

1 学习目标

- 掌握fastDFS的安装,使用
- 将文件上传到fastDFS服务中
- 完成商品的CRUD

2 修改文件上传服务

2.1 FastDFS

2.1.1 什么是分布式文件系统

先思考一下，之前上传的功能，有没有什么问题？

上传本身没有任何问题，问题出在保存文件的方式，我们是保存在服务器机器，就会有下面的问题：

- 单机器存储，存储能力有限
- 无法进行水平扩展，因为多台机器的文件无法共享,会出现访问不到的情况
- 数据没有备份，有单点故障风险
- 并发能力差

这个时候，最好使用分布式文件存储来代替本地文件存储。

分布式文件系统（Distributed File System）是指文件系统管理的物理存储资源不一定直接连接在本地节点上，而是通过计算机网络与节点相连。

通俗来讲：

- 传统文件系统管理的文件就存储在本机。
- 分布式文件系统管理的文件存储在很多机器，这些机器通过网络连接，要被统一管理。无论是上传或者访问文件，都需要通过管理中心来访问

2.1.2 什么是FastDFS

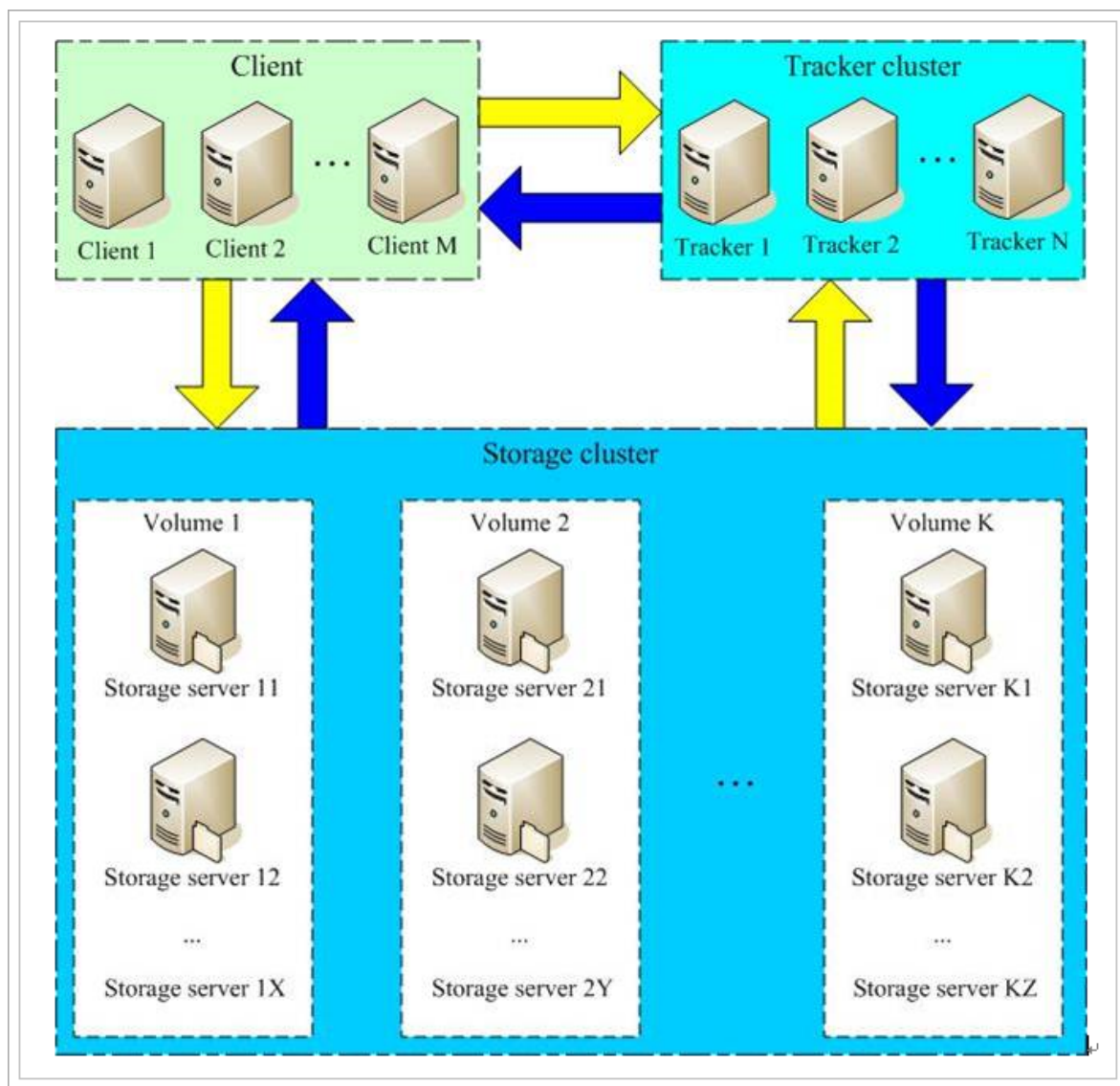
FastDFS是由淘宝的余庆先生所开发的一个轻量级、高性能的开源分布式文件系统。用纯C语言开发，功能丰富：

