日志注解，最后再通过.value（）方法获取自定义注解中日志信息的值，这样在记录日志的时候就可以显示更加人性化的信息。

三 、项目的生命周期

1. 需求分析

2. 概要设计

3. 详细设计（用例图，流程图，类图）

4. 数据库设计（powerdesigner）

5. 代码开发（编写）

6. 单元测试（junit 白盒测试）（开发人员）

7. 集成测试（黑盒测试，loadrunner）（编写测试脚本）（高级测试）

8. 上线试运行（用户自己体验）

9. 压力测试（loadrunner）

10. 正式上线

11. 维护

4、springboot运行原理

我们从springboot项目的启动类中可以看到最核心的两行代码，

@SpringBootApplication和SpringApplication.run方法。

在@SpringBootApplication的内部包含了3个注解

@Configurtion

@EnableAutoConfiguration

@ComponenScan

@configuration是基于javaConfifig形式的Spring Ioc容器的配置类，可以把它看成xml配置文件中的beans标签。@ConfifiguraƟon写到类上面，在类中的方法上如果写了@bean注解，那么它的返回值将作为一个bean注册到Spring的Ioc容器，方法名默认作为bean的id。

@ComponentScan这个注解对应xml配置中的context:component‐scan元素，说白了它的作用就是自动扫描并加载符合条件的组件，比如@Component和@Service等或者bean定义，最终将这些bean定义加载到ioc容器中。我们可以通过basePackages来指定@ComponentScan自动扫描的范围，如果不指定，则默认Spring框架实现会从声明@ComponentScan所在类的package进行扫描，这也是SpringBoot的启动类最好是放在root package下的原因。

@EnableAutoConfiguration这个注解是借助@Import的帮助，将所有符合自动配置条件的bean定义加载到Ioc容器中

负载均衡及分布式session

我们项目在最后上线部署的时候，采用的是nginx+3个tomcat进行负载均衡，在我们不进行负载均衡之前，所有的请求都有一台toncat进行处理，这样会使我们的tomcat0所承受的压力增大，而我们进行负载均衡之后，同样数量的请求经过nginx将其分发给多台toncat进行处理，从而降低toncat所承受的压力，而且当其中一台机器宕机时，其他机器还可以继续提供服务，保证服务不间断。

咱们都知道nginx有不同的负载均衡策略，默认是轮询策略，说白了就是将请求的分发到不同的tomcat上，比如我们有两台tomcat，分别是t1和t2，当4个请求到来的时候，第一个请求被nginx发送到t1上，第二个请求被nginx发送到t2上，第三个请求被发送到t1上，第四个请求又发送到t2上，这就是nginx默认的轮询策略，当其中一台tomcat宕机时，会被自动剔除掉。

再者在企业中不同tomcat所在机器配置和性能都不一样，这个时候如果用默认的轮询策略就显的比较浪费资源，我们可以通过weight这个关键字配置加权轮询，比如将性能高的那台tomcat的权重设置为4，将普通配置的那台tomcat设置为1，这样就能达到让性能该的那个tomcat处理更多的请求。

除此之外nginx还有一个IP\_Hash的负载均衡策略，这个策略说白了就是根据访问的客户端的ip地址，对其进行hash化，然后如果该客户端第一次请求被发送到t1上了，那么它就会和t1进行绑定，以后的每次请求也会被发送到t1上，只有宕机了，请求才会被发送到其他tomcat上。所以我们就用ip\_hash这种负载均衡策略来代替默认的轮询策略，这样就解决了验证码错误问题。但这里面还有一个问题，如果被客户端绑定的这台tomcat宕机了，那么请求就会

被发送到其他tomcat上，这个时候在用户进行操作时，请求会被拦截器拦截，拦截器中的代码会验证session中是否有当前登录的用户信息，如果没有则跳转到登录页面，因为这台tomcat上的session中不存在用户登录的信息，所以就会跳转到登录页面，这就会导致用户的体验度非常不好，感觉也会怪异。

