🤓 elasticSearch

学习目标

1.elasticSearch创建的数据结构

2. 倒排索引

3.ik分词器自定义

4.基本RestFul命令

5.操作命令

- 5.1.创建文档（不符合规则,应当先定义规则，在插入数据）
- 5.2.elasticSearch的数据类型
- 5.3.创建索引，定义规则
- 5.4.get方式查询信息
- 5.5.修改信息
- 5.6.操作文档

6.elasticSearch查询

- 6.1简单查询
- 6.2复杂查询
- 6.3.小总结
- 6.4.注意JSON结果

7.springboot集成se

8.爬虫:

9.搜索引擎接口

9.前后端交互vue渲染

> 模板作者@guihaole

日期：1月21日 星期五

学习目标

1.elasticSearch回顾及其知识点

2.学习视频，狂神说

1.elasticSearch创建的数据结构

- index索引-------数据库创建库
- types类型-------数据库创建表（弃用）
- documents文档--------数据库记录
- fields------------数据库的列

2. 倒排索引

1. 什么叫倒排索引?

列：有三个文档【high ,low,mid,haha,to,forerver】,【high,text,date,low,data】,【a,b,text,date】

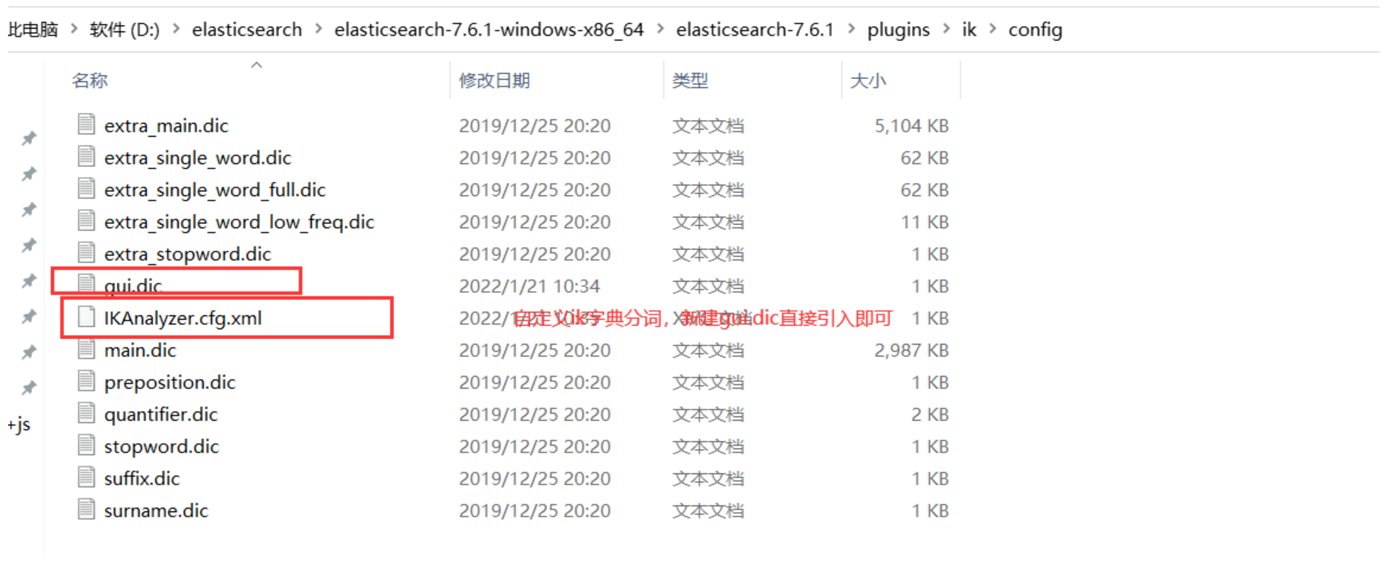
high	1,2
low	1,2
mid	1
haha	1
a	3
forerver	1
text	2,3
date	2,3

将所有的数据进行分类，过滤掉无关的数据，每个数据在不同的文档中权重比列也不相同，这就叫倒排索引

2. elasticSeach的索引

elasticSeach的底层使用了luece，实质上也就是使用了luece倒排索引，一个索引可以有多个分片，elasticSeach已启动就是一个集群，每一个索引分片可以在集群中移动，索引相当于type的集合又是多个文档的集合，文档为最小的单位。

