三范式 Vs反三范式.

student id name cid

user id classname

selcet * from student,user where sutdent.cid=user.id;

需要内联查.会产生迪尔卡级.就是好多重复数据.

还需要mysql处理完之后在进行重复数据的筛选.

会降低性能.

优化 --> 直接将class班级名称写到一张表中.可以优化查询效率不需要再多关联表.

studen id name cid classname;

前台新学的函数.map返回新的数组.

!!0== fasle !!-1==true;

后端分页插件.

Pagehelder(起始参数.每页数据)

PageInfo封装结果集 并返回，可以通过PageInfo返回总条数.

批量新增extends insertListMapper ;

调用PageHelder.insertList<>

也可以调用Example对象进行批量新增.

toCharArray() String的方法把字符串转换为char类型数组.

.valueOf()返回包装数据类型..parse返回的是基本数据类型.

sterm().forEch()

sterm().map();

一,电商

1.架构：

前端:vue+vuetify+iview+element-ui

本地配置host环境.

前端请求后台需要经过nginx代理.

配置了.mange.mrshop.com 监控80端口 转发到127.0.0.1:9001前端web项目

配置了.api.mrshop.com 转发到127.0.0.1:8088

转发到zuul.zuul负责将请求到路由指定的server

父级项目:

baidu-shop-parent

baidu-shop-basics

baidu-shop-erurka-server

erurka-server:用来提供服务注册中心

baidu-shop-upload-server

upload-server:mvc图片上传

重要点:1:解决跨域问题.

2:调用mvc的上传功能.file.transferTo(图片的路径);//上传

baidu-shop-zuul-server

zuul-server:

解决跨域名访问.通过CorsConfiguration对象.

zuul:负责路由 请求过滤.

ribbon:负载均衡.

hystrix:负责熔断

baidu-shop-commons

baidu-shop-common-core

base:公共的返回规范.

dto:公共的分页类.

global:全局异常处理

@RestControllerAdvice// 加在类上表示请求拦截 类似于aop afterpostProcess增强方法.

@ExceptionHandler(value= MethodArgumentNotValidException.class)加在方法上表示要拦截的是那种类型的异常

utils:公共的工具类

Json转换工具类

转换大小写拼音工具类.

公共验证标识符

baidu-shop-servers

baidu-shop-category-server 接口的实现类 控制层/业务层/持久层.

category-server:分类管理.

baidu-shop-brand-server

brand-server:品牌管理

baidu-shop-server-api :api方法 swaager2配置

baidu-shop-categroy-api

category-api:存放分类管理的接口.

baidu-shop-brand-api

brand-api:存放品牌管理的接口

2.后台

2.1后台管理

商品管理：商品分类，品牌，商品规格等

销售管理：订单统计，订单退款，促销活动

用户管理：用户控制，冻结，解冻等

权限管理：整个项目的权限控制

2.1.2后台技术

SpringMVC，Spring，Mybatis，SpringBoot，SpringCloud

Redis，RabbitMQ，ES（Elasticsearch），nginx，FastDFS，MyCat，Thymeleaf，JWT

3.前台

3.1前台管理

搜索商品，购物车，订单，评论

3.2前台技术

HTML，CSS，Java，JQuery

Vuetify，WebPack（构建工具），NPM（安装包工具）,Vue-cli（脚手架）,vue-router（路由），axios，quil-editor

##

4.开发环境

JDK（JDK1.8）+IDE（idea）+maven+git

5.域名

我们在开发的过程中，为了保证以后的生产、测试环境统一。尽量都采用域名来访问项目。

一级域名：www.mrshop.com

二级域名：manage.mrshop.com , api.mrshop.com

6.项目结构

- **mingrui-shop-parent**

mingrui-shop-basics

    mingrui-shop-config-server

    mingrui-shop-eureka-server 8761

    mingrui-shop-zuul-server 8088 -->

    mingrui-shop-upload-server 8200

- **mingrui-shop-commons**

  mingrui-shop-common-core

- **mingrui-shop-services**

  mingrui-shop-service-categoty

  mingrui-shop-service-brand

  mingrui-shop-service-spec

- **mingrui-shop-services-api**

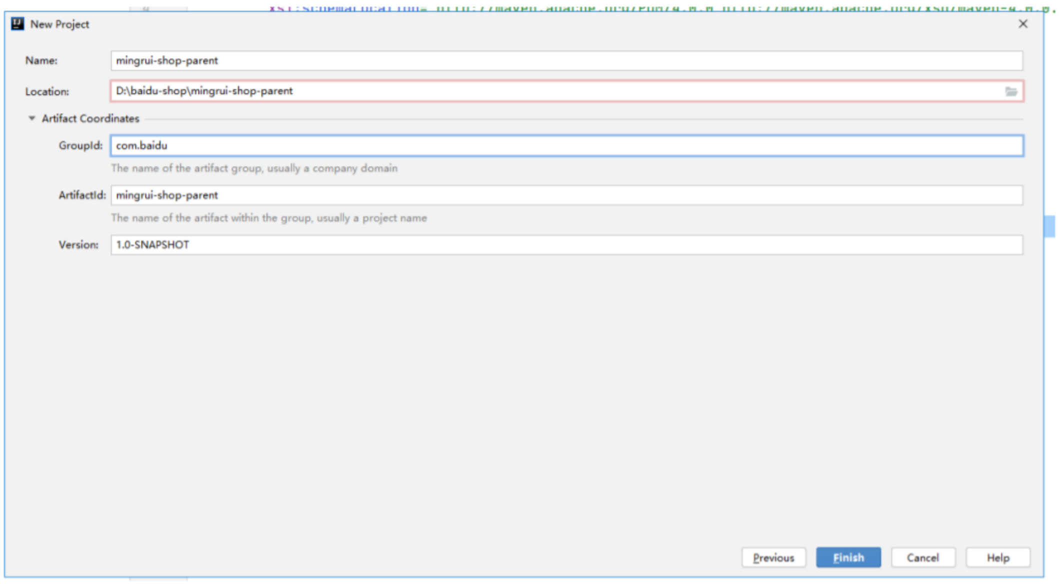
  mingrui-shop-service-api-categoty

  mingrui-shop-service-api-brand

  mingrui-shop-service-api-spec

7.创建项目

搭建父类项目



New Project

Name: mingrui-shop-parent

Location: D:\baidu-shop\mingrui-shop-parent

Artifact Coordinates

GroupId: com.baidu

The name of the artifact group, usually a company domain

ArtifactId: mingrui-shop-parent

The name of the artifact within the group, usually a project name

Version: 1.0-SNAPSHOT

删除src文件夹

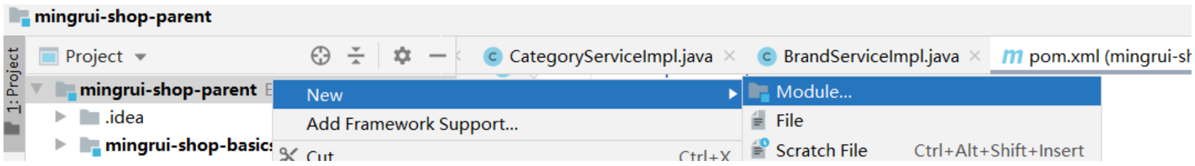
pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.baidu</groupId>
    <artifactId>mingrui-shop-parent</artifactId>
    <version>1.0-SNAPSHOT</version>

    <!--父级项目不需要打包所有packging的类型为pom-->
    <packaging>pom</packaging>

    <properties>
        <!--项目构建编码-->
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

 <project.reporting.outputEncoding>UTF8</project.reporting.outputEncoding>

        <!--声明JDK版本-->
        <java.version>1.8</java.version>
        <!--spring cloud 版本.注意此版本是建立在boot2.2.2版本上的-->
        <mr.spring.cloud.version>Hoxton.SR1</mr.spring.cloud.version>
    </properties>


    <!--boot 版本-->
    <parent>
```

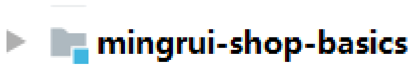
```xml
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.3.1.RELEASE</version>
        <!--始终从仓库中获取，不从本地路径获取-->
        <relativePath />
    </parent>
    <dependencies>
        <!-- 集成commons工具类 -->
        <dependency>
            <groupId>org.apache.commons</groupId>
            <artifactId>commons-lang3</artifactId>
        </dependency>
        <!-- 集成lombok 框架 -->
        <dependency>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
        </dependency>
        <!--junit测试-->
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
        </dependency>
        <!-- SpringBoot整合eureka客户端 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>
        </dependency>
        <!--boot 测试模块-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <!-- 项目依赖,子级模块可以继承依赖-->
    <dependencyManagement>
        <dependencies>
            <!--cloud 依赖-->
            <dependency>
                <groupId>org.springframework.cloud</groupId>
                <artifactId>spring-cloud-dependencies</artifactId>
                <version>${mr.spring.cloud.version}</version>
                <type>pom</type>
                <!--解决maven单继承的问题-->
                <scope>import</scope>
            </dependency>
        </dependencies>
    </dependencyManagement>
    <!-- 注意： 这里必须要添加， 否者各种依赖有问题 -->
    <repositories>
        <repository>
            <id>spring-milestones</id>
            <name>Spring Milestones</name>
            <url>https://repo.spring.io/libs-milestone</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
    </repositories>


</project>
```

创建基本父级项目

在mingrui-shop-parent 工程名上右键-->new-->module



项目名为: mingrui-shop-basics



删除src文件夹

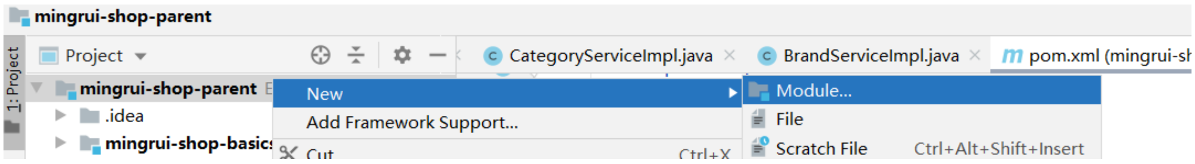
pom.xml

只需要把打包方式设置为pom即可,暂时先不要引入其他依赖

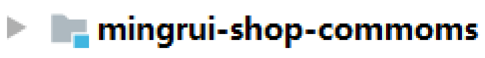
```
<!--父级项目不需要打包所有packging的类型为pom-->
<packaging>pom</packaging>
```

创建工具工程

在mingrui-shop-parent 工程名上右键-->new-->module



项目名为: mingrui-shop-commoms



删除src文件夹

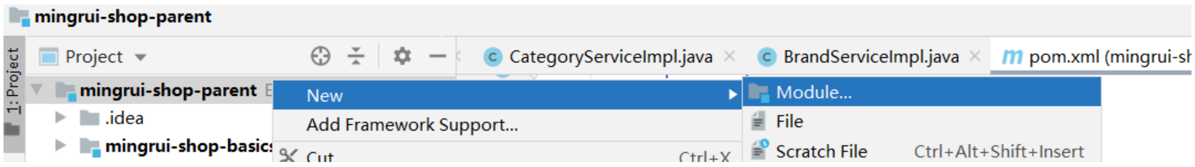
pom.xml

只需要把打包方式设置为pom即可,暂时先不要引入其他依赖

```
<!--父级项目不需要打包所有packging的类型为pom-->
<packaging>pom</packaging>
```

创建服务实现工程

在mingrui-shop-parent 工程名上右键-->new-->module



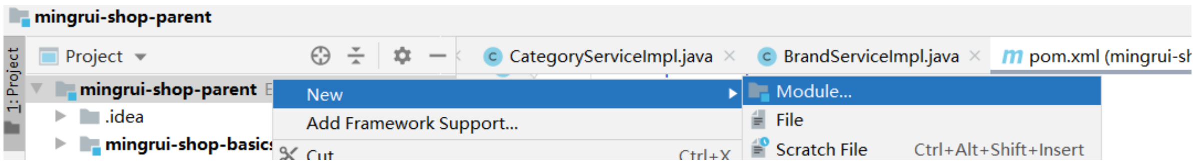
项目名为: mingrui-shop-service

pom.xml

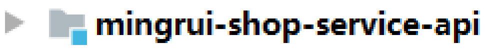
```
<!--父级项目不需要打包所有packging的类型为pom-->
    <packaging>pom</packaging>
    <dependencies>
        <!-- SpringBoot-整合Web组件 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!-- springcloud feign组件 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
    </dependencies>
```

创建服务接口工程

在mingrui-shop-parent 工程名上右键-->new-->module



项目名为: mingrui-shop-service-api

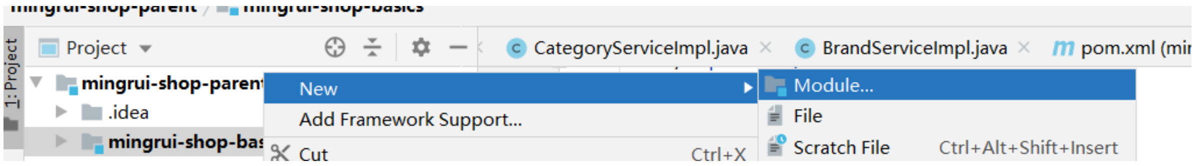


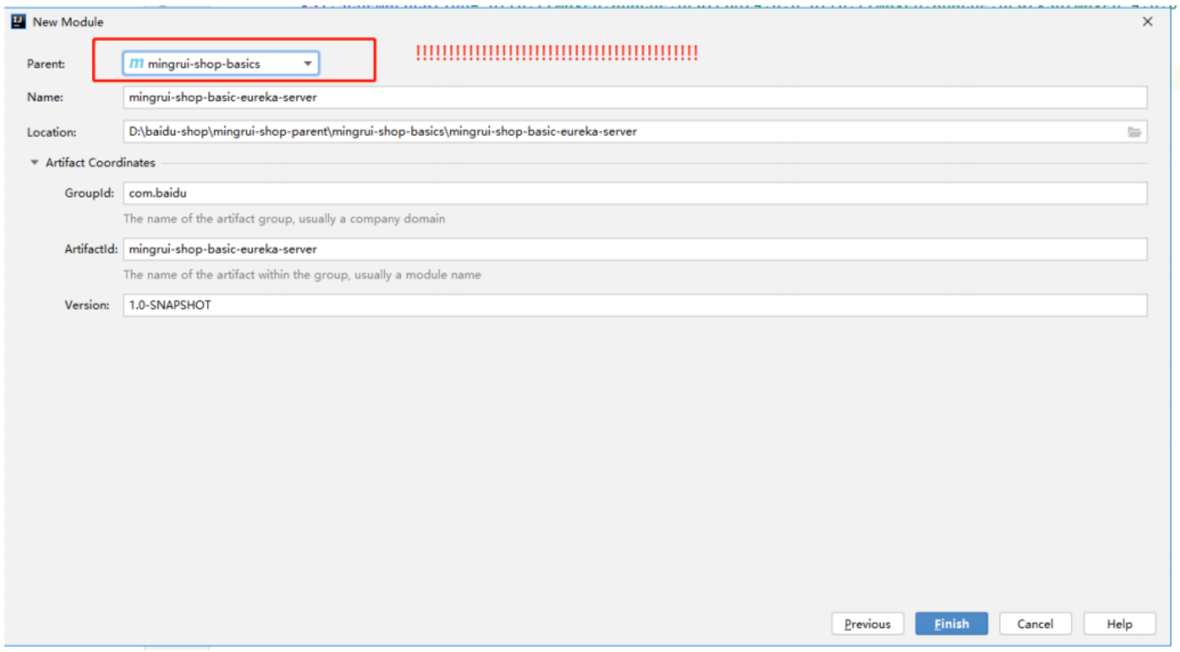
pom.xml

```
<!--父级项目不需要打包所有packging的类型为pom-->
<packaging>pom</packaging>

<dependencies>
    <!-- SpringBoot-整合Web组件 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
</dependencies>
```

创建eureka服务

在mingrui-shop-basics工程名上右键-->new-->module





Parent: mingrui-shop-basics

Name: mingrui-shop-basic-eureka-server

Location: D:\baidu-shop\mingrui-shop-parent\mingrui-shop-basics\mingrui-shop-basic-eureka-server

Artifact Coordinates

GroupId: com.baidu

The name of the artifact group, usually a company domain

ArtifactId: mingrui-shop-basic-eureka-server

The name of the artifact within the group, usually a module name

Version: 1.0-SNAPSHOT

注意:在点击finish之前一定要确认一遍当前创建工程的父工程是mingrui-shop-basics,

接下来的项目创建的时候也是一样的

pom.xml

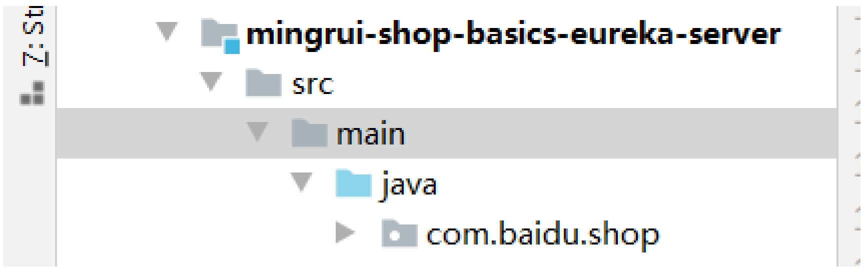
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <parent>
        <artifactId>mingrui-shop-basics</artifactId>
        <groupId>com.baidu</groupId>
        <version>1.0-SNAPSHOT</version>
    </parent>
    <modelVersion>4.0.0</modelVersion>

    <artifactId>mingrui-shop-basics-eureka-server</artifactId>

    <dependencies>
        <!--eureka 服务依赖-->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-netflix-eureka-server</artifactId>
        </dependency>
    </dependencies>

</project>
```

application.yml

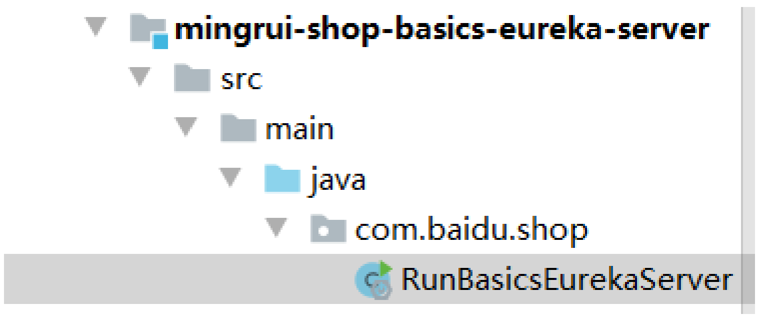
```
server:
  port: 8761
spring:
  application:
    name: eureka-server
eureka:
  client:
    # eureka服务url,值为map集合默认key为defaultZone
    service-url:
      defaultZone: http://${eureka.instance.hostname}:${server.port}/eureka
    # 当前服务是否同时注册
    register-with-eureka: false
    # 去注册中心获取其他服务的地址
    fetch-registry: false
  instance:
    hostname: localhost
    # 定义服务续约任务（心跳）的调用间隔，单位：秒 默认30
    lease-renewal-interval-in-seconds: 1
    # 定义服务失效的时间，单位：秒 默认90
    lease-expiration-duration-in-seconds: 2
  server:
    # 测试时关闭自我保护机制，保证不可用服务及时踢出
    enable-self-preservation: false
```

创建启动类

首先在java包下创建com.baidu.shop



其次在com.baidu.shop上创建启动类

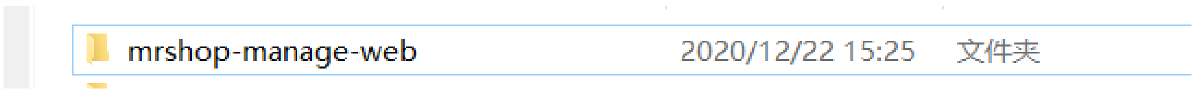


启动类：

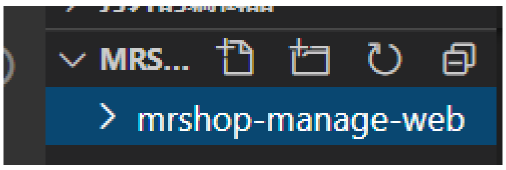
```
package com.baidu.shop;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.server.EnableEurekaServer;
```

```
@SpringBootApplication
@EnableEurekaServer
public class RunBasicsEurekaServer {
    public static void main(String[] args) {
        SpringApplication.run(RunBasicsEurekaServer.class);
    }
}
```

8.前台项目


mrshop-manage-web 2020/12/22 15:25 文件夹

找到此文件从Visual Studio Code打开



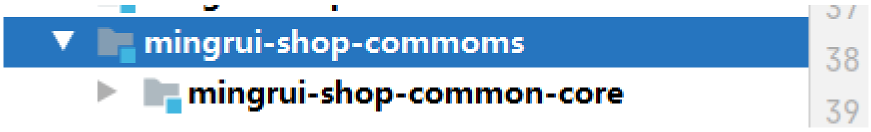
项目搭建完成！！！

二，分类管理

1.搭建

mingrui-shop-common-core工程

在mingrui-shop--commons项目下新建mingrui-shopcommon-core项目



pom.xml

```
<!--处理json与各种数据类型或文档类型的转换-->
        <dependency>
            <groupId>com.google.code.gson</groupId>
            <artifactId>gson</artifactId>
            <version>2.8.5</version>
        </dependency>

        <!--json对象序列化和反序列化的支持-->
        <dependency>
            <groupId>org.codehaus.jackson</groupId>
            <artifactId>jackson-core-lgpl</artifactId>
            <version>1.9.13</version>
        </dependency>
        <dependency>
        <groupId>org.codehaus.jackson</groupId>
            <artifactId>jackson-mapper-lgpl</artifactId>
            <version>1.9.13</version>
        </dependency>
        <!--java对象和json对象之间的转换-->
        <dependency>
            <groupId>org.codehaus.jackson</groupId>
            <artifactId>jackson-core-asl</artifactId>
            <version>1.9.13</version>
        </dependency>
        <dependency>
            <groupId>org.codehaus.jackson</groupId>
            <artifactId>jackson-mapper-asl</artifactId>
            <version>1.9.13</version>
        </dependency>

        <!--alibaba的json处理工具-->
        <dependency>
            <groupId>com.alibaba</groupId>
            <artifactId>fastjson</artifactId>
            <version>1.2.62</version>
        </dependency>
        <dependency>
            <groupId>com.fasterxml.jackson.core</groupId>
            <artifactId>jackson-databind</artifactId>
            <version>2.11.2</version>
        </dependency>
```

新建包com.baidu.shop.utils创建JSONUtil

```
import com.alibaba.fastjson.JSONObject;
import org.codehaus.jackson.JsonParseException;
import org.codehaus.jackson.map.JsonMappingException;
import org.codehaus.jackson.map.ObjectMapper;
import org.codehaus.jackson.type.TypeReference;
import com.google.gson.Gson;
import com.google.gson.GsonBuilder;
import com.google.gson.JsonArray;
import com.google.gson.JsonElement;
import com.google.gson.JsonObject;
import com.google.gson.JsonParser;
import com.google.gson.reflect.TypeToken;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;


public class JSONUtil {
    private static Gson gson = null;
    static {
        gson = new GsonBuilder().setDateFormat("yyyy-MM-dd
HH:mm:ss").create();// todo yyyy-MM-dd HH:mm:ss
    }
    public static synchronized Gson newInstance() {
        if (gson == null) {
```

```java
            gson = new GsonBuilder().setDateFormat("yyyy-MM-dd
HH:mm:ss").create();
        }
        return gson;
    }
    public static String toJsonString(Object obj) {
        return gson.toJson(obj);
    }
    public static <T> T toBean(String json, Class<T> clz) {
        return gson.fromJson(json, clz);
    }
    public static <T> Map<String, T> toMap(String json, Class<T> clz) {
        Map<String, JsonObject> map = gson.fromJson(json, new
                TypeToken<Map<String, JsonObject>>() {
                }.getType());
        Map<String, T> result = new HashMap<String, T>();
        for (String key : map.keySet()) {
            result.put(key, gson.fromJson(map.get(key), clz));
        }
        return result;
    }
    public static Map<String, Object> toMap(String json) {
        Map<String, Object> map = gson.fromJson(json, new TypeToken<Map<String,
                Object>>() {
        }.getType());
        return map;
    }
    public static <T> List<T> toList(String json, Class<T> clz) {
        JsonArray array = new JsonParser().parse(json).getAsJsonArray();
        List<T> list = new ArrayList<T>();
        for (final JsonElement elem : array) {
            list.add(gson.fromJson(elem, clz));
        }
        return list;
    }
    /**
     * 从json字符串中获取需要的值
     *
     * @param json
     * @param clazz 要转换的类型
     * @return
     */
    public static <T> Object getObjectByKey(String json, Class<T> clazz) {
        if (json != null && !"".equals(json)) {
            return JSONObject.parseObject(json, clazz);
        }
        return null;
    }
    /**
     * 从json字符串中获取需要的值
     *
     * @param json
     * @param clazz 要转换的类型
     * @return
     */
    public static <T> List<T> getListByKey(String json, Class<T> clazz) {
        if (json != null && !"".equals(json)) {
            return JSONObject.parseArray(json, clazz);
        }
        return null;
    }
    /**
     * 从json字符串中获取需要的值
     *
     * @param json
     * @param key
     * 键
     * @return
     */
    public static String getStrByKey(String json, String key) {
        String str = "";
        if (json != null && !"".equals(json)) {
            JSONObject j = JSONObject.parseObject(json);
            if (j.get(key) != null) {
                str = j.get(key).toString();
            }
        }
        return str;
    }
    /**
     * 向文件中写数据
     *
     * @param _sDestFile
     * @param _sContent
     * @throws IOException
     */
    public static void writeByFileOutputStream(String _sDestFile, String
            _sContent) throws IOException {
        FileOutputStream fos = null;
        try {
            fos = new FileOutputStream(_sDestFile);
            fos.write(_sContent.getBytes());
        } catch (Exception ex) {
            ex.printStackTrace();
        } finally {
            if (fos != null) {
                fos.close();
                fos = null;
            }
        }
    }
    /**
     * 非空
     *
     * @param str
     * @return true:不为空 false: 空
     */
    public static boolean noEmpty(String str) {
        boolean flag = true;
        if ("".equals(str)) {
            flag = false;
        }
        return flag;
    }
    /**
     * 将"%"去掉
     *
     * @param str
     * @return
     */
    public static double getDecimalByPercentage(String str) {
        double fuse = 0.0;
        if (!"".equals(str) && str != null) {
            if (str.split("%").length > 0) {
                fuse = Double.parseDouble(str.split("%")[0]);
                return fuse;
            }
        }
        return 0.0;
    }
    /**
     * 保留2位小数
     *
     * @param number
     * @return
     */
    public static double ConversionFraction(double number) {
        return Math.floor(number * 100 + 0.5) / 100;
    }
    public static float ConversionM(double number) {
        return (float) JSONUtil.ConversionFraction(number / 1024 / 1024);
    }
    public static String getErrorText(String s) {
        JSONObject j = JSONObject.parseObject(s);
        return

j.getJSONObject(j.keySet().iterator().next()).get("errortext").toString();
    }
    public static String getSingleJobId(String s) throws Exception {
        JSONObject j = JSONObject.parseObject(s);
        try {
            return

j.getJSONObject(j.keySet().iterator().next()).get("jobid").toString();
        } catch (Exception e) {
            try {
                return

j.getJSONObject(j.keySet().iterator().next()).get("errortext").toString();
            } catch (Exception e1) {
                throw new Exception(e1.getMessage());
            }
        }
    }
    public static <T> T readValue(String jsonStr, TypeReference type)
            throws JsonParseException, JsonMappingException, IOException {
        ObjectMapper mapper = new ObjectMapper();
        return mapper.readValue(jsonStr, type);
    }
    public static JSON_TYPE getJSONType(String str) {
        if (null == str || "".equals(str)) {
            return JSON_TYPE.JSON_TYPE_ERROR;
        }
        final char[] strChar = str.substring(0, 1).toCharArray();
        final char firstChar = strChar[0];
        if (firstChar == '{') {
            return JSON_TYPE.JSON_TYPE_OBJECT;
        } else if (firstChar == '[') {
            return JSON_TYPE.JSON_TYPE_ARRAY;
        } else {
            return JSON_TYPE.JSON_TYPE_ERROR;
        }
    }
    public enum JSON_TYPE {
        /** JSONObject */
        JSON_TYPE_OBJECT,
        /** JSONArray */
        JSON_TYPE_ARRAY,
        /** 不是JSON格式的字符串 */
        JSON_TYPE_ERROR
    }
}
```

