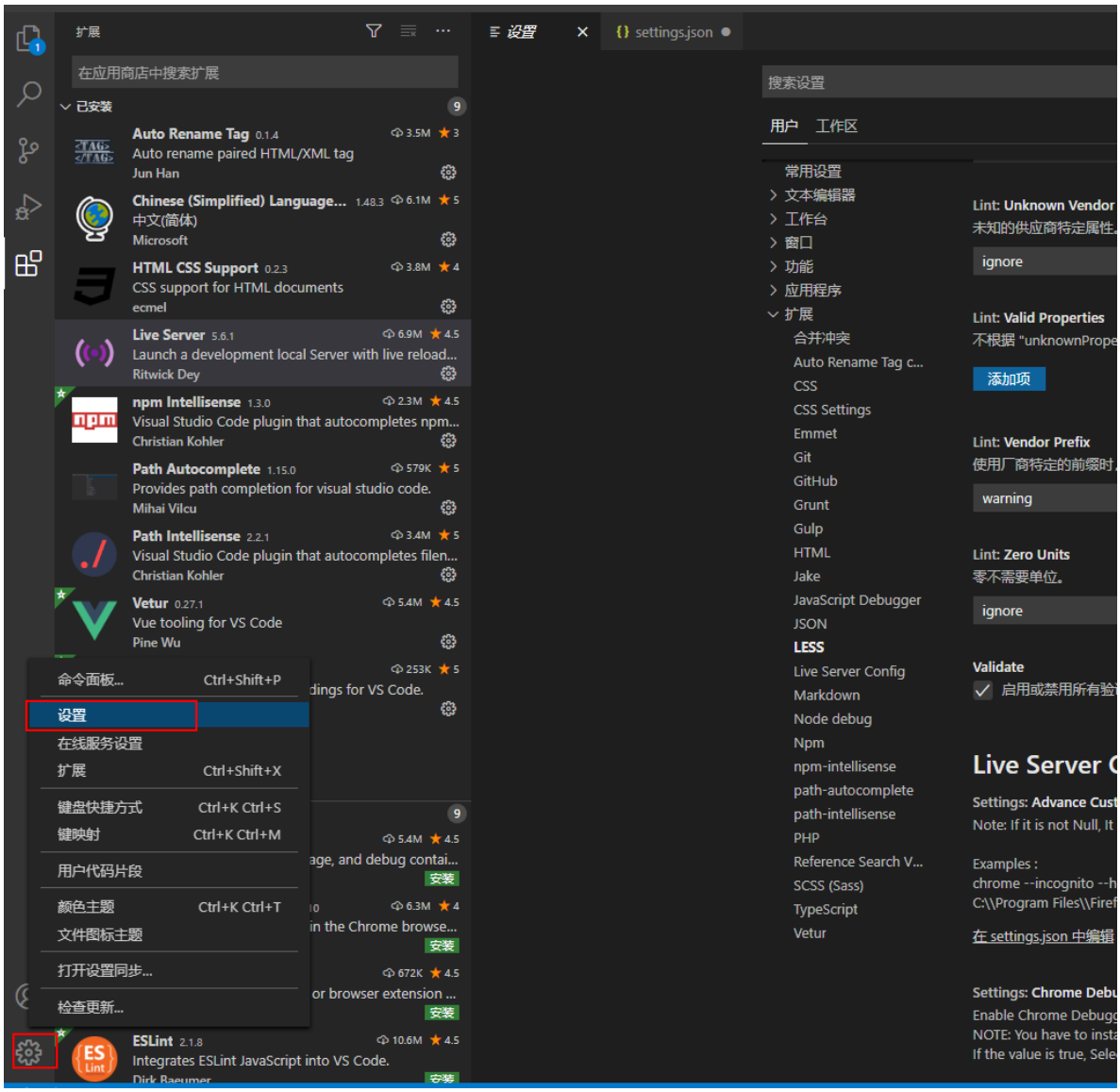
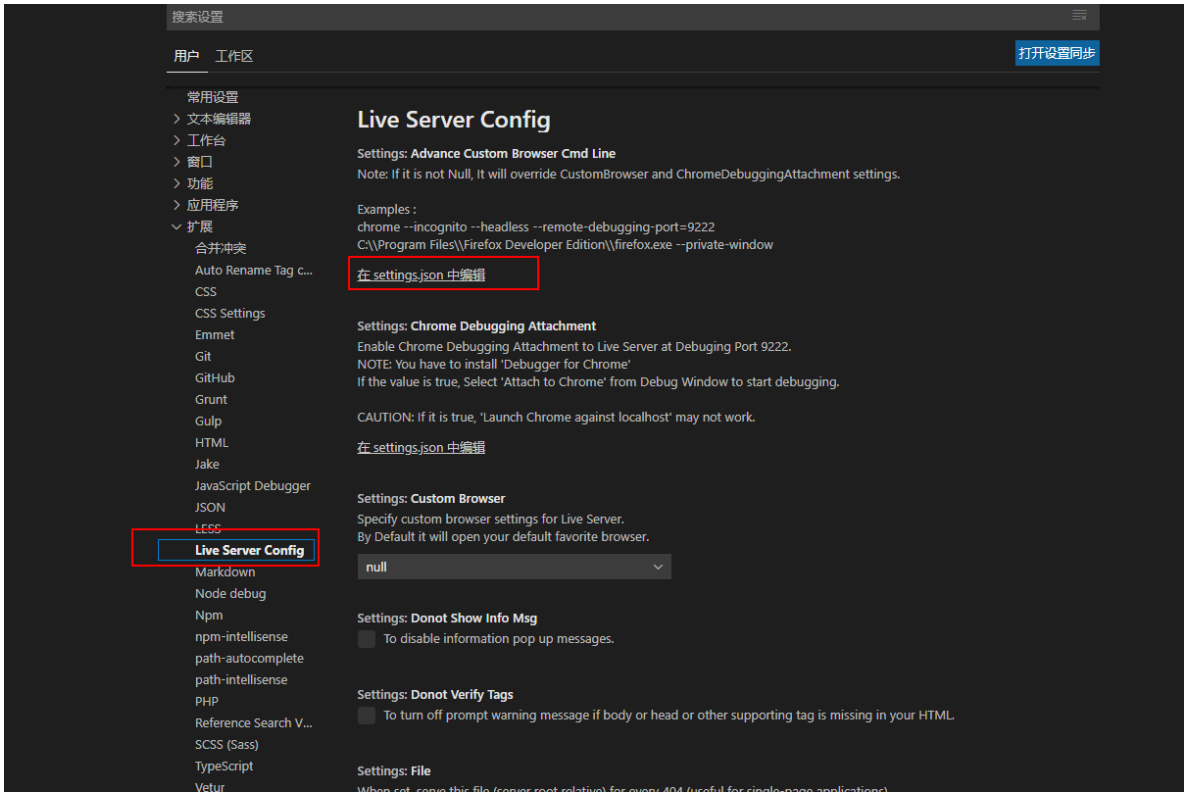
1 学习目标

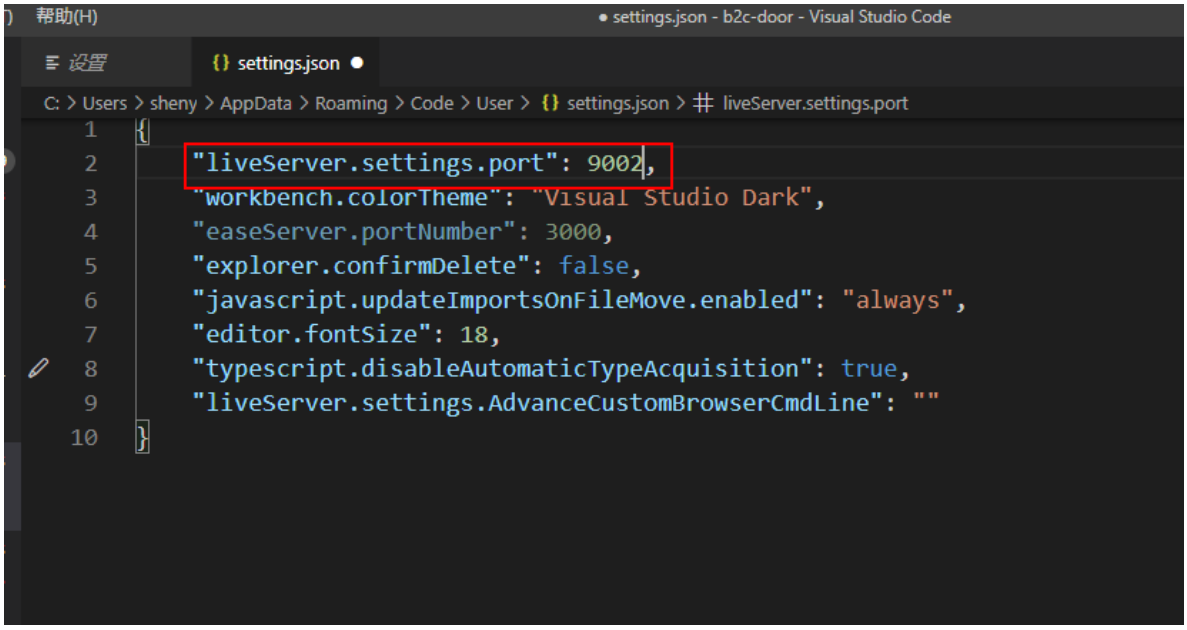
- 独立编写数据导入功能
- 独立实现基本搜索
- 独立实现页面分页
- 独立实现结果排序