- 文件存储
- 文件同步
- 文件访问（上传、下载）
- 存取负载均衡
- 在线扩容

适合有大容量存储需求的应用或系统。同类的分布式文件系统有谷歌的GFS、HDFS（Hadoop）、TFS（淘宝）等。

2.1.3 FastDFS的架构

2.1.3.1 架构图



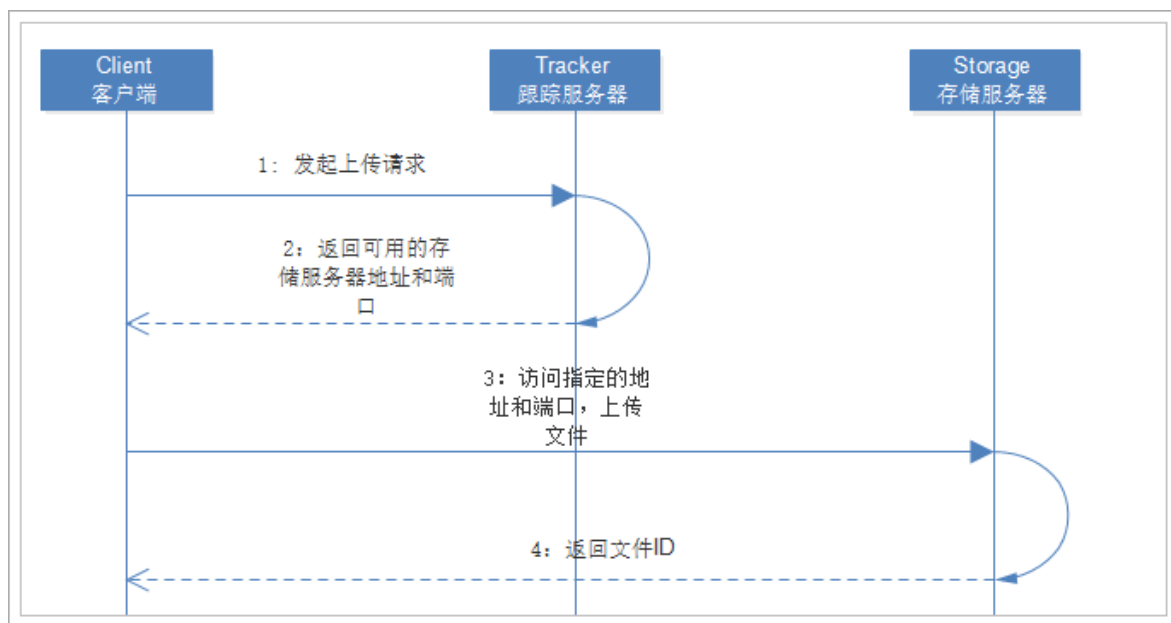
FastDFS两个主要的角色：Tracker Server 和 Storage Server 。