新建包com.baidu.shop.status创建HTTPStatus

```

public class HTTPStatus {
    public static final int OK = 200;//成功
    public static final int ERROR = 500;//失败
}
```

新建包com.baidu.shop.base创建Result

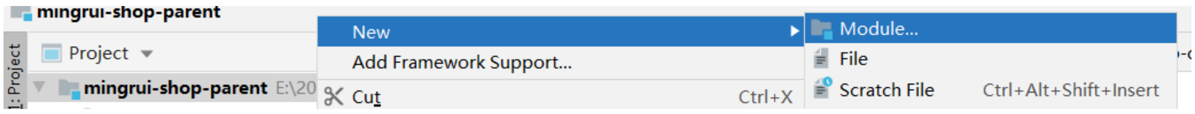
```
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
public class Result<T> {
    private Integer code;//返回码
    private String message;//返回消息
    private T data;//返回数据
    public Result(Integer code, String message, Object data) {
        this.code = code;
        this.message = message;
        this.data = (T) data;
    }
}
```

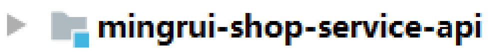
com.baidu.shop.status创建BaseApiService

```
import com.baidu.shop.utils.JSONUtil;
import com.baidu.shop.status.HTTPStatus;
import lombok.Data;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Component;

@Data
@Component
@Slf4j
```

```
public class BaseApiService<T> {
    public Result<T> setResultError(Integer code, String msg) {
        return setResult(code, msg, null);
    }
    // 返回错误，可以传msg
    public Result<T> setResultError(String msg) {
        return setResult(HTTPStatus.ERROR, msg, null);
    }
    // 返回成功，可以传data值
    public Result<T> setResultSuccess(T data) {
        return setResult(HTTPStatus.OK, HTTPStatus.OK + "", data);
    }
    // 返回成功，沒有data值
    public Result<T> setResultSuccess() {
        return setResult(HTTPStatus.OK, HTTPStatus.OK + "", null);
    }
    // 返回成功，沒有data值
    public Result<T> setResultSuccess(String msg) {
        return setResult(HTTPStatus.OK, msg, null);
    }
    // 通用封装
    public Result<T> setResult(Integer code, String msg, T data) {
        log.info(String.format("{code : %s , message : %s , data :
%s}",code,msg,
                JSONUtil.toJsonString(data)));
        return new Result<T>(code, msg, data);
    }
}
```

在mingrui-shop-parent新建mingrui-shop-service-api工程





mingrui-shop-service-api/pom.xml

```
<!-- SpringBoot-整合Web组件 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--Entity 中的@Table 和@Id需要次注解-->
        <dependency>
            <groupId>javax.persistence</groupId>
            <artifactId>persistence-api</artifactId>
            <version>1.0.2</version>
        </dependency>
        <!--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>
        <!--引入common工程代码-->
        <dependency>
            <groupId>com.baidu</groupId>
            <artifactId>mingrui-shop-common-core</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
```

在mingrui-shop-service-api项目下新建mingrui-shopservice-api-xxx项目

pom.xml

```
<!--帮助开发人员快速生成API文档-->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>
        <!--提供可视化的API文档-->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>
```

mingrui-shopservice-api-xxx项目新建包com.baidu.shop.entity

```
import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.persistence.Id;
import javax.persistence.Table;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
com.baidu.shop.entity
@ApiModel(value = "分类实体类")
@Data
@Table(name = "tb_category")
public class CategoryEntity {
    @Id
    @ApiModelProperty(value = "分类主键",example = "1")
    private Integer id;

    @ApiModelProperty(value = "分类名称")
    private String name;

    @ApiModelProperty(value = "父级分类",example = "1")
    private Integer parentId;

    @ApiModelProperty(value = "是否是父级节点",example = "1")
    private Integer isParent;

    @ApiModelProperty(value = "排序",example = "1")
    private Integer sort;
}
```

com.baidu.shop.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;

@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();
    }
}
```

2.商品查询

后台

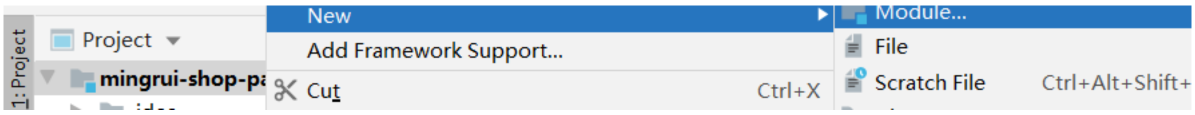
定义商品分类接口

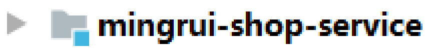
mingrui-shopservice-api-xxx项目创建com.baidu.shop.CategoryService

```
import com.baidu.shop.base.Result;
import com.baidu.shop.entity.CategoryEntity;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Api(tags = "商品分类接口")
public interface CategoryService {
    @ApiOperation(value = "通过查询商品分类")
    @GetMapping(value = "category/list")
    Result<List<CategoryEntity>> getCategoryByPid(Integer pid);
}
```

service工程

在mingrui-shop-parent创建mingrui-shop-service

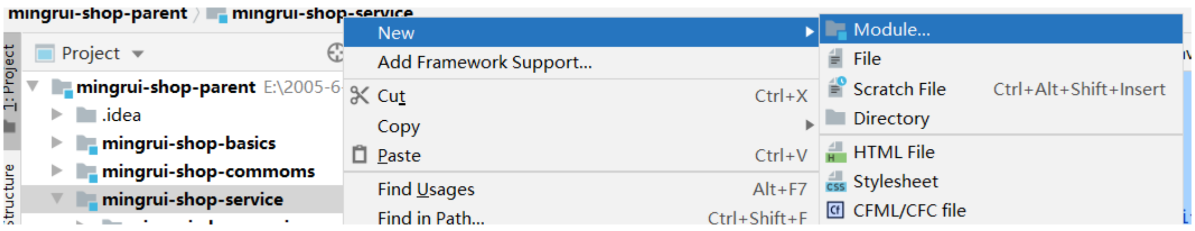


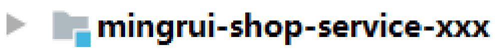


mingrui-shop-service/pom.xml

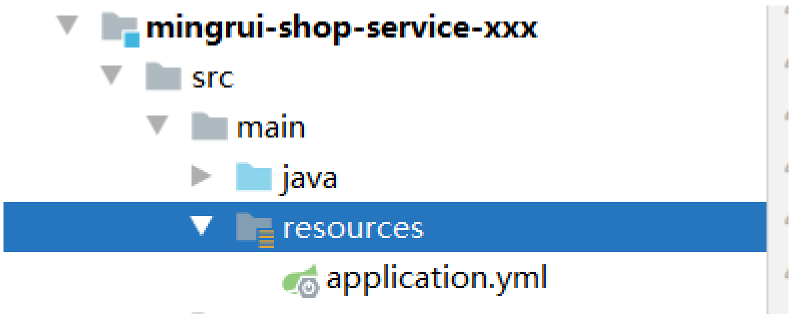
```
<dependencies>
        <!-- SpringBoot-整合Web组件 -->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!-- springcloud feign组件 -->
        <dependency>
            <groupId>org.springframework.cloud</groupId>
            <artifactId>spring-cloud-starter-openfeign</artifactId>
        </dependency>
        <!--mysql数据库连接-->
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope><!--项目运行阶段使用-->
        </dependency>
        <!--通用mapper-->
        <dependency>
            <groupId>tk.mybatis</groupId>
            <artifactId>mapper-spring-boot-starter</artifactId>
            <version>2.1.5</version>
        </dependency>
        <!--分页工具-->
        <dependency>
            <groupId>com.github.pagehelper</groupId>
            <artifactId>pagehelper-spring-boot-starter</artifactId>
            <version>1.2.10</version>
        </dependency>
        <!--将mingrui-shop-service-api-xxx引入项目-->
        <dependency>
            <groupId>com.baidu</groupId>
            <artifactId>mingrui-shop-service-api-xxx</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

在mingrui-shop-service工程下新建mingrui-shop-service-xxx工程





在resources创建application.yml



application.yml

```
server:
  port: 8100
spring:
  application:
    name: xxx-server
  # 配置数据源
  datasource:
    # 数据源名称，任意
    name: mysql
    url: jdbc:mysql://127.0.0.1:3306/mr-shop?
useSSL=true&nullNamePatternMatchesAll=true&serverTimezone=GMT%2B8&useUnicode=tru
e&characterEncoding=utf8
    # 数据库连接用户
    username: root
    # 数据库连接密码
    password: root
    # 驱动名称
    driver-class-name: com.mysql.cj.jdbc.Driver
    # boot2.0+使用hikari作为默认数据库连接池
    type: com.zaxxer.hikari.HikariDataSource
    hikari:
      # 是否自动提交事务 默认
      auto-commit: true
      # 允许的最小连接数
      minimum-idle: 5
      # 连接池内最大连接数
      maximum-pool-size: 10
      # 验证连接的sql语句
      connection-test-query: SELECT 1 FROM DUAL
```

```
        # 连接超时时间 默认30000 毫秒 如果小于250毫秒，则被重置回30秒
        connection-timeout: 30000
        # 验证超时时间默认5000毫秒 如果小于250毫秒，则会被重置回5秒
        validation-timeout: 5000
        # 设置连接在连接池中的存货时间 如果不等于0且小于30秒则会被重置回30分钟
        max-lifetime: 1800000
# 通用mapper
mapper:
  mappers: tk.mybatis.mapper.common.Mapper
  identity: MYSQL
#日志设置
logging:
  level:
    # 打印与我们程序相关的日志信息
    com.baidu.shop: debug
# eureka配置
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
```

新建包com.baidu新建启动类RunShopServicexxx.class

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.openfeign.EnableFeignClients;
import tk.mybatis.spring.annotation.MapperScan;

@SpringBootApplication
@EnableFeignClients
@MapperScan("com.baidu.shop.mapper")
public class RunShopServicexxx {
    public static void main(String[] args) {
        SpringApplication.run(RunShopServicexxx.class);
    }
}
```

在com.baidu下新建包shop.mapper创建mapper

```
import com.baidu.shop.entity.CategoryEntity;
import org.apache.ibatis.annotations.Select;
import tk.mybatis.mapper.common.Mapper;

import java.util.List;

public interface CategoryMapper extends Mapper<CategoryEntity> {
    @Select(value = "select id,name from tb_category where id in (select
category_id from tb_category_brand where brand_id=#{brandId})")
    List<CategoryEntity> getByBrandId(Integer brandId);
}
```

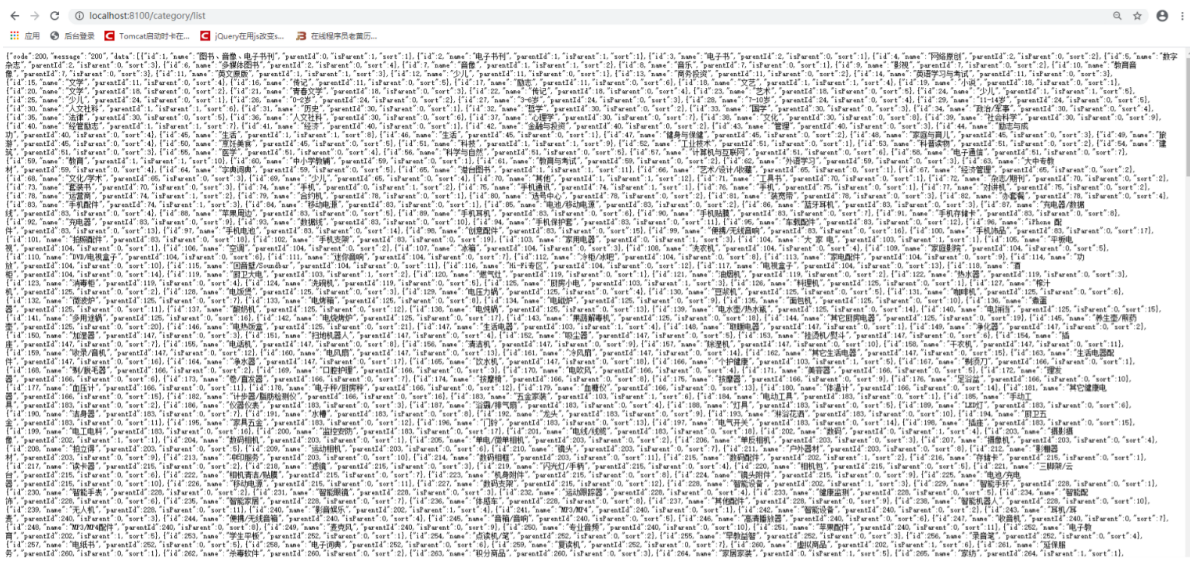
在com.baidu下新建包shop.service.impl新建实现类

```
import com.baidu.shop.mapper.CategoryMapper;
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
```

```java
import com.baidu.shop.entity.CategoryEntity;
import com.baidu.shop.service.CategoryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RestController;
import java.util.List;

@RestController
public class CategoryServiceImpl extends BaseApiService implements
CategoryService {

    @Resource
    private CategoryMapper categoryMapper;

    //商品查询
    @Override
    public Result<List<CategoryEntity>> getCategoryByPid(Integer pid) {
        CategoryEntity categoryEntity = new CategoryEntity();
        categoryEntity.setParentId(pid);
        List<CategoryEntity> list = categoryMapper.select(categoryEntity);
        return this.setResultSuccess(list);
    }
}
```

浏览器输入

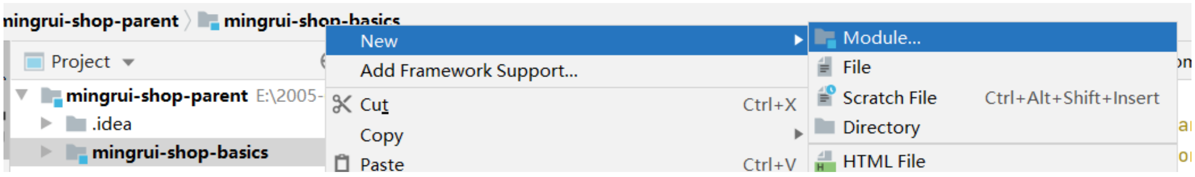
```
http://localhost:8100/category/list
```



```
{"code":200,"message":"200","data":[{"id":1,"name":"图书、音像、电子书刊","parentId":0,"isParent":1,"sort":1},{"id":2,"name":"电子书刊","parentId":1,"isParent":1,"sort":1},{"id":3,"name":"电子书","parentId":2,"isParent":0,"sort":1},{"id":4,"name":"网络原创","parentId":2,"isParent":0,"sort":2},{"id":5,"name":"数字
杂志","parentId":2,"isParent":0,"sort":3},{"id":6,"name":"多媒体图书","parentId":2,"isParent":0,"sort":4},{"id":7,"name":"音像","parentId":1,"isParent":1,"sort":2},{"id":8,"name":"音乐","parentId":7,"isParent":0,"sort":1},{"id":9,"name":"影视","parentId":7,"isParent":0,"sort":2},{"id":10,"name":"教育音
像","parentId":7,"isParent":0,"sort":3},{"id":11,"name":"英文原版","parentId":1,"isParent":1,"sort":3},{"id":12,"name":"少儿","parentId":11,"isParent":0,"sort":1},{"id":13,"name":"商务投资","parentId":11,"isParent":0,"sort":2},{"id":14,"name":"英语学习与考试","parentId":11,"isParent":0,"sort":3},
{"id":15,"name":"文学","parentId":11,"isParent":0,"sort":4},{"id":16,"name":"传记","parentId":11,"isParent":0,"sort":5},{"id":17,"name":"励志","parentId":11,"isParent":0,"sort":6},{"id":18,"name":"文艺","parentId":1,"isParent":1,"sort":4},{"id":19,"name":"小说","parentId":18,"isParent":0,"sort":1},
{"id":20,"name":"文学","parentId":18,"isParent":0,"sort":2},{"id":21,"name":"青春文学","parentId":18,"isParent":0,"sort":3},{"id":22,"name":"传记","parentId":18,"isParent":0,"sort":4},{"id":23,"name":"艺术","parentId":18,"isParent":0,"sort":5},{"id":24,"name":"少儿","parentId":1,"isParent":1,"sort":5},
{"id":25,"name":"少儿","parentId":24,"isParent":0,"sort":1},{"id":26,"name":"0-2岁","parentId":24,"isParent":0,"sort":2},{"id":27,"name":"3-6岁","parentId":24,"isParent":0,"sort":3},{"id":28,"name":"7-10岁","parentId":24,"isParent":0,"sort":4},{"id":29,"name":"11-14岁","parentId":24,"isParent":0,"sort":5},
{"id":30,"name":"人文社科","parentId":1,"isParent":1,"sort":6},{"id":31,"name":"历史","parentId":30,"isParent":0,"sort":1},{"id":32,"name":"哲学","parentId":30,"isParent":0,"sort":2},{"id":33,"name":"国学","parentId":30,"isParent":0,"sort":3},{"id":34,"name":"政治/军事","parentId":30,"isParent":0,"sort":4},
{"id":35,"name":"法律","parentId":30,"isParent":0,"sort":5},{"id":36,"name":"人文社科","parentId":30,"isParent":0,"sort":6},{"id":37,"name":"心理学","parentId":30,"isParent":0,"sort":7},{"id":38,"name":"文化","parentId":30,"isParent":0,"sort":8},{"id":39,"name":"社会科学","parentId":30,"isParent":0,"sort":9},
{"id":40,"name":"经管励志","parentId":1,"isParent":1,"sort":7},{"id":41,"name":"经济","parentId":40,"isParent":0,"sort":1},{"id":42,"name":"金融与投资","parentId":40,"isParent":0,"sort":2},{"id":43,"name":"管理","parentId":40,"isParent":0,"sort":3},{"id":44,"name":"励志与成
功","parentId":40,"isParent":0,"sort":4},{"id":45,"name":"生活","parentId":1,"isParent":1,"sort":8},{"id":46,"name":"生活","parentId":45,"isParent":0,"sort":1},{"id":47,"name":"健身与保健","parentId":45,"isParent":0,"sort":2},{"id":48,"name":"家庭与育儿","parentId":45,"isParent":0,"sort":3},{"id":49,"name":"旅
游","parentId":45,"isParent":0,"sort":4},{"id":50,"name":"烹饪美食","parentId":45,"isParent":0,"sort":5},{"id":51,"name":"科技","parentId":1,"isParent":1,"sort":9},{"id":52,"name":"工业技术","parentId":51,"isParent":0,"sort":1},{"id":53,"name":"科普读物","parentId":51,"isParent":0,"sort":2},{"id":54,"name":"建
筑","parentId":51,"isParent":0,"sort":3},{"id":55,"name":"医学","parentId":51,"isParent":0,"sort":4},{"id":56,"name":"科学与自然","parentId":51,"isParent":0,"sort":5},{"id":57,"name":"计算机与互联网","parentId":51,"isParent":0,"sort":6},{"id":58,"name":"电子通信","parentId":51,"isParent":0,"sort":7},
{"id":59,"name":"教育","parentId":1,"isParent":1,"sort":10},{"id":60,"name":"中小学教辅","parentId":59,"isParent":0,"sort":1},{"id":61,"name":"教育与考试","parentId":59,"isParent":0,"sort":2},{"id":62,"name":"外语学习","parentId":59,"isParent":0,"sort":3},{"id":63,"name":"大中专教
材","parentId":59,"isParent":0,"sort":4},{"id":64,"name":"字典词典","parentId":59,"isParent":0,"sort":5},{"id":65,"name":"港台图书","parentId":1,"isParent":1,"sort":11},{"id":66,"name":"艺术/设计/收藏","parentId":65,"isParent":0,"sort":1},{"id":67,"name":"经济管理","parentId":65,"isParent":0,"sort":2},
{"id":68,"name":"文化/学术","parentId":65,"isParent":0,"sort":3},{"id":69,"name":"少儿","parentId":65,"isParent":0,"sort":4},{"id":70,"name":"其他","parentId":1,"isParent":1,"sort":12},{"id":71,"name":"工具书","parentId":70,"isParent":0,"sort":1},{"id":72,"name":"杂志/期刊","parentId":70,"isParent":0,"sort":2},
{"id":73,"name":"套装书","parentId":70,"isParent":0,"sort":3},{"id":74,"name":"手机","parentId":0,"isParent":1,"sort":2},{"id":75,"name":"手机通讯","parentId":74,"isParent":1,"sort":1},{"id":76,"name":"手机","parentId":75,"isParent":0,"sort":1},{"id":77,"name":"对讲机","parentId":75,"isParent":0,"sort":2},
{"id":78,"name":"运营商","parentId":74,"isParent":1,"sort":2},{"id":79,"name":"合约机","parentId":78,"isParent":0,"sort":1},{"id":80,"name":"选号中心","parentId":78,"isParent":0,"sort":2},{"id":81,"name":"装宽带","parentId":78,"isParent":0,"sort":3},{"id":82,"name":"办套餐","parentId":78,"isParent":0,"sort":4},
{"id":83,"name":"手机配件","parentId":74,"isParent":1,"sort":3},{"id":84,"name":"移动电源","parentId":83,"isParent":0,"sort":1},{"id":85,"name":"电池/移动电源","parentId":83,"isParent":0,"sort":2},{"id":86,"name":"蓝牙耳机","parentId":83,"isParent":0,"sort":3},{"id":87,"name":"充电器/数据
线","parentId":83,"isParent":0,"sort":4},{"id":88,"name":"苹果周边","parentId":83,"isParent":0,"sort":5},{"id":89,"name":"手机耳机","parentId":83,"isParent":0,"sort":6},{"id":90,"name":"手机贴膜","parentId":83,"isParent":0,"sort":7},{"id":91,"name":"手机存储卡","parentId":83,"isParent":0,"sort":8},
{"id":92,"name":"充电器","parentId":83,"isParent":0,"sort":9},{"id":93,"name":"数据线","parentId":83,"isParent":0,"sort":10},{"id":94,"name":"手机保护套","parentId":83,"isParent":0,"sort":11},{"id":95,"name":"车载配件","parentId":83,"isParent":0,"sort":12},{"id":96,"name":"iPhone 配
件","parentId":83,"isParent":0,"sort":13},{"id":97,"name":"手机电池","parentId":83,"isParent":0,"sort":14},{"id":98,"name":"创意配件","parentId":83,"isParent":0,"sort":15},{"id":99,"name":"便携/无线音响","parentId":83,"isParent":0,"sort":16},{"id":100,"name":"手机饰品","parentId":83,"isParent":0,"sort":17},
{"id":101,"name":"拍照配件","parentId":83,"isParent":0,"sort":18},{"id":102,"name":"手机支架","parentId":83,"isParent":0,"sort":19},{"id":103,"name":"家用电器","parentId":0,"isParent":1,"sort":3},{"id":104,"name":"大 家 电","parentId":103,"isParent":1,"sort":1},{"id":105,"name":"平板电
视","parentId":104,"isParent":0,"sort":1},{"id":106,"name":"空调","parentId":104,"isParent":0,"sort":2},{"id":107,"name":"冰箱","parentId":104,"isParent":0,"sort":3},{"id":108,"name":"洗衣机","parentId":104,"isParent":0,"sort":4},{"id":109,"name":"家庭影院","parentId":104,"isParent":0,"sort":5},
{"id":110,"name":"DVD/电视盒子","parentId":104,"isParent":0,"sort":6},{"id":111,"name":"迷你音响","parentId":104,"isParent":0,"sort":7},{"id":112,"name":"冷柜/冰吧","parentId":104,"isParent":0,"sort":8},{"id":113,"name":"家电配件","parentId":104,"isParent":0,"sort":9},{"id":114,"name":"功
放","parentId":104,"isParent":0,"sort":10},{"id":115,"name":"回音壁/Soundbar","parentId":104,"isParent":0,"sort":11},{"id":116,"name":"Hi-Fi专区","parentId":104,"isParent":0,"sort":12},{"id":117,"name":"电视盒子","parentId":104,"isParent":0,"sort":13},{"id":118,"name":"酒
柜","parentId":104,"isParent":0,"sort":14},{"id":119,"name":"厨卫大电","parentId":103,"isParent":1,"sort":2},{"id":120,"name":"燃气灶","parentId":119,"isParent":0,"sort":1},{"id":121,"name":"油烟机","parentId":119,"isParent":0,"sort":2},{"id":122,"name":"热水器","parentId":119,"isParent":0,"sort":3},
{"id":123,"name":"消毒柜","parentId":119,"isParent":0,"sort":4},{"id":124,"name":"洗碗机","parentId":119,"isParent":0,"sort":5},{"id":125,"name":"厨房小电","parentId":103,"isParent":1,"sort":3},{"id":126,"name":"料理机","parentId":125,"isParent":0,"sort":1},{"id":127,"name":"榨汁
机","parentId":125,"isParent":0,"sort":2},{"id":128,"name":"电饭煲","parentId":125,"isParent":0,"sort":3},{"id":129,"name":"电压力锅","parentId":125,"isParent":0,"sort":4},{"id":130,"name":"豆浆机","parentId":125,"isParent":0,"sort":5},{"id":131,"name":"咖啡机","parentId":125,"isParent":0,"sort":6},
{"id":132,"name":"微波炉","parentId":125,"isParent":0,"sort":7},{"id":133,"name":"电烤箱","parentId":125,"isParent":0,"sort":8},{"id":134,"name":"电磁炉","parentId":125,"isParent":0,"sort":9},{"id":135,"name":"面包机","parentId":125,"isParent":0,"sort":10},{"id":136,"name":"煮蛋
器","parentId":125,"isParent":0,"sort":11},{"id":137,"name":"酸奶机","parentId":125,"isParent":0,"sort":12},{"id":138,"name":"电炖锅","parentId":125,"isParent":0,"sort":13},{"id":139,"name":"电水壶/热水瓶","parentId":125,"isParent":0,"sort":14},{"id":140,"name":"电饼铛","parentId":125,"isParent":0,"sort":15},
{"id":141,"name":"多用途锅","parentId":125,"isParent":0,"sort":16},{"id":142,"name":"电烧烤炉","parentId":125,"isParent":0,"sort":17},{"id":143,"name":"果蔬解毒机","parentId":125,"isParent":0,"sort":18},{"id":144,"name":"其它厨房电器","parentId":125,"isParent":0,"sort":19},{"id":145,"name":"养生壶/煎药
壶","parentId":125,"isParent":0,"sort":20},{"id":146,"name":"电热饭盒","parentId":125,"isParent":0,"sort":21},{"id":147,"name":"生活电器","parentId":103,"isParent":1,"sort":4},{"id":148,"name":"取暖电器","parentId":147,"isParent":0,"sort":1},{"id":149,"name":"净化器","parentId":147,"isParent":0,"sort":2},
{"id":150,"name":"加湿器","parentId":147,"isParent":0,"sort":3},{"id":151,"name":"扫地机器人","parentId":147,"isParent":0,"sort":4},{"id":152,"name":"吸尘器","parentId":147,"isParent":0,"sort":5},{"id":153,"name":"挂烫机/熨斗","parentId":147,"isParent":0,"sort":6},{"id":154,"name":"插
座","parentId":147,"isParent":0,"sort":7},{"id":155,"name":"电话机","parentId":147,"isParent":0,"sort":8},{"id":156,"name":"清洁机","parentId":147,"isParent":0,"sort":9},{"id":157,"name":"除湿机","parentId":147,"isParent":0,"sort":10},{"id":158,"name":"干衣机","parentId":147,"isParent":0,"sort":11},
{"id":159,"name":"收录/音机","parentId":147,"isParent":0,"sort":12},{"id":160,"name":"电风扇","parentId":147,"isParent":0,"sort":13},{"id":161,"name":"冷风扇","parentId":147,"isParent":0,"sort":14},{"id":162,"name":"其它生活电器","parentId":147,"isParent":0,"sort":15},{"id":163,"name":"生活电器配
件","parentId":147,"isParent":0,"sort":16},{"id":164,"name":"净水器","parentId":147,"isParent":0,"sort":17},{"id":165,"name":"饮水机","parentId":147,"isParent":0,"sort":18},{"id":166,"name":"个护健康","parentId":103,"isParent":1,"sort":5},{"id":167,"name":"剃须刀","parentId":166,"isParent":0,"sort":1},
{"id":168,"name":"剃/脱毛器","parentId":166,"isParent":0,"sort":2},{"id":169,"name":"口腔护理","parentId":166,"isParent":0,"sort":3},{"id":170,"name":"电吹风","parentId":166,"isParent":0,"sort":4},{"id":171,"name":"美容器","parentId":166,"isParent":0,"sort":5},{"id":172,"name":"理发
器","parentId":166,"isParent":0,"sort":6},{"id":173,"name":"卷/直发器","parentId":166,"isParent":0,"sort":7},{"id":174,"name":"按摩椅","parentId":166,"isParent":0,"sort":8},{"id":175,"name":"按摩器","parentId":166,"isParent":0,"sort":9},{"id":176,"name":"足浴盆","parentId":166,"isParent":0,"sort":10},
{"id":177,"name":"血压计","parentId":166,"isParent":0,"sort":11},{"id":178,"name":"电子秤/厨房秤","parentId":166,"isParent":0,"sort":12},{"id":179,"name":"血糖仪","parentId":166,"isParent":0,"sort":13},{"id":180,"name":"体温计","parentId":166,"isParent":0,"sort":14},{"id":181,"name":"其它健康电
器","parentId":166,"isParent":0,"sort":15},{"id":182,"name":"计步器/脂肪检测仪","parentId":166,"isParent":0,"sort":16},{"id":183,"name":"五金家装","parentId":103,"isParent":1,"sort":6},{"id":184,"name":"电动工具","parentId":183,"isParent":0,"sort":1},{"id":185,"name":"手动工
具","parentId":183,"isParent":0,"sort":2},{"id":186,"name":"仪器仪表","parentId":183,"isParent":0,"sort":3},{"id":187,"name":"浴霸/排气扇","parentId":183,"isParent":0,"sort":4},{"id":188,"name":"灯具","parentId":183,"isParent":0,"sort":5},{"id":189,"name":"LED灯","parentId":183,"isParent":0,"sort":6},
{"id":190,"name":"洁身器","parentId":183,"isParent":0,"sort":7},{"id":191,"name":"水槽","parentId":183,"isParent":0,"sort":8},{"id":192,"name":"龙头","parentId":183,"isParent":0,"sort":9},{"id":193,"name":"淋浴花洒","parentId":183,"isParent":0,"sort":10},{"id":194,"name":"厨卫五
金","parentId":183,"isParent":0,"sort":11},{"id":195,"name":"家具五金","parentId":183,"isParent":0,"sort":12},{"id":196,"name":"门铃","parentId":183,"isParent":0,"sort":13},{"id":197,"name":"电气开关","parentId":183,"isParent":0,"sort":14},{"id":198,"name":"插座","parentId":183,"isParent":0,"sort":15},
{"id":199,"name":"电工电料","parentId":183,"isParent":0,"sort":16},{"id":200,"name":"监控安防","parentId":183,"isParent":0,"sort":17},{"id":201,"name":"电线/线缆","parentId":183,"isParent":0,"sort":18},{"id":202,"name":"数码","parentId":0,"isParent":1,"sort":4},{"id":203,"name":"摄影摄
像","parentId":202,"isParent":1,"sort":1},{"id":204,"name":"数码相机","parentId":203,"isParent":0,"sort":1},{"id":205,"name":"单电/微单相机","parentId":203,"isParent":0,"sort":2},{"id":206,"name":"单反相机","parentId":203,"isParent":0,"sort":3},{"id":207,"name":"摄像机","parentId":203,"isParent":0,"sort":4},
{"id":208,"name":"拍立得","parentId":203,"isParent":0,"sort":5},{"id":209,"name":"运动相机","parentId":203,"isParent":0,"sort":6},{"id":210,"name":"镜头","parentId":203,"isParent":0,"sort":7},{"id":211,"name":"户外器材","parentId":203,"isParent":0,"sort":8},{"id":212,"name":"影棚器
材","parentId":203,"isParent":0,"sort":9},{"id":213,"name":"冲印服务","parentId":203,"isParent":0,"sort":10},{"id":214,"name":"数码相框","parentId":203,"isParent":0,"sort":11},{"id":215,"name":"数码配件","parentId":202,"isParent":1,"sort":2},{"id":216,"name":"存储卡","parentId":215,"isParent":0,"sort":1},
{"id":217,"name":"读卡器","parentId":215,"isParent":0,"sort":2},{"id":218,"name":"滤镜","parentId":215,"isParent":0,"sort":3},{"id":219,"name":"闪光灯/手柄","parentId":215,"isParent":0,"sort":4},{"id":220,"name":"相机包","parentId":215,"isParent":0,"sort":5},{"id":221,"name":"三脚架/云
台","parentId":215,"isParent":0,"sort":6},{"id":222,"name":"相机清洁/贴膜","parentId":215,"isParent":0,"sort":7},{"id":223,"name":"机身附件","parentId":215,"isParent":0,"sort":8},{"id":224,"name":"镜头附件","parentId":215,"isParent":0,"sort":9},{"id":225,"name":"电池/充电
器","parentId":215,"isParent":0,"sort":10},{"id":226,"name":"移动电源","parentId":215,"isParent":0,"sort":11},{"id":227,"name":"数码支架","parentId":215,"isParent":0,"sort":12},{"id":228,"name":"智能设备","parentId":202,"isParent":1,"sort":3},{"id":229,"name":"智能手环","parentId":228,"isParent":0,"sort":1},
{"id":230,"name":"智能手表","parentId":228,"isParent":0,"sort":2},{"id":231,"name":"智能眼镜","parentId":228,"isParent":0,"sort":3},{"id":232,"name":"运动跟踪器","parentId":228,"isParent":0,"sort":4},{"id":233,"name":"健康监测","parentId":228,"isParent":0,"sort":5},{"id":234,"name":"智能配
饰","parentId":228,"isParent":0,"sort":6},{"id":235,"name":"智能家居","parentId":228,"isParent":0,"sort":7},{"id":236,"name":"体感车","parentId":228,"isParent":0,"sort":8},{"id":237,"name":"其他配件","parentId":228,"isParent":0,"sort":9},{"id":238,"name":"智能机器人","parentId":228,"isParent":0,"sort":10},
{"id":239,"name":"无人机","parentId":228,"isParent":0,"sort":11},{"id":240,"name":"影音娱乐","parentId":202,"isParent":1,"sort":4},{"id":241,"name":"MP3/MP4","parentId":240,"isParent":0,"sort":1},{"id":242,"name":"智能设备","parentId":240,"isParent":0,"sort":2},{"id":243,"name":"耳机/耳
麦","parentId":240,"isParent":0,"sort":3},{"id":244,"name":"便携/无线音箱","parentId":240,"isParent":0,"sort":4},{"id":245,"name":"音箱/音响","parentId":240,"isParent":0,"sort":5},{"id":246,"name":"高清播放器","parentId":240,"isParent":0,"sort":6},{"id":247,"name":"收音机","parentId":240,"isParent":0,"sort":7},
{"id":248,"name":"MP3/MP4配件","parentId":240,"isParent":0,"sort":8},{"id":249,"name":"麦克风","parentId":240,"isParent":0,"sort":9},{"id":250,"name":"专业音频","parentId":240,"isParent":0,"sort":10},{"id":251,"name":"苹果配件","parentId":240,"isParent":0,"sort":11},{"id":252,"name":"电子教
育","parentId":202,"isParent":1,"sort":5},{"id":253,"name":"学生平板","parentId":252,"isParent":0,"sort":1},{"id":254,"name":"点读机/笔","parentId":252,"isParent":0,"sort":2},{"id":255,"name":"早教益智","parentId":252,"isParent":0,"sort":3},{"id":256,"name":"录音笔","parentId":252,"isParent":0,"sort":4},
{"id":257,"name":"电纸书","parentId":252,"isParent":0,"sort":5},{"id":258,"name":"电子词典","parentId":252,"isParent":0,"sort":6},{"id":259,"name":"复读机","parentId":252,"isParent":0,"sort":7},{"id":260,"name":"虚拟商品","parentId":202,"isParent":1,"sort":6},{"id":261,"name":"延保服
务","parentId":260,"isParent":0,"sort":1},{"id":262,"name":"杀毒软件","parentId":260,"isParent":0,"sort":2},{"id":263,"name":"积分商品","parentId":260,"isParent":0,"sort":3},{"id":264,"name":"家居家装","parentId":0,"isParent":1,"sort":5},{"id":265,"name":"家纺","parentId":264,"isParent":1,"sort":1},
```

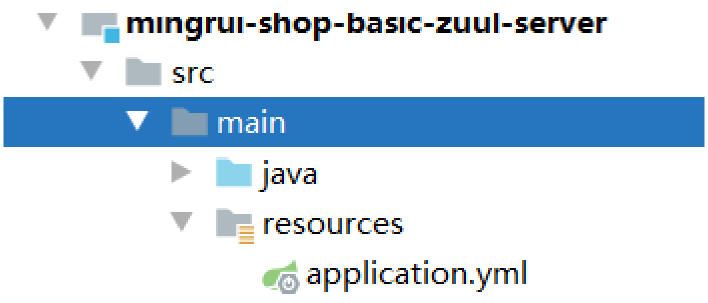
网管服务

在mingrui-shop-basic项目下新建mingrui-shop-basic-zuul-server



pom.xml