2 门户网站

- 在vs-code工作空间下解压mrshop-internet-portal.zip
- vscode打开刚刚解压的项目
- VScode设置








```
"liveServer.settings.port": 9002
```

- hosts文件

```
127.0.0.1 www.mrshop.com
```

- nginx.conf

```
    server {
        listen       80;
        server_name  www.mrshop.com;
        location / {
            proxy_pass http://127.0.0.1:9002;
            proxy_connect_timeout 600;
            proxy_read_timeout 600;
```

```
        }

        error_page   500 502 503 504  /50x.html;
        location = /50x.html {
            root   html;
        }
    }
```

3 后台项目搭建

注意spring-boot版本必须为2.3.1

3.1 mingrui-shop-service-api

3.1.1 pom.xml

```
        <!--2.3版本之后web删除了验证插件-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-validation</artifactId>
        </dependency>
        <!--feign get请求需要使用@SpringQueryMap注解-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
```

3.1.2 新建项目mingrui-shop-service-api-search

3.1.2.1 pom.xml

```
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-elasticsearch</artifactId>
        </dependency>
    </dependencies>
```

3.1.2.2 新建包com.baidu.shop.document

3.1.2.3 在包下新建类GoodsDoc

```
import lombok.Data;
import org.springframework.data.annotation.Id;
import org.springframework.data.elasticsearch.annotations.Document;
import org.springframework.data.elasticsearch.annotations.Field;
import org.springframework.data.elasticsearch.annotations.FieldType;

import java.util.Date;
import java.util.List;
import java.util.Map;

/**
```

```
 * @ClassName GoodsDoc
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/16
 * @Version V1.0
 **/
@Document(indexName = "goods",shards = 1,replicas = 0)
@Data
public class GoodsDoc {

    @Id
    private Long id;

    @Field(type = FieldType.Text, analyzer = "ik_max_word")
    private String title;

    @Field(type = FieldType.Text, analyzer = "ik_max_word")
    private String brandName;

    @Field(type = FieldType.Text, analyzer = "ik_max_word")
    private String categoryName;

    @Field(type = FieldType.Keyword, index = false)
    private String subTitle;

    private Long brandId;

    private Long cid1;

    private Long cid2;

    private Long cid3;

    private Date createTime;

    private List<Long> price;

    @Field(type = FieldType.Keyword, index = false)
    private String skus;

    //规格
    private Map<String, Object> specs;

}
```

3.1.2.4 com.baidu.shop下新建包service

3.1.2.5 在service包下新建ShopElasticsearchService

```
import com.alibaba.fastjson.JSONObject;
import com.baidu.shop.base.Result;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;

/**
```

```
 * @ClassName ShopElasticsearchServicec
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/5
 * @Version V1.0
 **/
@Api(tags = "es接口")
public interface ShopElasticsearchService {

    @ApiOperation(value = "获取商品信息测试")
    @GetMapping(value = "es/goodsInfo")
    Result<JSONObject> esGoodsInfo();
}
```

3.1.2.6 在com.baidu.shop下新建config/MrSwagger2Config

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

/**
 * @ClassName MrSwagger2Config
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/17
 * @Version V1.0
 **/
@Configuration
@EnableSwagger2
public class MrSwagger2Config {

    @Bean
    public Docket createRestApi(){
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(this.apiInfo())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.baidu"))
                .paths(PathSelectors.any())
                .build();
    }

    private ApiInfo apiInfo(){
        return new ApiInfoBuilder()
                //标题
                .title("明瑞SWAGGER2标题")
                //条款地址
                .termsOfServiceUrl("http://www.baidu.com")
                //联系方式-->有String参数的方法但是已经过时，所以不推荐使用
```

```
                .contact(new
Contact("shenyaqi","baidu.com","shenyaqiii@163.com"))
                //版本
                .version("v1.0")
                //项目描述
                .description("描述")
                //创建API基本信息
                .build();
    }
}
```

3.2 mingrui-shop-common-core

3.2.1 pom.xml

```
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-elasticsearch</artifactId>
        </dependency>
```

3.2.2 utils包下新建ESHighLightUtil

```
import org.elasticsearch.search.fetch.subphase.highlight.HighlightBuilder;
import org.springframework.data.elasticsearch.core.SearchHit;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.stream.Collectors;

/**
 * @ClassName ESHighLightUtil
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2021/3/1
 * @Version V1.0
 **/
public class ESHighLightUtil {

    //构建高亮字段buiilder
    public static HighlightBuilder getHighlightBuilder(String ...highLightField)
{

        HighlightBuilder highlightBuilder = new HighlightBuilder();

        Arrays.asList(highLightField).forEach(hlf -> {
            HighlightBuilder.Field field = new HighlightBuilder.Field(hlf);

            field.preTags("<span style='color:red'>");
            field.postTags("</span>");
```

```
            highlightBuilder.field(field);//这个值不会被覆盖,看源码
        });

        return highlightBuilder;
    }

    //将返回的内容替换成高亮
    public static <T> List<SearchHit<T>> getHighLightHit(List<SearchHit<T>>
list){

        return list.stream().map(hit -> {
            //得到高亮字段
            Map<String, List<String>> highlightFields =
hit.getHighlightFields();

            highlightFields.forEach((key,value) -> {
                try {
                    T content = hit.getContent();//当前文档 T为当前文档类型

                    //content.getClass()获取当前文档类型,并且得到排序字段的set方法
                    //注意这种首字母大写的方式效率非常低,大数据环境下绝对不允许,但是可以实现
效果
                    //可以参考ascll表来实现首字母大写
                    Method method = content.getClass().getMethod("set" +
String.valueOf(key.charAt(0)).toUpperCase() + key.substring(1),String.class);

                    //执行set方法并赋值
                    method.invoke(content,value.get(0));
                } catch (NoSuchMethodException e) {
                    e.printStackTrace();
                } catch (IllegalAccessException e) {
                    e.printStackTrace();
                } catch (InvocationTargetException e) {
                    e.printStackTrace();
                }

            });

            return hit;
        }).collect(Collectors.toList());

    }

    //首字母大写,效率最高!
    private static String firstCharUpperCase(String name){

        char[] chars = name.toCharArray();
        chars[0] -= 32;
        return String.valueOf(chars);
    }
}
```

3.3 mingrui-shop-service

3.3.1 新建项目mingrui-shop-service-search

3.3.2 pom.xml

```
    <dependencies>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-elasticsearch</artifactId>
        </dependency>

        <dependency>
            <groupId>com.baidu</groupId>
            <artifactId>mingrui-shop-service-api-search</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>

    </dependencies>
```

3.3.3 application.yml

```
server:
  port: 8300

spring:
  elasticsearch:
    rest:
      uris: 119.45.191.248:9200
  application:
    name: search-server
```

3.3.4 新建包com.baidu

3.3.5 新建启动类RunSearchServerApplication

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.jdbc.DataSourceAutoConfiguration;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.openfeign.EnableFeignClients;

/**
 * @ClassName RunSearchServerApplication
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/16
 * @Version V1.0
 **/
//exclude = {DataSourceAutoConfiguration.class} 不加载数据源的配置
@SpringBootApplication(exclude = {DataSourceAutoConfiguration.class})
@EnableEurekaClient
@EnableFeignClients
public class RunSearchServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(RunSearchServerApplication.class);
```

```
    }
}
```

3.3.5 新建com.baidu.shop.feign

3.3.6 包下新建GoodsFeign

```
import com.baidu.shop.service.GoodsService;
import org.springframework.cloud.openfeign.FeignClient;

/**
 * @ClassName GoodsFeign
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/5
 * @Version V1.0
 **/
@FeignClient(value = "xxx-server")
public interface GoodsFeign extends GoodsService {
}
```