为了最终解决这个问题，我通过上网查资料，就决定通过采用基于redis的分布式session来解决session共享问题。

要想完成分布式session的编码工作，就必须需要搞清楚单机版的session是如何工作的，这样我们就可以模拟它的工作流程和机制，并且把相关的信息存储到redis这个缓存服务器中。

单机版session的工作原理是这样的，当访问页面时如果遇到getSession（）这个方法，那么就会在服务端的内存中开辟一块空间用来存储以后要放到session中的数据，并且会生成一个唯一的session作为和这块空间关联的依据，写入到客户端浏览器的会话Cookie中，说的通俗一点可以这样理解，就相当于我们去超市哦购物的时候一个个的存储柜，其中session就是这个存储柜上的钥匙，而存储在session中的数据就是存储到这一个个存储中的物品，一把

钥匙只能开一个柜子门，这样就保证了不同人拿着他们唯一的那把钥匙去开各自的柜子门去到属于自己的物品。

当知道了这些后，我们就清楚，这里面最主要涉及到会话cookie的操作以及将数据存储到redis中并且还得让会话cookie和redis中的key产生关联，这样每次发送请求时，会将本地存储cookie的域名和当前访问的域名进行对比，如果匹配则会将cookie自动提交到服务器端，这样就可以根据cookie中存储哪个自定义的sessionld在redis中找到对应的数据。

首先我封装一个cookie的工具类，然后在里面写了常见的写cookie，读coodie，删cookie的常见操作。这里面要注意的是，cookie被分为会话cookie和持久化cookie，他们的区别就是会话cookie的生命周期和浏览器保持一致，一旦浏览器关闭了，就会消失而持久化cookie是存储在硬盘上，它的生命周期取决于在编程的时候设置的maxAge属性，而持久化cookie要根据项目的要求设置maxAge；

再者在设置maxAge时候还需要设置域名以及Path，通常将域名设置为当前域名，将path设置为/；代表网站的跟目录，写cookie通过的是response的addCookie方法，而读cookie用的是request的getCookies方法，返回一个cookie比较简单，说白了就是将指定name的cookie的maxAge设置为0，然后重新通过response的addCookie方法写入到客户端浏览器中，这样就完成了对指定cookie的删除工作。

在项目中我是通过jedis来作为中间的桥梁操作redis服务器的，并且考虑到性能问题配置了jedis连接池，这样就可以在大并发访问的时候提高性能，并且把对redis的常规操作也都封装成了工具类，方便后续的使用。

我将项目中单机版的session的地方都改为了通过redis的进行信息的存储。首先在生成验证码的地方，我首先通过CookieUƟl读取指定name的cookie，如果没有读取则通过uuid生成一个唯一的标识作为cookie的值，用指定的Loginld作为cookie的name将其通过CookieUƟl的writeCookie方法写入到客户端浏览器中，这样就保证了最开始没有Loginld的时候会生成新的，如果Loginld存在则无论将Cookie的值作为redis中的key，将生成的随机数验证码作为值，

调用redisUƟl工具类存储到redis中，并设置过期时间为5分钟，因为验证码美必要一致存在占用内存空间。

在登录方法中首先通过CookieUtil的readCookie方法根据Loginld作为key，读取指定的cookie，获取cookie中的值，将该值作为redis中key，读取对应的验证码信息，和用户输入的验证码进行对比，如果一致则验证码正确进行后面的验证，否则验证码输入错误。在各项验证都成功后将获取的用户信息通过Gson转换为json格式的字符串作为值，用user+Loginld作为key，将用户信息存入到redis中，并设置过期时间为30分钟，同时删除redis中的验证码，释放空间。

最后就是在拦截器中通过CookieUtil的readCookie方法读取到会话cookie中存储的Loginld的值，用user+Loginld作为redis中的key读取用户信息，如果为null则证明用户没有成功登录跳转到登录页面，否则证明用户成功登录，调用redisUƟl中的expire方法给redis中的用户信息续命，重新设置为30分钟，这也是参考本地session的工作机制，这样就完成将整个项目中的本地session切换为基于本地session的工作机制。这样就完成将整个项目的本地session切换为基于自定义cookie+redis的分布式session。就解决了在进行负载均衡配置后，如果一台tomcat宕机，用户也无序重新登录的问题。

