项目一： 车车车险 2018/8 -2019/5 Spring Boot+SpringCloud+Spring MVC

项目二： 阿保在线保险商城系统2017/9 - 2018/7 Spring+SpringMVC+Mybatis+dubbo+zookeeper

项目三： 保险电商服务平台2017/1- 2017/9 Spring+ SpringMVC + Mybatis + redis Nginx+MySQL

项目四：环辉租车2016/3 - 2017/1 Spring+SpringMVC+Mybatis+Mysql

项目五: 车险综合服务平台 2015/7 - 2016/3 Spring+SpringMVC+Mybatis+Mysql

第一个项目做了9个月，

第二个项目做了10个月，

第三个项目做了8个月，

第四个月做了9个月，

第五个项目做了8个月。

优点：对多线程比较感兴趣，所以有深入的了解，对java集合框架有深入研究。比较有上进心，遇到不会或者不明白的问题愿意去探索。

缺点：比较腼腆，有点代码洁癖

对加班怎么看：加班是很正常的一件事，毕竟我们是给别人工作的，人家要得急我们当然要去赶进度。对加班不反感，没有太大的意见。

家住在哪？住在海淀区，沙河那一片，到这需要两个小时左右的车程，到公司还是有点远的，如果可以在咱们公司的话，住的地方也快到期了，那就正好在附近的地方租一套方便上下班。

简单的介绍一下自己，我叫罗佳，来自河南省，在北京毕业后一直在从事java行业，已经三年多了，工作期间做过五个项目，负责过保险类的，租车的和保险管理后台的项目，最近的一个项目是车保类的项目，用的是SpringBoot+SpringCloud的分布式框架，数据库使用的是MySQL关系型数据库，redis这些非关系型数据库。