3.3.7 新建com.baidu.shop.service.impl

3.3.8 在包下新建ShopElasticsearchServiceImpl

```
import com.alibaba.fastjson.JSONObject;
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.SpuDTO;
import com.baidu.shop.feign.GoodsFeign;
import com.baidu.shop.service.ShopElasticsearchService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;

import java.util.Map;

/**
 * @ClassName ShopElasticsearchServiceImpl
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/5
 * @Version V1.0
 **/
@RestController
public class ShopElasticsearchServiceImpl extends BaseApiService implements
ShopElasticsearchService {

    @Autowired
    private GoodsFeign goodsFeign;

    @Override
    public Result<JSONObject> esGoodsInfo() {
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setPage(1);
```

```
        spuDTO.setRows(5);
        Result<Map<String, Object>> spuInfo = goodsFeign.getSpuInfo(spuDTO);
        System.out.println(spuInfo);

        return null;
    }
}
```

3.4 mingrui-shop-service-xxx

3.4.1 GoodsServiceImpl

getSpuInfo需要增加分页信息的判断

```
if(spuDTO.getPage() != null && spuDTO.getRows() != null)
PageHelper.startPage(spuDTO.getPage(),spuDTO.getRows());
```

4 mysql数据迁移到es(数据准备)

前置说明:从此功能模块开始,不涉及到新项目不会在将项目名写的那么详细,看不懂文档说明你不应该在这个阶段,建议末班

4.1 spu和sku数据填充

4.1.1 GoodsService

```
    @ApiOperation(value = "获取spu详情信息")
    @GetMapping(value = "goods/getSpuDetailBydSpu")
    public Result<SpuDetailEntity> getSpuDetailBydSpu(@RequestParam Integer
spuId);

    @ApiOperation(value = "获取sku信息")
    @GetMapping(value = "goods/getSkuBySpuId")
    Result<List<SkuDTO>> getSkuBySpuId(@RequestParam Integer spuId);
```

4.1.2 ShopElasticsearchServiceImpl

```
    @Override
    public Result<JSONObject> esGoodsInfo() {
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setPage(1);
        spuDTO.setRows(5);
        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(spuDTO);

        log.info("goodsFeign.getSpuInfo --> {}",spuInfo);

        if (spuInfo.getCode() == 200) {

            List<GoodsDoc> docList = spuInfo.getData().stream().map(spu -> {

                Integer spuId = spu.getId();
```

```
                GoodsDoc goodsDoc = new GoodsDoc();
                //spu信息填充
                goodsDoc.setId(spuId.longValue());
                goodsDoc.setCid1(spu.getCid1().longValue());
                goodsDoc.setCid2(spu.getCid2().longValue());
                goodsDoc.setCid3(spu.getCid3().longValue());
                goodsDoc.setCreateTime(spu.getCreateTime());
                goodsDoc.setSubTitle(spu.getSubTitle());
                //可搜索的数据
                goodsDoc.setTitle(spu.getTitle());
                goodsDoc.setBrandName(spu.getBrandName());
                goodsDoc.setCategoryName(spu.getCategoryName());


                //sku数据填充
                Result<List<SkuDTO>> skuResult =
goodsFeign.getSkuBySpuId(spuId);
                if(skuResult.getCode() == 200){
                    List<SkuDTO> skuList = skuResult.getData();

                    List<Long> priceList = new ArrayList<>();

                    List<Map<String, Object>> skuListMap =
skuList.stream().map(sku -> {
                        Map<String, Object> map = new HashMap<>();
                        map.put("id", sku.getId());
                        map.put("title", sku.getTitle());
                        map.put("image", sku.getImages());
                        map.put("price", sku.getPrice());

                        priceList.add(sku.getPrice().longValue());

                        return map;
                    }).collect(Collectors.toList());

                    goodsDoc.setPrice(priceList);
                    goodsDoc.setSkus(JSONUtil.toJsonString(skuListMap));
                }

                //规格数据填充

                return goodsDoc;
            }).collect(Collectors.toList());

            log.info("docListInfo --> {}",docList);
        }

        return null;
    }
```

4.2 规格数据填充

现在会出现一个问题,那就是feign 重复的问题

多个feign的name相同肯定是不行的,这个问题就相当于有多个相同的@RequestMapping一样

理论上如果大家的电脑内存足够大,然后我们每一个模块就是一个服务,是绝对不会出现这个问题的

就比如说category-service,brand-service,.....

但是我们得尊重现实

所以我们需要解决一下这个问题

在@FeignClient注解上增加属性contextId声明一下上下文的id,不管value属性的值是否一样,只要contextId不一样就没有问题

但是强烈不推荐大家这么做,如果大家在公司搭建springcloud框架的话,一定是将模块服务化

```
@FeignClient(value = "xxx-server",contextId = "SpecificationService")
public interface SpecificationFeign extends SpecificationService {
}
```

```
@FeignClient(contextId = "GoodsService", value = "xxx-server")
@FeignClient(contextId = "SpecificationService", value = "xxx-server")
```

4.2.1 SpecificationService

```
    @ApiOperation(value = "查询规格参数")
    @GetMapping(value = "specparam/getSpecParamInfo")
    public Result<List<SpecParamEntity>> getSpecParamInfo(@SpringQueryMap
SpecParamDTO specParamDTO);
```

4.2.2 JSONUtil

```
    public static Map<String, String> toMapValueString(String json) {

        Map<String, String> map = gson.fromJson(json, new TypeToken<Map<String,
String>>() {
        }.getType());
        return map;
    }

    public static Map<String, List<String>> toMapValueStrList(String json) {

        Map<String, List<String>> map = gson.fromJson(json, new
TypeToken<Map<String, List<String>>>() {}.getType());

        return map;
    }
```

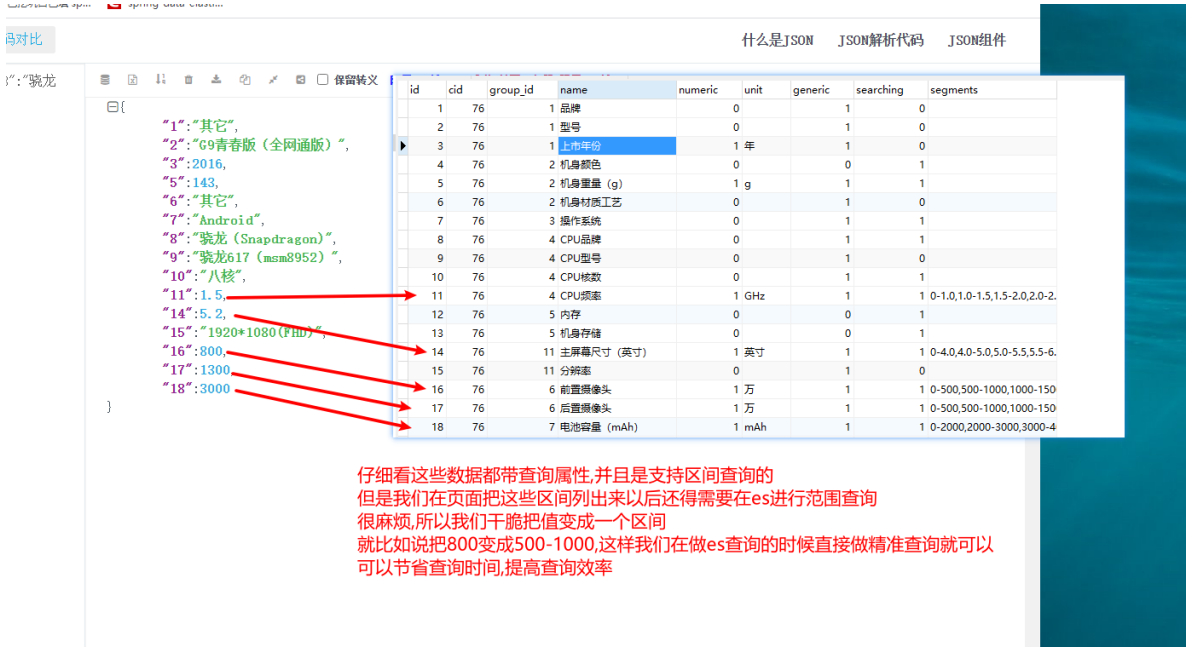
4.3.3 ShopElasticsearchServiceImpl

```
    @Override
    public Result<JSONObject> esGoodsInfo() {
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setPage(1);
        spuDTO.setRows(5);
        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(spuDTO);

        log.info("goodsFeign.getSpuInfo --> {}",spuInfo);
```

```
        if (spuInfo.getCode() == 200) {

            List<GoodsDoc> docList = spuInfo.getData().stream().map(spu -> {

                Integer spuId = spu.getId();

                GoodsDoc goodsDoc = new GoodsDoc();
                //spu信息填充
                goodsDoc.setId(spuId.longValue());
                goodsDoc.setCid1(spu.getCid1().longValue());
                goodsDoc.setCid2(spu.getCid2().longValue());
                goodsDoc.setCid3(spu.getCid3().longValue());
                goodsDoc.setCreateTime(spu.getCreateTime());
                goodsDoc.setSubTitle(spu.getSubTitle());
                //可搜索的数据
                goodsDoc.setTitle(spu.getTitle());
                goodsDoc.setBrandName(spu.getBrandName());
                goodsDoc.setCategoryName(spu.getCategoryName());


                //sku数据填充
                Result<List<SkuDTO>> skuResult =
goodsFeign.getSkuBySpuId(spuId);
                if(skuResult.getCode() == 200){
                    List<SkuDTO> skuList = skuResult.getData();

                    List<Long> priceList = new ArrayList<>();

                    List<Map<String, Object>> skuListMap =
skuList.stream().map(sku -> {
                        Map<String, Object> map = new HashMap<>();
                        map.put("id", sku.getId());
                        map.put("title", sku.getTitle());
                        map.put("image", sku.getImages());
                        map.put("price", sku.getPrice());

                        priceList.add(sku.getPrice().longValue());

                        return map;
                    }).collect(Collectors.toList());

                    goodsDoc.setPrice(priceList);
                    goodsDoc.setSkus(JSONUtil.toJsonString(skuListMap));
                }

                //规格数据填充
                //获取规格参数
                SpecParamDTO specParamDTO = new SpecParamDTO();
                specParamDTO.setCid(spu.getCid3());
                specParamDTO.setSearching(1);//只查询是为查询属性的规格参数
                Result<List<SpecParamEntity>> specParamInfo =
specificationFeign.getSpecParamInfo(specParamDTO);

                if(specParamInfo.getCode() == 200){

                    //获取Spudetail数据
                    Result<SpuDetailEntity> spuDetailBydSpu =
goodsFeign.getSpuDetailBydSpu(spuId);
```

```
                    if(spuDetailBydSpu.getCode() == 200){

                        SpuDetailEntity spuDetail = spuDetailBydSpu.getData();

                        //将通用规格转换为Map
                        String genericSpec = spuDetail.getGenericSpec();
                        Map<String, String> genericSpecMap =
JSONUtil.toMapValueString(genericSpec);

                        //特殊规格转换为map,值为List,因为有可能会有多个例如:颜色
                        String specialSpec = spuDetail.getSpecialSpec();
                        Map<String, List<String>> specialSpecMap =
JSONUtil.toMapValueStrList(specialSpec);

                        //存放数据的map集合
                        Map<String, Object> specs = new HashMap<>();
                        specParamInfo.getData().stream().forEach(specParam -> {

                            //将对应的规格名称和值放到Map集合中
                            if(specParam.getGeneric() == 1){//通用规格


 specs.put(specParam.getName(),genericSpecMap.get(specParam.getId() + ""));

                            }else{//特殊规格

 specs.put(specParam.getName(),specialSpecMap.get(specParam.getId() + ""));
                            }
                        });

                        goodsDoc.setSpecs(specs);
                    }
                }

                return goodsDoc;
            }).collect(Collectors.toList());

            log.info("docListInfo --> {}",docList);
        }

        return null;
    }
```