1. **后台系统管理**

我当时就负责过项目中的系统管理模块，这个模块最主要是为了控制系统的权限资源，让不同角色的用户登录后看到不同的权限，我们当时将权限控制到了按钮级别。其中这个模块最主要的就是授权和验权！

在这个系统管理模块中包含了用户管理，角色管理，菜单管理以及给用户赋角色，给角色赋权限；涉及到的表有用户表，角色表，用户角色关联表，资源表，角色资源关联表。

在资源管理模块采用ztree进行资源的增删改查操作，为了将权限控制到按钮级别我们在进行资源管理时会设置该资源是属于按钮还是普通菜单，通过在数据库中增加一个type类型的字段来实现，如type为1则是普通菜单，type为2则是按钮。

这样用户在登陆的时候根据用户名和密码到用户表验证信息是否合法，如果合法 则获取用户信息，之后根据用户id再到用户角色关联表中得到相关连的角色id集合,之后根据角色id再到角色资源关联表中获取该角色所拥有的资源id集合，然后再根据资源id集合到资源表中获取具体的菜单，展现给当前登录用户，从而达到不同的用户看到不同的菜单资源。

为了防止用户不登录而直接访问后台资源我通过在项目中加入LoginInterceptor拦截器对未经认证的用户进行拦截，如果未经认证则跳转到登陆页面使用户再次登陆，除此之外还加入了PermissionInterceptor拦截器来对用户无权访问的资源进行拦截，如果无权限访问则跳转到没有访问权限页面。

在登录拦截器中我们验证session中是否有用户的信息，如果有则证明用户是经过正常途径成功登录的，那就放行；因为在登录成功的时候我们会将用户信息放入session中，这样可以作为拦截中衡量用户是否成功登录的一个标志。如果没有则证明用户没有正常成功登录则重新跳转到登录页面。

在权限拦截器中，我们首先判断如果该用户的角色是管理员则直接放行，因为管理员可以访问所有的资源，在用户成功登录的时候我们就会获取用户的角色将其存入到session中，这样在拦截器中就可以直接从session中获取角色信息，提高性能。

再者我们会判断用户访问的资源是否在资源表中存在，如果不存在也直接放行，因为这类资源可以被所有用户访问到，所以才没有添加到资源表中进行管理，资源表也是被缓存起来的，这样可以避免在拦截器中频繁查询数据库。

最后我们还会判断用户访问的资源是否在他所拥有的资源权限列表内，如果是也直接放行，

当然考虑到性能，用户所拥有的资源权限列表也是在用户成功登录时候存入到缓存中的。

如果以上这些都不能满足则证明用户无权访问资源，跳转到没有权限页面。在我们后续的测试中普通的请求是符合上面的流程判断的，但是对于ajax请求则不能进行页面的正常跳转。所以当时就在权限拦截器中通过request来获取http的X-Requested-With这个header,判断是否为XMLHttpRequest，如果是则证明是ajax请求，我们针对ajax请求会返回一个json串，前台在接受到这个json串后跳转到没有权限访问页面。

在用户登录的时候我们考虑到安全性又加入了 一天内连续三次密码错误，则将当前用户锁定的功能。具体在实现的时候我们是这样做的，首先我们在用户表中增加了 密码错误时的登录时间，密码错误时的登录次数，以及当前用户的状态这三个字段。每次用户密码输入错误的时候我们就会让错误登录的次数加1，并且更新错误登录时的时间，当登录错误次数为3时我们就会将用户的状态设置为锁定状态。

增加错误登录时间的目的是为了解决怎么验证是一天内的错误，因为如果用户账号没被锁定，则第二天又有三次机会，我们将登录错误的时间和当期登录时间都精确到天，这时候可以根据当前时间和登录错误的时间做对比，如果当前时间在登录错误时间的后面则证明是第二天，此时将登录错误次数重置为0，这样就保证了在账号被锁定前，用户第二天又有3次机会。