```xml
<!-- zuul -->
<dependency>
    <groupId>org.springframework.cloud</groupId>
    <artifactId>spring-cloud-starter-netflix-zuul</artifactId>
</dependency>
```

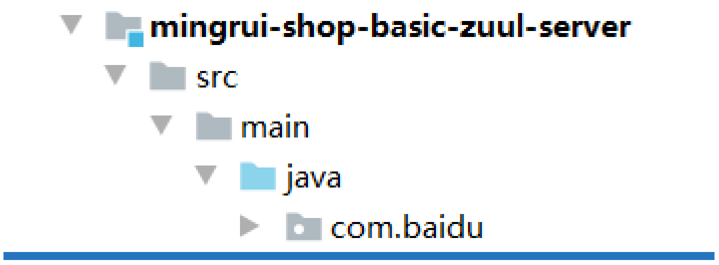


```
mingrui-shop-basic-zuul-server
  src
    main
      java
      resources
        application.yml
```

application.yml

```yaml
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-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/
```

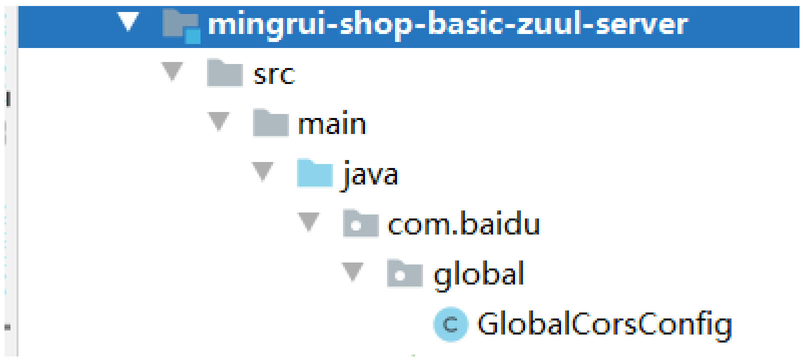
新建包com.baidu



新建启动类RunBasicZuulServer.class

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;
import org.springframework.cloud.netflix.zuul.EnableZuulProxy;

@SpringBootApplication
@EnableZuulProxy
@EnableEurekaClient
public class RunBasicZuulServer {
    public static void main(String[] args) {
        SpringApplication.run(RunBasicZuulServer.class);
    }
}
```

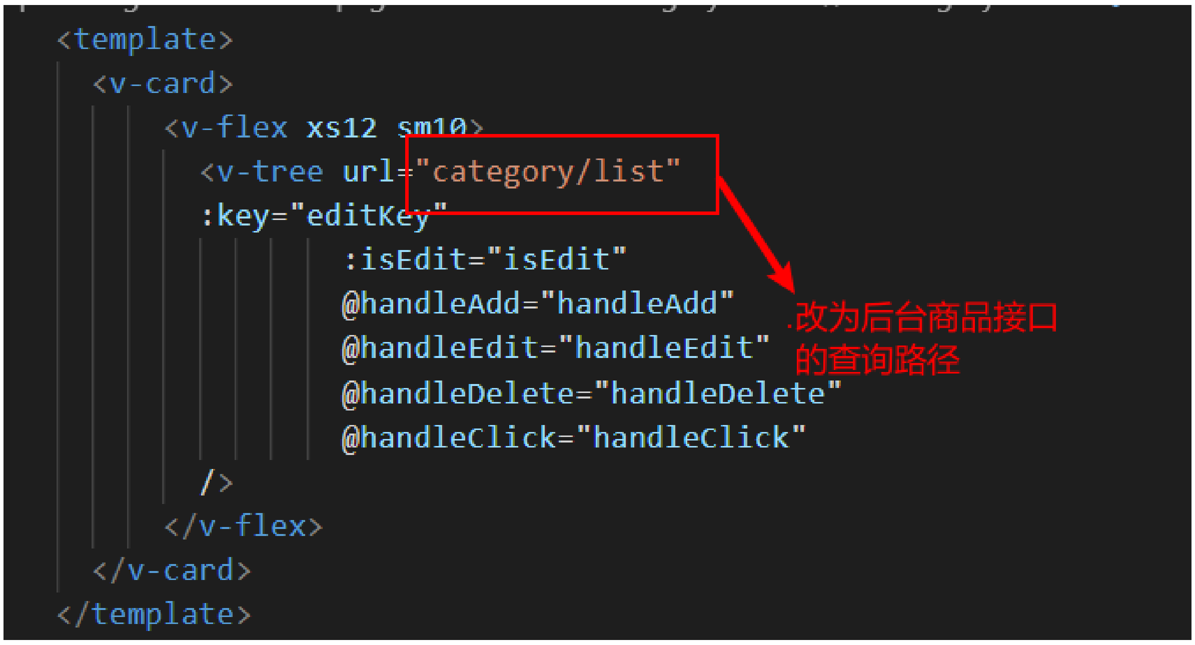
在com.baidu下新建global.GlobalCorsConfig



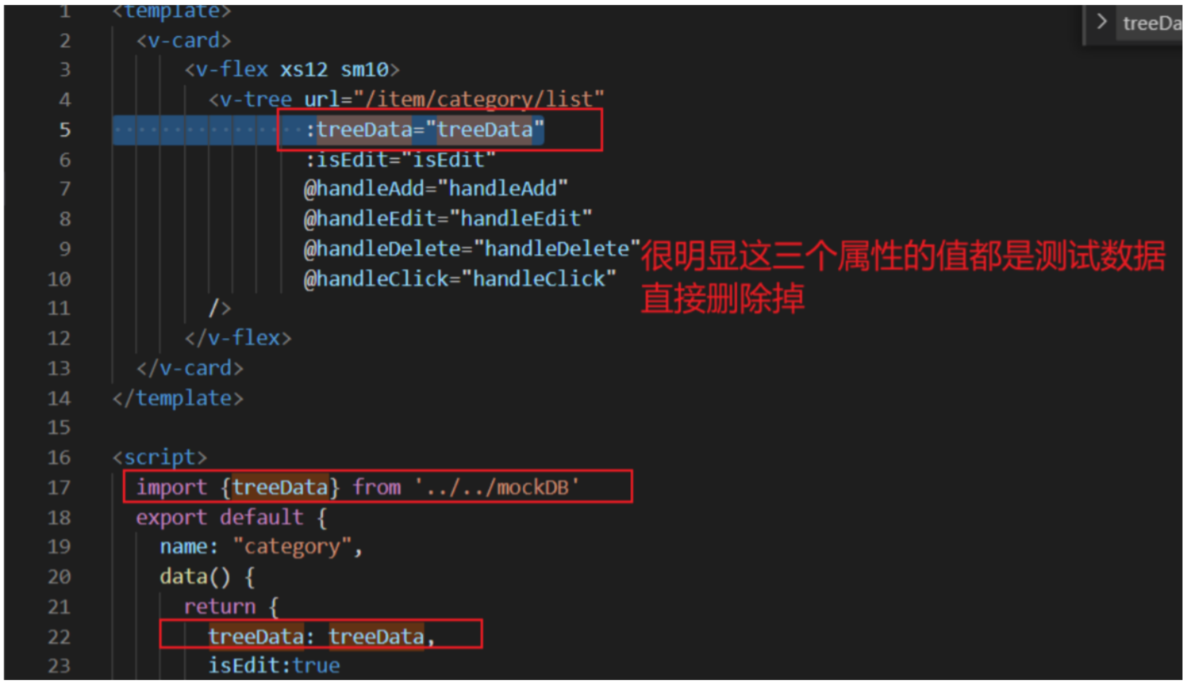
```

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;

@Configuration
public class GlobalCorsConfig {
    @Bean
    public CorsFilter corsFilter() {
        final UrlBasedCorsConfigurationSource source = new
            UrlBasedCorsConfigurationSource();
        final CorsConfiguration config = new CorsConfiguration();
        config.setAllowCredentials(true); // 允许cookies跨域
        config.addAllowedOrigin("*");// 允许向该服务器提交请求的URI，*表示全部允许。。这里尽量限制来源域，比如http://xxxx:8080 ,以降低安全风险。。
        config.addAllowedHeader("*");// 允许访问的头信息,*表示全部
        config.setMaxAge(18000L);// 预检请求的缓存时间（秒），即在这个时间段里，对于相同的跨域请求不会再预检了
        config.addAllowedMethod("*");// 允许提交请求的方法，*表示全部允许，也可以单独设置GET、PUT等
        config.addAllowedMethod("HEAD");
        config.addAllowedMethod("GET");// 允许Get的请求方法
        config.addAllowedMethod("PUT");
        config.addAllowedMethod("POST");
        config.addAllowedMethod("DELETE");
        config.addAllowedMethod("PATCH");
        source.registerCorsConfiguration("/**", config);//3.返回新的CorsFilter.
        return new CorsFilter(source);
    }
}
```

前台

Category.vue



```
<template>
  <v-card>
    <v-flex xs12 sm10>
      <v-tree url="category/list"
      :key="editKey"
              :isEdit="isEdit"
              @handleAdd="handleAdd"
              @handleEdit="handleEdit"
              @handleDelete="handleDelete"
              @handleClick="handleClick"
      />
    </v-flex>
  </v-card>
</template>
```



```
1   <template>
2     <v-card>
3       <v-flex xs12 sm10>
4         <v-tree url="/item/category/list"
5                 :treeData="treeData"
6                 :isEdit="isEdit"
7                 @handleAdd="handleAdd"
8                 @handleEdit="handleEdit"
9                 @handleDelete="handleDelete"
10                @handleClick="handleClick"
11        />
12      </v-flex>
13    </v-card>
14  </template>
15
16  <script>
17    import {treeData} from '../../mockDB'
18    export default {
19      name: "category",
20      data() {
21        return {
22          treeData: treeData,
23          isEdit:true
```

很明显这三个属性的值都是测试数据
直接删除掉

删除line:5和line:22

```
<template>
    <v-card>
        <v-flex xs12 sm10>
            <v-tree url="/category/list"
            :isEdit="isEdit"
            @handleAdd="handleAdd"
            @handleEdit="handleEdit"
            @handleDelete="handleDelete"
            @handleClick="handleClick"
            />
        </v-flex>
    </v-card>
</template>
<script>
    //import {treeData} from '../../mockDB'
    export default {
        name: "category",
            data() {
                return {
                isEdit:true
                }
            },
        methods: {
            handleAdd(node) {
                console.log("add .... ");
                console.log(node);
            },
            handleEdit(id, name) {
                console.log("edit... id: " + id + ", name: " + name)
            },
            handleDelete(id) {
                console.log("delete ... " + id)
            },
            handleClick(node) {
                console.log(node)
                3.5.2 Tree.vue
```

```
                  getData方法重新制定resp回调
              }
          }
      };
  </script>
  <style scoped>
  </style>
```

Tree.vue

```
<template>
  <v-list class="pt-0 pb-0" dense>
    <TreeItem
      class="item" :model="model" v-for="(model, index) in db" :key="index"
      :url="url"
      :isEdit="isEdit"
      :nodes="nodes"
      @handleAdd="handleAdd"
      @handleEdit="handleEdit"
      @handleDelete="handleDelete"
      @handleClick="handleClick"
    />
  </v-list>
</template>

<script>
  import TreeItem from './TreeItem';

  export default {
    name: "vTree",
    props: {
      url: String,
      isEdit: {
        type: Boolean,
        default: false
      },
      treeData:{
        type:Array
      }
    },
    data() {
      return {
        db: [],
        nodes:{
          opened:null,
          selected:{isSelected:false}
        }
      }
    },
    components: {
      TreeItem
    },
    created() {
      if(this.treeData && this.treeData.length > 0){
        this.db = this.treeData;
        return;
      }
      this.getData();
    },
    methods: {
      getData() {
        this.$http.get(this.url, {params: {pid: 0}}).then(resp => {
          console.log(resp)
          this.db = resp.data.data;
          this.db.forEach(n => n['path'] = [n.name])
        })
      },
      handleAdd(node) {
        this.$emit("handleAdd", this.copyNodeInfo(node));
      },
      handleEdit(id, name) {
        this.$emit("handleEdit", id, name)
      },
      handleDelete(id) {
        this.deleteById(id, this.db);
        this.$emit("handleDelete", id);
      },
      handleClick(node){
        this.$emit("handleClick", this.copyNodeInfo(node))
      },
      // 根据id删除
      deleteById(id, arr) {
        let src = arr & this.db;
        for (let i in src) {
          let d = src[i];
          if (d.id === id) {
            src.splice(i, 1)
            return;
          }
          if (d.children && d.children.length > 0) {
            this.deleteById(id, d.children)
          }
        }
      },
      copyNodeInfo(node){
        const o = {};
        for(let i in node){
          o[i] = node[i];
        }
        return o;
      }
    },
    watch: {}
  }
</script>

<style scoped>
  .item {
    cursor: pointer;
  }
</style>
```

TreeItem.vue

```
<template>
  <div>
    <v-list-tile
      @click="toggle" class="level1 pt-0 pb-0 mt-0 mb-0" :class="
{'selected':isSelected}">
      <v-list-tile-avatar>
        <v-icon v-if="model.isParent">{{open ? 'folder_open' : 'folder'}}</v-
icon>
        <v-icon v-if="!model.isParent">insert_drive_file</v-icon>
      </v-list-tile-avatar>
      <v-list-tile-content>
        <v-list-tile-title v-show="!beginEdit">
          <span >{{model.name}}</span>
        </v-list-tile-title>
        <input v-show="beginEdit" @click.stop="" :ref="model.id" v-
model="model.name"
               @blur="afterEdit" @keydown.enter="afterEdit"/>
      </v-list-tile-content>
      <v-list-tile-action v-if="isEdit">
        <v-btn icon @mouseover="c1='primary'" @mouseout="c1=''" :color="c1"
@click.stop="addChild">
          <i class="el-icon-plus"/>
        </v-btn>
      </v-list-tile-action>
      <v-list-tile-action v-if="isEdit">
        <v-btn icon @mouseover="c2='success'" @mouseout="c2=''" :color="c2"
@click.stop="editChild">
          <i class="el-icon-edit"/>
        </v-btn>
      </v-list-tile-action>
      <v-list-tile-action v-if="isEdit">
        <v-btn icon @mouseover="c3='error'" @mouseout="c3=''" :color="c3"
@click.stop="deleteChild">
          <i class="el-icon-delete"/>
        </v-btn>
      </v-list-tile-action>
    </v-list-tile>

    <v-list v-if="isFolder" v-show="open" class="ml-4 pt-0 pb-0" dense
transition="scale-transition">
      <tree-item
        class="item"
        v-for="(model, index) in model.children"
        :key="index"
        :model="model"
        :url="url"
        :isEdit="isEdit"
        :nodes="nodes"
        :parentState="open"
        @handleAdd="handleAdd"
        @handleEdit="handleEdit"
        @handleDelete="handleDelete"
        @handleClick="handleClick"
      >
      </tree-item>
    </v-list>
  </div>
```