- Tracker Server：跟踪服务器，主要负责调度storage节点与client通信，在访问上起负载均衡的作用，和记录storage节点的运行状态，是连接client和storage节点的枢纽。
- Storage Server：存储服务器，保存文件和文件的meta data（元数据），每个storage server会启动一个单独的线程主动向Tracker cluster中每个tracker server报告其状态信息，包括磁盘使用情况，文件同步情况及文件上传下载次数统计等信息
- Group：文件组，多台Storage Server的集群。上传一个文件到同组内的一台机器上后，FastDFS会将该文件即时同步到同组内的其它所有机器上，起到备份的作用。不同组的服务器，保存的数据不同，而且相互独立，不进行通信。
- Tracker Cluster：跟踪服务器的集群，有一组Tracker Server（跟踪服务器）组成。
- Storage Cluster：存储集群，有多个Group组成。

2.1.3.2 上传和下载流程

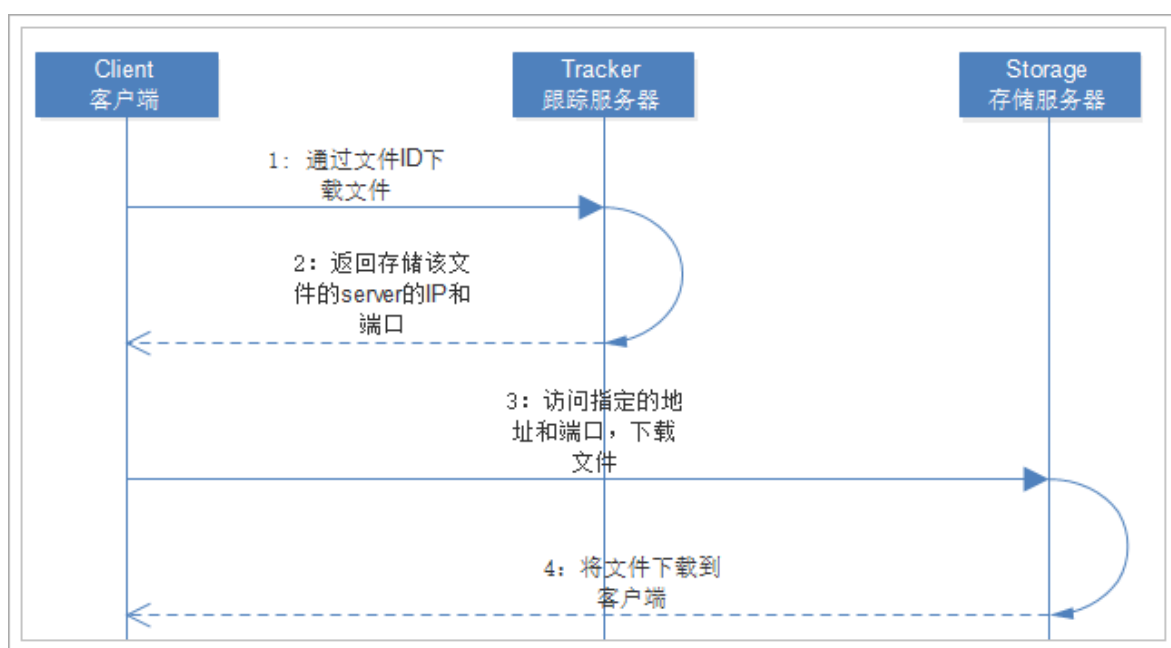
上传

时序图(UML)



1. Client通过Tracker server查找可用的Storage server。
2. Tracker server向Client返回一台可用的Storage server的IP地址和端口号。
3. Client直接通过Tracker server返回的IP地址和端口与其中一台Storage server建立连接并进行文件上传。
4. 上传完成，Storage server返回Client一个文件ID，文件上传结束。






下载



1. Client通过Tracker server查找要下载文件所在的Storage server。
2. Tracker server向Client返回包含指定文件的某个Storage server的IP地址和端口号。
3. Client直接通过Tracker server返回的IP地址和端口与其中一台Storage server建立连接并指定要下载文件。
4. 下载文件成功。