再者考虑到是连续3次输入错误，所以在账号被锁定前，如果有1次登录成功，就会将错误登录的次数重置为0。

考虑到用户密码存储的安全性我们通过md5进行双次加密后存到数据库表中，并且考虑到用户密码在传输过程中的安全性，我们也通过md5的双次加密保证密码的安全传输。而且在用户注册成功的时候我们还会在用户表中随机生成一个salt，每个用户的salt都不一样，在进行md5时候，将salt融入其中，就更增加了安全性。

通过上面就完成了用户登录以及系统权限管理的整个过程。

在我们的程序中，我们的日志记录起到了一个至关重要的作用，因为我们可以通过日志记录直观的看到程序的一个运行情况，以及我们后期对程序的维护和升级都需要依赖于日志。

# Redis

1 什么是redis (数据库 关系型数据库 mysql orale db2 sqlserver 非关系型数据库（缓存） redis mogodb )

C语言 key-value存储系统。

Redis 优势

原子 – Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。

基于内存 读写快 数据类型丰富5种 持久化方案2种

读写快

性能极高 – Redis能读的速度是110000次/s,写的速度是81000次/s 。

数据类型丰富5种

丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作

string（字符串）

key value

命令 set 和get

设置值

set key value

get key

hash（哈希）

key filed value

设置值

hmset key filed value

取值

hget key filed

list（列表）

设置值 lpush key value，value

取值 lrange key 下标，下标

set（集合）

设置值

sadd key value ,value

取值

smembers key

zset(sorted set：有序集合)。

设置值

zadd key score value

zadd key 分 值

取值 zrange by score key 小分 大分

zrange key 小分 大分

Redis支持数据的备份

持久化方案2种

(rdb 按照一定时间的key改变 进行持久化 丢失数据会丢失一片

aof 日志 写操作(增删改操作)的指令会被记录 丢失只会丢失很少的数据)

使用场景 经常访问但不会轻易改变的数据 都可以使用来减轻数据库的压力

(购物车( 电商项目使用aof方案 ) 地区 热点数据 缓存)

三大问题 穿透 击穿 雪崩

缓存穿透 根本不存在的key 是穿透

把传过来的key 设置个失效时间

缓存击穿（热点数据集中失效）

缓存的key时间到期 直接去查数据库了

互斥锁 (mutex key)：这种方案思路比较简单，但是存在一定的隐患，如果构建缓存过程出现问题或者时间较长，可能会存在死锁和线程池阻塞的风险，但是这种方法能够较好的降低后端存储负载并在一致性上做的比较好。

” 永远不过期 “：这种方案由于没有设置真正的过期时间，实际上已经不存在热点 key 产生的一系列危害，但是会存在数据不一致的情况，同时代码复杂度会增大。

如何避免缓存雪崩？

如果缓存因某种原因发生了宕机，那么原本被缓存抵挡的海量查询请求就会像疯狗一样涌向数据库。此时数据库如果抵挡不了这巨大的压力，它就会崩溃。

使用缓存集群，保证缓存高可用

和飞机都有多个引擎一样，如果缓存层设计成高可用的，即使个别节点、个别机器、甚至是机房宕掉，依然可以提供服务，例如前面介绍过的 Redis Sentinel 和 Redis Cluster 都实现了高可用。

使用缓存集群 3种 主从复制 哨兵 集群

主从复制

主从 1主多从 读写分离 主服务提供写服务 从服务提供读功能

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。

修改配置文件

slaveof 主服务的IP 主服务端口

哨兵

监测主服务 主服务如果死亡 将从 从服务中选择一个作为主服务并告知其他从服务 主服务进行了变更

集群 (分片)

数据分区(或称数据分片)是Redis集群最核心的功能

一个 Redis 集群包含 16384 个哈希槽（hash slot)， 数据库中的每个键都属于这 16384 个哈希槽的其中一个，集群使用公式CRC16(key) % 16384 来计算键key属于哪个槽，其中 CRC16(key)语句用于计算键key的CRC16校验和 。集群中的每个节点负责处理一部分哈希槽。