北京车与车科技有限公司：北京市朝阳区酒仙桥路甲10号3号楼15层17层1701-76

北京市朝阳区北苑路甲13号院2号楼7层2-704

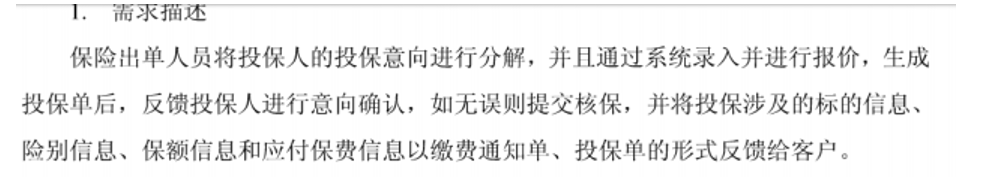
公司还可以，规模也挺大的，我们这个研发部大概有30来号人

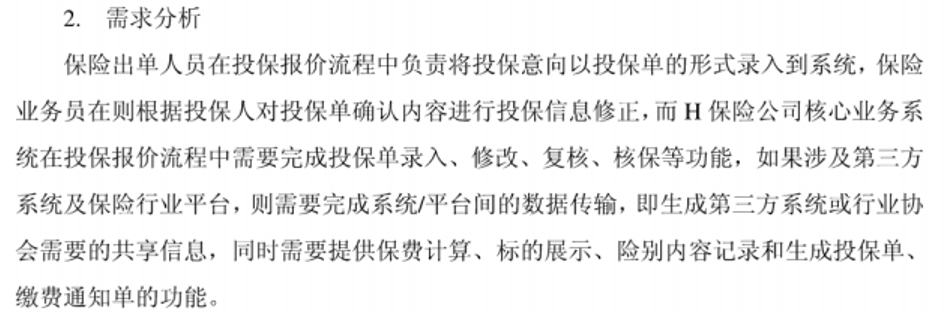
我们小组有9个人 ，一个小组长，两个测试，6各开发两个前端，四个后端

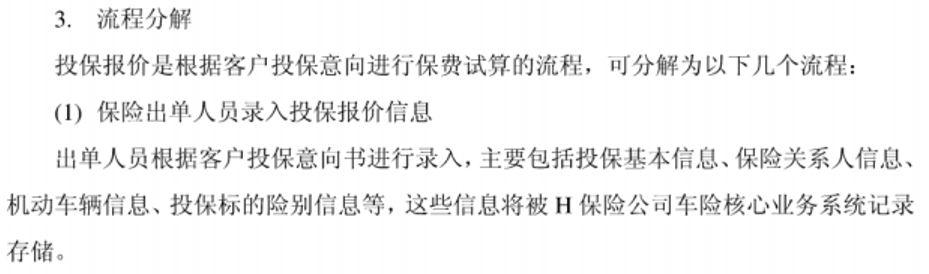
北京市朝阳区汤立路201号院4号楼21层1单元2506

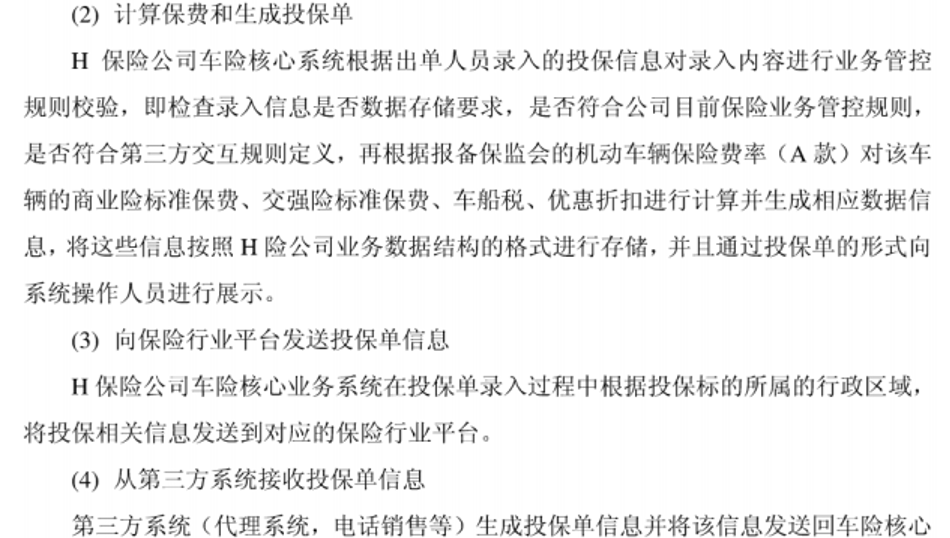
**北京市东城区美术馆后街大取灯胡同2号**

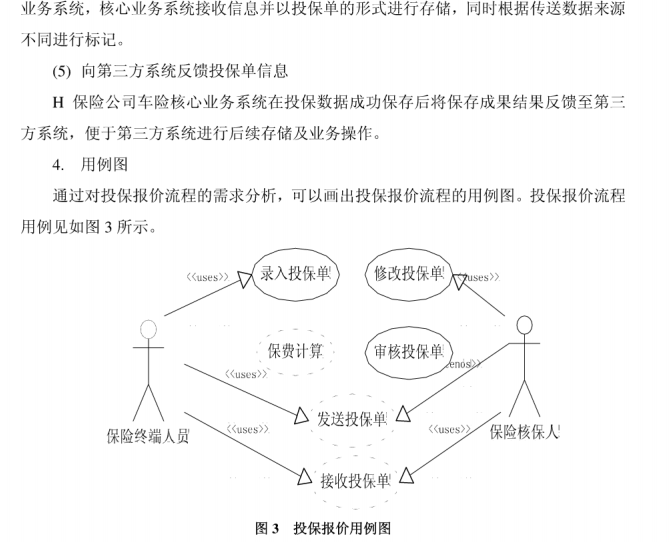
**投保报价模块:**

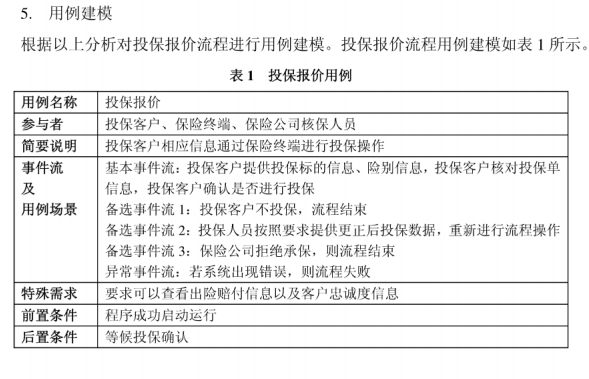












**如何保 证调用接口的安全性：**

我们和保险公司约定好，http请求头设置一个规则，用我们公司的名字和保险公司的名字进行加密，以此来保证我们接口的安全性。

**个案抽挡:**

参与者 总公司续期业管岗，分公司续期业管岗、中支续期内勤

输入 保单信息

流程描述

1．操作员输入保单号码等条件后，点击【查询】按钮，显示符合抽档条件的保单信息；

2．核对保单信息无误，并选择需要抽档的保单后，点击【个单续期个案催收】按钮，系统校验是否满足抽档条件，如不满足，提示相应的提示语“抽档错误”；如 满足，保单续期个案抽档成功，按险种生成应收保费的记录。