2.2 centos下安装fastDFS

解压fastDFS压缩包,得到压缩文件

 FastDFS_v5.08.tar.gz	2020/3/16 23:55	GZ 压缩文件	337 KB
 fastdfs-nginx-module_v1.16.tar.gz	2020/3/16 23:55	GZ 压缩文件	18 KB
 libevent-2.0.22-stable.tar.gz	2020/3/16 23:55	GZ 压缩文件	835 KB
 libfastcommon-master.tar	2020/8/24 15:45	TAR 压缩文件	669 KB
 nginx-1.10.0.tar.gz	2020/3/16 23:38	GZ 压缩文件	888 KB

在linux系统中自己的目录下新建fastDFS目录

```
mkdir fastDFS
```

```
[root@VM-0-5-centos jlz]# mkdir fastDFS
[root@VM-0-5-centos jlz]# ls
fastDFS  jdk  mysql  springboot-projects  tomcat
```

进入fastDFS目录

```
cd fastDFS/
```

将刚刚解压得到的压缩包全部上传到此文件内

 fastdfs-nginx-module_v1.16.tar.gz	17KB	GZ 压缩文...	2020/8/25, 14:11	-rw-r--r--	root
 FastDFS_v5.08.tar.gz	337KB	GZ 压缩文...	2020/8/25, 14:11	-rw-r--r--	root
 libevent-2.0.22-stable.tar.gz	835KB	GZ 压缩文...	2020/8/25, 14:11	-rw-r--r--	root
 libfastcommon-master.tar	669KB	TAR 压缩...	2020/8/25, 14:11	-rw-r--r--	root
 nginx-1.10.0.tar.gz	888KB	GZ 压缩文...	2020/8/25, 14:11	-rw-r--r--	root

```
[root@VM-0-5-centos fastDFS]# ls
fastdfs-nginx-module_v1.16.tar.gz  FastDFS_v5.08.tar.gz  libevent-2.0.22-stable.tar.gz  libfastcommon-master.tar  nginx-1.10.0.tar.gz
[root@VM-0-5-centos fastDFS]# ll
total 2756
-rw-r--r-- 1 root root 17510 Aug 25 14:11 fastdfs-nginx-module_v1.16.tar.gz
-rw-r--r-- 1 root root 344620 Aug 25 14:11 FastDFS_v5.08.tar.gz
-rw-r--r-- 1 root root 854987 Aug 25 14:11 libevent-2.0.22-stable.tar.gz
-rw-r--r-- 1 root root 685056 Aug 25 14:11 libfastcommon-master.tar
-rw-r--r-- 1 root root 908954 Aug 25 14:11 nginx-1.10.0.tar.gz
```

2.2.1 安装C/C++ 编译环境

```
yum -y install gcc gcc-c++ autoconf pcre pcre-devel make automake
```

```
Installed:
  autoconf.noarch 0:2.69-11.el7          automake.noarch 0:1.13.4-3.el7          gcc-c++.x86_64 0:4.8.5-39.el7          pcre-devel.x86_64 0:8.32-17.el7

Dependency Installed:
  libstdc++.devel.x86_64 0:4.8.5-39.el7  m4.x86_64 0:1.4.16-10.el7          perl-Data-Dumper.x86_64 0:2.145-3.el7  perl-Test-Harness.noarch 0:3.28-3.el7
  perl-Thread-Queue.noarch 0:3.02-2.el7

Complete!
```

2.2.2 安装libevent

```
yum -y install libevent
```

```
Total download size: 214 k
Installed size: 725 k
Downloading packages:
libevent-2.0.21-4.el7.x86_64.rpm
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : libevent-2.0.21-4.el7.x86_64
  Verifying  : libevent-2.0.21-4.el7.x86_64

Installed:
  libevent.x86_64 0:2.0.21-4.el7

Complete!
```