3.ik分词器自定义



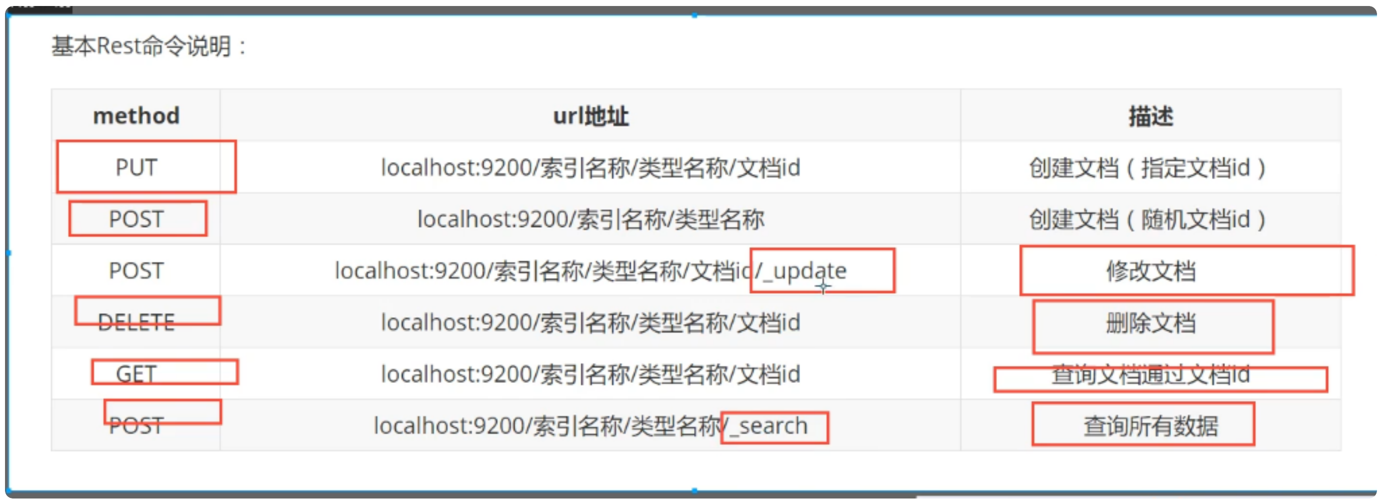
此电脑 › 软件 (D:) › elasticsearch › elasticsearch-7.6.1-windows-x86_64 › elasticsearch-7.6.1 › plugins › ik › config

名称	修改日期	类型	大小
extra_main.dic	2019/12/25 20:20	文本文档	5,104 KB
extra_single_word.dic	2019/12/25 20:20	文本文档	62 KB
extra_single_word_full.dic	2019/12/25 20:20	文本文档	62 KB
extra_single_word_low_freq.dic	2019/12/25 20:20	文本文档	11 KB
extra_stopword.dic	2019/12/25 20:20	文本文档	1 KB
gui.dic	2022/1/21 10:34	文本文档	1 KB
IKAnalyzer.cfg.xml	2022/[illegible]	[illegible]	1 KB
main.dic	2019/12/25 20:20	文本文档	2,987 KB
preposition.dic	2019/12/25 20:20	文本文档	1 KB
quantifier.dic	2019/12/25 20:20	文本文档	2 KB
stopword.dic	2019/12/25 20:20	文本文档	1 KB
suffix.dic	2019/12/25 20:20	文本文档	1 KB
surname.dic	2019/12/25 20:20	文本文档	1 KB

自定义ik字典分词，新建gui.dic直接引入即可

+js

4.基本RestFul命令



基本Rest命令说明：

method	url地址	描述
PUT	localhost:9200/索引名称/类型名称/文档id	创建文档（指定文档id）
POST	localhost:9200/索引名称/类型名称	创建文档（随机文档id）
POST	localhost:9200/索引名称/类型名称/文档id/_update	修改文档
DELETE	localhost:9200/索引名称/类型名称/文档id	删除文档
GET	localhost:9200/索引名称/类型名称/文档id	查询文档通过文档id
POST	localhost:9200/索引名称/类型名称/_search	查询所有数据