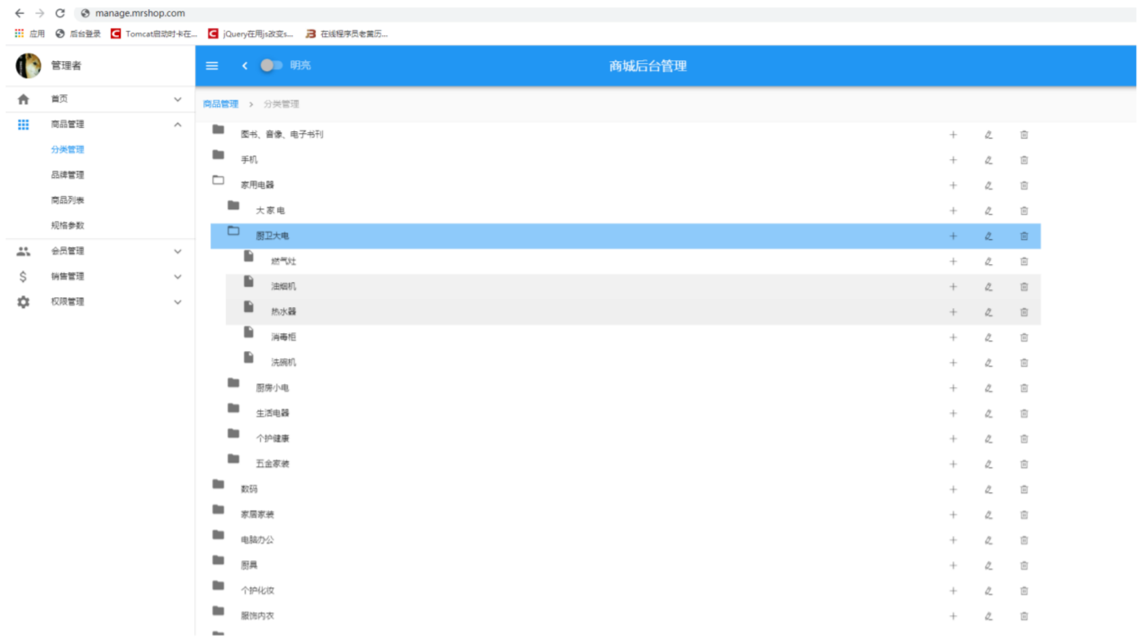
```
</template>

<script>
  import Vue from 'vue'

  export default {
    name: "tree-item",
    props: {
      model: Object,
      url: String,
      isEdit: {
        type: Boolean,
        default: false
      },
      nodes: Object,
      parentState:Boolean
    },
    created() {
    },
    data: function () {
      return {
        c1: '',
        c2: '',
        c3: '',
        isSelected: false,
        open:false,
        beginEdit:false
      }
    },
    watch:{
      parentState(val){
        if(!val){
          this.open = val;
        }
      }
    },
    computed: {
      isFolder: function () {
        return this.model.children &&
          this.model.children.length > 0
      }
    },
    methods: {
      toggle: function () {
        // 将其它被选中项取消选中
        this.nodes.selected.isSelected = false;
        // 当前项被选中
        this.isSelected = true;
        // 保存当前选中项
        this.nodes.selected = this

        // 客户自己的点击事件回调
        this.handleClick(this.model);

        // 判断是否为顶级节点，顶级节点需要记录和替换
        if(this.model.parentId == 0){
          // 判断打开节点是否是自己
          if(this.nodes.opened && this != this.nodes.opened){
```

```
          // 不是，则关闭原来的节点
          this.nodes.opened.open = false;
        }
        // 将自己记录为打开的节点
        this.nodes.opened = this;
      }
      // 切换开闭状态
      this.open = !this.open;
      // 如果已经是叶子节点,或者自己是关闭的，或者自己已经有儿子了，结束
      if (!this.model.isParent || this.isFolder || !this.open) {
        return;
      }
      // 展开后查询子节点
      this.$http.get(this.url, {params: {pid: this.model.id}})
        .then(resp => {
        Vue.set(this.model, 'children', resp.data.data);
        // 封装当前节点的路径
        this.model.children.forEach(n => {
          n['path'] = [];
          this.model.path.forEach(p => n['path'].push(p));
          n['path'].push(n.name);
        });
      }).catch( e => {
        console.log(e);
      });
    },
    addChild: function () {
      let child = {
        id: 0,
        name: '新的节点',
        parentId: this.model.id,
        isParent: false,
        sort:this.model.children? this.model.children.length + 1:1
      }
      if (!this.model.isParent) {
        Vue.set(this.model, 'children', [child]);
        this.model.isParent = true;
        this.open = true;
        this.handleAdd(child);
      } else {
        if (!this.isFolder) {
          this.$http.get(this.url, {params: {pid: this.model.id}}).then(resp
=> {
            Vue.set(this.model, 'children', resp.data);
            this.model.children.push(child);
            this.open = true;
            this.handleAdd(child);
          });
        } else {
          this.model.children.push(child);
          this.open = true;
          this.handleAdd(child);
        }
      }
    },
    deleteChild: function () {
      this.$message.confirm('此操作将永久删除数据，是否继续?', '提示', {
        confirmButtonText: '确定删除',
        cancelButtonText: '取消',
        type: 'warning'
      }).then(() => {
        this.handleDelete(this.model.id);
      }).catch(()=>{
        this.$message.info('已取消删除');
      })

    },
    editChild() {
      this.beginEdit = true;
      this.$nextTick(() => this.$refs[this.model.id].focus());
    },
    afterEdit() {
      if (this.beginEdit) {
        this.beginEdit = false;
        this.handleEdit(this.model.id, this.model.name);
      }
    },
    handleAdd(node) {
      this.$emit("handleAdd", node);
    },
    handleDelete(id) {
      this.$emit("handleDelete", id);
    },
    handleEdit(id, name) {
      this.$emit("handleEdit", id, name)
    },
    handleClick(node) {
      this.$emit("handleClick", node);
    }
  }
}
</script>

<style scoped>
.level1 {
  height: 40px;
}

.selected {
  background-color: rgba(105,184,249,0.75);
}
</style>
```

依次启动eureka-server,xxx-service,zuul-server,vue项目 保证自己的hosts,nginx没有问题

浏览器输入

```
http://manage.mrshop.com/
```



3.商品删除

后台

mingrui-shop-service-api-xxx

service：CategoryService

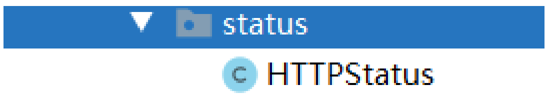
```
@ApiOperation(value = "删除商品分类")
@DeleteMapping(value = "category/del")
Result<JsonObject> delCategory(Integer id);
```

mingrui-shop-common-core

util：新建ObjectUtil(让代码更优雅)

```
package com.baidu.shop.utils;

public class ObjectUtil {
    public static  Boolean isNull(Object obj){
        return null == obj;
    }
    public static Boolean isNotNull(Object obj){
        return null != obj;
    }
}
```

mingrui-shop-service-api-xxx新建com.baidu.shop.status



HTTPStatus

```
package com.baidu.shop.status;

public class HTTPStatus {
    public static final int OK = 200;//成功
    public static final int ERROR = 500;//失败
}
```

实现类中要做的事:

 通过当前id查询分类信息

 判断是否有数据(安全)

判断当前节点是否是父级节点(安全)

判断当前节点的父节点下 除了当前节点是否还有别的节点(业务)

    没有:将当前节点的父节点isParent的值修改为0

通过id删除数据

代码实现CategoryServiceImpl

```
//商品删除
@Transactional//事务 增删改都需要
@Override
public Result<JsonObject> delCategory(Integer id) {

    String msg = "";

    System.out.println(id);
    //判断页面传递过来的id是否合法
    if(null != id && id >0){
        //通过id查询当前节点信息
        CategoryEntity categoryEntity = categoryMapper.selectByPrimaryKey(id);

        //判断当前节点是否为父节点(安全!)
        if (categoryEntity.getParentId() == 1)return this.setResultError("当前是父
节点不能删除");//return之后的代码不会执行

        //如果当前分类被品牌绑定的话不能被删除 --> 通过分类id查询中间表是否有数据 true :
当前分类不能被删除 false:继续执行
        Example example1 = new Example(CategoryBrandEntity.class);
        example1.createCriteria().andEqualTo("categoryId",id);
        List<CategoryBrandEntity> categoryBrandEntities =
categoryBrandMapper.selectByExample(example1);
        if (categoryBrandEntities.size() > 0){
            return this.setResultError("当前分类被品牌绑定不能被删除 ");
        }


        //通过当前节点的父节点id 查询 当前节点(将要被删除的节点)的父节点下是否还有其他子节点
        Example example = new Example(CategoryEntity.class);

 example.createCriteria().andEqualTo("parentId",categoryEntity.getParentId());
        List<CategoryEntity> categoryList =
categoryMapper.selectByExample(example);
```

```
        //如果size <= 1 --> 如果当前节点被删除的话 当前节点的父节点下没有节点了 --> 将当前节点的父节点状态改为叶子节点
        if (categoryList.size() <= 1){
            CategoryEntity updateCategoryEntity= new CategoryEntity();
            updateCategoryEntity.setParentId(0);
            updateCategoryEntity.setId(categoryEntity.getParentId());

            categoryMapper.updateByPrimaryKeySelective(updateCategoryEntity);
        }
        categoryMapper.deleteByPrimaryKey(id);
        return this.setResultSuccess();
    }
    return this.setResultError("id不合理");
}
```

前台

TreeItem.vue : line 164

```
deleteChild: function () {
        if(this.model.isParent == 1){
          this.$message.warning('当前为父节点,不能删除');
          return;
        }
        this.$message.confirm('此操作将永久删除数据, 是否继续?', '提示', {
          confirmButtonText: '确定删除',
          cancelButtonText: '取消',
          type: 'warning'
        }).then(() => {
          this.handleDelete(this.model.id);
        }).catch(()=>{
          this.$message.info('已取消删除');
        })

      },
```

Category.vue

```
handleDelete(id) {
        this.$http.delete('./category/del?id=' + id).then(resp=>{

            if(resp.data.code != 200){
              this.$message.error('删除失败: ' + resp.data.message);
              return;
            }

            this.editKey = new Date().getTime();
            this.$message.success('删除成功')
        }).catch(error => console.log(error))
      },
```

注意需要给v-tree组件绑定一个key的属性,默认值为0

```
<v-tree :key="key" url="/category/list">
    data() {
        return {
            key:0
        }
    }
```

4.商品修改

后台

mingrui-shop-service-api-xxx

service：CategoryService

```
@ApiOperation(value = "修改商品分类")
@PutMapping(value = "category/edit")
Result<JsonObject> editCategory( @RequestBody CategoryEntity entity);
```

mingrui-shop-service-xxx

代码实现CategoryServiceImpl

```
//修改商品分类
@Transactional
@Override
public Result<JsonObject> editCategory(CategoryEntity entity) {
    try {
        categoryMapper.updateByPrimaryKeySelective(entity);
    } catch (Exception e) {
        e.printStackTrace();
    }
    return this.setResultSuccess();
}
```

前台

将TreeItem.vue的afterEdit方法this.model.beginEdit改为 this.beginEdit

```
 afterEdit() {
        if (this.beginEdit) {
          this.beginEdit = false;
          this.handleEdit(this.model.id, this.model.name);
        }
      },
```

Category.vue

```
 handleEdit(id, name) {
        console.log("edit... id: " + id + ", name: " + name)
        this.$http.put('./category/edit',{
          id:id,
          name:name
        }).then(resp=>{
          this.$message.success('修改成功')
          this.editKey = new Date().getTime();
        }).catch(error => console.log(error))
      },
```

5.商品新增

后台

mingrui-shop-service-api-xxx

service：CategoryService

```
@ApiOperation(value = "增加商品分类")
@PostMapping(value = "category/add")
Result<JsonObject> addCategory(@RequestBody CategoryEntity entity);
```

mingrui-shop-service-xxx

实现代码：CategoryServiceImpl

```
//新增商品分类
@Transactional
@Override
public Result<JsonObject> addCategory(CategoryEntity entity) {

    CategoryEntity parentCategoryEntity = new CategoryEntity();
    parentCategoryEntity.setId(entity.getParentId());
    parentCategoryEntity.setIsParent(1);
    categoryMapper.updateByPrimaryKeySelective(parentCategoryEntity);

    categoryMapper.insertSelective(entity);

    return this.setResultSuccess();
}
```

前台

Category.vue

```
handleAdd(node) {
        console.log("add .... ");


        //处理boolean值
        node.isParent = node.isParent?1:0;
        //删除id属性
        delete node.id;
                console.log(node);
```

```
         //新增
         this.$http.post('./category/add',node).then(resp=>{
             this.$message.success('新增成功')
             this.editKey = new Date().getTime();
         }).catch(error=>console.log(error))
       },
```

项目不完美

参数校验

common-core工程新建包com.baidu.shop.validate.group

包下新建操作类MingruiOperation

```
package com.baidu.shop.validate.group;

public class MingruiOperation {
    public interface Add{}
    public interface Update{}
    public interface Search{}
}
```

mingrui-shop-service-api-xxx

com.baidu.shop.entity

```
import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.persistence.Id;
import javax.persistence.Table;
import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@ApiModel(value = "分类实体类")
@Data
@Table(name = "tb_category")
public class CategoryEntity {
    @Id
    @ApiModelProperty(value = "分类主键",example = "1")
    @NotNull(message = "ID不能为空",groups = {MingruiOperation.Update.class})
    private Integer id;

    @ApiModelProperty(value = "分类名称")
    @NotEmpty(message = "分类名称不能为空",groups =
{MingruiOperation.Add.class,MingruiOperation.Update.class})
    private String name;

    @ApiModelProperty(value = "父级分类",example = "1")
    @NotNull(message = "父类ID不能为空",groups = {MingruiOperation.Update.class})
    private Integer parentId;

    @ApiModelProperty(value = "是否是父级节点",example = "1")
    @NotNull(message = "是否为父节点不能为空",groups =
{MingruiOperation.Add.class})
    private Integer isParent;

    @ApiModelProperty(value = "排序",example = "1")
    @NotNull(message = "排序字段不能为空",groups = {MingruiOperation.Add.class})
    private Integer sort;
}
```

service :

```
package com.baidu.shop.service;

import com.baidu.shop.base.Result;
import com.baidu.shop.entity.CategoryEntity;
import com.baidu.shop.validate.group.MingruiOperation;
import com.google.gson.JsonObject;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.models.auth.In;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import java.util.List;


@Api(tags = "商品分类接口")
public interface CategoryService {
    @ApiOperation(value = "通过查询商品分类")
    @GetMapping(value = "category/list")
    Result<List<CategoryEntity>> getCategoryByPid(Integer pid);

    @ApiOperation(value = "删除商品分类")
    @DeleteMapping(value = "category/del")
    Result<JsonObject> delCategory(Integer id);

    @ApiOperation(value = "修改商品分类")
    @PutMapping(value = "category/edit")
    Result<JsonObject> editCategory(@Validated({MingruiOperation.Update.class})
@RequestBody CategoryEntity entity);

    @ApiOperation(value = "增加商品分类")
    @PostMapping(value = "category/add")
    Result<JsonObject>
addCategory(@Validated({MingruiOperation.Add.class})@RequestBody CategoryEntity
entity);

    @ApiOperation(value = "通过品牌id查询商品分类")
    @GetMapping(value = "category/getByBrand")
    Result<List<CategoryEntity>> getByBrand (Integer brandId);
}
```

全局异常处理

mingrui-shop-common-core

pom.xml

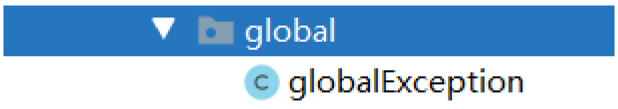
```
<!-- SpringBoot-整合Web组件 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

com.baidu.shop.staus:HTTPStatus

```
public static final int PARAMS_VALIDATE_ERROR = 5002;//参数校验失败
```

新建包com.baidu.shop.global

新建全局异常处理类GlobalException



```
package com.baidu.shop.global;

import com.baidu.shop.base.Result;
import com.baidu.shop.status.HTTPStatus;
import com.google.gson.JsonObject;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.MethodArgumentNotValidException;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.bind.annotation.RestControllerAdvice;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.stream.Collectors;

@RestControllerAdvice
@Slf4j
public class globalException {
    @ExceptionHandler(value = RuntimeException.class)
    public Result<JsonObject> testException(RuntimeException e){
        log.error("code : {}, message : {}", HTTPStatus.ERROR,e.getMessage());
        return new Result<JsonObject>(HTTPStatus.ERROR,e.getMessage(),null);
    }
    @ExceptionHandler(value = MethodArgumentNotValidException.class)
    public HashMap<String, Object> methodValid(MethodArgumentNotValidException
exception) throws Exception{
        //== ===区别？？
        HashMap<String, Object> map = new HashMap<>();
        map.put("code",HTTPStatus.PARAMS_VALIDATE_ERROR);

       List<String> msgList = new ArrayList<>();

       exception.getBindingResult().getFieldErrors().stream().forEach(error ->{
            msgList.add("Field -->" + error.getField() + ":" +
error.getDefaultMessage());
            log.error("Field -->" + error.getField() + ":" +
error.getDefaultMessage());
       });
        String message =
msgList.parallelStream().collect(Collectors.joining(","));

        map.put("massage",message);

        return map;
    }
}
```

三，品牌管理

1.品牌查询

后台

mingrui-shop-common-core

pom.xml

```
<!--帮助开发人员快速生成API文档-->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>
```

com.baidu.shop.validate.group

MingruiOperation

```
public interface Search{}
```

在com.baidu.shop.base包下新建BaseDTO

```
package com.baidu.shop.base;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

@Data
@ApiModel(value = "BaseDTO用于数据传输,其他dto需要继承此类")
public class BaseDTO {

    @ApiModelProperty(value = "当前页",example = "1")
    private Integer page;

    @ApiModelProperty(value = "每页显示多少条",example = "5")
    private Integer rows;

    @ApiModelProperty(value = "排序字段")
    private String sort;

    @ApiModelProperty(value = "是否升序")
    private String order;

    public String getOrder() {
        return sort + " " + (Boolean.valueOf(order) ? "desc":"asc");
    }
}
```

mingrui-shop-service-api

pom.xml

```
<!--分页工具-->
<dependency>
    <groupId>com.github.pagehelper</groupId>
    <artifactId>pagehelper-spring-boot-starter</artifactId>
    <version>1.2.10</version>
</dependency>
```

mingrui-shop-service-api-xxx

在com.baidu.shop下新建dto包

新建BrandDto

```
package com.baidu.shop.dto;

import com.baidu.shop.base.BaseDTO;
import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@Data
@ApiModel(value = "品牌DTO")
public class BrandDto extends BaseDTO {

    @ApiModelProperty(value = "品牌ID",example = "1")
    @NotNull(message = "主键不能为空",groups =
{MingruiOperation.Add.class,MingruiOperation.Update.class})
    private Integer id;

    @ApiModelProperty(value = "品牌名称",example = "1")
    @NotEmpty(message = "品牌名称不能为空",groups = {MingruiOperation.Add.class,
MingruiOperation.Update.class})
    private String name;

    @ApiModelProperty(value = "品牌图片")
    private String image;
    @ApiModelProperty(value = "品牌首字母")
    private Character letter;
}
```

com.baidu.shop.service包下新建BrandService

```
package com.baidu.shop.service;

import com.baidu.shop.base.Result;
import com.baidu.shop.dto.BrandDTO;
import com.baidu.shop.entity.BrandEntity;
import com.github.pagehelper.PageInfo;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.List;

@Api(tags = "品牌接口")
public interface BrandService {
    @GetMapping(value = "brand/list")
    @ApiOperation(value = "查询品牌列表")
    Result<List<BrandEntity>> getBrandInfo(BrandDto brandDto);
}
```

mingrui-shop-service-xxx

在mapper包下新建BrandMapper

```
package com.baidu.shop.mapper;

import com.baidu.shop.entity.BrandEntity;
import tk.mybatis.mapper.common.Mapper;

public interface BrandMapper extends Mapper<BrandEntity> {
}
```

com.baidu.shop.service.impl

新建BrandServiceImpl

```
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.BrandDTO;
import com.baidu.shop.entity.BrandEntity;
import com.baidu.shop.mapper.BrandMapper;
import com.baidu.shop.service.BrandService;
import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.RestController;
import tk.mybatis.mapper.entity.Example;
import javax.annotation.Resource;
import java.util.List;

@RestController
```

```
public class BrandServiceImpl extends BaseApiService implements BrandService{

    @Autowired
    private BrandMapper brandMapper;

    //查询品牌
    @Override
    public Result<List<BrandEntity>> getBrandInfo(BrandDto brandDto) {
        //mybatis如何自定义分页插件 --》 mybatis执行器
        PageHelper.startPage(brandDto.getPage(),brandDto.getRows());
        if
(!StringUtils.isEmpty(brandDto.getSort()))PageHelper.orderBy(brandDto.getOrder()
);

        BrandEntity brandEntity = BaiduBeanUtil.copyProperties(brandDto,
BrandEntity.class);

        Example example = new Example(BrandEntity.class);
        example.createCriteria().andLike("name","%" + brandEntity.getName() +
"%");

        List<BrandEntity> brandEntities = brandMapper.selectByExample(example);
        PageInfo<BrandEntity> pageInfo = new PageInfo<>(brandEntities);
        return this.setResultSuccess(pageInfo);
    }
```

前台

官网地址:

表格:https://v15.vuetifyjs.com/zh-Hans/components/data-tables

输入框:https://v15.vuetifyjs.com/zh-Hans/components/snackbars

按钮:https://v15.vuetifyjs.com/zh-Hans/components/buttons

在item包下新建MrBrand.vue

```
<template>
  <v-card>
    <v-card-title>
      <v-btn color="info" @click="dialog = true">新增</v-btn>
      <div class="text-xs-center">
        <v-dialog v-model="dialog" width="500">
          <v-card>
            <v-card-title class="headline grey lighten-2" primary-title>
            品牌 {{isEdit?'修改':'新增'}}
           </v-card-title>
            <mr-brand-form @closeDialog="closeDialog" :dialog="dialog"
:isEdit="isEdit" :brandDetail="brandDetail"/>
          </v-card>
        </v-dialog>
      </div>
      <!-- 调按钮和输入框之间的间距 -->
      <v-spacer />
      <!--append-icon : 图标 label : input默认值-->
      <v-text-field
        append-icon="search"
```

```
          label="品牌名称"
          @keyup.enter="getTableData()"
          v-model="search"
        ></v-text-field>
      </v-card-title>
      <v-data-table
        :headers="headers"
        :items="desserts"
        :pagination.sync="pagination"
        :total-items="total"
        class="elevation-1"
      >
        <!-- 注意此处不能只能用官网的属性,应当参照Brand.vue的写法 -->
        <template slot="items" slot-scope="props">
          <td class="text-xs-center">{{ props.item.id }}</td>
          <td class="text-xs-center">{{ props.item.name }}</td>
          <!-- :src 给元素绑定src属性 属性的值为props.item.image -->
          <td class="text-xs-center">
            <img width="200" :src="props.item.image" />
            </td>
          <td class="text-xs-center">{{ props.item.letter }}</td>
          <td class="text-xs-center">
              <v-btn falt icon  @click="editDialog(props.item)" color="green">
                  <v-icon>edit</v-icon>
              </v-btn>
              <v-btn falt icon @click="delDialog(props.item.id)" color="red">
                  <v-icon>delete</v-icon>
              </v-btn>
          </td>
        </template>
      </v-data-table>
    </v-card>
</template>
<script>
import MrBrandForm from './MrBrandForm';
export default {
  name: "MrBrand",
  components:{
    MrBrandForm
  },
  data() {
    return {
      brandDetail:{},
      isEdit:false,
      dialog:false,
      pagination: {},
      total: 0,
      search: "",
      headers: [
        {
          text: "id",
          align: "center",
          value: "id",
        },
        {
          text: "品牌名称",
          align: "center",
          value: "name",
        },
        {
          text: "品牌logo",
          align: "center",
          value: "image",
        },
        {
          text: "首字母",
          align: "center",
          value: "letter",
        },
        {
          text: "操作",
          align: "center",
          sortable: false,
          value: "id",
        }
      ],
      desserts: [],
    };
  },
  mounted() {
    this.getTableData();
  },
  methods: {
    closeDialog(){
      this.dialog = false;
      this.getTableData()
    },
    addDialog(){
      this.isEdit = false;
      this.dialog = true;
    },
    editDialog(obj){
      this.brandDetail = obj;
      this.isEdit = true;
      this.dialog = true;
    },
    delDialog(id){
      this.$http.delete('brand/del?id=' + id).then(resp =>{
        if(resp.data.code == 200){
            this.getTableData()
            this.$message.success('删除成功')
        }else{
          this.$message.error(resp.data.msg)
        }
      }).catch(error => console.log(error))
    },
    getTableData() {
      this.$http
        .get("/brand/list", {
          params: {
            page: this.pagination.page,
            rows: this.pagination.rowsPerPage,
            sort: this.pagination.sortBy,
            order: this.pagination.descending,
            name: this.search,
          },
        })
        .then((resp) => {
          this.desserts = resp.data.data.list;
          this.total = resp.data.data.total;
        })
        .catch((error) => console.log(error));
    },
  },
  watch: {
    pagination() {
      this.getTableData();
    },
  },
};
</script>
```

index.js line : 27

```
route("/item/brand",'/item/MrBrand',"MrBrand"),
```

2.品牌新增

后台

mingrui-shop-common-core

在utils包下新建BaiduBeanUtil

```
package com.baidu.shop.utils;

import org.springframework.beans.BeanUtils;

public class BaiduBeanUtil<T>{
    public static <T> T copyProperties(Object source,Class<T> tClass) {
        T t=null;
        try {
            t = tClass.newInstance();//创建实例
            BeanUtils.copyProperties(source, t);
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
        return t;
    }
}
```

mingrui-shop-service-api-xxx

com.baidu.shop.dto BrandDto

```
@ApiModelProperty(value = "品牌分类信息")
@NotEmpty(message = "品牌分类信息不能为空",groups = {MingruiOperation.Add.class})
private String categories;
```

com.baidu.shop.entity包下新建CategoryBrandEntity

```
package com.baidu.shop.entity;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import javax.persistence.Table;

@Data
@Table(name = "tb_category_brand")
@NoArgsConstructor
@AllArgsConstructor
public class CategoryBrandEntity {
    private Integer categoryId;

    private Integer brandId;
}
```

com.baidu.shop.service：BrandService

```
@PostMapping(value = "brand/save")
@ApiOperation(value = "新增品牌")
Result<JsonObject> save(@RequestBody BrandDto brandDto);
```

mingrui-shop-service-xxx

在mapper包下新建CategoryBrandMapper

```
package com.baidu.shop.mapper;

import com.baidu.shop.entity.CategoryBrandEntity;
import tk.mybatis.mapper.common.Mapper;
import tk.mybatis.mapper.common.special.InsertListMapper;

public interface CategoryBrandMapper extends
InsertListMapper<CategoryBrandEntity>, Mapper<CategoryBrandEntity> {
}
```

com.baidu.shop.service.impl：BrandServiceImpl

```
//新增品牌
    @Transactional
    @Override
    public Result<JsonObject> save(BrandDto brandDto) {

        //新增返回主键?
        //两种方式实现 select-key insert加两个属性
        BrandEntity brandEntity =
BaiduBeanUtil.copyProperties(brandDto,BrandEntity.class);
        //处理品牌首字母

 brandDto.setLetter(PinyinUtil.getUpperCase(String.valueOf(brandDto.getName().ch
arAt(0)),PinyinUtil.TO_FUUL_PINYIN).charAt(0));
```

```
        brandMapper.insertSelective(brandEntity);
        //维护中间表数据

 this.insetrCategoryBrandList(brandDto.getCategories(),brandEntity.getId());
        return this.setResultSuccess();
    }
```

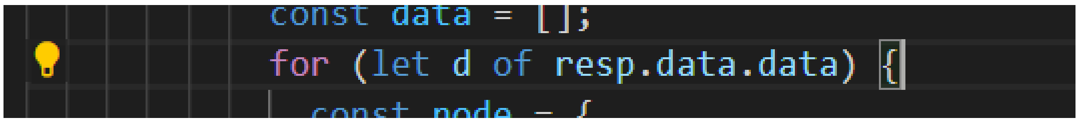
```
//新增修改整合
private void insetrCategoryBrandList (String categories,Integer brandId){
    if (StringUtils.isEmpty(categories))throw new RuntimeException("分类信息不能为
空");

    //判断分类集合字符串中是否包含,
    if(categories.contains(",")){
        categoryBrandMapper.insertList(
            //数组转list
            Arrays.asList(categories.split(","))
                    //获取stream流, 流: 对一次数据进行操作
                    .stream()
                    //遍历list中所有数据
                    .map(categoryById->new
CategoryBrandEntity(Integer.valueOf(categoryById),brandId))
                    //最终转换成list
                    .collect(Collectors.toList())
        );
    }else{
        CategoryBrandEntity categoryBrandEntity = new CategoryBrandEntity();
        categoryBrandEntity.setCategoryId(Integer.valueOf(categories));
        categoryBrandEntity.setBrandId(brandId);

        categoryBrandMapper.insertSelective(categoryBrandEntity);
    }
}
```

前台

```
<template>
  <v-card>
    <v-card-title>
      <v-btn color="info" @click="dialog = true">新增</v-btn>
      <div class="text-xs-center">
        <v-dialog v-model="dialog" width="500">
          <v-card>
            <v-card-title class="headline grey lighten-2" primary-title>
            品牌 {{isEdit?'修改':'新增'}}
           </v-card-title>
            <mr-brand-form @closeDialog="closeDialog" :dialog="dialog"
:isEdit="isEdit" :brandDetail="brandDetail"/>
          </v-card>
        </v-dialog>
      </div>
      <!-- 调按钮和输入框之间的间距 -->
      <v-spacer />
      <!--append-icon : 图标 label : input默认值-->
      <v-text-field
```