2.2.3 安装 libfastcommon

```
tar -xvf libfastcommon-master.tar #解压压缩包
cd libfastcommon-master/ #进入目录
./make.sh #编译
```

```
[root@VM-0-5-centos fastDSF]# cd libfastcommon-master/
[root@VM-0-5-centos libfastcommon-master]# ls
HISTORY  INSTALL  libfastcommon.spec  make.sh  php-fastcommon  README  src
[root@VM-0-5-centos libfastcommon-master]# ./make.sh
```

```
./make.sh install #安装
cp /usr/lib64/libfastcommon.so /usr/lib #复制镜像
```

```
[root@VM-0-5-centos libfastcommon-master]# cp /usr/lib64/libfastcommon.so /usr/lib
cp: overwrite '/usr/lib/libfastcommon.so' ? y
```

2.2.4 安装FastDFS

```
cd .. #进入fastDFS目录
tar -zxvf FastDFS_v5.08.tar.gz #解压压缩包
cd FastDFS/ #进入目录
./make.sh #编译
./make.sh install #安装
```

2.2.4.1 启动tracker

```
cd /etc/fdfs #进入fastDFS配置目录
cp tracker.conf.sample tracker.conf #复制文件
vi tracker.conf #编辑刚刚复制的配置文件
```

修改配置

```
base_path=/shop/fdfs/tracker
```

```
# the base path to store data and log files
base_path=/shop/fdfs/tracker
```

保存并退出文件

创建刚刚设置的目录

```
mkdir -p /shop/fdfs/tracker
```

启动tracker

```
service fdfs_trackerd start #停止换成stop
```

```
[root@VM-0-5-centos fdfs]# mkdir -p /shop/fdfs/tracker
[root@VM-0-5-centos fdfs]# service fdfs_trackerd start
Reloading systemd: [ OK ]
Starting fdfs_trackerd (via systemctl): [ OK ]
```

2.2.4.2 启动storage

```
cp storage.conf.sample storage.conf
vi storage.conf
```

修改配置

```
base_path=/shop/fdfs/storage
store_path0=/shop/fdfs/storage
tracker_server=本机ip地址:22122 #注意不能写localhost,需要写ip地址
```

```
# the base path to store data and log files
base_path=/shop/fdfs/storage
```

```
# store_path#, based 0, if store_path0 not exists, it's value is base_path
# the paths must be exist
store_path0=/shop/fdfs/storage
#store_path1=/home/yuqing/fastdfs2
```

```
# tracker_server can occur more than once, and tracker_server format is
# "host:port", host can be hostname or ip address
tracker_server=119.45.188.14:22122
```

保存并退出

创建刚刚设置的目录

```
mkdir -p /shop/fdfs/storage
```

启动storage

```
service fdfs_storaged start
```

```
[root@VM-0-5-centos fdfs]# mkdir -p /shop/fdfs/storage
[root@VM-0-5-centos fdfs]# service fdfs_storaged start
Starting fdfs_storaged (via systemctl): [ OK ]
```

查看进程

```
ps -ef | grep fdfs
```

```
[root@VM-0-5-centos fdfs]# ps -ef | grep fdfs
root      8475      1  0 14:33 ?        00:00:00 /usr/bin/fdfs_trackerd /etc/fdfs/tracker.conf
root      9582      1  2 14:41 ?        00:00:00 /usr/bin/fdfs_storaged /etc/fdfs/storage.conf
root      9670    5986  0 14:41 pts/1    00:00:00 grep --color=auto fdfs
```

2.2.5 安装FastDFS的Nginx模块

进入压缩包所在的目录

```
[root@VM-0-5-centos fastDFS]# ls
FastDFS  fastdfs-nginx-module_v1.16.tar.gz  FastDFS_v5.08.tar.gz  libevent-2.0.22-stable.tar.gz  libfastcommon-master  libfastcommon-master.tar  nginx-1.10.0.tar.gz
```

```
tar -zxvf fastdfs-nginx-module_v1.16.tar.gz #解压压缩包
cd fastdfs-nginx-module/src/ #进入目录
vi config
//直接输入下面的命令,注意冒号需要手敲
: --> %s+/usr/local/+/usr/+g --> enter --> esc --> :wq --> enter
```