5.操作命令

5.1.创建文档（不符合规则,应当先定义规则，在插入数据）

JSON　复制代码

```json
PUT /text1/type1/1
{
  "name":"归浩乐",
  "age":18
}
```

5.2.elasticSearch的数据类型

除了java中的基本数据类型还有text,keyword(不可分割)

5.3.创建索引，定义规则

```json
PUT /test2
{
  "mappings": {
    "properties": {
      "name":{
        "type": "text"
      },
      "age":{
        "type": "long"
      },
      "birthday":{
        "type": "date"
      }
    }
  }
}
```

5.4.get方式查询信息

- GET text1　查询索引，类型，文档
- GET _cat/health　_cat可以查看es的很多信息
- GET _cat/indices?v

5.5.修改信息

1. 通过普通put的方式

```json
PUT /text1/type1/1
{
  "name":"归浩乐",
  "age":18
}
```

2. 通过POST的方式

JSON 复制代码

```json
POST /text1/type1/1/_update
{
  "doc":{
    "name": "小红书包"
  }
}
```

5.6.操作文档

JSON 复制代码

```
# 练习
PUT /book
{
  "settings": {
    "number_of_shards": 5, "number_of_replicas": 1
  },
  "mappings": {
      "properties":{
        "name":{
          "type": "text",
          "analyzer": "ik_max_word",
          "index": true,
          "store": false
        },
        "author":{
          "type": "keyword"
        },
        "count":{
          "type": "long"
        },
        "on-sale":{
          "type": "date",
          "format": "yyyy-MM-dd HH:mm:ss||yyyy-MM-dd||epoch_millis"
        },
        "descr":{
          "type": "text",
          "analyzer": "ik_max_word"
        }
      }
    }
}
#添加文档
POST /book/_doc/1
{
  "name":"小红书包",
  "author":"归浩乐",
  "count":100000,
  "on-sale":"2000-01-01",
  "descr":"归浩乐喜欢小红书包"
}
#添加文档
POST /book/_doc/2
{
  "name":"归浩乐",
  "author":"小红书包",
    "count":100000,
    "on-sale":"2000-01-01",
    "descr":"归浩乐喜欢小红书包,小红书包也爱归浩乐"
  }
  #修改文档
  POST /book/_doc/2/_update
  {
    "doc":{
      "descr":"小红书包爱归浩乐"
    }
  }
  DELETE /test1
  DELETE /test2
  DELETE /text1
```

6.elasticSearch查询

6.1简单查询

JSON 复制代码

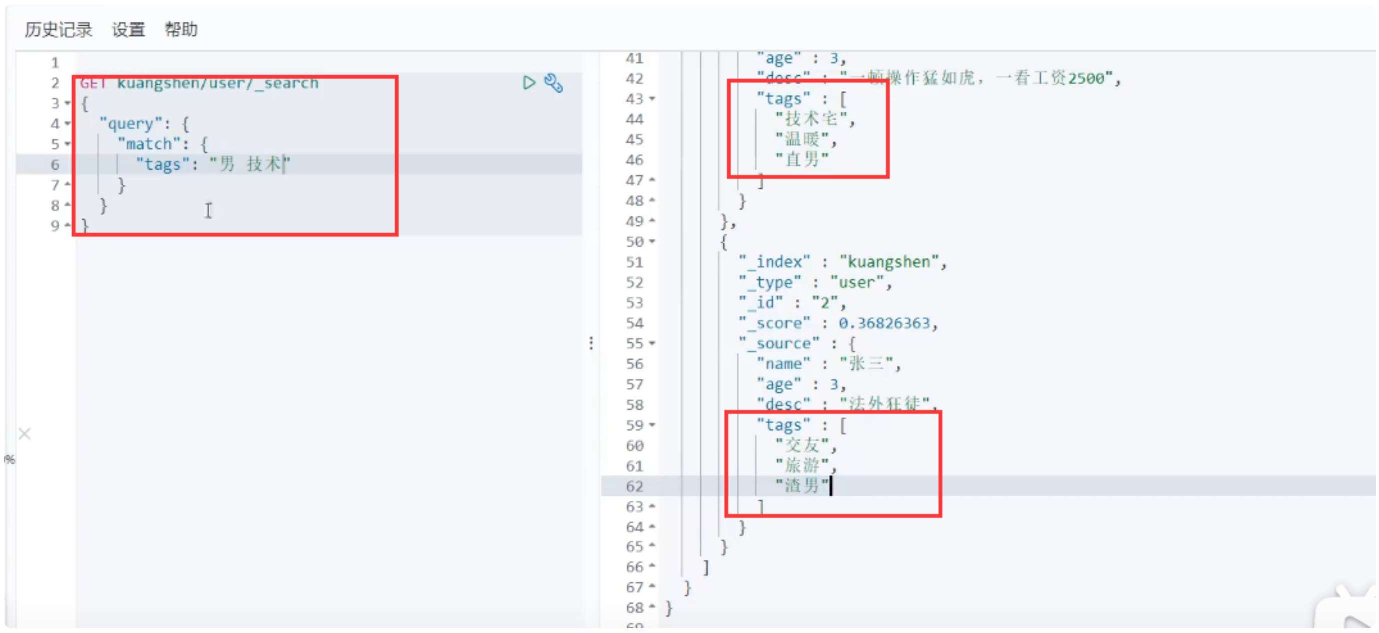
```
#PUT 添加  POST 修改   GET 查询  DELETE 删除
GET /book/_doc/_search?q=descr:小红书包
```