```
          append-icon="search"
          label="品牌名称"
          @keyup.enter="getTableData()"
          v-model="search"
        ></v-text-field>
      </v-card-title>
      <v-data-table
        :headers="headers"
        :items="desserts"
        :pagination.sync="pagination"
        :total-items="total"
        class="elevation-1"
      >
        <!-- 注意此处不能只能用官网的属性,应当参照Brand.vue的写法 -->
        <template slot="items" slot-scope="props">
          <td class="text-xs-center">{{ props.item.id }}</td>
          <td class="text-xs-center">{{ props.item.name }}</td>
          <!-- :src 给元素绑定src属性 属性的值为props.item.image -->
          <td class="text-xs-center">
            <img width="200" :src="props.item.image" />
            </td>
          <td class="text-xs-center">{{ props.item.letter }}</td>
          <td class="text-xs-center">
              <v-btn falt icon  @click="editDialog(props.item)" color="green">
                  <v-icon>edit</v-icon>
              </v-btn>
              <v-btn falt icon @click="delDialog(props.item.id)" color="red">
                  <v-icon>delete</v-icon>
              </v-btn>
          </td>
        </template>
      </v-data-table>
    </v-card>
  </template>
  <script>
  import MrBrandForm from './MrBrandForm';
  export default {
    name: "MrBrand",
    components:{
      MrBrandForm
    },
    data() {
      return {
        brandDetail:{},
        isEdit:false,
        dialog:false,
        pagination: {},
        total: 0,
        search: "",
        headers: [
          {
            text: "id",
            align: "center",
            value: "id",
          },
          {
            text: "品牌名称",
            align: "center",
            value: "name",
          },
          {
            text: "品牌logo",
            align: "center",
            value: "image",
          },
          {
            text: "首字母",
            align: "center",
            value: "letter",
          },
          {
            text: "操作",
            align: "center",
            sortable: false,
            value: "id",
          }
        ],
        desserts: [],
      };
    },
    mounted() {
      this.getTableData();
    },
    methods: {
      closeDialog(){
        this.dialog = false;
        this.getTableData()
      },
      addDialog(){
        this.isEdit = false;
        this.dialog = true;
      },
      editDialog(obj){
        this.brandDetail = obj;
        this.isEdit = true;
        this.dialog = true;
      },
      delDialog(id){
        this.$http.delete('brand/del?id=' + id).then(resp =>{
          if(resp.data.code == 200){
              this.getTableData()
              this.$message.success('删除成功')
          }else{
            this.$message.error(resp.data.msg)
          }
        }).catch(error => console.log(error))
      },
      getTableData() {
        this.$http
          .get("/brand/list", {
            params: {
              page: this.pagination.page,
              rows: this.pagination.rowsPerPage,
              sort: this.pagination.sortBy,
              order: this.pagination.descending,
              name: this.search,
            },
          })
          .then((resp) => {
            this.desserts = resp.data.data.list;
            this.total = resp.data.data.total;
          })
          .catch((error) => console.log(error));
      },
    },
    watch: {
      pagination() {
        this.getTableData();
      },
    },
  };
  </script>
```

Casecader.vue需要修改113行



```
const data = [];
for (let d of resp.data.data) {
  const node = {
```

MrBrand.vue

```
<template>
  <v-card>
    <v-card-title>
      <v-btn color="info" @click="dialog = true">新增</v-btn>
      <div class="text-xs-center">
        <v-dialog v-model="dialog" width="500">
          <v-card>
            <v-card-title class="headline grey lighten-2" primary-title>
            品牌 {{isEdit?'修改':'新增'}}
           </v-card-title>
            <mr-brand-form @closeDialog="closeDialog" :dialog="dialog"
:isEdit="isEdit" :brandDetail="brandDetail"/>
          </v-card>
        </v-dialog>
      </div>
      <!-- 调按钮和输入框之间的间距 -->
      <v-spacer />
      <!--append-icon : 图标 label : input默认值-->
      <v-text-field
        append-icon="search"
        label="品牌名称"
        @keyup.enter="getTableData()"
        v-model="search"
      ></v-text-field>
    </v-card-title>
    <v-data-table
      :headers="headers"
      :items="desserts"
      :pagination.sync="pagination"
      :total-items="total"
      class="elevation-1"
    >
```

```
      <!-- 注意此处不能只能用官网的属性,应当参照Brand.vue的写法 -->
      <template slot="items" slot-scope="props">
        <td class="text-xs-center">{{ props.item.id }}</td>
        <td class="text-xs-center">{{ props.item.name }}</td>
        <!-- :src 给元素绑定src属性 属性的值为props.item.image -->
        <td class="text-xs-center">
          <img width="200" :src="props.item.image" />
          </td>
        <td class="text-xs-center">{{ props.item.letter }}</td>
        <td class="text-xs-center">
            <v-btn falt icon  @click="editDialog(props.item)" color="green">
                <v-icon>edit</v-icon>
            </v-btn>
            <v-btn falt icon @click="delDialog(props.item.id)" color="red">
                <v-icon>delete</v-icon>
            </v-btn>
        </td>
      </template>
    </v-data-table>
  </v-card>
</template>
<script>
import MrBrandForm from './MrBrandForm';
export default {
  name: "MrBrand",
  components:{
    MrBrandForm
  },
  data() {
    return {
      brandDetail:{},
      isEdit:false,
      dialog:false,
      pagination: {},
      total: 0,
      search: "",
      headers: [
        {
          text: "id",
          align: "center",
          value: "id",
        },
        {
          text: "品牌名称",
          align: "center",
          value: "name",
        },
        {
          text: "品牌logo",
          align: "center",
          value: "image",
        },
        {
          text: "首字母",
          align: "center",
          value: "letter",
        },
        {
          text: "操作",
          align: "center",
          sortable: false,
          value: "id",
        }
      ],
      desserts: [],
    };
  },
  mounted() {
    this.getTableData();
  },
  methods: {
    closeDialog(){
      this.dialog = false;
      this.getTableData()
    },
    addDialog(){
      this.isEdit = false;
      this.dialog = true;
    },
    editDialog(obj){
      this.brandDetail = obj;
      this.isEdit = true;
      this.dialog = true;
    },
    delDialog(id){
      this.$http.delete('brand/del?id=' + id).then(resp =>{
        if(resp.data.code == 200){
            this.getTableData()
            this.$message.success('删除成功')
        }else{
          this.$message.error(resp.data.msg)
        }
      }).catch(error => console.log(error))
    },
    getTableData() {
      this.$http
        .get("/brand/list", {
          params: {
            page: this.pagination.page,
            rows: this.pagination.rowsPerPage,
            sort: this.pagination.sortBy,
            order: this.pagination.descending,
            name: this.search,
          },
        })
        .then((resp) => {
          this.desserts = resp.data.data.list;
          this.total = resp.data.data.total;
        })
        .catch((error) => console.log(error));
    },
  },
  watch: {
    pagination() {
      this.getTableData();
    },
  },
};
</script>
```

新增遗留的问题

品牌首字母自动识别

后台

mingrui-shop-common-core

pom.xml

```
<dependency>
    <groupId>com.belerweb</groupId>
    <artifactId>pinyin4j</artifactId>
    <version>2.5.1</version>
</dependency>
```

com.baidu.shop.utils包下新建PinyinUtil

```
package com.baidu.shop.utils;

import net.sourceforge.pinyin4j.PinyinHelper;
import net.sourceforge.pinyin4j.format.HanyuPinyinOutputFormat;
import net.sourceforge.pinyin4j.format.HanyuPinyinToneType;
import java.util.regex.Matcher;
import java.util.regex.Pattern;


public class PinyinUtil {
    public static final Boolean TO_FUUL_PINYIN = true;
    public static final Boolean TO_FIRST_CHAR_PINYIN = false;
    /**
     * 获取汉字首字母或全拼大写字母
     *
     * @param chinese 汉字
     * @param isFull 是否全拼 true:表示全拼 false表示：首字母
     * @return 全拼或者首字母大写字符窜
     */
    public static String getUpperCase(String chinese, boolean isFull) {
        return convertHanzi2Pinyin(chinese, isFull).toUpperCase();
    }
    /**
     * 获取汉字首字母或全拼小写字母
     *
     * @param chinese 汉字
     * @param isFull 是否全拼 true:表示全拼 false表示：首字母
     * @return 全拼或者首字母小写字符窜
     */
    public static String getLowerCase(String chinese, boolean isFull) {
        return convertHanzi2Pinyin(chinese, isFull).toLowerCase();
    }
    /**
     * 将汉字转成拼音
     * <p>
     * 取首字母或全拼
     *
     * @param hanzi 汉字字符串
     * @param isFull 是否全拼 true:表示全拼 false表示：首字母
     * @return 拼音
     */
    private static String convertHanzi2Pinyin(String hanzi, boolean isFull) {
/***
 * ^[\u2E80-\u9FFF]+$ 匹配所有东亚区的语言
 * ^[\u4E00-\u9FFF]+$ 匹配简体和繁体
 * ^[\u4E00-\u9FA5]+$ 匹配简体
 */
        String regExp = "^[\u4E00-\u9FFF]+$";
        StringBuffer sb = new StringBuffer();
        if (hanzi == null || "".equals(hanzi.trim())) {
            return "";
        }
        String pinyin = "";
        for (int i = 0; i < hanzi.length(); i++) {
            char unit = hanzi.charAt(i);
//是汉字，则转拼音
            if (match(String.valueOf(unit), regExp)) {
                pinyin = convertSingleHanzi2Pinyin(unit);
                if (isFull) {
                    sb.append(pinyin);
                } else {
                    sb.append(pinyin.charAt(0));
                }
            } else {
                sb.append(unit);
            }
        }
        return sb.toString();
    }
    /**
     * 将单个汉字转成拼音
     *
     * @param hanzi 汉字字符
     * @return 拼音
     */
    private static String convertSingleHanzi2Pinyin(char hanzi) {
        HanyuPinyinOutputFormat outputFormat = new HanyuPinyinOutputFormat();
        outputFormat.setToneType(HanyuPinyinToneType.WITHOUT_TONE);
        String[] res;
        StringBuffer sb = new StringBuffer();
        try {
            res = PinyinHelper.toHanyuPinyinStringArray(hanzi, outputFormat);
            sb.append(res[0]);//对于多音字，只用第一个拼音
        } catch (Exception e) {
            e.printStackTrace();
            return "";
        }
        return sb.toString();
    }
    /***
     * 匹配
     * <P>
     * 根据字符和正则表达式进行匹配
     *
     * @param str 源字符串
    4.1.3 mingrui-shop-service-xxx
    4.1.3.1 BrandServiceImpl
     * @param regex 正则表达式
     *
     * @return true: 匹配成功 false: 匹配失败
     */
    private static boolean match(String str, String regex) {
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(str);
        return matcher.find();
    }

}
```

mingrui-shop-service-xxx

com.baidu.shop.service.impl

```
@Override
public Result<JSONObject> save(BrandDTO brandDTO) {
    //str.charAt(0)获取当前字符串第一个字符
    //String.valueOf()将对象转换为String类型的字符串
    //
    // char c = brandDTO.getName().charAt(0);
    // String s = String.valueOf(c);
    // String upperCase = PinyinUtil.getUpperCase(s,
    PinyinUtil.TO_FIRST_CHAR_PINYIN);
    // char c1 = upperCase.charAt(0);
    // brandDTO.setLetter(c1);

brandDTO.setLetter(PinyinUtil.getUpperCase(String.valueOf(brandDTO.getName().ch
    arAt(0)),PinyinUtil.TO_FIRST_CHAR_PINYIN).charAt(0));
    try {
        BrandEntity brandEntity = BaiduBeanUtil.copyProperties(brandDTO,
        BrandEntity.class);
        //通用mapper新增返回主键
        brandMapper.insertSelective(brandEntity);
            //绑定关系
            if(StringUtils.isEmpty(brandDTO.getCategories())){
                return this.setResultError("分类数据不能为空");
            }
            if(brandDTO.getCategories().contains(",")){
                List<CategoryBrandEntity> list = new ArrayList<>();
                String[] categoryArr = brandDTO.getCategories().split(",");
                Arrays.asList(categoryArr).stream().forEach(str -> {
                    CategoryBrandEntity categoryBrandEntity = new
                    CategoryBrandEntity();
                    categoryBrandEntity.setBrandId(brandEntity.getId());
                    categoryBrandEntity.setCategoryId(Integer.parseInt(str));
                    list.add(categoryBrandEntity);
                });
            categoryBrandMapper.insertList(list);
            }
    } catch (Exception e) {
        e.printStackTrace();
    }
    return this.setResultSuccess();
}
```

前台

将MrBrandForm.vue组件中所有关于首字母的内容删除掉

```
<template>
    <div>
            <v-form v-model="valid" ref="form">
                <v-text-field v-model="brand.name" label="品牌名称"
:rules="nameRules" required></v-text-field>
                <v-cascader url="/category/list" required v-
model="brand.categories" multiple label="商品分类"/>
                <!-- 文件上传 -->
                <v-layout row>
                    <v-flex xs3>
                      <span style="font-size: 16px; color: #444">品牌LOGO:</span>
                    </v-flex>
                    <v-flex>
                        <v-upload
                          v-model="brand.image"
                          url="/upload"
                          :multiple="false"
                          :pic-width="250"
                          :pic-height="100"
                        />
                    </v-flex>
                </v-layout>
            </v-form>
         <v-card-actions>
          <v-spacer></v-spacer>
          <v-btn small @click="cancel()">取消</v-btn>
          <v-btn small color="red"  @click="submitForm">确认</v-btn>
        </v-card-actions>
    </div>
</template>
<script>
  export default {
    name:"MrBrandForm",
    props:{
      //父组件传递过来的模态框状态
      dialog: Boolean,
      brandDetail: Object,
      isEdit: Boolean
    },
    watch:{
      dialog(val){
        if(val) this.$refs.form.reset();
      },
      brandDetail(val){
        if(this.isEdit){
            this.$http.get('category/getByBrand',{
              params:{
                brandId:val.id
              }
            }).then(resp =>{
              console.log(resp);
              let brand = val;
              brand.categories = resp.data.data;
              this.brand = brand;
            }).catch(error => console.log(error))
            // this.brand=val;
        }
      }
    },
    data () {
      //在js中 null == false , '' == false , undefined == false , 0 == false
      return {
        valid: true,
        nameRules:[
          (v) => !!v || '品牌名称不能为空',
          (v) => (v && v.length <= 10) || '品牌名称长度最长10'
        ],
        brand:{
            name:"",
            letter:"",
            categories:[],

        }
      }
    },
    methods:{
        cancel(){
            this.$emit("closeDialog");
        },
        submitForm(){
            if(!this.$refs.form.validate()){
                return;
            }
            let fromData = this.brand;
            let categoryIdArr = this.brand.categories.map(category
=>category.id);
            fromData.categories = categoryIdArr.join();

            //$.ajax() 和 $.get() | $.post()方法的区别?
            this.$http({
              url:'/brand/save',
              data:fromData,
              method:this.isEdit ? 'put':'post'
              }).then(resp=>{
                if(resp.data.code != 200){
                    return;
                }
                  //关闭模态框
                    this.cancel();
            }).catch(error => console.log(error))
        }
    }
  }
</script>
```

清空form和表单验证的问题

前台

MrBrand.vue

```
<mr-brand-form @closeDialog="closeDialog" :dialog="dialog" :isEdit="isEdit"
:brandDetail="brandDetail"/>
```

MrBrandFrom.vue

```
<template>
    <div>
            <v-form v-model="valid" ref="form">
                <v-text-field v-model="brand.name" label="品牌名称"
:rules="nameRules" required></v-text-field>
                <v-cascader url="/category/list" required v-
model="brand.categories" multiple label="商品分类"/>
                <!-- 文件上传 -->
                <v-layout row>
                    <v-flex xs3>
                      <span style="font-size: 16px; color: #444">品牌LOGO:</span>
                    </v-flex>
                    <v-flex>
                        <v-upload
                          v-model="brand.image"
                          url="/upload"
                          :multiple="false"
                          :pic-width="250"
                          :pic-height="100"
                        />
                    </v-flex>
                </v-layout>
            </v-form>
         <v-card-actions>
          <v-spacer></v-spacer>
          <v-btn small @click="cancel()">取消</v-btn>
          <v-btn small color="red"  @click="submitForm">确认</v-btn>
        </v-card-actions>
    </div>
</template>
<script>
  export default {
    name:"MrBrandForm",
    props:{
      //父组件传递过来的模态框状态
      dialog: Boolean,
      brandDetail: Object,
      isEdit: Boolean
    },
    watch:{
      dialog(val){
        if(val) this.$refs.form.reset();
      },
      brandDetail(val){
        if(this.isEdit){
            this.$http.get('category/getByBrand',{
              params:{
                brandId:val.id
              }
            }).then(resp =>{
              console.log(resp);
              let brand = val;
              brand.categories = resp.data.data;
              this.brand = brand;
            }).catch(error => console.log(error))
            // this.brand=val;
        }
      }
    },
    data () {
      //在js中 null == false , '' == false , undefined == false , 0 == false
      return {
        valid: true,
        nameRules:[
          (v) => !!v || '品牌名称不能为空',
          (v) => (v && v.length <= 10) || '品牌名称长度最长10'
        ],
        brand:{
            name:"",
            letter:"",
            categories:[],

        }
      }
    },
    methods:{
        cancel(){
            this.$emit("closeDialog");
        },
        submitForm(){
            if(!this.$refs.form.validate()){
                return;
            }
            let fromData = this.brand;
            let categoryIdArr = this.brand.categories.map(category
=>category.id);
            fromData.categories = categoryIdArr.join();

            //$.ajax() 和 $.get() | $.post()方法的区别?
            this.$http({
              url:'/brand/save',
              data:fromData,
              method:this.isEdit ? 'put':'post'
              }).then(resp=>{
                if(resp.data.code != 200){
                    return;
                }
                   //关闭模态框
                    this.cancel();
            }).catch(error => console.log(error))
        }
    }
  }
```

```
</script>
```

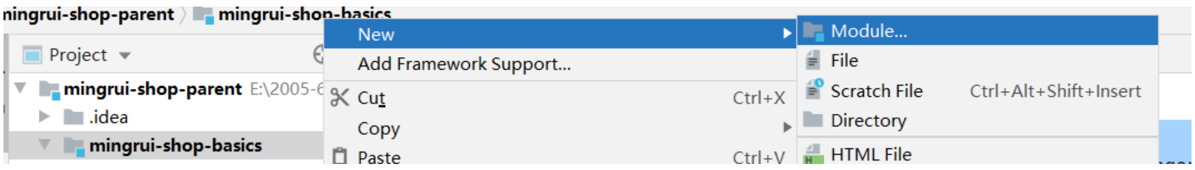
Cascader.vue(bug) line:162

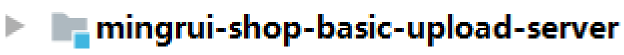
```
if(val && this.showAllLevels && !this.multiple){
           this.selected = [val.map(o => o[this.itemText]).join("/")]
         } else if (this.multiple && typeof val == 'object') {
           this.selected = val.map(o => {
             return {
               label: o[this.itemText],
               value: o[this.itemValue]
             }
           })
         } else {
           if(typeof val == 'object'){
             this.selected = [val[this.itemText]]
           }
         }
```

上传图片

后台

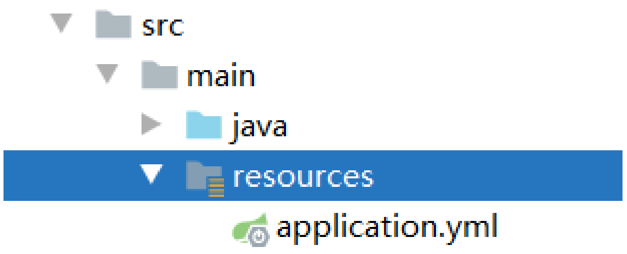
在mingrui-shop-basics下新建mingrui-shop-basic-uploadserver





pom.xml

```
<dependencies>
    <!-- SpringBoot-整合Web组件 -->
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>com.baidu</groupId>
        <artifactId>mingrui-shop-common-core</artifactId>
        <version>1.0-SNAPSHOT</version>
    </dependency>
</dpendencies>
```



application.yml

```
server:
  port: 8200
spring:
  application:
    name: upload-server
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka/
mingrui:
  upload:
    path:
      windows: E:\\images
      linux: /lihuimin/images
    img:
      host: http://image.mrshop.com
```

com.baidu新建启动类RunBasicUploadServer

```

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication
@EnableEurekaClient
public class RunBasicUploadServer {
    public static void main(String[] args) {
        SpringApplication.run(RunBasicUploadServer.class);
    }
}
```

新建包com.baidu.shop.upload.controller新建UploadController

```
package com.baidu.shop.upload.controller;

import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.status.HTTPStatus;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;
import java.io.File;
import java.io.IOException;
import java.util.UUID;


@RestController
@RequestMapping(value = "upload")
public class UploadController extends BaseApiService {

    //linux系统的上传目录
    @Value(value = "${mingrui.upload.path.windows}")
    private String windowsPath;
    //window系统的上传目录
    @Value(value = "${mingrui.upload.path.linux}")
    private String linuxPath;
    //图片服务器的地址
    @Value(value = "${mingrui.upload.img.host}")
    private String imgHost;

    @PostMapping
    public Result<String> uploadImg(@RequestParam MultipartFile file) {
        if(file.isEmpty()) return this.setResultError("上传的文件为空");//判断上传的
文件是否为空
        String filename = file.getOriginalFilename();//获取文件名
        String path = "";
        String os = System.getProperty("os.name").toLowerCase();
        if(os.indexOf("win") != -1){
            path = windowsPath;
        }else if(os.indexOf("lin") != -1){
            path = linuxPath;
        }
        filename = UUID.randomUUID() + filename;//防止文件名重复
//创建文件 路径+分隔符(linux和window的目录分隔符不一样)+文件名
        File dest = new File(path + File.separator + filename);
//判断文件夹是否存在,不存在的话就创建
        if(!dest.getParentFile().exists()) dest.getParentFile().mkdirs();
        try {
            file.transferTo(dest);//上传
        } catch (IllegalStateException e) {
// TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
// TODO Auto-generated catch block
            e.printStackTrace();
        }
        return this.setResult(HTTPStatus.OK,"upload success!!!",imgHost + "/" +
                filename);//将文件名返回页面用于页面回显
    }
}
```

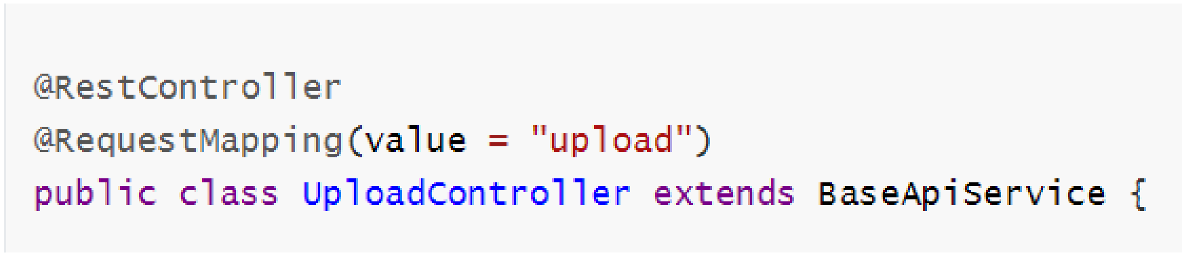
新建包:com.baidu.global新建GlobalCorsConfig

```
package com.baidu.global;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;


@Configuration
public class GlobalCorsConfig {
    @Bean
    public CorsFilter corsFilter() {
        final UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        final CorsConfiguration config = new CorsConfiguration();
        config.setAllowCredentials(true); // 允许cookies跨域
        config.addAllowedOrigin("*");// 允许向该服务器提交请求的URI，*表示全部允许。。
这里尽量限制来源域，比如http://xxxx:8080 ,以降低安全风险。。
```

```
        config.addAllowedHeader("*");// 允许访问的头信息,*表示全部
        config.setMaxAge(18000L);// 预检请求的缓存时间（秒），即在这个时间段里，对于相同的跨域请求不会再预检了
        config.addAllowedMethod("*");// 允许提交请求的方法，*表示全部允许，也可以单独设置GET、PUT等
        config.addAllowedMethod("HEAD");
        config.addAllowedMethod("GET");// 允许Get的请求方法
        config.addAllowedMethod("PUT");
        config.addAllowedMethod("POST");
        config.addAllowedMethod("DELETE");
        config.addAllowedMethod("PATCH");
        source.registerCorsConfiguration("/**", config);
        //3.返回新的CorsFilter.
        return new CorsFilter(source);
    }

}
```

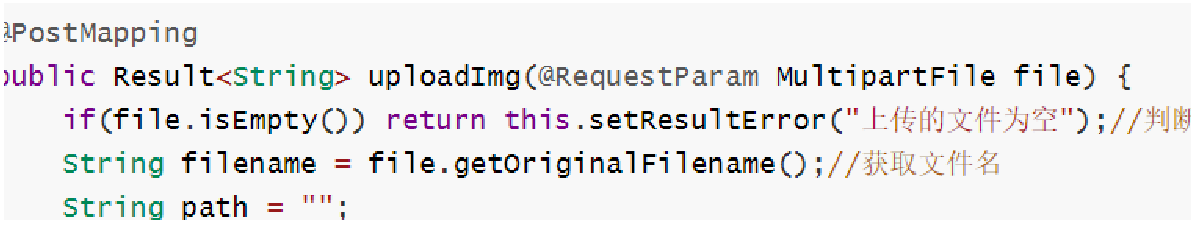
使用postman测试上传

post请求localhost:8200(端口)/upload(UploadController方法名)/file(UploadController参数)



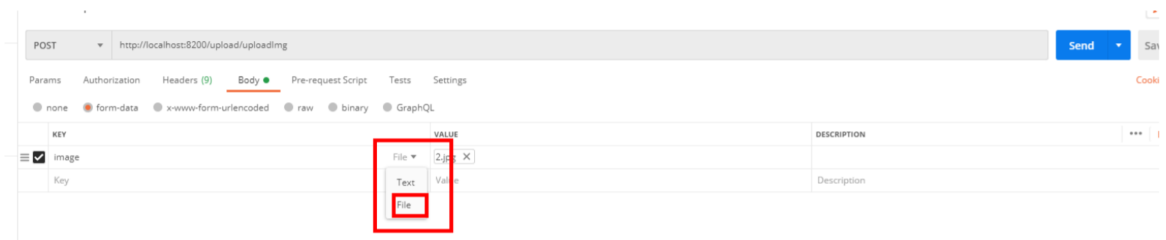
```
@RestController
@RequestMapping(value = "upload")
public class UploadController extends BaseApiService {
```

upload



```
@PostMapping
public Result<String> uploadImg(@RequestParam MultipartFile file) {
    if(file.isEmpty()) return this.setResultError("上传的文件为空");//判断
    String filename = file.getOriginalFilename();//获取文件名
    String path = "";
```

file



POST http://localhost:8200/upload/uploadImg

Body: form-data

KEY	VALUE	DESCRIPTION
image	File 2.jpg	
Key	Value	Description

hosts文件中新增