3．如果不录入应收起期和止期默认为当年度的应收起期和止期。

4．客户交费完当年度的保单，且保单核销之后，该保单下一年度需要继续交费的，列入应收起期和止期，可以对下一年度保单进行抽档。个案抽档只能在上次保费实收之后才能抽取下一年度应收保费。例如2018年已经交费核销，可以抽档2019年度的保单，不能抽档2020的保单。

输出 抽档成功的应收数据。

A、操作员输入保单号码等条件后，点击【查询】按钮，显示符合条件的保单信息。

B、核对保单信息无误，并选择需要抽档的保单后，点击【个单续期个案催收】按钮，系统校验是否满足抽档条件，如不满足，提示”抽档错误+具体的抽档被校验原因”；如满足，保单续期个案抽档成功。

C、个案抽档不受时间限制。

D、处于自垫状态的保单，如果上期自垫为整期自垫，则本次仍参与抽档；如果上期自垫为非整期自垫，则本次不允许续期抽档。

E、对于整期自垫后再次抽档的保单，抽档后续做如下特殊处理：

（1）不发送续期交费提醒短信和续期催交提醒短信；

（2）不生成续期交费通知书，在第三方文件生成时和综合打印中均不支持查询和打印续期交费通知书；不向E服务推送续期通知书和续期交费信息；

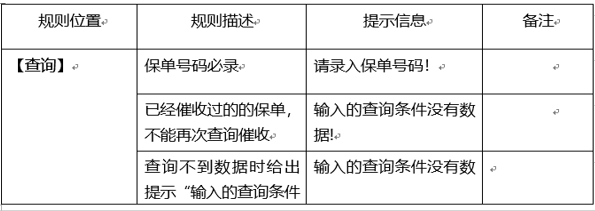
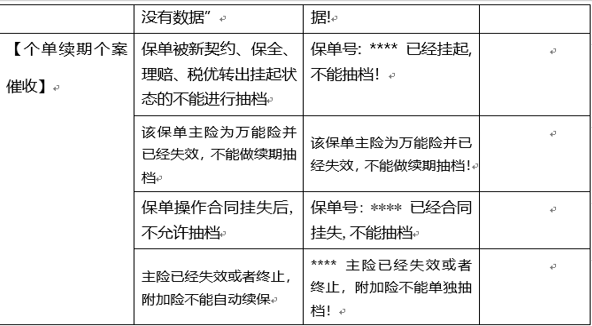
（3）不体现在当期的应收清单中，包括应收已收清单、应收未收清单和应收清单；

（4）不体现在销管应收表laspayperson中，不计入续期达成率统计。

F、如果不录入应收起期和止期默认为当年度的应收起期和止期。

G、客户交费完当年度的保单，且保单核销之后，该保单下一年度需要继续交费的，列入应收起期和止期，可以对下一年度保单进行抽档。个案抽档只能上次保费实收之后才能抽取下一年度应收保费。例如2018年已经交费核销，可以抽档2019年度的保单，不能抽档2020年度的保单。

H、保单失效、终止不能个案抽档

**登录注册：**

JSON Web Token（JWT）是一个非常轻巧的规范。这个规范允许我们使用JWT在用 户和服务器之间传递安全可靠的信息。

一个JWT实际上就是一个字符串，它由三部分组成，头部、载荷与签名。

头部（**Header**） 头部用于描述关于该JWT的最基本的信息，例如其类型以及签名所用的算法等。这也可以 被表示成一个JSON对象。 在头部指明了签名算法是HS256算法。 我们进行BASE64编 码http://base64.xpcha.com/，编码后的字符串如下： eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9 小知识：Base64是一种基于64个可打印字符来表示二进制数据的表示方法。由于2 的6次方等于64，所以每6个比特为一个单元，对应某个可打印字符。三个字节有24 个比特，对应于4个Base64单元，即3个字节需要用4个可打印字符来表示。JDK 中 提供了非常方便的 BASE64Encoder 和 BASE64Decoder，用它们可以非常方便的 完成基于 BASE64 的编码和解码