4.3 数值范围数据处理



处理带有范围的属性值

```
    @Override
    public Result<JSONObject> esGoodsInfo() {
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setPage(1);
        spuDTO.setRows(5);
        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(spuDTO);

        log.info("goodsFeign.getSpuInfo --> {}",spuInfo);

        if (spuInfo.getCode() == 200) {

            List<GoodsDoc> docList = spuInfo.getData().stream().map(spu -> {

                Integer spuId = spu.getId();

                GoodsDoc goodsDoc = new GoodsDoc();
                //spu信息填充
                goodsDoc.setId(spuId.longValue());
                goodsDoc.setCid1(spu.getCid1().longValue());
                goodsDoc.setCid2(spu.getCid2().longValue());
                goodsDoc.setCid3(spu.getCid3().longValue());
                goodsDoc.setCreateTime(spu.getCreateTime());
                goodsDoc.setSubTitle(spu.getSubTitle());
                //可搜索的数据
                goodsDoc.setTitle(spu.getTitle());
                goodsDoc.setBrandName(spu.getBrandName());
                goodsDoc.setCategoryName(spu.getCategoryName());


                //sku数据填充
                Result<List<SkuDTO>> skuResult =
goodsFeign.getSkuBySpuId(spuId);
                if(skuResult.getCode() == 200){
                    List<SkuDTO> skuList = skuResult.getData();

                    List<Long> priceList = new ArrayList<>();
```

```
                    List<Map<String, Object>> skuListMap =
skuList.stream().map(sku -> {
                        Map<String, Object> map = new HashMap<>();
                        map.put("id", sku.getId());
                        map.put("title", sku.getTitle());
                        map.put("image", sku.getImages());
                        map.put("price", sku.getPrice());

                        priceList.add(sku.getPrice().longValue());

                        return map;
                    }).collect(Collectors.toList());

                    goodsDoc.setPrice(priceList);
                    goodsDoc.setSkus(JSONUtil.toJsonString(skuListMap));
                }

                //规格数据填充
                //获取规格参数
                SpecParamDTO specParamDTO = new SpecParamDTO();
                specParamDTO.setCid(spu.getCid3());
                specParamDTO.setSearching(1);
                Result<List<SpecParamEntity>> specParamInfo =
specificationFeign.getSpecParamInfo(specParamDTO);

                if(specParamInfo.getCode() == 200){

                    //获取Spudetail数据
                    Result<SpuDetailEntity> spuDetailBydSpu =
goodsFeign.getSpuDetailBydSpu(spuId);
                    if(spuDetailBydSpu.getCode() == 200){

                        SpuDetailEntity spuDetail = spuDetailBydSpu.getData();

                        //将通用规格转换为Map
                        String genericSpec = spuDetail.getGenericSpec();
                        Map<String, String> genericSpecMap =
JSONUtil.toMapValueString(genericSpec);

                        //特殊规格转换为map,值为List,因为有可能会有多个例如:颜色
                        String specialSpec = spuDetail.getSpecialSpec();
                        Map<String, List<String>> specialSpecMap =
JSONUtil.toMapValueStrList(specialSpec);

                        //遍历map数据
                        Map<String, Object> specs = new HashMap<>();
                        specParamInfo.getData().stream().forEach(specParam -> {

                            //将对应的规格名称和值放到Map集合中
                            if(specParam.getGeneric() == 1){//通用规格

                                String value =
genericSpecMap.get(specParam.getId() + "");
                                //将具体的值变为区间对应页面的范围
                                if(specParam.getNumeric() == 1){


 if(!StringUtils.isEmpty(specParam.getSegments())){
```

```
                                        //将值变成区间
                //防止单位为null的情况
                                        value =
this.chooseSegment(value,specParam.getSegments(),specParam.getUnit() ==
null?"":specParam.getUnit());
                                    }
                                }

                                specs.put(specParam.getName(),value);

                            }else{//特殊规格

 specs.put(specParam.getName(),specialSpecMap.get(specParam.getId() + ""));
                            }
                        });

                        goodsDoc.setSpecs(specs);
                    }
                }

                return goodsDoc;
            }).collect(Collectors.toList());

            log.info("docListInfo --> {}",docList);
        }

        return null;
    }
```

```
    private String chooseSegment(String value, String segments, String unit) {
        double val = NumberUtils.toDouble(value);
        String result = "其它";
        // 保存数值段
        for (String segment : segments.split(",")) {
            String[] segs = segment.split("-");
            // 获取数值范围
            double begin = NumberUtils.toDouble(segs[0]);
            double end = Double.MAX_VALUE;
            if(segs.length == 2){
                end = NumberUtils.toDouble(segs[1]);
            }
            // 判断是否在范围内
            if(val >= begin && val < end){
                if(segs.length == 1){
                    result = segs[0] + unit + "以上";
                }else if(begin == 0){
                    result = segs[1] + unit + "以下";
                }else{
                    result = segment + unit;
                }
                break;
            }
        }
        return result;
    }
```