```
127.0.0.1 image.mrshop.com
```

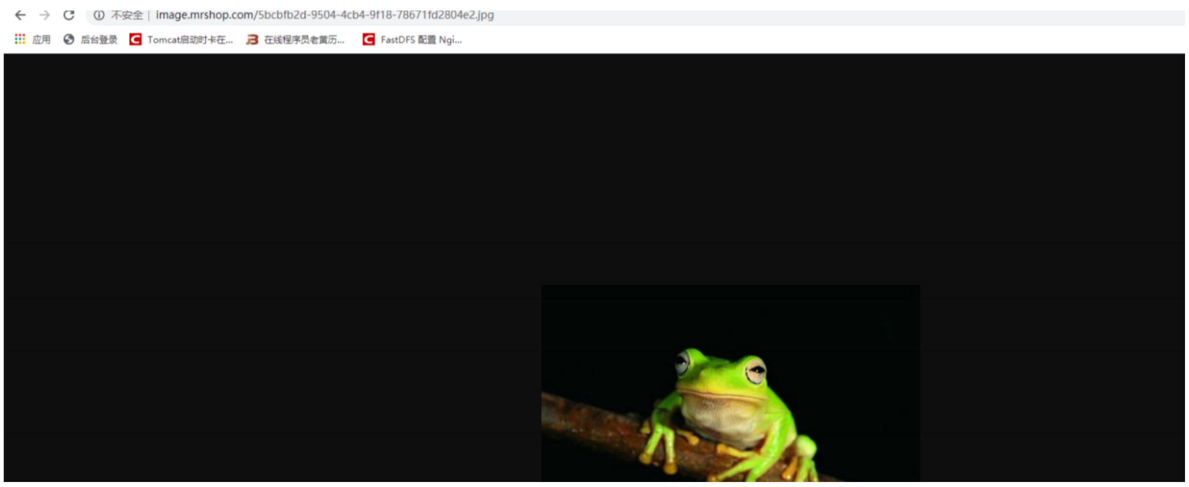
nginx-home/conf/nginx.conf新增

```
server {
    listen 80;
    server_name image.mrshop.com;
    proxy_set_header X-Forwarded-Host $host;
    proxy_set_header X-Forwarded-Server $host;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    location ~ .*\.(gif|jpg|pdf|jpeg|png)$ {
        #root D:/nginx-1.15.5/temp/images/;#指定图片存放路径(可以放在nginx文件夹路 径
里也可以放其他p盘) root                 E:\images;
    }
        location / {
            root html;
            index index.html index.htm;
        }
    }
```

重启nginx

```
nginx.exe -s reload
```

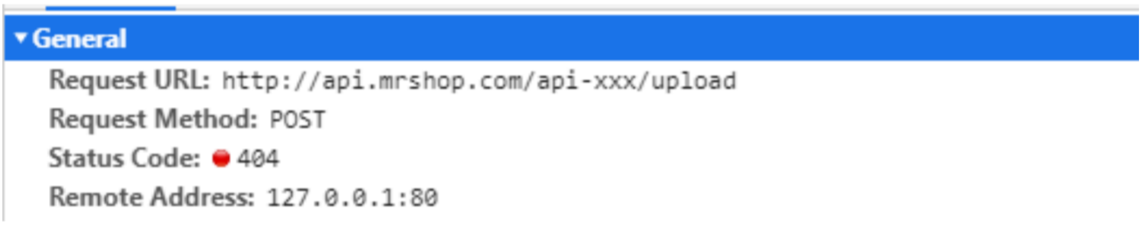
浏览器输入http://image.mrshop.com/imageName.jpg测试



启动vue项目测试能否成功 ？

不能成功,还是请求了网关,

网关会吧api-xxx的请求转发到service-xxx的服务中,而service-xxx并没有 upload服务



▾ General
Request URL: http://api.mrshop.com/api-xxx/upload
Request Method: POST
Status Code: ● 404
Remote Address: 127.0.0.1:80

所以我想只要包含/upload请求的我都不进行路由

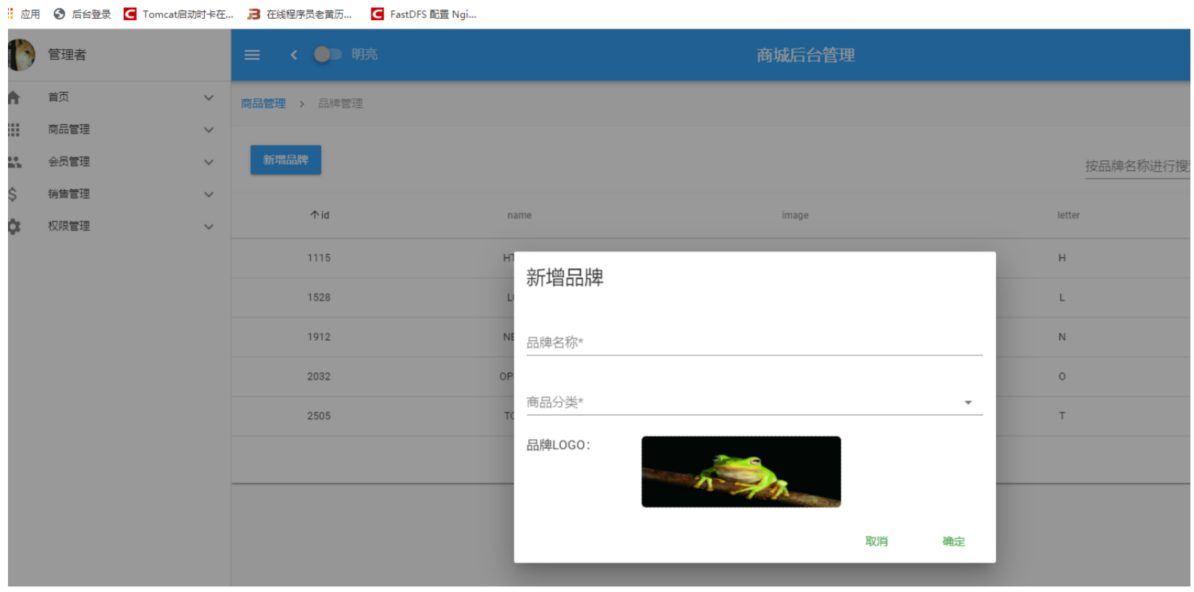
mingrui-shop-basic-zuul-server的application.yml

```


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-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
```

在nginx-home/conf/nginx.conf 添加api.mrshop.com的代理配置

```
# 上传路径的映射
# 只要包含/api-xxx/upload 都会把请求映射到8200服务
# rewrite "^/api-xxx/(.*)$" /$1 break;
# 将/api-xxx 替换成/
location /api-xxx/upload {
    proxy_pass http://127.0.0.1:8200;
    proxy_connect_timeout 600;
    proxy_read_timeout 600; rewrite "^/api-xxx/(.*)$" /$1 break;
}
```

重启nginx



3.品牌修改

回显

后台

mingrui-shop-service-api-xxx

Service：CategoryService

```
@ApiOperation(value = "通过品牌id查询商品分类")
@GetMapping(value = "category/getByBrand")
Result<List<CategoryEntity>> getByBrand (Integer brandId);
```

mingrui-shop-service-xxx

com.baidu.shop.mapper：CategoryMapper

```
@Select(value = "select id,name from tb_category where id in (select category_id
from tb_category_brand where brand_id=#{brandId})")
```

com.baidu.shop.service.impl：CategoryServiceImpl

```
//通过品牌查询商品id
@Override
public Result<List<CategoryEntity>> getByBrand(Integer brandId) {
    List<CategoryEntity> byBrandId = categoryMapper.getByBrandId(brandId);
    return this.setResultSuccess(byBrandI);
}
```

前台

MrBrand.vue

```
<template>
  <v-card>
    <v-card-title>
      <v-btn color="info" @click="dialog = true">新增</v-btn>
      <div class="text-xs-center">
        <v-dialog v-model="dialog" width="500">
          <v-card>
            <v-card-title class="headline grey lighten-2" primary-title>
            品牌 {{isEdit?'修改':'新增'}}
           </v-card-title>
            <mr-brand-form @closeDialog="closeDialog" :dialog="dialog"
:isEdit="isEdit" :brandDetail="brandDetail"/>
          </v-card>
        </v-dialog>
      </div>
      <!-- 调按钮和输入框之间的间距 -->
      <v-spacer />
      <!--append-icon : 图标 label : input默认值-->
      <v-text-field
        append-icon="search"
        label="品牌名称"
        @keyup.enter="getTableData()"
        v-model="search"
      ></v-text-field>
    </v-card-title>
    <v-data-table
      :headers="headers"
      :items="desserts"
      :pagination.sync="pagination"
      :total-items="total"
      class="elevation-1"
    >
      <!-- 注意此处不能只能用官网的属性,应当参照Brand.vue的写法 -->
      <template slot="items" slot-scope="props">
        <td class="text-xs-center">{{ props.item.id }}</td>
        <td class="text-xs-center">{{ props.item.name }}</td>
        <!-- :src 给元素绑定src属性 属性的值为props.item.image -->
        <td class="text-xs-center">
```

```
          <img width="200" :src="props.item.image" />
          </td>
        <td class="text-xs-center">{{ props.item.letter }}</td>
        <td class="text-xs-center">
            <v-btn falt icon  @click="editDialog(props.item)" color="green">
                <v-icon>edit</v-icon>
            </v-btn>
            <v-btn falt icon @click="delDialog(props.item.id)" color="red">
                <v-icon>delete</v-icon>
            </v-btn>
        </td>
      </template>
    </v-data-table>
  </v-card>
</template>
<script>
import MrBrandForm from './MrBrandForm';
export default {
  name: "MrBrand",
  components:{
    MrBrandForm
  },
  data() {
    return {
      brandDetail:{},
      isEdit:false,
      dialog:false,
      pagination: {},
      total: 0,
      search: "",
      headers: [
        {
          text: "id",
          align: "center",
          value: "id",
        },
        {
          text: "品牌名称",
          align: "center",
          value: "name",
        },
        {
          text: "品牌logo",
          align: "center",
          value: "image",
        },
        {
          text: "首字母",
          align: "center",
          value: "letter",
        },
        {
          text: "操作",
          align: "center",
          sortable: false,
          value: "id",
        }
      ],
      desserts: [],
    };
  },
  mounted() {
    this.getTableData();
  },
  methods: {
    closeDialog(){
      this.dialog = false;
      this.getTableData()
    },
    addDialog(){
      this.isEdit = false;
      this.dialog = true;
    },
    editDialog(obj){
      this.brandDetail = obj;
      this.isEdit = true;
      this.dialog = true;
    },
    delDialog(id){
      this.$http.delete('brand/del?id=' + id).then(resp =>{
        if(resp.data.code == 200){
            this.getTableData()
            this.$message.success('删除成功')
        }else{
          this.$message.error(resp.data.msg)
        }
      }).catch(error => console.log(error))
    },
    getTableData() {
      this.$http
        .get("/brand/list", {
          params: {
            page: this.pagination.page,
            rows: this.pagination.rowsPerPage,
            sort: this.pagination.sortBy,
            order: this.pagination.descending,
            name: this.search,
          },
        })
        .then((resp) => {
          this.desserts = resp.data.data.list;
          this.total = resp.data.data.total;
        })
        .catch((error) => console.log(error));
    },
  },
  watch: {
    pagination() {
      this.getTableData();
    },
  },
};
</script>
```

MrBrandForm.vue

```
<template>
    <div>
            <v-form v-model="valid" ref="form">
                <v-text-field v-model="brand.name" label="品牌名称"
:rules="nameRules" required></v-text-field>
                <v-cascader url="/category/list" required v-
model="brand.categories" multiple label="商品分类"/>
                <!-- 文件上传 -->
                <v-layout row>
                    <v-flex xs3>
                      <span style="font-size: 16px; color: #444">品牌LOGO:</span>
                    </v-flex>
                    <v-flex>
                        <v-upload
                          v-model="brand.image"
                          url="/upload"
                          :multiple="false"
                          :pic-width="250"
                          :pic-height="100"
                        />
                    </v-flex>
                </v-layout>
            </v-form>
         <v-card-actions>
          <v-spacer></v-spacer>
          <v-btn small @click="cancel()">取消</v-btn>
          <v-btn small color="red"  @click="submitForm">确认</v-btn>
        </v-card-actions>
    </div>
</template>
<script>
  export default {
    name:"MrBrandForm",
    props:{
      //父组件传递过来的模态框状态
      dialog: Boolean,
      brandDetail: Object,
      isEdit: Boolean
    },
    watch:{
      dialog(val){
        if(val) this.$refs.form.reset();
      },
      brandDetail(val){
        if(this.isEdit){
            this.$http.get('category/getByBrand',{
              params:{
                brandId:val.id
              }
            }).then(resp =>{
              console.log(resp);
              let brand = val;
              brand.categories = resp.data.data;
              this.brand = brand;
            }).catch(error => console.log(error))
            // this.brand=val;
        }
      }
    },
    data () {
      //在js中 null == false , '' == false , undefined == false , 0 == false
      return {
        valid: true,
        nameRules:[
          (v) => !!v || '品牌名称不能为空',
          (v) => (v && v.length <= 10) || '品牌名称长度最长10'
        ],
        brand:{
            name:"",
            letter:"",
            categories:[],

        }
      }
    },
    methods:{
        cancel(){
            this.$emit("closeDialog");
        },
        submitForm(){
            if(!this.$refs.form.validate()){
                return;
            }
            let fromData = this.brand;
            let categoryIdArr = this.brand.categories.map(category
=>category.id);
            fromData.categories = categoryIdArr.join();

            //$.ajax() 和 $.get() | $.post()方法的区别?
            this.$http({
              url:'/brand/save',
              data:fromData,
              method:this.isEdit ? 'put':'post'
              }).then(resp=>{
                if(resp.data.code != 200){
                    return;
                }
                   //关闭模态框
                    this.cancel();
            }).catch(error => console.log(error))
        }
    }
  }
</script>
```

修改

后台

mingrui-shop-service-api-xxx

com.baidu.shop.service：BrandService

```
@PutMapping(value = "brand/save")
@ApiOperation(value = "修改品牌")
Result<JsonObject> eidt(@RequestBody BrandDto brandDto);
```

mingrui-shop-service-xxx

com.baidu.shop.service.impl：BrandServiceImpl

```
//修改品牌
@Transactional
@Override
public Result<JsonObject> eidt(BrandDto brandDto) {


    BrandEntity brandEntity = BaiduBeanUtil.copyProperties(brandDto,
BrandEntity.class);


 brandEntity.setLetter(PinyinUtil.getUpperCase(String.valueOf(brandEntity.getNam
e().toCharArray()[0]),false).toCharArray()[0]);
    brandMapper.updateByPrimaryKeySelective(brandEntity);
    //先通过brandId删除中间表的数据
    this.delteCategoryByBrandId(brandEntity.getId());
    //批量新增 / 新增
    this.insetrCategoryBrandList(brandDto.getCategories(),brandEntity.getId());
    return this.setResultSuccess();
}
```

```
//新增修改整合
private void insetrCategoryBrandList (String categories,Integer brandId){
    if (StringUtils.isEmpty(categories))throw new RuntimeException("分类信息不能为
空");

    //判断分类集合字符串中是否包含,
    if(categories.contains(",")){
        categoryBrandMapper.insertList(
            //数组转list
            Arrays.asList(categories.split(","))
                    //获取stream流, 流: 对一次数据进行操作
                    .stream()
                    //遍历list中所有数据
                    .map(categoryById->new
CategoryBrandEntity(Integer.valueOf(categoryById),brandId))
                    //最终转换成list
                    .collect(Collectors.toList())
        );
    }else{
        CategoryBrandEntity categoryBrandEntity = new CategoryBrandEntity();
        categoryBrandEntity.setCategoryId(Integer.valueOf(categories));
        categoryBrandEntity.setBrandId(brandId);
        categoryBrandMapper.insertSelective(categoryBrandEntity);
    }
}
```

前台

```
submitForm(){
            if(!this.$refs.form.validate()){
                return;
            }
            let fromData = this.brand;
            let categoryIdArr = this.brand.categories.map(category
=>category.id);
            fromData.categories = categoryIdArr.join();

            //$.ajax() 和 $.get() | $.post()方法的区别?
            this.$http({
              url:'/brand/save',
              data:fromData,
              method:this.isEdit ? 'put':'post'
              }).then(resp=>{
                if(resp.data.code != 200){
                    return;
                }
                   //关闭模态框
                    this.cancel();
            }).catch(error => console.log(error))
        }
```

4.品牌删除

后台

mingrui-shop-service-api-xxx

com.baidu.shop.service：BrandService

```
@DeleteMapping(value = "brand/del")
@ApiOperation(value = "删除品牌")
Result<JsonObject> del( Integer id);
```

mingrui-shop-service-xxx：BrandServiceImpl

```
//删除品牌
@Override
public Result<JsonObject> del(Integer id) {
    //删除品牌信息
    brandMapper.deleteByPrimaryKey(id);
    this.delteCategoryByBrandId(id);
    return this.setResultSuccess();
}

private void  delteCategoryByBrandId (Integer brandId){
    Example example = new Example(CategoryBrandEntity.class);
```

```
        example.createCriteria().andEqualTo("brandId",brandId);
        categoryBrandMapper.deleteByExample(example);
    }
```

前台

```
<v-btn falt icon @click="delDialog(props.item.id)" color="red">
                <v-icon>delete</v-icon>
            </v-btn>
```

```
delDialog(id){
      this.$http.delete('brand/del?id=' + id).then(resp =>{
        if(resp.data.code == 200){
            this.getTableData()
            this.$message.success('删除成功')
        }else{
          this.$message.error(resp.data.msg)
        }
      }).catch(error => console.log(error))
    },
```

四，规格参数

1.规格组(查询)

后台

mingrui-shop-service-api-xxx

com.baidu.shop.entity新建：SpecGroupEntity

```java
import lombok.Data;

import javax.persistence.Id;
import javax.persistence.Table;

@Table(name = "tb_spec_group")
@Data
public class SpecGroupEntity {
    @Id
    private Integer id;

    private Integer cid;

    private String name;
}
```

com.baidu.shop.dto新建：SpecGroupDto

```java
import com.baidu.shop.base.BaseDTO;
import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@ApiModel(value = "规格分组")
@Data
public class SpecGroupDto extends BaseDTO {

    @ApiModelProperty(value = "主键",example = "1")
    @NotNull(message = "主键不能为空",groups = {MingruiOperation.Update.class})
    private Integer id;

    @ApiModelProperty(value = "类型id",example = "1")
    @NotNull(message = "类型id不能为空",groups = {MingruiOperation.Add.class})
    private Integer cid;

    @ApiModelProperty(value = "规格组名称",example = "1")
    @NotEmpty(message = "规格组名称不能为空",groups = {MingruiOperation.Add.class})
    private String name;
}
```

com.baidu.shop.service新建：SpecGroupService

```
package com.baidu.shop.service;

import com.baidu.shop.base.Result;
import com.baidu.shop.dto.SpecGroupDto;
import com.baidu.shop.entity.SpecGroupEntity;
import com.google.gson.JsonObject;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiModelProperty;
import org.springframework.validation.annotation.Validated;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Api(value ="规格组接口")
public interface SpecGroupService {

    @ApiModelProperty(value = "规格组查询")
    @GetMapping(value = "specGroup/getSpecGroupInfo")
    Result<List<SpecGroupEntity>> getSpecGroupInfo(SpecGroupDto specGroupDto);
}
```

mingrui-shop-service-xxx

com.baidu.shop.mapper新建：SpecGroupMapper

```
package com.baidu.shop.mapper;

import com.baidu.shop.entity.SpecGroupEntity;
import tk.mybatis.mapper.common.Mapper;

public interface SpecGroupMapper extends Mapper<SpecGroupEntity> {
}
```

com.baidu.shop.service.impl新建：SpecGroupServiceImpl

```
import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.SpecGroupDTO;
import com.baidu.shop.entity.SpecGroupEntity;
import com.baidu.shop.mapper.SpecGroupMapper;
import com.baidu.shop.service.SpecificationService;
import com.baidu.shop.utils.ObjectUtil;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.RestController;
import tk.mybatis.mapper.entity.Example;
import javax.annotation.Resource;
import java.util.List;

@RestController
public class SpecGroupServiceImpl extends BaseApiService implements
SpecGroupService {

    @Resource
    private SpecGroupMapper specGroupMapper;

    //规格组查询
    @Override
    public Result<List<SpecGroupEntity>> getSpecGroupInfo(SpecGroupDto
specGroupDto) {
        Example example = new Example(SpecGroupEntity.class);

        if (ObjectUtil.isNotNull(specGroupDto.getCid()))
                example
                .createCriteria()
                .andEqualTo("cid",
BaiduBeanUtil.copyProperties(specGroupDto,SpecGroupEntity.class).getCid());

        List<SpecGroupEntity> specGroupExample =
specGroupMapper.selectByExample(example);
        return this.setResultSuccess(specGroupExample);
    }
}
```

前台

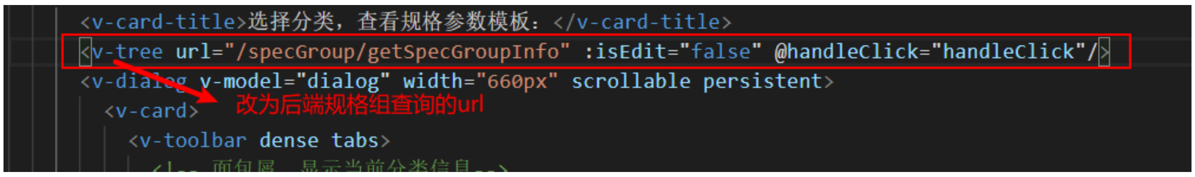
加载分类信息

index.js line:29

```
route("/item/specification",'/item/specification/Specification',"Specification")
,
```

specification/Specification.vue

```
<v-tree url="/specGroup/getSpecGroupInfo" :isEdit="false"
@handleClick="handleClick"/>
```



```
<v-card-title>选择分类，查看规格参数模板：</v-card-title>
<v-tree url="/specGroup/getSpecGroupInfo" :isEdit="false" @handleClick="handleClick"/>
<v-dialog v-model="dialog" width="660px" scrollable persistent>
  <v-card>
    <v-toolbar dense tabs>
```

规格组(查询)

SpecGroup.vue

```
loadData(){
          this.$http.get("/specGroup/getSpecGroupInfo",{
              params:{
                  cid:this.cid
              }
          })
          .then((resp) => {
              this.groups = resp.data.data;
          })
          .catch(() => {
              this.groups = [];
          })
      },
```

2.规格组(新增,删除,修改)

后台

mingrui-shop-service-api-xxx

com.baidu.shop.service：SpecGroupService

```
@ApiModelProperty(value = "新增规格组")
@PostMapping(value = "specGroup/save")
Result<JsonObject> saveSpecGroup(@RequestBody SpecGroupDto specGroupDto);

@ApiModelProperty(value = "修改规格组")
@PutMapping(value = "specGroup/save")
Result<JsonObject> editSpecGroup(@RequestBody SpecGroupDto specGroupDto);

@ApiModelProperty(value = "删除规格组")
@DeleteMapping("specGroup/delete/{id}")
Result<JsonObject> delSpecRoupById(@PathVariable Integer id);
```

mingrui-shop-service-xxx

com.baidu.shop.service.impl：SpecGroupServiceImpl

```
//规格组新增
@Transactional
@Override
public Result<JsonObject> saveSpecGroup(SpecGroupDto specGroupDto) {

 specGroupMapper.insertSelective(BaiduBeanUtil.copyProperties(specGroupDto,SpecG
roupEntity.class));
    return this.setResultSuccess();
}
```

```
//规格组修改
@Override
public Result<JsonObject> editSpecGroup(SpecGroupDto specGroupDto) {

 specGroupMapper.updateByPrimaryKeySelective(BaiduBeanUtil.copyProperties(specGr
oupDto,SpecGroupEntity.class));
    return this.setResultSuccess();
}

//规格组删除

@Override
public Result<JsonObject> delSpecRoupById(Integer id) {
    //删除规格组之前要先判断一下当前规格组下是否有规格参数
    //true : 不能被删除
    //false -->
    Example example = new Example(SpecParamEntity.class);
    example.createCriteria().andEqualTo("groupId",id);
    List<SpecParamEntity> specParamEntities =
specParamMapper.selectByExample(example);
    if (specParamEntities.size() >0){
        return this.setResultError("当前规格组有数据不能被删除");
    }

    specGroupMapper.deleteByPrimaryKey(id);
    return this.setResultSuccess();
}
```

前台

SpecGroup.vue

```
 save(){
          this.$http({
           method: this.isEdit ? 'put' : 'post',
           url: 'specGroup/save',
           data: this.group
         }).then((resp) => {
           // 关闭窗口
           if(resp.data.code == 200){
               this.show = false;
               this.$message.success("保存成功！");
               this.loadData();
           }

         }).catch(() => {
             this.$message.error("保存失败！");
           });
     },
     deleteGroup(id){
         this.$message.confirm("确认要删除分组吗？")
         .then(() => {
           this.$http.delete("/specGroup/delete/ " + id)
               .then((resp) => {
                   if(resp.data.code!=200){
                       this.$message.error(resp.data.message);
```

```
                        this.loadData();
                        return
                    }
                    this.loadData();
                    this.$message.success("删除成功")
                })
          })
      },
```

3. 规格参数(查询)

后台

mingrui-shop-service-api-xxx

com.baidu.shop.entity新建:SpecParamEntity

```java
package com.baidu.shop.entity;

import lombok.Data;

import javax.persistence.Column;
import javax.persistence.Id;
import javax.persistence.Table;

@Table(name = "tb_spec_param")
@Data
public class SpecParamEntity {

    @Id
    private Integer id;

    private Integer cid;

    private Integer groupId;

    private String name;

    @Column(name = "`numeric`")
    private Boolean numeric;

    private String unit;

    private Boolean generic;

    private Boolean searching;

    private String segments;
}
```

com.baidu.shop.dto新建:SpecParamDto

```java
package com.baidu.shop.dto;

```

```java
import com.baidu.shop.base.BaseDTO;
import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;


import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;

@Data
@ApiModel(value = "规格参数")
public class SpecParamDto extends BaseDTO {
    @ApiModelProperty(value = "主键",example = "1")
    @NotNull(message = "主键不能为空",groups = {MingruiOperation.Update.class})
    private Integer id;

    @ApiModelProperty(value = "分类id",example = "1")
    @NotNull(message = "分类id不能为空",groups = {MingruiOperation.Update.class})
    private Integer cid;

    @ApiModelProperty(value = "规格组id",example = "1")
    @NotNull(message = "规格组id不能为空",groups = {MingruiOperation.Update.class})
    private Integer groupId;

    @ApiModelProperty(value = "规格组参数名称",example = "1")
    @NotEmpty(message = "规格组参数名称不能为空",groups =
{MingruiOperation.Add.class})
    private String name;

    @ApiModelProperty(value = "是否是数字类型参数，1->true或0->false",example = "0")
    @NotNull(message = "是否是数字类型参数不能为空",groups =
{MingruiOperation.Add.class,MingruiOperation.Update.class})
    private Boolean numeric;

    @ApiModelProperty(value = "数字类型参数的单位，非数字类型可以为空",example = "1")
    @NotEmpty(message = "数字类型参数的单位不能为空",groups =
{MingruiOperation.Add.class})
    private String unit;

    @ApiModelProperty(value = "是否是sku通用属性，1->true或0->false",example = "0")
    @NotNull(message = "是否是sku通用属性不能为空",groups =
{MingruiOperation.Add.class,MingruiOperation.Update.class})
    private Boolean generic;

    @ApiModelProperty(value = "是否用于搜索过滤，true或false",example = "0")
    @NotNull(message = "是否用于搜索过滤不能为空",groups =
{MingruiOperation.Add.class,MingruiOperation.Update.class})
    private Boolean searching;