载荷（**playload**） 载荷就是存放有效信息的地方。这个名字像是特指飞机上承载的货品，这些有效信息包 含三个部分 {"typ":"JWT","alg":"HS256"}

（1）标准中注册的声明（建议但不强制使用） iss: jwt签发者 sub: jwt所面向的用户 aud: 接收jwt的一方 exp: jwt的过期时间，这个过期时间必须要大于签发时间 nbf: 定义在什么时间之前，该jwt都是不可用的. iat: jwt的签发时间 jti: jwt的唯一身份标识，主要用来作为一次性token,从而回避重放攻击。

（2）公共的声明 公共的声明可以添加任何的信息，一般添加用户的相关信息或其他业务需要的必要信息. 但不建议添加敏感信息，因为该部分在客户端可解密.

（3）私有的声明 私有声明是提供者和消费者所共同定义的声明，一般不建议存放敏感信息，因为base64 是对称解密的，意味着该部分信息可以归类为明文信息。这个指的就是自定义的claim。比如前面那个结构举例中的admin和name都属于自定的 claim。这些claim跟JWT标准规定的claim区别在于：JWT规定的claim，JWT的接收方在 拿到JWT之后，都知道怎么对这些标准的claim进行验证(还不知道是否能够验证)；而 private claims不会验证，除非明确告诉接收方要对这些claim进行验证以及规则才行。 定义一个payload: {"sub":"1234567890","name":"John Doe","admin":true}然后将其进行base64编码，得到Jwt的第二部分：

eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9

签证（**signature**） jwt的第三部分是一个签证信息，这个签证信息由三部分组成： header (base64后的) payload (base64后的) secret

这个部分需要base64加密后的header和base64加密后的payload使用.连接组成的字符 串，然后通过header中声明的加密方式进行加盐secret组合加密，然后就构成了jwt的第三部分：

TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7HgQ

将这三部分用.连接成一个完整的字符串,构成了最终的jwt:

eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.TJVA95OrM7E2cBab30RMHrHDcEfxjoYZgeFONFh7Hg Q

注意：secret是保存在服务器端的，jwt的签发生成也是在服务器端的，secret就是用 来进行jwt的签发和jwt的验证，所以，它就是你服务端的私钥，在任何场景都不应该流 露出去。一旦客户端得知这个secret, 那就意味着客户端是可以自我签发jwt了。  
 在做注册的时候，我们根据公司提供的API文档，找到对应的发送短信的接口写入到我们的controller当中，在我们的service层中生成验证码，这个验证码使用random随机数生成，当时我们控制这个随机数为六位数这个范围，也就是大于等于100000小于等于999999，得到验证码后，用redisTemplate这个类调用opsforvalue方法，将手机号码和验证码以key，value的形式存入到redis中（String）设置过期时间：TimeUnit.MINUTES，这个验证码并不会一直都有用，用户使用一次之后，就没有什么太大的用户了，处理好这些后，我们将用户的手机号和验证码放到一个map集合当中，然后将整个集合发送到rabbitMQ指定的队列上等待被消费，这里用rabbitTemplate这个模板调用converAndSend这个方法便可以直接塞入到指定的队列当中，我们在调用阿里云的短信接口的时候，因为我们这个是一个微服务的项目，所以我们专门写了一个工程，这个工程就是专门为了获取rabbitMQ队列上的数据调用阿里大于给我们客户发送短信用的，在这里我们写了一个消息监听类，引入@RabbitListener和@Component以及@RabbitHandler便可消费我们rabbitMQ上的消息，我们往rabbmtMQ传的时候用的MAP集合装的，获取的时候当然也就用Map集合来进行获取，

在使用阿里云发送短信时，他就提供给我们一个短信工具类，这个短信工具来就可以自动的帮我们发送短信了，我们的短信模板，短信签名，手机号在阿里云官网就可以直接弄好了，他就可一直接给我们指定的手机发送短信，然后我们的短信监听类在使用阿里云提供的的短信工具类这些模板和签名的时候，直接使用@Autowired注入就好了，然后调用工具类里的发送短信方法，便可成功发送短信，我们用户在进行注册的时候，输入验证码后，去拿着我们验证码和我们redis缓存当中的验证码做对比，如果验证码为空和输入有误，则提示用户验证码有问题，如果正常的话则对我们的用户密码进行加密，这里加密的时候采用的是Bcrypt加密算法。