至此所有的查询已经全部完毕,但是查询的函数已经太多代码

4.4 查询函数拆分

拆函数的时候尽量倒着拆

```
import com.alibaba.fastjson.JSONObject;
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.document.GoodsDoc;
import com.baidu.shop.dto.SkuDTO;
import com.baidu.shop.dto.SpecParamDTO;
import com.baidu.shop.dto.SpuDTO;
import com.baidu.shop.entity.SpecParamEntity;
import com.baidu.shop.entity.SpuDetailEntity;
import com.baidu.shop.feign.GoodsFeign;
import com.baidu.shop.feign.SpecificationFeign;
import com.baidu.shop.service.ShopElasticsearchService;
import com.baidu.shop.utils.JSONUtil;
import lombok.extern.slf4j.Slf4j;
import org.apache.commons.lang3.math.NumberUtils;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.RestController;

import java.util.*;
import java.util.stream.Collectors;

/**
 * @ClassName ShopElasticsearchServiceImpl
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/5
 * @Version V1.0
 **/
@RestController
@Slf4j
public class ShopElasticsearchServiceImpl extends BaseApiService implements
ShopElasticsearchService {

    @Autowired
    private GoodsFeign goodsFeign;

    @Autowired
    private SpecificationFeign specificationFeign;

    @Override
    public Result<JSONObject> esGoodsInfo() {//此函数只是单纯的获取数据商品数据,对用户
是不可见的
        //分页的信息肯定不能写死,所以让调用方传递过来数据即可
        SpuDTO spuDTO = new SpuDTO();
        spuDTO.setPage(1);
        spuDTO.setRows(5);
        Result<List<SpuDTO>> spuInfo = goodsFeign.getSpuInfo(spuDTO);

        log.info("goodsFeign.getSpuInfo --> {}",spuInfo);

        if (spuInfo.getCode() == 200) {
```

```
            List<GoodsDoc> docList = spuInfo.getData().stream().map(spu -> {

                Integer spuId = spu.getId();
                GoodsDoc goodsDoc = new GoodsDoc();
                //spu信息填充
                goodsDoc.setId(spuId.longValue());
                goodsDoc.setCid1(spu.getCid1().longValue());
                goodsDoc.setCid2(spu.getCid2().longValue());
                goodsDoc.setCid3(spu.getCid3().longValue());
                goodsDoc.setCreateTime(spu.getCreateTime());
                goodsDoc.setSubTitle(spu.getSubTitle());
                //可搜索的数据
                goodsDoc.setTitle(spu.getTitle());
                goodsDoc.setBrandName(spu.getBrandName());
                goodsDoc.setCategoryName(spu.getCategoryName());

                //sku数据填充
                List<Map<String, Object>> skuList = this.skus(spu.getId());

                //价格在skuList里面
                //理论上不应该这么做,性能受影响
                //但是这个理论上用户不会操作的
                goodsDoc.setPrice(this.getPrices(skuList));
                goodsDoc.setSkus(JSONUtil.toJsonString(skuList));
                //规格数据填充
                //获取规格参数
                Map<String, Object> specs = this.getSpecs(spu);
                goodsDoc.setSpecs(specs);

                return goodsDoc;
            }).collect(Collectors.toList());

            log.info("docListInfo --> {}",docList);
        }

        return null;
    }

    //获取map类型的Sku信息
    private List<Map<String, Object>> skus(Integer spuId){
        Result<List<SkuDTO>> skuResult = goodsFeign.getSkuBySpuId(spuId);
        if(skuResult.getCode() == 200){
            List<SkuDTO> skuList = skuResult.getData();

            List<Map<String, Object>> skuListMap = skuList.stream().map(sku -> {

                Map<String, Object> map = new HashMap<>();
                map.put("id", sku.getId());
                map.put("title", sku.getTitle());
                map.put("image", sku.getImages());
                map.put("price", sku.getPrice());

                return map;
            }).collect(Collectors.toList());
            return skuListMap;
        }
        return null;
    }
```

```
    private List<Long> getPrices(List<Map<String, Object>> skus){

        return skus.stream().map(sku -> {
            Integer price = (Integer) sku.get("price");

            return price.longValue();
        }).collect(Collectors.toList());
    }

    //获取Spec数据
    private Map<String, Object> getSpecs(SpuDTO spuDTO){

        SpecParamDTO specParamDTO = new SpecParamDTO();
        specParamDTO.setCid(spuDTO.getCid3());
        specParamDTO.setSearching(1);
        Result<List<SpecParamEntity>> specParamInfo =
specificationFeign.getSpecParamInfo(specParamDTO);

        if(specParamInfo.getCode() == 200){
            //获取Spudetail数据
            Result<SpuDetailEntity> spuDetailBydSpu =
goodsFeign.getSpuDetailBydSpu(spuDTO.getId());
            if(spuDetailBydSpu.getCode() == 200){

                Map<String, Object> specs = this.getSpecs(spuDetailBydSpu,
specParamInfo);

                return specs;
            }
        }

        return null;
    }

    //方法重载
    private Map<String, Object> getSpecs(Result<SpuDetailEntity> spuDetailBydSpu
            , Result<List<SpecParamEntity>> specParamInfo){

        SpuDetailEntity spuDetail = spuDetailBydSpu.getData();

        //将通用规格转换为Map
        String genericSpec = spuDetail.getGenericSpec();
        Map<String, String> genericSpecMap =
JSONUtil.toMapValueString(genericSpec);

        //特殊规格转换为map,值为List,因为有可能会有多个例如:颜色
        String specialSpec = spuDetail.getSpecialSpec();
        Map<String, List<String>> specialSpecMap =
JSONUtil.toMapValueStrList(specialSpec);

        //遍历map数据
        Map<String, Object> specs = new HashMap<>();
        specParamInfo.getData().stream().forEach(specParam -> {

            //将对应的规格名称和值放到Map集合中
            if(specParam.getGeneric() == 1){//通用规格
```

```
                String value = genericSpecMap.get(specParam.getId() + "");
                //将具体的值变为区间对应页面的范围
                if(specParam.getNumeric() == 1){

                    if(!StringUtils.isEmpty(specParam.getSegments())){
                        //将值变成区间                                                    //
防止单位为null的情况
                        value =
this.chooseSegment(value,specParam.getSegments(),specParam.getUnit() ==
null?"":specParam.getUnit());
                    }
                }

                specs.put(specParam.getName(),value);
            }else{//特殊规格

 specs.put(specParam.getName(),specialSpecMap.get(specParam.getId() + ""));
            }
        });

        return specs;
    }

    private String chooseSegment(String value, String segments, String unit) {
        double val = NumberUtils.toDouble(value);
        String result = "其它";
        // 保存数值段
        for (String segment : segments.split(",")) {
            String[] segs = segment.split("-");
            // 获取数值范围
            double begin = NumberUtils.toDouble(segs[0]);
            double end = Double.MAX_VALUE;
            if(segs.length == 2){
                end = NumberUtils.toDouble(segs[1]);
            }
            // 判断是否在范围内
            if(val >= begin && val < end){
                if(segs.length == 1){
                    result = segs[0] + unit + "以上";
                }else if(begin == 0){
                    result = segs[1] + unit + "以下";
                }else{
                    result = segment + unit;
                }
                break;
            }
        }
        return result;
    }
}
```

5 mysql数据迁移到es(入库)

5.1 提供操作ESApi

5.1.1 ShopElasticsearchService

- 将查询数据的接口删除掉
- 实现类中查询数据的@Override删除掉
- 查询数据的返回值为List,注意函数需要返回数据(return)

```
    //ES数据初始化-->索引创建,映射创建,mysql数据同步
    @ApiOperation(value = "ES商品数据初始化-->索引创建,映射创建,mysql数据同步")
    @GetMapping(value = "es/initGoodsEsData")
    Result<JSONObject> initGoodsEsData();

    @ApiOperation(value = "清空ES中的商品数据")
    @GetMapping(value = "es/clearGoodsEsData")
    Result<JSONObject> clearGoodsEsData();
```

5.1.2 新建repository包

5.1.3 新建GoodsRepository

```
import com.baidu.shop.document.GoodsDoc;
import
org.springframework.data.elasticsearch.repository.ElasticsearchRepository;

/**
 * @ClassName GoodsRepository
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/9/6
 * @Version V1.0
 **/
public interface GoodsRepository extends ElasticsearchRepository<GoodsDoc,Long>
{
}
```

5.1.4 ShopElasticsearchServiceImpl

```
    @Resource
    private ElasticsearchRestTemplate elasticsearchRestTemplate;

    @Resource
    private GoodsRepository goodsRepository;

    @Override
    public Result<JSONObject> clearGoodsEsData() {

        IndexOperations index =
elasticsearchRestTemplate.indexOps(GoodsDoc.class);
        if(index.exists()){
            index.delete();
        }

        return this.setResultSuccess();
    }
```

```
    @Override
    public Result<JSONObject> initGoodsEsData() {

        //创建索引和映射
        IndexOperations index =
elasticsearchRestTemplate.indexOps(GoodsDoc.class);
        if(!index.exists()){
            index.create();
            index.createMapping();
        }

        //查询数据
        List<GoodsDoc> goodsDocs = this.esGoodsInfo();
        //将得到的结果入库
        goodsRepository.saveAll(goodsDocs);
        //elasticsearchRestTemplate.save(goodsDocs);
        return this.setResultSuccess();
    }
```