6.2复杂查询

JSON | 复制代码

```
#匹配
GET /book/_doc/_search
{
  "query": {
    # 匹配
    "match": {
      "descr": "小红书包"
    }
  },
  #显示属性那列
  "_source":["descr","name"]
}
#分页 排序
GET /book/_doc/_search
{
  "query": {
    "match": {
      "descr": "喜欢"
    }
  },
  "sort":[
    {
      "count":{
          "order":"desc"
      }
    }
  ],
  #分页
  "from":0,
  "size":2
}
#bool查询多条件查询
#must(and)  should(or)  must_not(not)
GET /book/_doc/_search
{
  "query":{
    "bool":{
      "must":[
        {
          "match":{
            "descr":"喜欢"
          }
        },
        {
          "match":{
              "count":100000
            }
          }
        ]
      }
    }
}
#过滤查询
#gt 大于 gte大于等于 lt 小于  lte小于等于
GET /book/_doc/_search
{
  "query": {
    "bool":{
      "must":[
        {
          "match":{
            "descr":"喜欢"
          }
        }
      ],
     "filter":{
            "range":{
              "count":{
                "lt":5000
              }
            }
          }
      }
    }
}
```

6.2.1. 多条件匹配查询



```
GET kuangshen/user/_search
{
  "query": {
    "match": {
      "tags": "男 技术"
    }
  }
}
```

```
        "age" : 3,
        "desc" : "一顿操作猛如虎，一看工资2500",
        "tags" : [
          "技术宅",
          "温暖",
          "直男"
        ]
      }
    },
    {
      "_index" : "kuangshen",
      "_type" : "user",
      "_id" : "2",
      "_score" : 0.36826363,
      "_source" : {
        "name" : "张三",
        "age" : 3,
        "desc" : "法外狂徒",
        "tags" : [
          "交友",
          "旅游",
          "渣男"
        ]
      }
    }
  ]
}
}
```

精确查询是直接通过倒排索引指定的词条进程精确查找的！

关于分词：

- trem 直接查询精确的　语法trem替换match即可
- match，会使用分词器解析！（先分析文档，让后在通过分析的文档进行查询）
- text和keyword的区别要知道