当时也考虑过用MD5加密，但是后来发现这个东西能够被破解，而Bcrypt这个东西里面有时间戳，时间戳这个东西是几乎不可能被破解的，Bcrypt这个加密算法是Spring Security安全框架提供的，所以我们只需要引入Spring Security这个依赖便可以了，但是我们们引入这个安全框架之后，所有的地址就得被安全框架所控制了，而我们只是用单纯的加密算法，所用这个时候我们建立一个安全配置类，里面引入@Configuration和@EnableWebSecurity这两个注解，便可配置所有的地址匿名访问，在启动类里面bean注入BCryptPasswordEncdor就可以加密了，我们再进行登陆效验密码的时候，BCryptPasswordEncdor给我们提供了两个方法，encode和matches方法，用encode方法进行加密，matches方法进行用户前台传入的密码和我们数据库加密后的密码进行比对，我们在登录的时候，选择了基于JWT的Token认证机制，客户端使用用户名和密码进行请求登陆，服务端收到请求，去验证用户名和密码，验证成功后，服务端会发送一个Token，再把这个Token发送给客户端，客户端收到Token后把他存储起来，比如经他放到cookie当中，客户端每次向服务器请求资源的时候，都会带着这个Token串，服务端收到请求后，会拿着前台传过来的Token串和我们服务端的Token串进行对比验证，成功就返回给客户请求的数据，  
创建token方式首先导入JJWT依赖，设置签发时间，设置密钥，每次的token都是不一样的，因为载荷当中存在时间.

**为什么要使用rabbitMQ？**

解耦，公用，并发

**如何防止表单重复提交？**

在数据库里添加唯一约束或创建唯一索引，防止出现重复数据。这是最有效的防止重复提交数据的方法。

**如何保证token的安全性？**

在token加上规则，公司名首位字母加上项目名称进行加密拼接在token上

**如何设置信息过期时间？**

后台设置了一个单次ip请求的次数限制。在服务器端限制单个 IP 在有效的时间内的请求次数，一旦用户请求次数(包括失败请求次数)超出我们设定的次数，则暂停对该 IP 一段时间内的请求。若情节特别严重，可以将 IP 加入黑名单，禁止该 IP 的访问请求。这样就可以限制一个 IP 地址的大量请求，避免攻击者通过同一个 IP 对大量用户进行攻击，增加了攻击难度，保障了业务的正常开展。

**搜索模块：**  
***项目的需求ElasticSearch***  
因为ES它在实时搜索的时候效率是高于solr的  Solr更适合传统搜索应用在搜索大的数据的情况下 ES的效率会很高  solr需要配域而ES不需要 相对来说比较简单  
我们在做搜索的时候主要是根据用户的需求进行课程搜索，我们是使用docker搭建一个ES的公共服务  
一开始我们是用postman调用restapi新建索引和文档 在新建文档的时候 id是由系统自动生成的。在修改文档的时候是根据id来进行修改的 如果id不存在 它就会创建新的文档 删除文档的话也是根据id来进行删除的 通过这个我们发现 如果都是通过rest请求的方式使用ES未免太过麻烦，而且也不够人性化。然后就想到了使用图形化界面来实现ES的日常管理，最常用的就是Head插件  在安装好Head插件后 我们发现了一个问题 就是它在启动之后 点击连接按钮没有反应 按F12发现了一个错误

“No 'Access-Control-Allow-Origin' header is present on the requested resource” 这个错是由于ES默认不允许跨域调用，而es-head是属于前端工程，所以报错。  
然后我就修改了ES的配置 让它允许跨域调用  
我们在做测试的时候 发现它的分词不太智能 默认的中文分词是将每个字看成一个词 这样是不符合要求的 所以我们就用了IK分词器来解决这个问题 IK 分词器有两种算法 其中ik\_smart为最少切分，ik\_max为最细粒度划分   
我们还可以自定义词库   
然后 我们又要考虑一个问题 因为我们查询的数据是来自不同的数据库或者表 这样的话就很难以一个比较快的速度去查询数据 这个时候我们就想到了用这个logstash进行同步数据 同步数据库与索引库 （solr是用的定时器 而ES用的是logstash）logstash是一款轻量级的日志搜集处理框架， 咱们在公司里边用这个ES也是分两种 一种是增量索引更新一种是全量索引更新 当我们安装好logstash 需要在logstash的bin文件夹下创建一个.config结尾的文件，在里面配置mysql的链接还有查询课程表的sql语句、es的链接，最主要还是需要配置一个定时任务。这个定时任务就是精确到年月日时分。在这儿我配的是每24小时同步一次。它有点儿类似于一个定时器 可以让它定时的去读我的数据库里边的东西，读完之后再把他所读东西搜集起来指定给ES  这样的话就相当于隔24小时ES就被同步一次  
ES的核心搜索引擎是通过倒排索引实现的，说倒排索引前我先说一下正排索引，它是通过关键词去文档中直接查找相应的单词，而倒排索引它是先把文档中的内容切分成词条，再经过词干还原、不区分大小写等方式转变成词项存储在倒排索引的数据结构中。倒排索引的数据结构包括由单词组成的词典、单词索引以及关键词出现过的文档编号。当用户在文本框中输入关键词后，es搜索引擎会先去词典中进行查找，找到后会根据单词索引展示文档内容。