一般是3主 3从 主服务死亡 从服务顶上 如果都死亡 其他的主服务将 死亡的服务占据的哈希槽 在进行分配

# 多线程

然后新建我们的定时器类TimerTask

在需要定时执行的方法上添加我们的注解 来设置我们的定时时间

@Scheduled(cron = "0 30 22 ? \* \*")

设置上之后 这个方法的方法体就变成了我们定时任务的执行体 等定时时间达到之后 就会去执行方法体里面的内容

注意类上需要添加我们的component注解 component注解是吧我们这个类 交给aop进行管理

1. 发送邮件

创建一个线程池来发送邮箱 线程池核心线程数根据自己的用户数量和服务器配置来均衡一个中间点，我们这用到了三个核心线程数。

循环我们的用户数据，创建我们的线程穿过去几个参数1.我们邮件的标题2.邮件的内容3.邮件接收者，之后再我们的线程执行体中（run方法中）去调用一下我们邮件的工具类，工具类中配置号我们发送方的邮箱号，设置我们的发送方的邮箱类型（**smtp.qq.com**）配置发件人的邮箱密码 （密码是我们在账号设置中设置的smtp的密匙）

设置发送人，接收人，标题，内容，这就实现了我们的邮箱发送

3.分布式锁

我们需要在定时任务的执行体里面添加一个分布式锁，目的是为了防止多台服务器集群同时触发我们的定时任务造成的邮件重复问题，这个分布式锁 我们用到了redis中的getset方法，

redis的getset方法redis中string类型下的 她会存进去新的值并且返回我们原来的值，原来没有值的话返回null；

通过getset方法把我们的当前时间作为key存到我们的redis中

1. **多线程**

为什么使用多线程？

创建线程的有哪些方式？

1）继承Thread类创建线程类

继承thread重写run方法

直接new 线程对象.start执行

2）通过Runnable接口创建线程类

实现runnable重写run方法

new thread

new 线程类 吧线程类放到thread中

然后通过start启动

3）通过线程池创建

线程池就可以理解为我们的线程容器

线程池的目的是为了减少我们线程的运行时间

不需要每次都去创建线程 执行后再销毁线程了 而是每次过来都去线程池中取一个线程，执行完之后吧线程归还给线程池中

//创建一个线程池

ExecutorService executorService = Executors.newFixedThreadPool(3);//设置主线程3

#### Java线程具有五中基本状态：

五个状态：新建状态，就绪状态，运行状态，阻塞状态，死亡状态

1）新建状态（New）：当线程对象对创建后，即进入了新建状态，如：Thread t = new MyThread()；

2）就绪状态（Runnable）：当调用线程对象的start()方法（t.start();），线程即进入就绪状态。处于就绪状态的线程，只是说明此线程已经做好了准备，随时等待CPU调度执行，并不是说执行了t.start()此线程立即就会执行；

3）运行状态（Running）：当CPU开始调度处于就绪状态的线程时，此时线程才得以真正执行，即进入到运行状态。注：就

绪状态是进入到运行状态的唯一入口，也就是说，线程要想进入运行状态执行，首先必须处于就绪状态中；

4）阻塞状态（Blocked）：处于运行状态中的线程由于某种原因，暂时放弃对CPU的使用权，停止执行，此时进入阻塞状态，直到其进入到就绪状态，才 有机会再次被CPU调用以进入到运行状态。

根据阻塞产生的原因不同，阻塞状态又可以分为三种：

a.等待阻塞：运行状态中的线程执行wait()方法，使本线程进入到等待阻塞状态；

b.同步阻塞 – 线程在获取synchronized同步锁失败(因为锁被其它线程所占用)，它会进入同步阻塞状态；

c.其他阻塞 – 通过调用线程的sleep()或join()或发出了I/O请求时，线程会进入到阻塞状态。当sleep()状态超时、join()等待线程终止或者超时、或者I/O处理完毕时，线程重新转入就绪状态。