5.2 mysql数据库全部迁移

上述入库操作也只是将部分数据入库了而已,在真实的环境中我们需要考虑实际情况来进行数据迁移

查询全部mysql数据肯定是不合适的(数据库压力太大)

一次查询五条数据????(这样会增加数据库和es服务的io)

所以得考虑硬件条件 以及 mysql 和 es的性能

当前我们是学习阶段,数据量不是很大,所以可以直接将mysql的数据全部查询出来,然后全部入库

5.2.1 initGoodsEsData方法

```
    @Override
    public Result<JSONObject> initGoodsEsData() {

        //创建索引和映射
        IndexOperations index =
elasticsearchRestTemplate.indexOps(GoodsDoc.class);
        if(!index.exists()){
            index.createMapping();
        }
        List<GoodsDoc> goodsDocs = this.esGoodsInfo();
        //将得到的结果入库
        if(!goodsDocs.isEmpty()){
            goodsRepository.saveAll(goodsDocs);
        }

        return this.setResultSuccess();
    }
```

5.2.2 esGoodsInfo方法

将分页注释掉

5.2.3 重置es数据

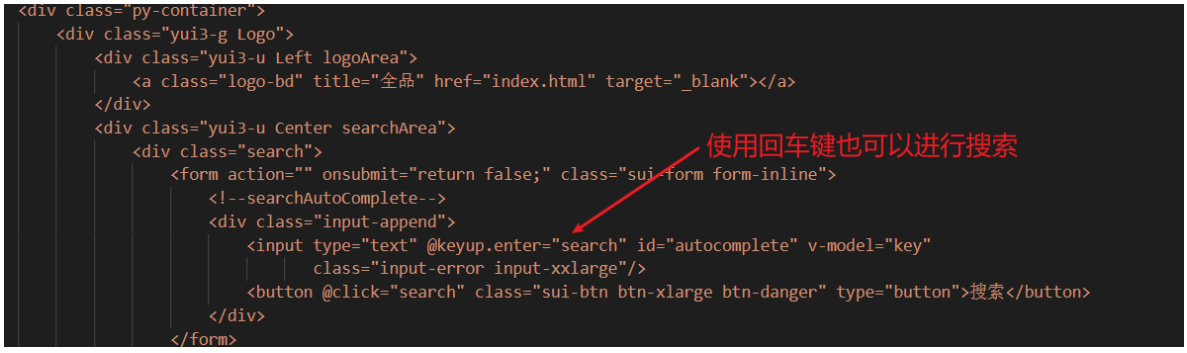
swagger-ui调用clearGoodsEsData接口清空掉es数据

swagger-ui调用initGoodsEsData重新初始化数据

6 搜索

6.1 top.js

top.html并没有什么用(在top.html写完代码后直接将这个html代码复制到了top.js的template中)



> 我们将template属性的值改一下:将top.html中的html代码复制到template的值中,需要注意的是:不要忘记加`<shortcut/>`标签

```
const b2cTop = {
    template: `<div class="nav-top">
    <!--这个千万不能忘了加-->
    <shortcut/>
    <!--页面顶部白条条，由js动态加载-->

    <!--头部-->
    <div class="header" id="headApp">
```

```
        <div class="py-container">
            <div class="yui3-g Logo">
                <div class="yui3-u Left logoArea">
                    <a class="logo-bd" title="全品" href="index.html"
target="_blank"></a>
                </div>
                <div class="yui3-u Center searchArea">
                    <div class="search">
                        <!--
                        onsubmit="return false;" 控制表单不提交
                        解决点击回车页面刷新的问题
                        -->
                        <form onsubmit="return false;" class="sui-form form-
inline">
                            <!--searchAutoComplete-->
                            <div class="input-append">
                                <!--type="text"属性重复的问题
                                @keyup.enter="search"回车也可以进行搜索
                                -->
                                <input @keyup.enter="search" id="autocomplete"
type="text" v-model="key"
                                       class="input-error input-xxlarge"/>
                                <button @click="search" class="sui-btn btn-
xlarge btn-danger" type="button">搜索</button>
                            </div>
                        </form>
                    </div>
                    <div class="hotwords">
                        <ul>
                            <li class="f-item">全品首发</li>
                            <li class="f-item">亿元优惠</li>
                            <li class="f-item">9.9元团购</li>
                            <li class="f-item">每满99减30</li>
                            <li class="f-item">亿元优惠</li>
                            <li class="f-item">9.9元团购</li>
                            <li class="f-item">办公用品</li>

                        </ul>
                    </div>
                </div>
                <div class="yui3-u Right shopArea">
                    <div class="fr shopcar">
                        <div class="show-shopcar" id="shopcar">
                            <span class="car"></span>
                            <a class="sui-btn btn-default btn-xlarge"
href="cart.html" target="_blank">
                                <span>我的购物车</span>
                                <i class="shopnum">0</i>
                            </a>
                            <div class="clearfix shopcarlist" id="shopcarlist"
style="display:none">
                                <p>"啊哦，你的购物车还没有商品哦！"</p>
                                <p>"啊哦，你的购物车还没有商品哦！"</p>
                            </div>
                        </div>
                    </div>
                </div>
            </div>
```

```
            <div class="yui3-g NavList">
                <div class="yui3-u Left all-sort">
                    <h4>全品精品</h4>
                </div>
                <div class="yui3-u Center navArea">
                    <ul class="nav">
                        <li class="f-item">服装城</li>
                        <li class="f-item">美妆馆</li>
                        <li class="f-item">品优超市</li>
                        <li class="f-item">全球购</li>
                        <li class="f-item">闪购</li>
                        <li class="f-item">团购</li>
                        <li class="f-item">有趣</li>
                        <li class="f-item"><a href="seckill-index.html"
target="_blank">秒杀</a></li>
                    </ul>
                </div>
                <div class="yui3-u Right"></div>
            </div>
        </div>
    </div>
    </div>`,
    name:'b2c-top',
    data() {
        return {
            key: "",
            query: location.search
        }
    },
    methods: {
        search() {
            window.location = 'search.html?key=' + this.key;
        },
        getUrlParam: function (name) {
            var reg = new RegExp("(^|&)" + name + "=([^&]*)(&|$)", "i");
            var r = window.location.search.substr(1).match(reg);
            if (r != null) {
                return decodeURI(r[2]);
            }
            return null;
        }
    },
    created() {
        this.key = this.getUrlParam("key");
    },
    components: {
        shortcut:() => import('./shortcut.js')
    }
}
export default b2cTop;
```

6.2 search.html

```
<script type="text/javascript">
    var vm = new Vue({
        el: "#searchApp",
        data: {
            goodsList:[],
            mrshop
        },
        components:{
            b2cTop: () => import("./js/pages/top.js")

        },
        //勾子函数
        created () {
            //获取url路径上的内容
            const search = mrshop.parse(location.search.substring(1))
            mrshop.http.get('/es/search',{
                params:{
                    search:search.key
                }
            }).then(resp => {

                const goodsList = resp.data.data.map(goods => {
                    goods.skus = JSON.parse(goods.skus);
                    goods.selected = goods.skus[0];
                    return goods;
                })

                this.goodsList = goodsList;
            }).catch(error => console.log(error));
        }
    });
```

引入common.js中的mrshop对象

获取url中?后面中的内容
并将其转换为json格式

将返回数据中的skus转换成json数据

设置第一个默认选中

```
    <div class="goods-list">
        <ul class="yui3-g">
            <li class="yui3-u-1-5" v-for="(goods,index) in goodsList" :key="index">
                <div class="list-wrap">
                    <div class="p-img">
                        <a href="item.html" target="_blank"><img :src="goods.selected.image" height="200"/></a>
                        <ul class="skus">
                            <li :class="{selected: sku.id == goods.selected.id}"
                            @click="goods.selected=sku"
                            v-for="(sku,index) in goods.skus"
                            :key="index">
                                <img :src="sku.image">
                            </li>
                            <!-- <li class=""><img src="img/_/mobile01.png"></li>
                            <li class=""><img src="img/_/mobile01.png"></li> -->
                        </ul>
                    </div>
                    <div class="clearfix"></div>
                    <div class="price">
                        <strong>
                            <em>¥</em>
                            <i>{{ mrshop.formatPrice(goods.selected.price) }}</i>
                        </strong>
                    </div>
                    <div class="attr">
                        <em v-html="goods.title"></em>
                    </div>
                    <div class="cu">
                        <em><span>促</span>满一件可参加超值换购</em>
                    </div>
                    <div class="commit">
                        <i class="command">已有2000人评价</i>
                    </div>
```