***Solr搜索引擎***

1.场景：用户登陆我们的网站，显示的是前台网站的首页。用户进来一般有两种情况，

无目的的随意看看和有目的的商品搜索，而第二个就是我们搜索引擎的存在的意义。

在首页中，一般会有一个搜索框，当用户在搜索框内输入关键字并点击搜索，则进入商品搜索页面。可以说首页的这个搜索框就是我们整个搜索模块的入口。

2.业务：在搜索页面中，我们会根据用户的关键字进行查询，整个页面中展示了包括关键字所涉及的所有分类，以及默认分类下的所有商品品牌、规格、规格属性，还有所有涉及的商品的列表。用户根据需求选择相应的属性，得到一个细化过后的spu属性，点击这个spu属性，则进入freemarker的静态页获取具体的sku商品。

3.实现：我们采用了solr搜索引擎，来实现搜索模块。同级别的其他搜索引擎还有ElasticSearch,简称es。Es也是一款基于lucence的搜索引擎。

Solr跟Es相比：solr查询快，更新索引相对慢，es更新索引快，查询相对慢。所以我们采用了solr来做搜索引擎。

在做商品搜索模块的时候我们考虑到用户频繁的使用搜索模块，而如果我们以like模糊查询的我们的数据库，如果访问量大数据库压力太大数据库将无法承受这个压力，这个时候我们就考虑到了用solr搜索引擎来处理这个问题，

缺模块具体业务

为了减轻对服务器的压力，我们在这块呢，给搜索起了一个单独的服务，同时呢，我们这个商品的数量还是非常多的，为了减少对数据库的压力，我们使用solr这个搜索引擎来解决这个问题，

Solr这一块是如何做的:

Solr是基于luncne开发的全文搜索引擎，是Apache开发的一个开源的web工程

项目中使用solr之前，我们要对solr进行相关的配置

先到官方下载solr工程的war包，将这个war包部署到服务器当中

Solr默认的中文分词器对中文的支持不是很好，所以我们给solr配置一个中文分词器，我们选用的是ik分词器，ik分词器是一个开源的基于java语言开发的轻量级的中文分词工具包

分词器的配置：将分词器的jar包拷贝到solr/web-inf/lib下

然后在solr/web-inf/下面创建一个文件夹，名为classes，在这个文件夹中，我们放扩展词典和扩展停止词典，

接着创建存放solr数据的文件夹，solrhome，我们可以存放在D盘下

将这个solrhome的地址，在solr的web.xml中配置一下

Schema.xml也是在这个solrhome里面的，这个schema.xml主要配置的就是solr数据库中的字段中文分词器也是在这里配置的 配置一个fieldType 让solr可以使用中文分词器

配置域：域就相当于 实体中的属性

复制域：将某个field中的数据复制到另一个域中

动态域：当我们需要动态扩充字段时，我们需要使用动态域

这些关于solr配置完毕之后，我们就可以结合项目开发了

首先，我们要将数据库里面的数据，一次性的导入到solr里面。

那我们在做sol导入数据，有两种方式，一种是导入插件，但是这种插件，只能做一些简单的业务逻辑，所以我们在这个项目当中呢，是自己用代码进行导入的，先从数据库中将数据查询出来，在pom.xml中配置springDataSolr依赖，然后利用solrtemplate来将数据导入到solr里面。