    @ApiModelProperty(value = "数值类型参数",example = "1")
    @NotEmpty(message = "数值类型参数不能为空",groups =
{MingruiOperation.Add.class})
    private String segments;
}
```

com.baidu.shop.service新建:SpecParamService

```
import org.springframework.web.bind.annotation.*;

import java.util.List;

@Api(value = "规格参数接口")
public interface SpecParamService {
    @ApiOperation(value = "查询规格参数")
    @GetMapping(value = "specParam/getSpecParamInfo")
    Result<List<SpecParamEntity>> getSpecParamInfo(SpecParamDto specParamDto);
}
```

mingrui-shop-service-xxx

com.baidu.shop.mapper包下新建SpecParamMapper

```
package com.baidu.shop.mapper;

import com.baidu.shop.entity.SpecParamEntity;
import tk.mybatis.mapper.common.Mapper;

public interface SpecParamMapper extends Mapper<SpecParamEntity> {

}
```

com.baidu.shop.service.impl新建:SpecParamServiceImpl

```
@RestController
public class SpecParamServiceImpl extends BaseApiService implements
SpecParamService {

    @Resource
    private SpecParamMapper specParamMapper;

    //查询规格参数
    @Override
    public Result<List<SpecParamEntity>> getSpecParamInfo(SpecParamDto
specParamDto) {
        Example example = new Example(SpecParamEntity.class);

 example.createCriteria().andEqualTo("groupId",specParamDto.getGroupId());
        List<SpecParamEntity> specParamEntities =
specParamMapper.selectByExample(example);
        return this.setResultSuccess(specParamEntities);
    }
}
```

前台

SpecParam.vue

```
 loadData() {
      this.$http
        .get("/specParam/getSpecParamInfo",{
           params:{
             groupId:this.group.id
           }
        })
        .then((resp) => {
          resp.data.data.forEach(p => {
              p.segments = p.segments ? p.segments.split(",").map(s =>
s.split("-")) : [];
          })

          this.params =resp.data.data;
        })
        .catch(() => {
          this.params = [];
        });
    },
```

4.规格参数(增,删,改)

后台

mingrui-shop-service-api-xxx

com.baidu.shop.service：SpecParamService

```
@ApiOperation(value = "新增规格参数")
@PostMapping(value = "specParam/save")
Result<JsonObject> saveSpecParam(@RequestBody SpecParamDto specParamDto);

@ApiOperation(value = "修改规格参数")
@PutMapping(value = "specParam/save")
Result<JsonObject> editSpecParam(@RequestBody SpecParamDto specParamDto);

@ApiOperation(value = "删除规格参数")
@DeleteMapping(value = "specParam/delete")
Result<JsonObject> delSpecParam(Integer id);
```

mingrui-shop-service-xxx

com.baidu.shop.service.impl：SpecParamServiceImpl

```

//新增规格参数
@Transactional
@Override
public Result<JsonObject> saveSpecParam(SpecParamDto specParamDto) {

 specParamMapper.insertSelective(BaiduBeanUtil.copyProperties(specParamDto,SpecP
aramEntity.class));
    return this.setResultSuccess();
}

//修改规格参数
@Transactional
@Override
public Result<JsonObject> editSpecParam(SpecParamDto specParamDto) {
 specParamMapper.updateByPrimaryKeySelective(BaiduBeanUtil.copyProperties(specParamDto,SpecParamEntity.class));
    return this.setResultSuccess();
}

//删除规格参数
@Transactional
@Override
public Result<JsonObject> delSpecParam(Integer id) {
    specParamMapper.deleteByPrimaryKey(id);
    return this.setResultSuccess();
}
```

前台

SpecParam.vue

```
methods: {
    loadData() {
      this.$http
        .get("/specParam/getSpecParamInfo",{
           params:{
             groupId:this.group.id
           }
        })
        .then((resp) => {
          resp.data.data.forEach(p => {
              p.segments = p.segments ? p.segments.split(",").map(s =>
s.split("-")) : [];
          })

          this.params =resp.data.data;
        })
        .catch(() => {
          this.params = [];
        });
    },
    editParam(param) {
        this.param = param;
        this.isEdit = true;
        this.show = true;
    },
    addParam() {
      this.param = {
          cid: this.group.cid,
          groupId: this.group.id,
          segments:[],
          numeric:false,
          searching:false,
          generic:false}
      this.show = true;
    },
    deleteParam(id) {
        this.$message.confirm("确认要删除该参数吗？")
```

```
          .then(() => {
              this.$http.delete("/specParam/delete?id=" + id)
              .then((resp) => {
                if(resp.data.code == 200){
                    this.$message.success("删除成功");
                    this.loadData();
                }

              })
              .catch(() => {
                  this.$message.error("删除失败");
              })
          })
      },
      formatBoolean(boo) {
        return boo ? "是" : "否";
      },
      save(){
          const p = {};
          Object.assign(p, this.param);
          p.segments = p.segments.map(s => s.join("-")).join(",")
          this.$http({
              method: this.isEdit ? 'put' : 'post',
              url: '/specParam/save',
              data: p,
          }).then((resp) => {
              // 关闭窗口
              if(resp.data.code == 200){
                  this.show = false;
                  this.$message.success("保存成功！");
                  this.loadData();
              }
            }).catch(() => {
                this.$message.error("保存失败！");
              });
      }
    }
```

五,商品管理

1.商品查询

后台

新建GoodsEntity

```
import lombok.Data;

import javax.persistence.Id;
import javax.persistence.Table;
import java.util.Date;

/**
 * 2 * @ClassName GoodsEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/5
 * 6 * @Version V1.0
 * 7
 **/
@Table(name = "tb_spu")
@Data
public class GoodsEntity {
    @Id
    private Integer id;


    private String title;

    private String subTitle;


    private Integer cid1;

    private Integer cid2;

    private Integer cid3;

    private Integer brandId;

    private Integer saleable;

    private Integer valid;

    private Date createTime;

    private Date lastUpdateTime;
}
```

新建GoodsDTO

```
import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.util.Date;

/**
 * 2 * @ClassName GoodsEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/5
 * 6 * @Version V1.0
 * 7
 **/
@ApiModel(value = "商品参数DTO")
@Data
public class GoodsDTO  extends  BaseDTO{
    @ApiModelProperty(value = "spuId")
    @NotNull(groups = {MingruiOperation.update.class})
    private Integer id;

    @ApiModelProperty(value = "标题")
    @NotEmpty(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private String title;

    @ApiModelProperty(value = "子标题")
    private String subTitle;

    @ApiModelProperty(value = "一级类目id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer cid1;

    @ApiModelProperty(value = "二级类目id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer cid2;

    @ApiModelProperty(value = "三级类目id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer cid3;

    @ApiModelProperty(value = "商品所属品牌id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer brandId;

    @ApiModelProperty(value = "是否上架，0下架，1上架")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer saleable;

    @ApiModelProperty(value = "是否有效，0已删除，1有效")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer valid;

    @ApiModelProperty(value = "添加时间")
    private Date createTime;

    @ApiModelProperty(value = "最后修改时间")
    private Date lostUpdateTime;

    private String categoryName;

    private String brandName;

     private  SpecDetailDTO spuDetail;

    private List<SpecSkuDTO> skus;
}
```

新建GoodsService

```java
/**
 * 2 * @ClassName GoodsService
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/5
 * 6 * @Version V1.0
 * 7
 **/
@Api(tags = "商品管理接口")
public interface GoodsService {
    @ApiOperation(value = "获取商品列表")
    @GetMapping("goods/list")
    Result<List<GoodsEntity>> getGoodsList(GoodsDTO goodsDTO);
}
```

新建GoodsServiceImpl

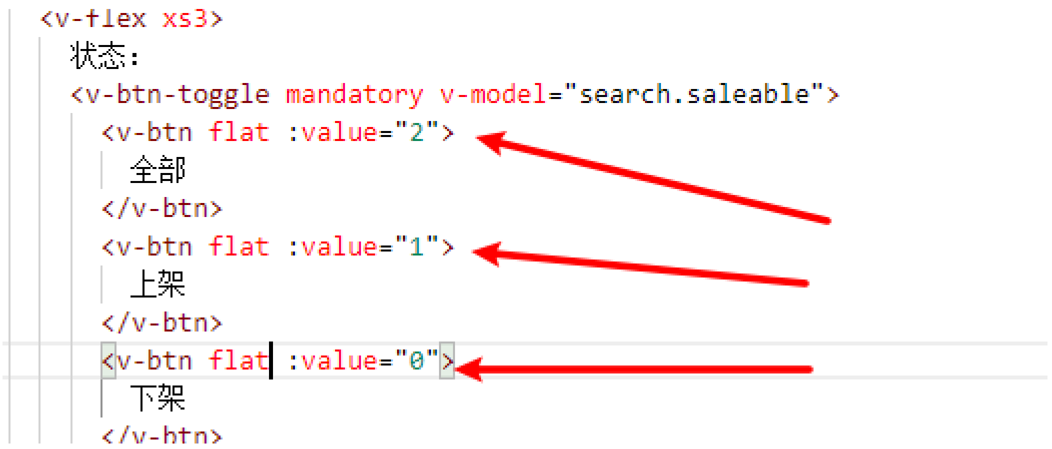
```java
package com.baidu.shop.service.impl;

import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.GoodsDTO;
import com.baidu.shop.entity.BrandEntity;
import com.baidu.shop.entity.GoodsEntity;
import com.baidu.shop.mapper.BrandMapper;
import com.baidu.shop.mapper.CategoryMapper;
import com.baidu.shop.mapper.GoodsMapper;
import com.baidu.shop.service.GoodsService;
import com.baidu.shop.status.HTTPStatus;
import com.baidu.shop.utils.ObjectEqUtil;
import com.baidu.shop.utils.TenXunBeanUtil;
import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.RestController;
import tk.mybatis.mapper.entity.Example;

import javax.annotation.Resource;
import java.util.Arrays;
import java.util.List;
import java.util.stream.Collectors;

/**
 * 2 * @ClassName GoodsServiceImpl
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/5
 * 6 * @Version V1.0
 * 7
 **/
@RestController
public class GoodsServiceImpl extends BaseApiService implements GoodsService {
    @Resource
    private GoodsMapper goodsMapper;

    @Resource
    private BrandMapper brandMapper;

    @Resource
    private CategoryMapper categoryMapper;


    @Override
    public Result<List<GoodsEntity>> getGoodsList(GoodsDTO goodsDTO) {

        if(ObjectEqUtil.isNull(goodsDTO.getPage())||
ObjectEqUtil.isNull(goodsDTO.getRows())) return this.setResultError("分页参数错
误");
            if(!StringUtils.isEmpty(goodsDTO.getSort()) &&
!StringUtils.isEmpty(goodsDTO.getOrder())){
                PageHelper.orderBy(goodsDTO.getSort()+"  "+
(Boolean.valueOf(goodsDTO.getOrder())?"desc":"asc"));
        }

        PageHelper.startPage(goodsDTO.getPage(),goodsDTO.getRows());
        Example example = new Example(GoodsEntity.class);
        Example.Criteria criteria = example.createCriteria();
        if(ObjectEqUtil.isNotNull(goodsDTO.getSaleable()) &&
goodsDTO.getSaleable() <2){
            criteria.andEqualTo("saleable",goodsDTO.getSaleable());
        }
        if(StringUtils.isEmpty(goodsDTO.getTitle())){
            criteria.andLike("title","%"+goodsDTO.getTitle()+"%");
        }



        List<GoodsEntity> goodsEntities = goodsMapper.selectByExample(example);


        List<GoodsDTO> collect = goodsEntities.stream().map(goodsEntity -> {

            String
categroyName=categoryMapper.selectCategoryNameById(Arrays.asList(goodsEntity.get
Cid1(), goodsEntity.getCid2(), goodsEntity.getCid3()));
            GoodsDTO goodsDTO1 = TenXunBeanUtil.copyProperties(goodsEntity,
GoodsDTO.class);

            BrandEntity brandEntity =
brandMapper.selectByPrimaryKey(goodsEntity.getBrandId());
            goodsDTO1.setBrandName(brandEntity.getName());
            goodsDTO1.setCategoryName(categroyName);

            return goodsDTO1;
        }).collect(Collectors.toList());
        PageInfo<GoodsEntity> pageInfo =new PageInfo<>(goodsEntities);
        return this.setResult(HTTPStatus.OK,pageInfo.getTotal()+"",collect);
    }
}
```

新建GoodsMapper

```java
import com.baidu.shop.entity.GoodsEntity;
import tk.mybatis.mapper.common.Mapper;

public interface GoodsMapper extends Mapper<GoodsEntity> {
}
```

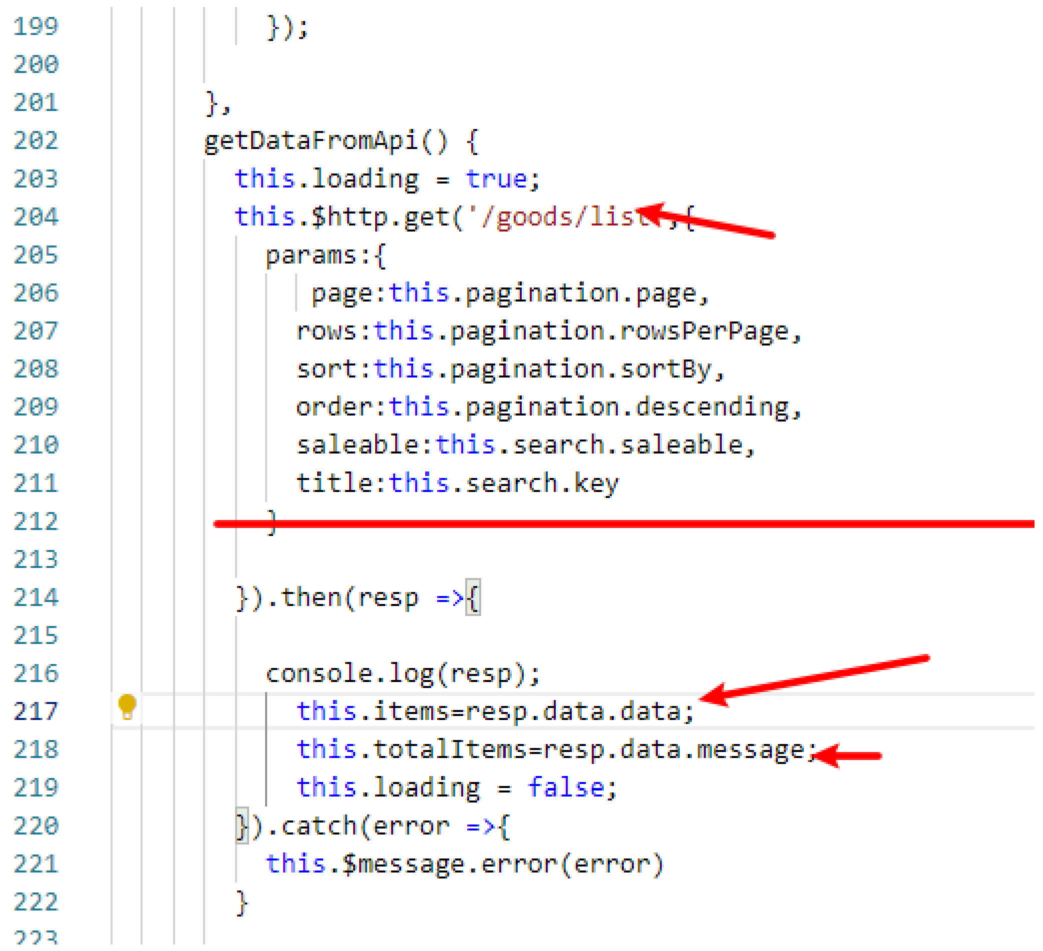
前台

修改goods.vue line 7



```html
<v-flex xs3>
  状态:
  <v-btn-toggle mandatory v-model="search.saleable">
    <v-btn flat :value="2">
      全部
    </v-btn>
    <v-btn flat :value="1">
      上架
    </v-btn>
    <v-btn flat :value="0">
      下架
    </v-btn>
```

修改 204



```javascript
199             });
200 
201         },
202         getDataFromApi() {
203           this.loading = true;
204           this.$http.get('/goods/list',{
205             params:{
206               page:this.pagination.page,
207               rows:this.pagination.rowsPerPage,
208               sort:this.pagination.sortBy,
209               order:this.pagination.descending,
210               saleable:this.search.saleable,
211               title:this.search.key
212             }
213 
214           }).then(resp =>{
215 
216             console.log(resp);
217               this.items=resp.data.data;
218               this.totalItems=resp.data.message;
219               this.loading = false;
220           }).catch(error =>{
221             this.$message.error(error)
222           }
223 
```

2.商品新增

后台

新建SpecDetailEntity

```
package com.baidu.shop.entity;

import lombok.Data;

import javax.persistence.Id;
import javax.persistence.Table;

/**
 * 2 * @ClassName SpecDetailEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/7
 * 6 * @Version V1.0
 * 7
 **/
@Table(name = "tb_spu_detail")
@Data
public class SpecDetailEntity {
    @Id
    private Integer spuId;

    private String description;

    private String genericSpec;

    private String specialSpec;

    private  String packingList;

    private String afterService;

}
```

新建SpecDetailDTO

```
package com.baidu.shop.dto;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

/**
 * 2 * @ClassName SpecDetailEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/7
 * 6 * @Version V1.0
 * 7
 **/

@Data
@ApiModel(value = "商品标题传输DTO")
public class SpecDetailDTO extends BaseDTO {
    @ApiModelProperty(value = "商品spuId")
    private Integer spuId;
    @ApiModelProperty(value = "商品描述信息")
    private String description;
    @ApiModelProperty(value = "通用规格参数数据")
    private String genericSpec;
    @ApiModelProperty(value = "特有规格参数及可选值信息，json格式")
    private String specialSpec;
    @ApiModelProperty(value = "包装清单")
    private  String packingList;
    @ApiModelProperty(value = "售后服务")
    private String afterService;

}
```

新建SpecSkuEntity

```
package com.baidu.shop.entity;

import lombok.Data;

import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
import java.util.Date;

/**
 * 2 * @ClassName SpecSkuEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/7
 * 6 * @Version V1.0
 * 7
 **/
@Table(name ="tb_sku")
@Data
public class SpecSkuEntity {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private Integer spuId;

    private String title;

    private String images;

    private Integer price;

    private  String indexes;

```

```java
    private String ownSpec;

    private Boolean enable;

    private Date  createTime;

    private Date lastUpdateTime;
}
```

新建SpecSkuDTO

```java
package com.baidu.shop.dto;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import java.util.Date;

/**
 * 2 * @ClassName SpecSkuEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/7
 * 6 * @Version V1.0
 * 7
 **/
@ApiModel(value = "商品sku传输DTO")
@Data
public class SpecSkuDTO extends BaseDTO {
    @ApiModelProperty(value = " sku id ")
    private Long id;
    @ApiModelProperty(value = "商品spuId ")
    private Integer spuId;
    @ApiModelProperty(value = " 商品标题 ")
    private String title;
    @ApiModelProperty(value = " 商品的图片，多个图片以‘,’分割")
    private String images;
    @ApiModelProperty(value = "销售价格，单位为分")
    private Integer price;
    @ApiModelProperty(value = " 特有规格属性在spu属性模板中的对应下标组合 ")
    private  String indexes;
    @ApiModelProperty(value = " sku的特有规格参数键值对，json格式，反序列化时请使用
linkedHashMap，保证有序 ")
    private String ownSpec;
    @ApiModelProperty(value = " 是否有效，0无效，1有效 ")
    private Boolean enable;
    @ApiModelProperty(value = " 添加时间 ")
    private Date  createTime;
    @ApiModelProperty(value = " 最后修改时间 ")
    private Date lastUpdateTime;
     @ApiModelProperty(value= "库存剩余")
    private Integer stock;
}
```

新建SpecStockEntity

```java
package com.baidu.shop.entity;

import lombok.Data;

import javax.persistence.Id;
import javax.persistence.Table;

/**
 * 2 * @ClassName SpecStockEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/7
 * 6 * @Version V1.0
 * 7
 **/
@Table(name = "tb_stock")
@Data
public class SpecStockEntity {
    @Id
    private Long skuId;

    private Integer seckillStock;

    private Integer seckillTotal;

    private Integer stock;




}
```

新建SpecStockDTO

```java
package com.baidu.shop.dto;

import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.persistence.Id;

/**
 * 2 * @ClassName SpecStockEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/7
 * 6 * @Version V1.0
 * 7
 **/

@Data
@ApiModel(value = "库存传输DTO")
```

```
public class SpecStockDTO {
    @Id
    @ApiModelProperty(value = "库存对应的商品sku id")
    private Long skuId;
    @ApiModelProperty(value = "可秒杀库存")
    private Integer seckillStock;
    @ApiModelProperty(value = "秒杀总数量")
    private Integer seckillTotal;
    @ApiModelProperty(value = "库存数量")
    private Integer stock;




}
```

修改goodsDTO

```
package com.baidu.shop.dto;

import com.baidu.shop.validate.group.MingruiOperation;
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;

import javax.validation.constraints.NotEmpty;
import javax.validation.constraints.NotNull;
import java.util.Date;
import java.util.List;

/**
 * 2 * @ClassName GoodsEntity
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/5
 * 6 * @Version V1.0
 * 7
 **/
@ApiModel(value = "商品参数DTO")
@Data
public class GoodsDTO  extends  BaseDTO{
    @ApiModelProperty(value = "spuId")
    @NotNull(groups = {MingruiOperation.update.class})
    private Integer id;

    @ApiModelProperty(value = "标题")
    @NotEmpty(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private String title;

    @ApiModelProperty(value = "子标题")
    private String subTitle;

    @ApiModelProperty(value = "一级类目id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer cid1;
    @ApiModelProperty(value = "二级类目id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer cid2;

    @ApiModelProperty(value = "三级类目id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer cid3;

    @ApiModelProperty(value = "商品所属品牌id")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer brandId;

    @ApiModelProperty(value = "是否上架，0下架，1上架")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer saleable;

    @ApiModelProperty(value = "是否有效，0已删除，1有效")
    @NotNull(groups =
{MingruiOperation.update.class,MingruiOperation.add.class})
    private Integer valid;

    @ApiModelProperty(value = "添加时间")
    private Date createTime;

    @ApiModelProperty(value = "最后修改时间")
    private Date lostUpdateTime;

    private String categoryName;


    private String brandName;

    private  SpecDetailDTO spuDetail;

    private List<SpecSkuDTO> skus;
}
```

新建SpecDetailMapper

```
package com.baidu.shop.mapper;

import com.baidu.shop.entity.SpecDetailEntity;
import tk.mybatis.mapper.common.Mapper;

public interface SpecDetailMapper extends Mapper<SpecDetailEntity> {
}
```

新建SpecSkuMapper

```java
package com.baidu.shop.mapper;

import com.baidu.shop.entity.SpecSkuEntity;
import tk.mybatis.mapper.common.Mapper;

public interface SpecSkuMapper extends Mapper<SpecSkuEntity> {
}
```

新建SpecStockMapper

```java
package com.baidu.shop.mapper;

import com.baidu.shop.entity.SpecStockEntity;
import tk.mybatis.mapper.common.Mapper;

public interface SpecStockMapper extends Mapper<SpecStockEntity> {
}
```

新建goods新增接口

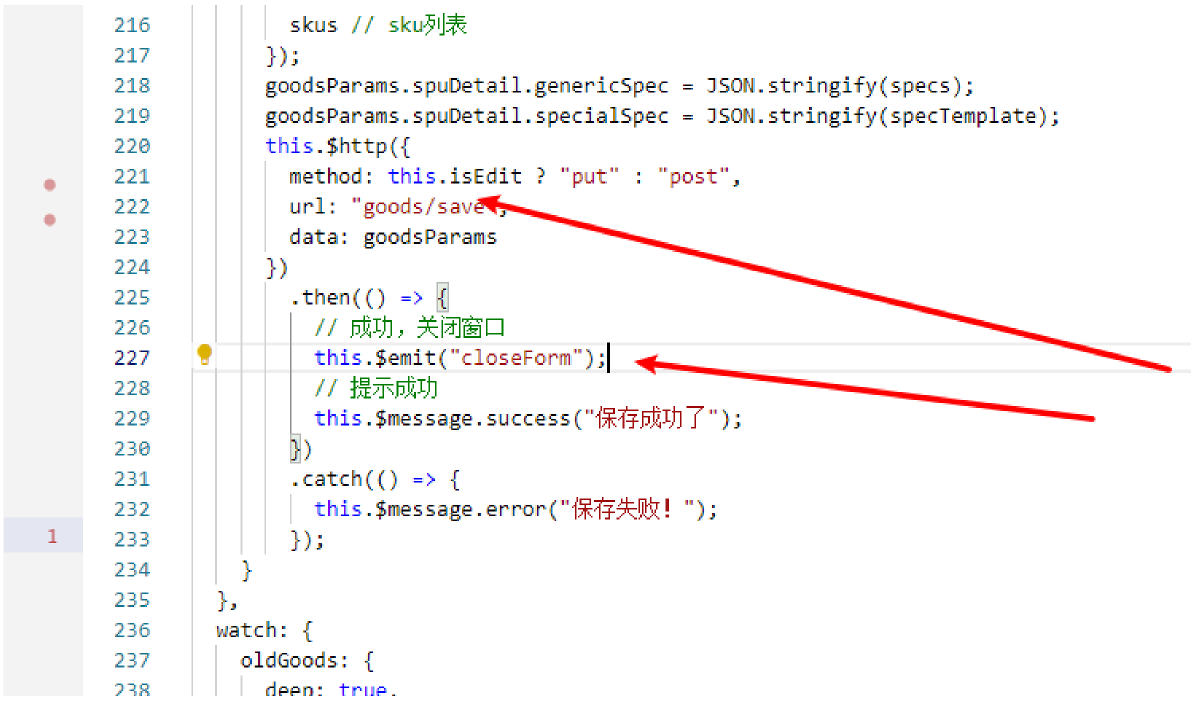
```java
package com.baidu.shop.service;

import com.baidu.shop.base.Result;
import com.baidu.shop.dto.GoodsDTO;
import com.baidu.shop.entity.GoodsEntity;
import com.google.gson.JsonObject;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;

import java.util.List;

/**
 * 2 * @ClassName GoodsService
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/5
 * 6 * @Version V1.0
 * 7
 **/
@Api(tags = "商品管理接口")
public interface GoodsService {
    @ApiOperation(value = "获取商品列表")
    @GetMapping("goods/list")
    Result<List<GoodsEntity>> getGoodsList(GoodsDTO goodsDTO);

    @ApiOperation(value = "新增商品")
    @PostMapping("goods/save")
    Result<JsonObject>  saveGoods(@RequestBody GoodsDTO goodsDTO);
}
```

实现新增商品接口

```java
package com.baidu.shop.service.impl;

import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.GoodsDTO;
import com.baidu.shop.dto.SpecDetailDTO;
import com.baidu.shop.dto.SpecSkuDTO;
import com.baidu.shop.entity.*;
import com.baidu.shop.mapper.*;
import com.baidu.shop.service.GoodsService;
import com.baidu.shop.status.HTTPStatus;
import com.baidu.shop.utils.ObjectEqUtil;
import com.baidu.shop.utils.TenXunBeanUtil;
import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;
import com.google.gson.JsonObject;
import org.springframework.transaction.annotation.Transactional;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.RestController;
import tk.mybatis.mapper.entity.Example;

import javax.annotation.Resource;
import java.util.Arrays;
import java.util.Date;
import java.util.List;
import java.util.stream.Collectors;