遍历数据

显示当前商品显示的sku

设置选中样式

点击图片显示大图

遍历所有的sku

必须使用v-html

```
            <div class="goods-list">
                <ul class="yui3-g">
                    <li class="yui3-u-1-5" v-for="(goods,index) in goodsList"
:key="index">
                        <div class="list-wrap">
                            <div class="p-img">

                                <!--大图片 设置成当前选中的图片-->
                                <a href="item.html" target="_blank"><img
:src="goods.selected.image" height="200"/></a>

                                <!--设置图片-->
                                <ul class="skus">
                                    <!--判断当前sku是否是选中的数据-->
```

```
                                    <li :class="{selected: sku.id ==
goods.selected.id}"
                                    @click="goods.selected=sku"
                                    v-for="(sku,index) in goods.skus"
                                    :key="index">
                                        <img :src="sku.image">
                                    </li>
                                    <!-- <li class=""><img
src="img/_/mobile01.png"></li>
                                    <li class=""><img src="img/_/mobile01.png">
</li> -->
                                </ul>
                            </div>
                            <div class="clearfix"></div>
                            <div class="price">
                                <strong>
                                    <em>¥</em>
                                    <!--将分转为元-->
                                    <i>{{
mrshop.formatPrice(goods.selected.price) }}</i>
                                </strong>
                            </div>
                            <div class="attr">
                                <!--注意:此处不能使用{{}}-->
                                <em v-html="goods.title"></em>
                            </div>
                            <div class="cu">
                                <em><span>促</span>满一件可参加超值换购</em>
                            </div>
                            <div class="commit">
                                <i class="command">已有2000人评价</i>
                            </div>
                            <div class="operate">
                                <a href="success-cart.html" target="_blank"
class="sui-btn btn-bordered btn-danger">加入购物车</a>
                                <a href="javascript:void(0);" class="sui-btn
btn-bordered">对比</a>
                                <a href="javascript:void(0);" class="sui-btn
btn-bordered">关注</a>
                            </div>
                        </div>
                    </li>
                </ul>
            </div>
```

```
<script type="text/javascript">
    var vm = new Vue({
        el: "#searchApp",
        data: {
            goodsList:[],//商品信息
            mrshop//不将此参数绑定到当前页面上,html代码不能使用mrshop.***函数
        },
        components:{
            b2cTop: () => import("./js/pages/top.js")
        },
        created () {
```

```
            const search = mrshop.parse(location.search.substring(1));//将
key=value转为key:value
            //查询请求
            mrshop.http.get('es/search',{
                params:{
                    search:search.key//将查询的内容传递到后台
                }
            }).then(resp => {
                //处理sku
                const goodsList = resp.data.data.map(goods => {

                    goods.skus = JSON.parse(goods.skus);//将字符串转为json
                    goods.selected = goods.skus[0];//设置当前选中
                    return goods;
                })

                this.goodsList = goodsList;

            }).catch(error => console.log(error))
        }
    });
</script>
```

6.2 ShopElasticsearchService

```
    @ApiOperation(value = "搜索")
    @GetMapping(value = "es/search")
    Result<List<GoodsDoc>> search(@RequestParam String search);
```

6.3 ShopElasticsearchServiceImpl

> 1：需要根据标题,分类名称,品牌名称进行查询
>
> 2：需要设置高亮字段

```
    @Override
    public Result<List<GoodsDoc>> search(String search) {

        NativeSearchQueryBuilder nativeSearchQueryBuilder = new
NativeSearchQueryBuilder();
        if(StringUtils.isEmpty(search)) return this.setResultError("搜索内容不能为
空");
        //多字段同时查询

 nativeSearchQueryBuilder.withQuery(QueryBuilders.multiMatchQuery(search,"title"
,"brandName","categoryName"));
        //设置高亮字段

 nativeSearchQueryBuilder.withHighlightBuilder(ESHighLightUtil.getHighlightBuild
er("title"));

        SearchHits<GoodsDoc> searchHits =
elasticsearchRestTemplate.search(nativeSearchQueryBuilder.build(),
GoodsDoc.class);
```

```
        List<SearchHit<GoodsDoc>> highLightHit =
ESHighLightUtil.getHighLightHit(searchHits.getSearchHits());
        List<GoodsDoc> goodsDocs = highLightHit.stream().map(hit ->
hit.getContent()).collect(Collectors.toList());

        return this.setResultSuccess(goodsDocs);
    }
```

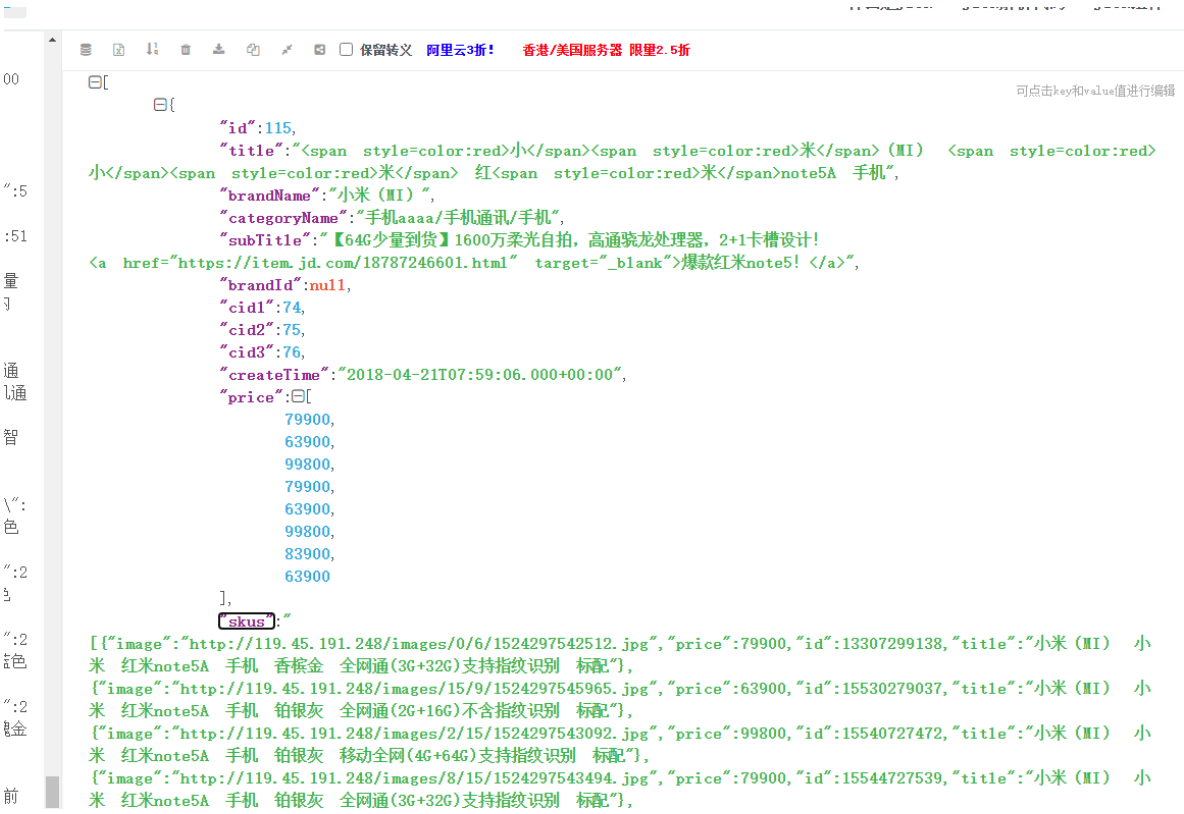
6.4 mingrui-shop-basic-zuul-serer/application.yml