要想实现高亮，要在那个字段高亮，怎么去亮。我们自己设置一个标签，例如:<em style=“color:red”>属性</em>

先创建一个高亮的设置选项，设置到那个域来高亮怎么高亮，设置样式，要在这个属性前边加个前缀，在属性后边价格后缀，

在哪个域上查询这个效果，进行封装，设置完之后就可以根据这个条件进行查询。得到处理过的数据列表之后，高亮并没有直接显示，我们高亮的字符串取出来在设置。进行循环。

而我们在做高亮的时候，显示的html代码会原样输出，这是angulars为了防止html攻击采取的安全机制，我们如何显示html的结果呢，我们用到了$sce服务的trustAsHtml方法来进行转换。（过滤器）

而我们在做用关键字来查询列表的时候，

当用户输入关键字搜索后，除了显示列表结果外，还应该显示通过这个关键字搜索到的记录都有哪些商品分类。

根据第一个商品分类查询对应的模板，根据模板查询出品牌列表

根据第一个商品分类查询对应的模板，根据模板查询出规格列表

当用户点击搜索面板的商品分类时，显示按照这个关键字查询结果的基础上，筛选此分类的结果。

当用户点击搜索面板的品牌时，显示在以上结果的基础上，筛选此品牌的结果

当用户点击搜索面板的规格时，显示在以上结果的基础上，筛选此规格的结果

当用户点击价格区间时，显示在以上结果的基础上，按价格进行筛选的结果

当用户点击搜索面板的相应条件时，隐藏已点击的条件。

首先我们在使用关键字之后怎么根据关键词查询出分类表。

搜索面板的商品分类需要使用Spring Data Solr的分组查询来实现

为了能够提高查询速度，我们需要把查询面板的品牌、规格数据提前放入redis

查询条件的构建、面板的隐藏需要使用angularJS来实现

后端的分类、品牌、规格、价格区间查询需要使用过滤查询来实现

首先我们商品分类是根据关键字的出来的，查出几种就是几种，我们用到了Solr里的Spring Data Solr 里的一个group选项，可以设置以哪一个域作为搜索条件，根据结果组装程一个集合展示出来。

而我们做品牌和规格的时候，我们存几种数据，通过商品分类找到模板id，

将商品分类数据、品牌数据、和规格数据都放入Redis存储。

当用户进入运营商后台的商品分类页面时，将商品分类数据放入缓存（Hash）。以分类名称作为key ,以模板ID作为值

当用户进入运营商后台的模板管理页面时，分别将品牌数据和规格数据放入缓存（Hash）。以模板ID作为key,以品牌列表和规格列表作为值

7.索引库的更新

最开始我们做搜索模块时，第一次运营商后台查询分类+模板列表等数据时,就一次性将数据加入索引库。可以省却临时搜索临时去数据库找数据的过程，能够省时省事。

后来经过更新，选择在第一次运营商后台查询分类+模板列表等数据时，转由rabbitmq，转交给solr索引库。查询更加方便快捷。

**技术问题：**

**跨域**

问题：说出几种你常见的解决跨域的方法？

答案：通过jsonp跨域；

使用window.name来进行跨域；

window.postMessage方法来跨域；

代理。

【扩展】： 通过修改document.domain来跨子域、flash

**Git**

问题：简单说一下你在做项目的时候一个git的流程？

答案：首先我们在做一个功能的时候，从自己的分支上拉一个feature分支，一般命名都是以这个功能块的名字来命名；功能做好之后，向老大发送一个pull请求，把代码合并到自己的分支上

【扩展】

提测时，从自己的分支上拉一个release分支，发现有bug时，在从release 分支上拉一个fixbug分支，修复bug之后，合并fixbug分支到release 支，最后合并到自己的分支上

代码冲突时，如果说就几个js文件，我们找见代码冲突的地方，把冲突的代码删除，换成自己的代码，最后提交；如果说有几百个文件，而我们又修改过好多，完全不知道是哪一块期待吗冲突了，这个时候我们要拉一个分支，把我们的代码保存一下，通过git log回滚到上一个版本，我们之前提交的起冲突代码就没有了，最后回到我们保存代码的分支，重新merge到我的分支：

保存 =》 回滚 =》 合并。

**三.solr同步索引**：数据库中数据会发生改变。如果我们不同步到Solr中的话就搜索不出最新的数据。所以我们再项目中会配置定时任务，定时的触发增量同步。增量同步是根据Solr中的dataimport.properties中记录的上一次同步时间和数据库中数据最后更新时间做对比，然后找出需要同步的数据。在同步到Solr时，这些数据会经过分词器，分词后以词作为数据的索引，就好比字典中的目录。查询数据时候根据这个索引找数据。就好比根据目录找字。所以速度快