6.2.2.高亮查询

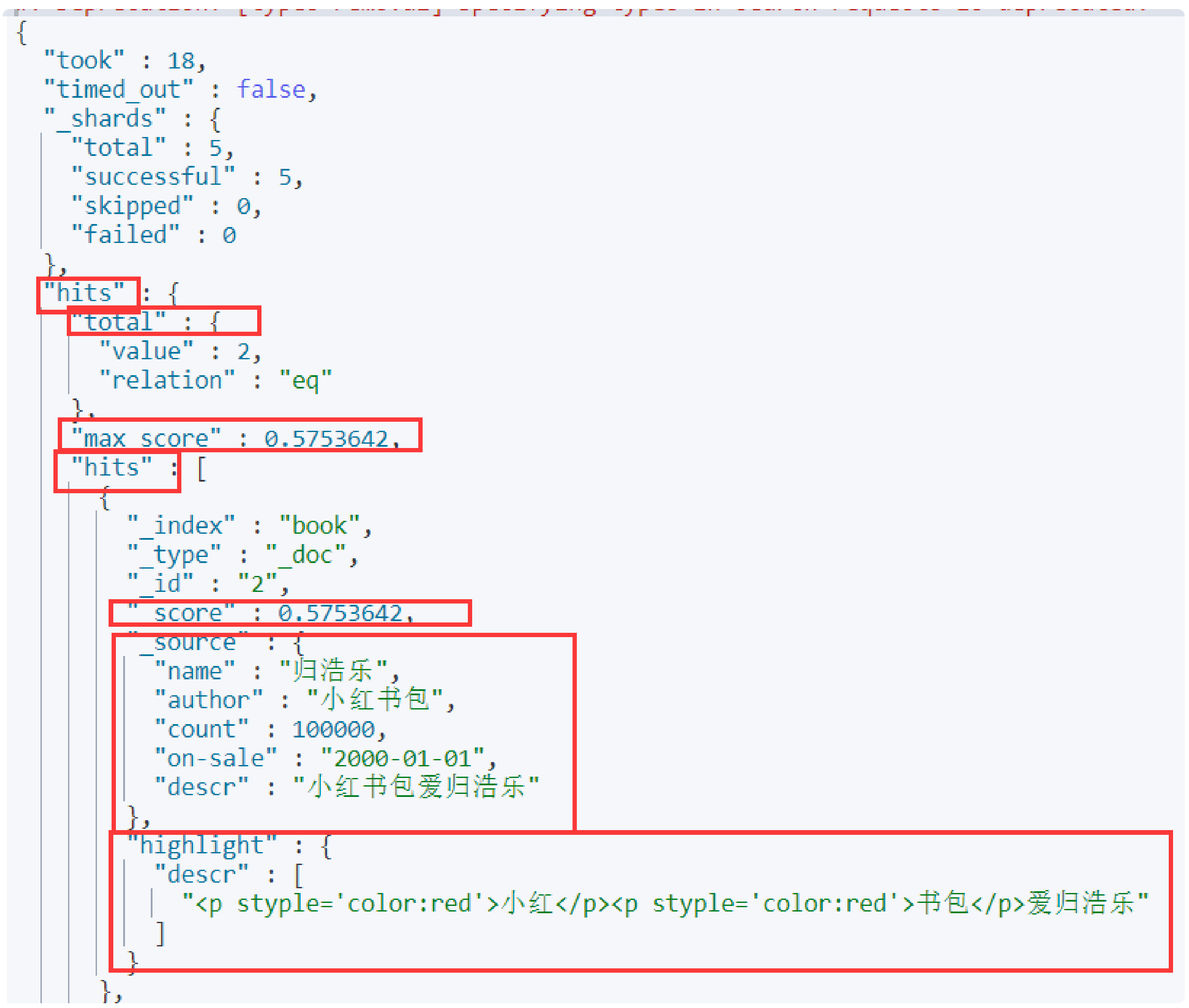
```JSON
#高亮查询
GET /book/_doc/_search
{
  "query":{
    "match":{
      "descr":"小红书包"
    }
  },
  "highlight":{
    "pre_tags":"<p styple='color:red'>",
    "post_tags":"</p>",
    "fields":{
      "descr":{}
    }
  }
}
```

6.3.小总结

es操作手册

- 创建索引数据结构: put /book{setting,mapping-->protities}
- 删除索引或者文档: delete /book ,/book/_doc/id
- 修改文档 post /book/_doc/id/_update {doc{"",""}}
- 增加文档 put /book/_doc/id {}
- 查询：get /book/_doc/_search{ "query":{ } }
- 模糊匹配 match 分页 from ,size 高亮 highlight: pre_tags,post_tags fields
- 多条件匹配查询：bool-->must(must_not,should)-->match(多个) 排序 sort 过滤filter---lt,lte,gt,gte

6.4.注意JSON结果



```
{
  "took" : 18,
  "timed_out" : false,
  "_shards" : {
    "total" : 5,
    "successful" : 5,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2,
      "relation" : "eq"
    },
    "max_score" : 0.5753642,
    "hits" : [
      {
        "_index" : "book",
        "_type" : "_doc",
        "_id" : "2",
        "_score" : 0.5753642,
        "_source" : {
          "name" : "归浩乐",
          "author" : "小红书包",
          "count" : 100000,
          "on-sale" : "2000-01-01",
          "descr" : "小红书包爱归浩乐"
        },
        "highlight" : {
          "descr" : [
            "<p styple='color:red'>小红</p><p styple='color:red'>书包</p>爱归浩乐"
          ]
        }
      },
```