/**
 * 2 * @ClassName GoodsServiceImpl
 * 3 * @Description: TODO
 * 4 * @Author zzx
 * 5 * @Date 2021/1/5
 * 6 * @Version V1.0
 * 7
 **/
@RestController
public class GoodsServiceImpl extends BaseApiService implements GoodsService {
    @Resource
    private GoodsMapper goodsMapper;

    @Resource
    private BrandMapper brandMapper;

    @Resource
    private CategoryMapper categoryMapper;

    @Resource
    private SpecSkuMapper specSkuMapper;


    @Resource
    private SpecStockMapper specStockMapper;

    @Resource
    private  SpecDetailMapper specDetailMapper;
    @Override
    public Result<List<GoodsEntity>> getGoodsList(GoodsDTO goodsDTO) {
        //判断分页参数是否正确
        if(ObjectEqUtil.isNull(goodsDTO.getPage())||
ObjectEqUtil.isNull(goodsDTO.getRows())) return this.setResultError("分页参数错
误");
        PageHelper.startPage(goodsDTO.getPage(),goodsDTO.getRows());
        if(!StringUtils.isEmpty(goodsDTO.getSort()) &&
!StringUtils.isEmpty(goodsDTO.getOrder())){
                PageHelper.orderBy(goodsDTO.getSort()+"  "+
(Boolean.valueOf(goodsDTO.getOrder())?"desc":"asc"));
        }
        //创建example对象进行条件查询
        Example example = new Example(GoodsEntity.class);
        Example.Criteria criteria = example.createCriteria();
        if(ObjectEqUtil.isNotNull(goodsDTO.getSaleable()) &&
goodsDTO.getSaleable() <2){
            criteria.andEqualTo("saleable",goodsDTO.getSaleable());
        }
        if(StringUtils.isEmpty(goodsDTO.getTitle())){
            criteria.andLike("title","%"+goodsDTO.getTitle()+"%");
        }




        List<GoodsEntity> goodsEntities = goodsMapper.selectByExample(example);


        List<GoodsDTO> collect = goodsEntities.stream().map(goodsEntity -> {
            //根据分类id查询分类名称
            String
categroyName=categoryMapper.selectCategoryNameById(Arrays.asList(goodsEntity.get
Cid1(), goodsEntity.getCid2(), goodsEntity.getCid3()));

            GoodsDTO goodsDTO1 = TenXunBeanUtil.copyProperties(goodsEntity,
GoodsDTO.class);
            //根据品牌id查询品牌名称
            BrandEntity brandEntity =
brandMapper.selectByPrimaryKey(goodsEntity.getBrandId());
            goodsDTO1.setBrandName(brandEntity.getName());
            goodsDTO1.setCategoryName(categroyName);

            return goodsDTO1;
        }).collect(Collectors.toList());

        PageInfo<GoodsEntity> pageInfo =new PageInfo<>(goodsEntities);

        return this.setResult(HTTPStatus.OK,pageInfo.getTotal()+"",collect);
    }

    @Override
    @Transactional
    public Result<JsonObject> saveGoods(GoodsDTO goodsDTO) {

        final Date date = new Date();
        GoodsEntity spuEntity = TenXunBeanUtil.copyProperties(goodsDTO,
GoodsEntity.class);
        spuEntity.setSaleable(1);
        spuEntity.setValid(1);
        spuEntity.setCreateTime(date);
        spuEntity.setLastUpdateTime(date);
        //新增spu数据
        goodsMapper.insertSelective(spuEntity);


        SpecDetailDTO specDetailDTO = goodsDTO.getSpuDetail();
        SpecDetailEntity specDetailEntity =
TenXunBeanUtil.copyProperties(specDetailDTO, SpecDetailEntity.class);
        specDetailEntity.setSpuId(spuEntity.getId());
            //新增detail
        specDetailMapper.insertSelective(specDetailEntity);


        List<SpecSkuDTO> specSkuDTOS = goodsDTO.getSkus();
        specSkuDTOS.stream().forEach(sku ->{
            SpecSkuEntity specSkuEntity = TenXunBeanUtil.copyProperties(sku,
SpecSkuEntity.class);
            specSkuEntity.setSpuId(spuEntity.getId());
            specSkuEntity.setCreateTime(date);
            specSkuEntity.setLastUpdateTime(date);
            //新增sku
            specSkuMapper.insertSelective(specSkuEntity);
            SpecStockEntity specStockEntity = new SpecStockEntity();
            specStockEntity.setSkuId(specSkuEntity.getId());
            specStockEntity.setStock(sku.getStock());
            //新增stock
            specStockMapper.insertSelective(specStockEntity);
        });

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

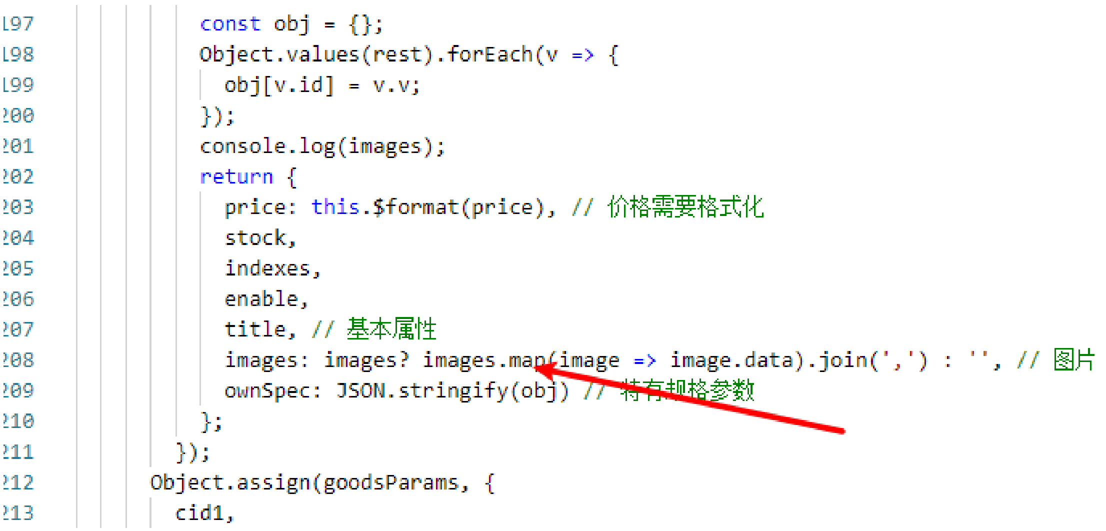
前台

修改line 221



```
216            skus // sku列表
217          });
218          goodsParams.spuDetail.genericSpec = JSON.stringify(specs);
219          goodsParams.spuDetail.specialSpec = JSON.stringify(specTemplate);
220          this.$http({
221            method: this.isEdit ? "put" : "post",
222            url: "goods/save",
223            data: goodsParams
224          })
225            .then(() => {
226              // 成功，关闭窗口
227              this.$emit("closeForm");
228              // 提示成功
229              this.$message.success("保存成功了");
230            })
231            .catch(() => {
232              this.$message.error("保存失败！");
233            });
234        }
235      },
236      watch: {
237        oldGoods: {
238          deep: true,
```

修改line 208

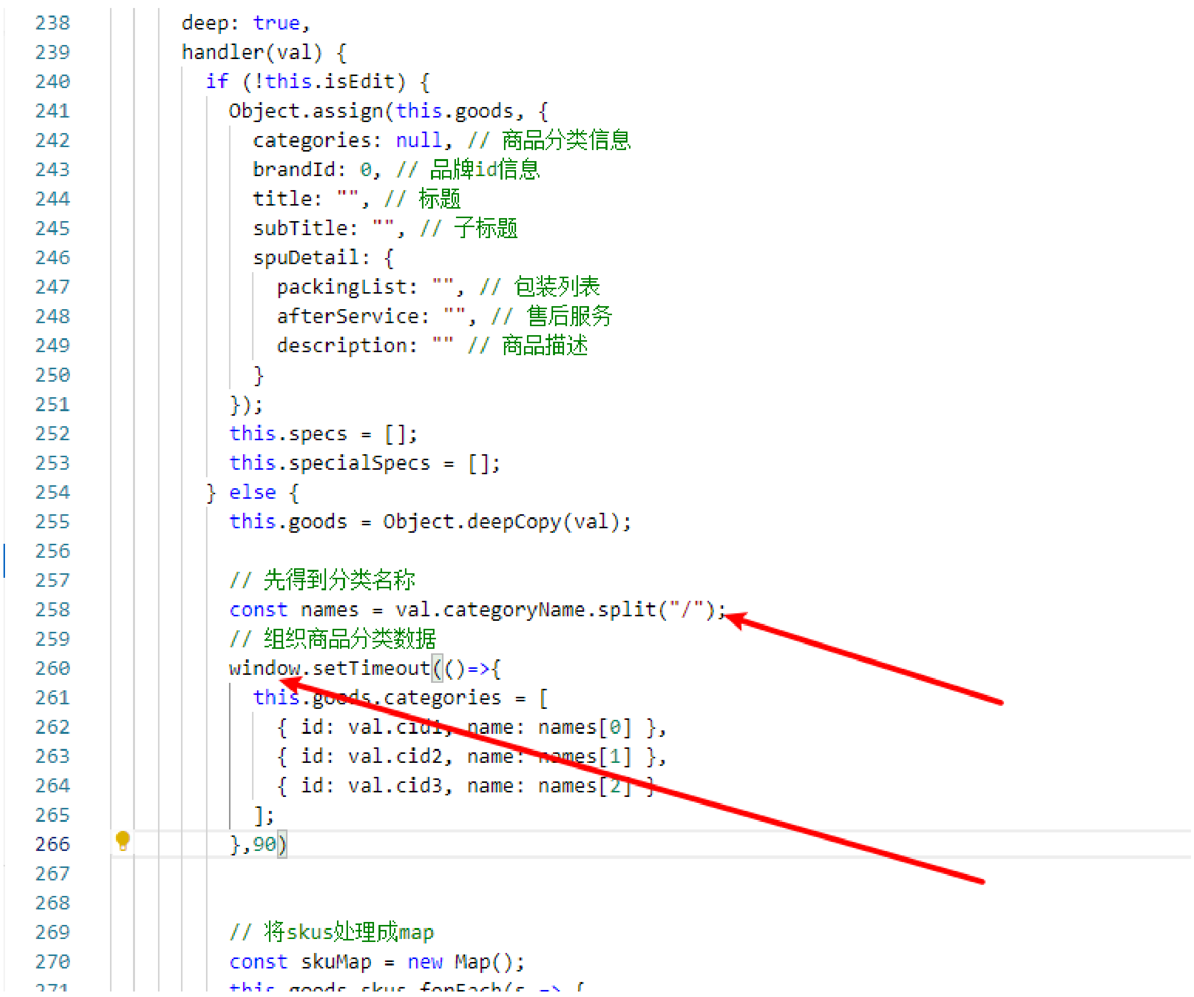


```
197              const obj = {};
198              Object.values(rest).forEach(v => {
199                obj[v.id] = v.v;
200              });
201              console.log(images);
202              return {
203                price: this.$format(price), // 价格需要格式化
204                stock,
205                indexes,
206                enable,
207                title, // 基本属性
208                images: images? images.map(image => image.data).join(',') : '', // 图片
209                ownSpec: JSON.stringify(obj) // 特有规格参数
210              };
211            });
212          Object.assign(goodsParams, {
213            cid1,
```

3.修改商品

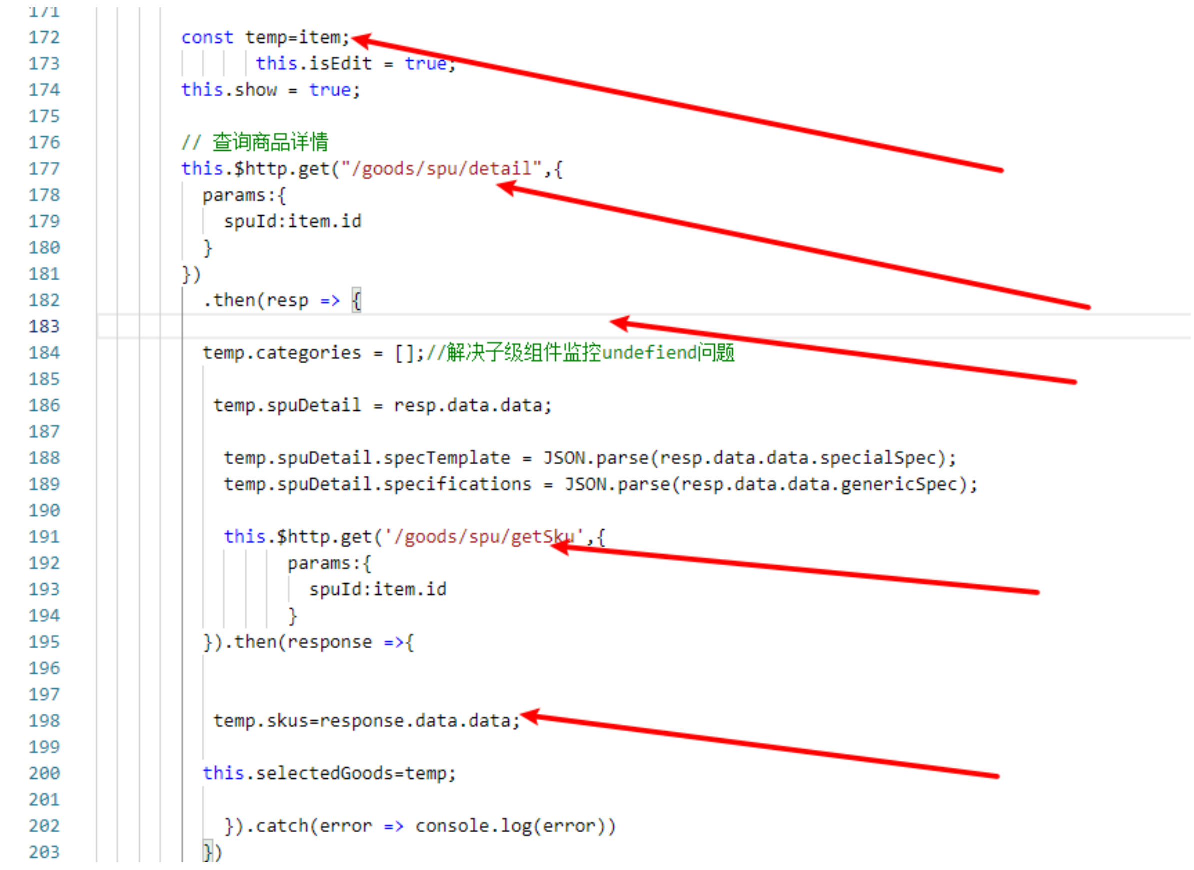
前台

修改goodForm.vue



```
        deep: true,
        handler(val) {
          if (!this.isEdit) {
            Object.assign(this.goods, {
              categories: null, // 商品分类信息
              brandId: 0, // 品牌id信息
              title: "", // 标题
              subTitle: "", // 子标题
              spuDetail: {
                packingList: "", // 包装列表
                afterService: "", // 售后服务
                description: "" // 商品描述
              }
            });
            this.specs = [];
            this.specialSpecs = [];
          } else {
            this.goods = Object.deepCopy(val);

            // 先得到分类名称
            const names = val.categoryName.split("/");
            // 组织商品分类数据
            window.setTimeout(()=>{
              this.goods.categories = [
                { id: val.cid1, name: names[0] },
                { id: val.cid2, name: names[1] },
                { id: val.cid3, name: names[2] }
              ];
            },90)


            // 将skus处理成map
            const skuMap = new Map();
            this.goods.skus.forEach(s => {
```

修改goods.vue



```
        const temp=item;
              this.isEdit = true;
        this.show = true;

        // 查询商品详情
        this.$http.get("/goods/spu/detail",{
          params:{
            spuId:item.id
          }
        })
          .then(resp => {

          temp.categories = [];//解决子级组件监控undefiend问题

           temp.spuDetail = resp.data.data;

            temp.spuDetail.specTemplate = JSON.parse(resp.data.data.specialSpec);
            temp.spuDetail.specifications = JSON.parse(resp.data.data.genericSpec);

            this.$http.get('/goods/spu/getSku',{
                  params:{
                    spuId:item.id
                  }
          }).then(response =>{


           temp.skus=response.data.data;

          this.selectedGoods=temp;

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

后台

创建通过spuId查询detail和查询sku的接口

```
@ApiOperation(value="通过spuId查询detail数据")
@GetMapping("/goods/spu/detail")
Result<SpecDetailDTO> getSpecDetailBySpuId(Integer spuId);
```

```
@GetMapping("/goods/spu/getSku")
@ApiOperation(value = "通过spuId查询sku参数")
Result<List<SpecSkuDTO>> getSkuBySpu(Integer spuId);
```

创建修改接口

```
@ApiOperation(value = "修改商品")
@PutMapping("goods/save")
Result<JsonObject>  editGoods( @Validated( {MingruiOperation.update.class})
@RequestBody GoodsDTO goodsDTO);
```

新建修改接口实现

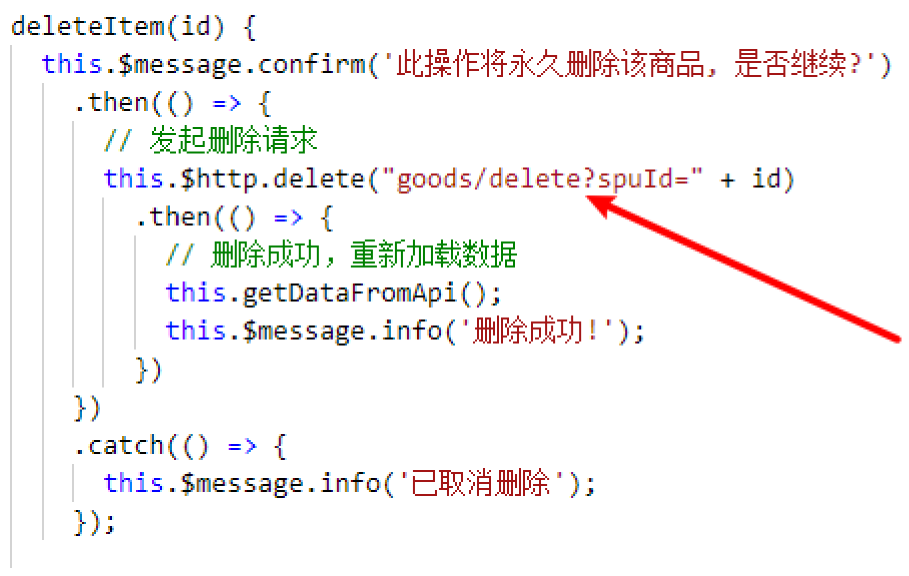
```
@Transactional
public Result<JsonObject> editGoods(GoodsDTO goodsDTO) {
    final Date date = new Date();
    //通过id删除spu数据
    GoodsEntity goodsEntity = TenXunBeanUtil.copyProperties(goodsDTO,
GoodsEntity.class);
    goodsEntity.setLastUpdateTime(date);
    goodsMapper.updateByPrimaryKeySelective(goodsEntity);
    //通过spuId删除detail中的数据
    SpecDetailEntity specDetailEntity =
TenXunBeanUtil.copyProperties(goodsDTO.getSpuDetail(), SpecDetailEntity.class);
    specDetailMapper.updateByPrimaryKeySelective(specDetailEntity);
    ////根据spuId查询出skuId通过skuId删除sku信息和stock信息
    deleteSkuAndStockBySpuId(goodsDTO.getId());
    //通过前台传来的数据新增sku列表以及库存
    saveSkuAndStockFunction(goodsDTO,goodsDTO.getId(),date);

    return this.setResultSuccess();
}


    private void deleteSkuAndStockBySpuId(Integer spuId){
        //根据spuId查询sku信息.
        Example example = new Example(SpecSkuEntity.class);
        example.createCriteria().andEqualTo("spuId",spuId);
        List<SpecSkuEntity> specSkuEntityList =
specSkuMapper.selectByExample(example);
        //获取skuId列表
        List<Long> longList = specSkuEntityList.stream().map(sku ->
sku.getId()).collect(Collectors.toList());
        //通过skuId列表删除sku数据
        specSkuMapper.deleteByIdList(longList);
        //通过skuId列表删除stock数据
        specStockMapper.deleteByIdList(longList);
    }


 private  void saveSkuAndStockFunction(GoodsDTO goodsDTO,Integer spuId, Date
date){
        List<SpecSkuDTO> specSkuDTOS = goodsDTO.getSkus();
        specSkuDTOS.stream().forEach(sku ->{
            SpecSkuEntity specSkuEntity = TenXunBeanUtil.copyProperties(sku,
SpecSkuEntity.class);
            specSkuEntity.setSpuId(spuId);
            specSkuEntity.setCreateTime(date);
            specSkuEntity.setLastUpdateTime(date);
            //新增sku
            specSkuMapper.insertSelective(specSkuEntity);
            SpecStockEntity specStockEntity = new SpecStockEntity();
            specStockEntity.setSkuId(specSkuEntity.getId());
            specStockEntity.setStock(sku.getStock());
            //新增stock
            specStockMapper.insertSelective(specStockEntity);
        });
    }

 @Override
    public Result<SpecDetailDTO> getSpecDetailBySpuId(Integer spuId) {
        SpecDetailEntity specDetailEntity =
specDetailMapper.selectByPrimaryKey(spuId);
        return this.setResultSuccess(specDetailEntity);
    }

    @Override
    public Result<List<SpecSkuDTO>> getSkuBySpu(Integer spuId) {
       List<SpecSkuDTO> list= specSkuMapper.getSkuAndStockBySpuId(spuId);
        return this.setResultSuccess(list);
    }
```

goodsMapper接口

```
import com.baidu.shop.dto.SpecSkuDTO;
import com.baidu.shop.entity.SpecSkuEntity;
import org.apache.ibatis.annotations.Select;
import tk.mybatis.mapper.additional.idlist.DeleteByIdListMapper;
import tk.mybatis.mapper.common.Mapper;

import java.util.List;

public interface SpecSkuMapper extends Mapper<SpecSkuEntity>,
DeleteByIdListMapper<SpecSkuEntity,Long> {
    @Select(value = "select sku.*,stock.stock from tb_sku sku,tb_stock stock
where sku.id=stock.sku_id and sku.spu_id=#{spuId}")
    List<SpecSkuDTO> getSkuAndStockBySpuId(Integer4 spuId);
}
```

4.删除商品

前台



```
deleteItem(id) {
  this.$message.confirm('此操作将永久删除该商品，是否继续?')
    .then(() => {
      // 发起删除请求
      this.$http.delete("goods/delete?spuId=" + id)
        .then(() => {
          // 删除成功，重新加载数据
          this.getDataFromApi();
          this.$message.info('删除成功!');
        })
    })
    .catch(() => {
      this.$message.info('已取消删除');
    });
```

后台

接口

```
@ApiOperation(value = "通过spuId删除商品")
@DeleteMapping("goods/delete")
Result<JsonObject>  deleteGoods(Integer spuId);
```

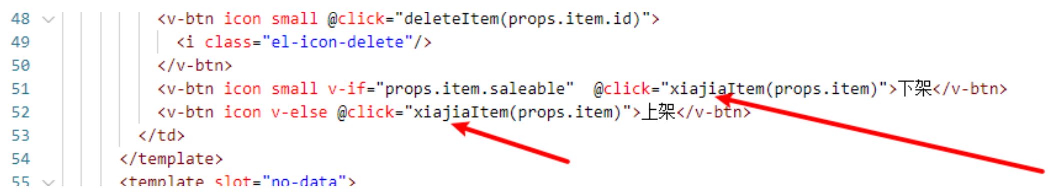
接口实现

```
@Override
@Transactional
public Result<JsonObject> deleteGoods(Integer spuId) {
    //根据id删除spu数据
    goodsMapper.deleteByPrimaryKey(spuId);
    //根据spuId删除detail数据
    specDetailMapper.deleteByPrimaryKey(spuId);
    //根据spuId查询出skuId通过skuId删除sku信息和stock信息
    deleteSkuAndStockBySpuId(spuId);
    return this.setResultSuccess();
}
```

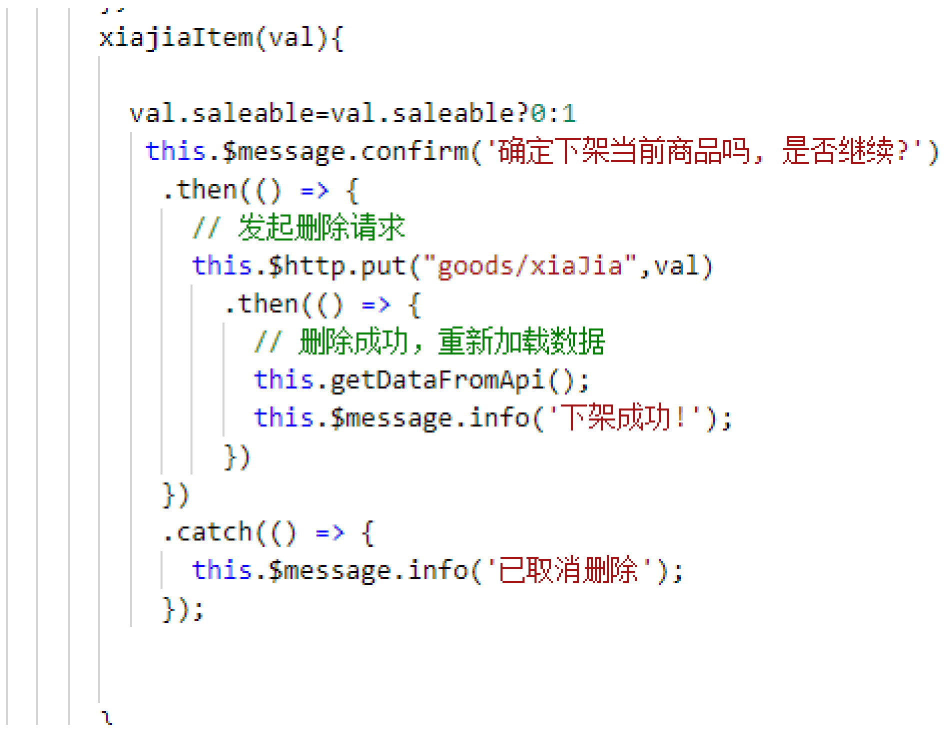
5.上下商品

前台

goods.vue新建



```
        <v-btn icon small @click="deleteItem(props.item.id)">
          <i class="el-icon-delete"/>
        </v-btn>
        <v-btn icon small v-if="props.item.saleable"  @click="xiajiaItem(props.item)">下架</v-btn>
        <v-btn icon v-else @click="xiajiaItem(props.item)">上架</v-btn>
      </td>
    </template>
    <template slot="no-data">
```



```
xiajiaItem(val){

  val.saleable=val.saleable?0:1
   this.$message.confirm('确定下架当前商品吗, 是否继续?')
    .then(() => {
      // 发起删除请求
      this.$http.put("goods/xiaJia",val)
        .then(() => {
          // 删除成功，重新加载数据
          this.getDataFromApi();
          this.$message.info('下架成功!');
        })
    })
    .catch(() => {
      this.$message.info('已取消删除');
    });
```

后台

新建xiajia接口

```
@ApiOperation(value = "修改商品上下架")
@PutMapping("goods/xiaJia")
Result<JsonObject>  xiaJiaGoods( @Validated( {MingruiOperation.update.class})
@RequestBody GoodsDTO goodsDTO);
```

新建接口实现

```
@Override
public Result<JsonObject> xiaJiaGoods(GoodsDTO goodsDTO) {
    GoodsEntity goodsEntity = TenXunBeanUtil.copyProperties(goodsDTO,
GoodsEntity.class);
    goodsMapper.updateByPrimaryKeySelective(goodsEntity);
    return this.setResultSuccess();
}
```