**四.高亮：**Solr在做查询时候我们项目导入了Spring-Boot-Starter-data-solr里面封装了SolrJ。查询时候创建一个SolrQuery对象。高亮的话需要通过setHighlighting开启高亮显示。然后设置高亮字段，还有高亮显示的前缀和后缀。想让匹配的文字在页面中变色，实际上就是在匹配的文字两端加上一对HTML标签。这个前缀后缀就是带着class的span标签，给class赋予样式。在查询时SolrJ就会返回给我们2组数据。一组是正常的数据集合。一组是以商品编号作为key以高亮信息作为值的Map。我们需要把这个Map和List进行融合，把高亮信息覆盖到集合中，然后用覆盖好的List展示。

**Redis**是一个内存数据库。查询速度快。里面可以存储5中类型的数据。String，Hash，List，Set，SortSet（zSet），我们可以通过Jedis或者SpringBoot提供的redisTemplate进行连接。对redis进行增删改查。因为它是内存数据库。为了防止数据丢失，redis中有两种持久化机制，一个是RDB一个是AOF。RDB是快照模式，按redis.conf中配置的规则进行快照，快照是一个rdb文件，存储再硬盘上。当服务器宕机或者重启后，会根据rdb文件进行恢复。但是这个快照两次之间有时间差，如果在两次之间宕机，就会造成数据丢失。效率高。但是不安全。AOF是日志形式，他会把我们数据库中的增删语句记录到日志中，日志存在硬盘上的AOF文件上，重启宕机后根据AOF文件恢复，虽然效率不如RDB，但是安全。我们一般用Redis来做数据缓存。在做定时器集群时也用到redis。还有就是防止短信频繁发送。也用redis来替代session存储用户信息。

*面试官问你还有什么需要问我的吗？*

对我这次的面试您有什么评价？晋升机制，入职后培训，福利待遇，最近在做什么项目，什么框架。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **车辆保费计算方法** | | | | | |
| 车价 | 国产 | | 进口 | | 备注 |
| 262000 | 保费价格 | 不计免赔 | 保费价格 | 不计免赔 |  |
| 车损： | 4560 | 684 | 4560 | 684 |  |
| 三者： |  | 0 |  | 0 | 5万710元； 10万1026元； 15万1169元； 20万1270元； 30万1434元； 50万1721元； 100万2234元； |
| 盗抢： | 1403.8 | 280.76 | 1403.8 | 280.76 |  |
| 自燃： | 393 | 58.95 | 393 | 58.95 |  |
| 玻璃： （无不计免赔） | 497.8 | 0 | 812.2 | 121.83 |  |
| 人员（司机+乘客\*1万） | 150 | 22.5 | 150 | 22.5 |  |
| 划痕： |  | 0 |  | 0 | 2000保额400元； 5000元保额（车价30万以下车辆）570元，（30万以上车辆50万以下车辆）900元，50万以上车辆1100元） |
| 合计 | 8050.81 | | 8487.04 | |  |
| 备注：此保费价格为参考价格，实际保费以保险公司出单核算为准。保费正常优惠通常为85折优惠，如果指定驾驶人（最多3位，根据驾驶人条件确定优惠系数）或省内行驶最多优惠到7折。 | | | | | |

|  |  |  |
| --- | --- | --- |
| 车损 | 车价\*1.5%+630元 | |
| 三者 | 5万710元；10万1026元；15万1169元；20万1270元；30万1434元；50万1721元；100万2234元 | |
| 盗抢 | 车价\*0.49%+120元 | |
| 自燃 | 车价\*0.15% | |
| 划痕 | 2000保额400元；5000元保额（车价30万以下车辆）570元，（30万以上车辆50万以下车辆）900元，50万以上车辆1100元） | |
| 玻璃 | 国产玻璃：车价\*0.19%；进口玻璃：车价\*0.31%（无不计免赔） | |
| 人员 | 司机：1万42元；乘客：1人/万\*4位 108元 | |
| 不计免赔 | 车损、三者、自燃、划痕、人员\*0.15%；盗抢\*0.2% | |
| **车船使用税（私家车）** | | |
| 1.0以下 |  | 240元 |
| 1.0以上——1.6（含） | | 360元 |
| 1.6以上——2.0（含） | | 420元 |
| 2.0以上——2.5（含） | | 900元 |
| 2.5以上——3.0（含） | | 1800元 |
| 3.0以上——4.0（含） | | 3000元 |
| 4.0以上 |  | 4500元 |