> 将search项目加入到zuul路由中

```
server:
  port: 8088

spring:
  application:
    name: eureka-zuul

zuul:
  # 声明路由
  routes:
    # 路由名称
    api-xxx:
      # 声明将所有以/api-ribbon/的请求都转发到eureka-ribbon的服务中
      path: /api-xxx/**
      serviceId: xxx-service
    # 搜索路由
    api-search:
      path: /api-search/**
      serviceId: search-server
  # 启用重试
  retryable: true
  # 包含此路径的不进行路由
  ignored-patterns: /upload/**
  # 忽略上传服务
  ignored-services:
    -upload-server
#配置负载
ribbon:
  ConnectTimeout: 250 # 连接超时时间(ms)
  ReadTimeout: 2000 # 通信超时时间(ms)
  OkToRetryOnAllOperations: true # 是否对所有操作重试
  MaxAutoRetriesNextServer: 2 # 同一服务不同实例的重试次数
  MaxAutoRetries: 1 # 同一实例的重试次数

hystrix:
  command:
    default:
      execution:
        isolation:
          thread:
            timeoutInMilliseconds: 10000 # 熔断超时时长：6000ms

eureka:
  client:
```

```
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

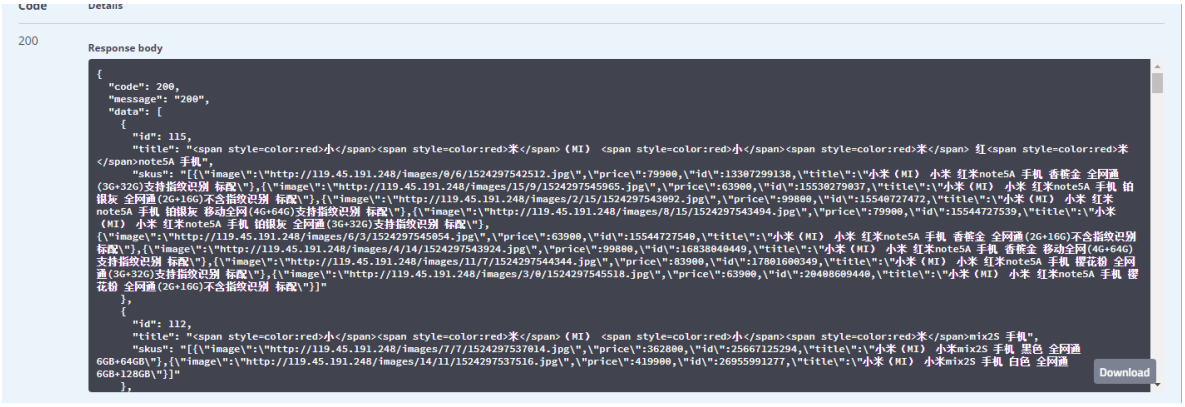
6.5 一点小小的问题



后台返回了好多没有用的数据,为了降低网络io的压力,所以我们处理一下

```
        //设置查询出来的内容,页面上做多只需要id,title,skus
        queryBuilder.withSourceFilter(new FetchSourceFilter(new String[]
{"id","title","skus"}, null));
```


```
spring:
  jackson:
    default-property-inclusion: non_null #空值不返回
```



7 搜索分页

7.1 search.html


```
                    <ul>
                        <!--处理上一页-->
                        <!--
                            prev:true 这个属性的值保持不变
                            disabled:page <= 1 当前页 <= 1的时候禁用上一页按钮
                            @click="page--" 点击上一页的时候当前页 - 1
                        -->
                        <li :class="{prev:true,disabled:page <= 1}"
@click="page--">
                            <!--
                                click.prevent="shopDefaultEvent" 阻止默认事件的发生
                                shopDefaultEvent为空函数
                            -->
                            <a href="#" @click.prevent="shopDefaultEvent" >«上一
页</a>
                        </li>
                        <!--页码-->
                        <!--
                            active:p==page 遍历页码 == 当前页此样式生效
                            @click="page=p" 点击页面的时候 设置当前页=当前点击页码
                            Math.min(3,totalPage) 参数A和参数B 返回值最小的
                        -->
                        <li :class="{active:p==page}" @click="page=p" v-for="
(p,index) in Math.min(3,totalPage)" :key="index">
                            <a href="#" @click.prevent="shopDefaultEvent">{{ p
}}</a>
```

```
                        </li>
                        <!--...是否显示-->
                        <!--
                            v-show="totalPage > 3" 总页数 > 3的时候显示...
                        -->
                        <li class="dotted" v-show="totalPage > 3"><span>...
</span></li>
                        <!--处理下一页-->
                        <li class="next" @click="page++">
                            <a href="#" @click.prevent="shopDefaultEvent">下一
页»</a>
                        </li>
                    </ul>
                    <div><span>共{{ totalPage }}页 </span><span>
```

```
<script type="text/javascript">
    var vm = new Vue({
        el: "#searchApp",
        data: {
            goodsList: [],
            mrshop,
            total: 0,
            totalPage: 0,
            page: 1
        },
        watch: {
            page() {
                this.search();
            }
        },
```

```
methods:{
    shopDefaultEvent () {
    },
    search () {
        const search = mrshop.parse(location.search.substring(1))
        mrshop.http.get('/es/search',{
            params:{
                search:search.key,
                page:this.page
            }
        }).then(resp => {

            const goodsList = resp.data.data.map(goods => {
                goods.skus = JSON.parse(goods.skus);
                goods.selected = goods.skus[0];
                return goods;
            })

            const msgJsonObj = JSON.parse(resp.data.message);
            this.total = msgJsonObj.total;
            this.totalPage = msgJsonObj.totalPage;

            this.goodsList = goodsList;
        }).catch(error => console.log(error));
    }
},
//勾子函数
created () {
    //获取url路径上的内容
    this.search();
}
```

```
            //发送搜索请求
            //axios
            mrshop.http.get('es/search', {
                params: {
                    search,
                    page: this.page
                }
            }).then(resp => {
                //判断code的值是否==200

                if (resp.data.code == 200) {
                    const goodsList = resp.data.data.map(goods => {
                        goods.skus = JSON.parse(goods.skus);
                        goods.selected = goods.skus[0];
                        return goods;
                    });

                    const msgJsonObj = JSON.parse(resp.data.message);
                    this.total = msgJsonObj.total;
                    this.totalPage = msgJsonObj.totalPage;

                    this.goodsList = goodsList;
                }else{}


            }).catch(error => console.log(error))
        }
```

```
var vm = new Vue({
    el: "#searchApp",
    data: {
        goodsList: [],
        mrshop,
        total: 0,
        totalPage: 0,
        page: 1
```

```
            },
            watch: {
                page() {
                    this.search();
                }
            },
            created() {
                this.search();
            },
            methods: {
                shopDefaultEvent() { },
                search() {
                    if (this.page < 1) {
                        return;
                    }
                    //得到要搜索的内容
                    const search = mrshop.getUrlParam('key');
                    //发送搜索请求
                    //axios
                    mrshop.http.get('es/search', {
                        params: {
                            search,
                            page: this.page
                        }
                    }).then(resp => {
                        //判断code的值是否==200

                        if (resp.data.code == 200) {
                            const goodsList = resp.data.data.map(goods => {
                                goods.skus = JSON.parse(goods.skus);
                                goods.selected = goods.skus[0];
                                return goods;
                            });

                            const msgJsonObj = JSON.parse(resp.data.message);
                            this.total = msgJsonObj.total;
                            this.totalPage = msgJsonObj.totalPage;

                            this.goodsList = goodsList;
                        }else{}


                    }).catch(error => console.log(error))
                }
            },
            components: {
                b2cTop: () => import("./js/pages/top.js")
            }
        });
```

7.2 ShopElasticsearchService

```
    @ApiOperation(value = "搜索")
    @GetMapping(value = "es/search")
    Result<List<GoodsDoc>> search(@RequestParam String search,@RequestParam
Integer page);
```

7.3 ShopElasticsearchServiceImpl

```
    @Override
    public Result<List<GoodsDoc>> search(String search, Integer page) {

        NativeSearchQueryBuilder queryBuilder = new NativeSearchQueryBuilder();

        if (!StringUtils.isEmpty(search)) {
            //多字段同时查询

 queryBuilder.withQuery(QueryBuilders.multiMatchQuery(search,"title","brandName"
,"categoryName"));
        }

        queryBuilder.withPageable(PageRequest.of(page-1,10));

        //设置查询出来的内容,页面上做多只需要id,title,skus
        queryBuilder.withSourceFilter(new FetchSourceFilter(new String[]
{"id","title","skus"}, null));
        //设置高亮字段

 queryBuilder.withHighlightBuilder(ESHighLightUtil.getHighlightBuilder("title"))
;

        SearchHits<GoodsDoc> hits =
elasticsearchRestTemplate.search(queryBuilder.build(), GoodsDoc.class);

        List<SearchHit<GoodsDoc>> highLightHit =
ESHighLightUtil.getHighLightHit(hits.getSearchHits());

        List<GoodsDoc> goodsDocs = highLightHit.stream().map(searchHit ->
searchHit.getContent()).collect(Collectors.toList());

//         long total = hits.getTotalHits();//总条数 47
//         Double totalD = Long.valueOf(total).doubleValue();//doule类型的总条数
//         double totalPageD = Math.ceil(totalD ;//如果有小数直接想上取整
//         int totalPage = Double.valueOf(totalPageD).intValue();//将double类型的值
转为int类型

        Map<String, Integer> map = new HashMap<>();
        map.put("total",Long.valueOf(hits.getTotalHits()).intValue());

 map.put("totalPage",Double.valueOf(Math.ceil(Long.valueOf(hits.getTotalHits()).
doubleValue()/ 10)).intValue());

        String message = JSONUtil.toJsonString(map);

        return this.setResult(HTTPStatus.OK,message,goodsDocs);
    }
```

当然,我知道这个页面代码有很多bug,我以流程为主,你们自己解决就可以了.....