5）死亡状态（Dead）：线程执行完了或者因异常退出了run()方法，该线程结束生命周期。

#### 5线程安全问题

在我们运行过程中可能会遇到线程安全问题

什么是线程安全问题？

（1）运行结果错误

我们多线程成一起对变量进行操作 进行i++ 一千个线程同时执行 得到的结果不一定是1000 会有得到重复数据的 是因为同时读到变量的值 （200）同时加一 最终的结果都是201

我们可以加一个synchronized锁 目的是为了让线程走到有锁的地方单个顺序执行

synchronized是加在容易产生线程安全问题的地方

但是枷锁也会碰到逻辑安全问题

死锁

死锁是一种状态，当两个（或多个）线程（或进程）相互持有对方所需要的资源，却又都不主动释放自己手中所持有的资源，导致大家都获取不到自己想要的资源，所有相关的线程（或进程）都无法继续往下执行，在未改变这种状态之前都不能向前推进，我们就把这种状态称为死锁状态，认为它们发生了死锁。

还有优先级不够 一直拿不到cpu资源 从而导致进程不运行

# 事务

# 1什么是事务？

事务就是一组有逻辑的操作单元，即一组sql语句，事务的操作要么都发生，要么都不发生

1. 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。

1. 事务的传播特性

传播特性

1. PROPAGATION\_REQUIRED(瑞快润特)

如果事务不存在则创建，如果存在使用存在的事务

1. PROPAGATION\_SUPPORTS(死普特死)

如果当前存在事务，则使用存在的事务，如果不存在则使用非事务方式处理。

1. PROPAGATION\_MANDATORY(慢的脆)

如果当前存在事务，则使用存在的事务，如果不存在则抛出异常。

4. PROPAGATION\_REQUIRES\_NEW(瑞快润)

永远创建新的事务，如果当前存在事务则挂起

5. PROPAGATION\_NOT\_SUPPORTED

永远使用非事务方式执行，如果当前存在事务则挂起

6. PROPAGATION\_NEVER

不支持事务执行方式，如果存在事务则抛出异常

7. PROPAGATION\_NESTED

如果存在一个活动的事务，则运行在一个嵌套的事务中. 如果没有活动事务, 安按照PROPAGATION\_REQUIRED 事务属性执行

1. 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。

# 基于token的登录

token是什么

token是一个认证令牌

为什么要用token

用token是为了确认用户是否登录，没有登录的话就不能让他访问接口，防止有人恶意访问我们的接口。

刚开始我们的项目里用的session，结果发现session因为是作用在，一个服务器的会话中，在分布式框架中不能获取到数据，然后就用了redis来解决这个问题，把数据存放到redis中，再从redis中取出数据，这样就解决了跨域问题，最后接触到了JWT技术，然后就把redis替换了。

用户在前端输入账号和密码，请求路径传到后端，会遇到一个跨域问题（跨域就是协议，端口，ip，三者之一 有一个不同就构成了跨域），在后端，我们配置了一个跨域拦截器，继承的ZuulFilter来解决跨域问题，然后我们是根据用户的账号查询数据库里是否有这个数据，没有的话说明他没有注册，然后重定向到注册页面让用户注册，反之，将传过来的密码进行MD5加密，用MD5加密密码是为了防止数据库密码泄露，然后判断密码是否正确，如果不正确返回一个状态码给前端，反之，将账号和密码放到map中，用JWT将map生成token令牌，再设置一个失效时间，然后将生成的token返回到前端。

前端我们用的是vue，在登录成功之后，我们把后端传过来的token存到sessionStorage中，然后我们在vue的main.js中设置了一个axios的全局设置，为了让每次的请求都添加上一个token参数，在axios的全局设置中判断本地的sessionStorage中是否有token，有的话就在请求上添加一个token参数，反之发请求的话会报没有token的错误。然后再配置一个axios全局设置的请求响应，判断返回回来的状态码，如果相同的话就返回到登录页面，让他登录。

session和token的区别就是session是存在一个服务器会话中，不能用在分布式框架中，相反token可以。