7.springboot集成se

官方高级api文档

https://www.elastic.co/guide/en/elasticsearch/client/java-rest/current/java-rest-high.html
狂神课程笔记
https://www.kuangstudy.com/bbs/1354069127022583809

8.爬虫:

数据问题？数据库获取，消息队列中获取中，都可以成为数据源，爬虫！
爬取数据：

1. 导入java相关的以来

```xml
        <!-- jsoup解析网页，tika用来爬取视频和音乐-->
        <dependency>
            <groupId>org.jsoup</groupId>
            <artifactId>jsoup</artifactId>
            <version>1.10.2</version>
        </dependency>
```

2. 抓取数据工具类

Java 复制代码

```java
//可以根据关键字获取相关的数据
public static List<Content> HtmlParse(String keyword) throws IOException {
        String url="https://search.jd.com/Search?keyword="+keyword;
        Document document = Jsoup.parse(new URL(url), 3000);
        Element elementById = document.getElementById("J_goodsList");
        Elements elements = elementById.getElementsByTag("li");
        ArrayList<Content> list = new ArrayList<>();
        for (Element element : elements) {
            String img = element.getElementsByTag("img").eq(0).attr("data-lazy-img");
            String price = element.getElementsByClass("p-price").eq(0).text();
            String title = element.getElementsByClass("p-name").eq(0).text();
            Content content = new Content();
            content.setPrice(price);
            content.setTitle(title);
            content.setUrl(img);
            list.add(content);
        }
        return list;
    }
```

9.搜索引擎接口

Java 复制代码

```java
@RestController
@RequestMapping("/content")
public class ContentController {
    @Autowired
    private ContentService contentService;
    @GetMapping("/add/{keyword}")
    public boolean add(@PathVariable("keyword") String keyword) throws IOException {
        boolean b = contentService.contentAdd(keyword);
        return b;
    }
    @GetMapping("/search/{keyword}/{pageNo}/{pageSize}")
    public List<Map<String,Object>> contentSearch(
            @PathVariable("keyword") String keyword,
            @PathVariable("pageNo") int pageNo,
            @PathVariable("pageSize") int pageSize) throws IOException{
        List<Map<String, Object>> maps = contentService.contentSearch(keyword, pageNo, pageSize);
        return maps;
    }
}


//service
@Service
public class ContentService {
    @Autowired
    private RestHighLevelClient restHighLevelClient;
    //将数据放入搜索引擎中
    public boolean contentAdd(String keyword) throws IOException {
        BulkRequest bulkRequest = new BulkRequest();
        List<Content> contents = HtmlParseUtil.HtmlParse(keyword);
        for (int i = 0; i <contents.size() ; i++) {
            bulkRequest.add(
                    new IndexRequest("jd_goods")
                    .source(JSON.toJSONString(contents.get(i)), XContentType.JSON)
            );
        }
        BulkResponse bulk = restHighLevelClient.bulk(bulkRequest, RequestOptions.DEFAULT);
        return !bulk.hasFailures();
    }
    public List<Map<String,Object>> contentSearch(String keyword,int pageNo,int pageSize) throws IOException {
        if(pageNo<0){
            pageNo=1;
        }
        SearchRequest searchRequest = new SearchRequest("jd_goods");
        SearchSourceBuilder sourceBuilder = new SearchSourceBuilder();
        sourceBuilder.from(pageNo);
        sourceBuilder.size(pageSize);
        TermQueryBuilder termQueryBuilder = QueryBuilders.termQuery("title", keyword);
        sourceBuilder.query(termQueryBuilder);
        sourceBuilder.timeout(new TimeValue(60, TimeUnit.SECONDS));
        searchRequest.source(sourceBuilder);
        SearchResponse searchResponse = restHighLevelClient.search(searchRequest, RequestOptions.DEFAULT);
        ArrayList<Map<String, Object>> list = new ArrayList<>();
        for (SearchHit hit : searchResponse.getHits().getHits()) {
            Map<String, Object> sourceAsMap = hit.getSourceAsMap();
            list.add(sourceAsMap);
        }
        return list;

    }
}

```

9.前后端交互vue渲染