保存并退出

```
cp mod_fastdfs.conf /etc/fdfs/ #复制文件
vi /etc/fdfs/mod_fastdfs.conf
```

修改默认配置

```
connect_timeout=10
tracker_server=本机ip地址:22122 #不能写localhost
url_have_group_name = true
store_path0=/shop/fdfs/storage
```

```
# connect timeout in second
# default value is 30s
connect_timeout=10
```

```
# FastDFS tracker_server can occur more than once, and tracker_server format is
# "host:port", host can be hostname or ip address
# valid only when load_fdfs_parameters_from_tracker is true
tracker_server=119.45.188.14:22122
```

```
# if the url / uri including the group name
# set to false when uri like /M00/00/00/xxx
# set to true when uri like ${group_name}/M00/00/00/xxx, such as group1/M00/xxx
# default value is false
url_have_group_name = true
```

```
# store_path#, based 0, if store_path0 not exists, it's value is base_path
# the paths must be exist
# must same as storage.conf
store_path0=/shop/fdfs/storage
#store_path1=/home/yuqing/fastdfs1
```

保存并退出

进入FastDFS/conf目录

```
[root@VM-0-5-centos conf]# pwd
/jlz/fastDFS/FastDFS/conf
[root@VM-0-5-centos conf]# ls
anti-steal.jpg  client.conf  http.conf  mime.types  storage.conf  storage_ids.conf  tracker.conf
[root@VM-0-5-centos conf]#
```

```
cp http.conf mime.types /etc/fdfs/ #复制文件
```

2.2.6 安装Nginx

进入压缩包所在的目录

```
[root@VM-0-5-centos fastDFS]# ls
FastDFS          fastdfs-nginx-module_v1.16.tar.gz  libevent-2.0.22-stable.tar.gz  libfastcommon-master.tar
fastdfs-nginx-module  FastDFS_v5.08.tar.gz              libfastcommon-master          nginx-1.10.0.tar.gz
[root@VM-0-5-centos fastDFS]#
```

```
tar -zxvf nginx-1.10.0.tar.gz #解压压缩包
yum -y install make zlib-devel libtool openssl openssl-devel #安装依赖
cd nginx-1.10.0/
./configure --prefix=/opt/nginx --sbin-path=/usr/bin/nginx --add-
module=/jlz/fastDFS/fastdfs-nginx-module/src
```

注意:

--prefix= --> nginx配置文件所在目录

--sbin-path= --> 执行程序文件所在目录

--add-module= --> **外部模块路径，启用对外部模块的支持 这个路径一定不能配置错,就写你自己 fastdfs-nginx-module的目录即可**

```
make && make install #编译并安装
```



```

    || cp conf/fastcgi_params '/opt/nginx/conf'
cp conf/fastcgi_params \
    '/opt/nginx/conf/fastcgi_params.default'
test -f '/opt/nginx/conf/fastcgi.conf' \
    || cp conf/fastcgi.conf '/opt/nginx/conf'
cp conf/fastcgi.conf '/opt/nginx/conf/fastcgi.conf.default'
test -f '/opt/nginx/conf/uwsgi_params' \
    || cp conf/uwsgi_params '/opt/nginx/conf'
cp conf/uwsgi_params \
    '/opt/nginx/conf/uwsgi_params.default'
test -f '/opt/nginx/conf/scgi_params' \
    || cp conf/scgi_params '/opt/nginx/conf'
cp conf/scgi_params \
    '/opt/nginx/conf/scgi_params.default'
test -f '/opt/nginx/conf/nginx.conf' \
    || cp conf/nginx.conf '/opt/nginx/conf/nginx.conf'
cp conf/nginx.conf '/opt/nginx/conf/nginx.conf.default'
test -d '/opt/nginx/logs' \
    || mkdir -p '/opt/nginx/logs'
test -d '/opt/nginx/logs' \
    || mkdir -p '/opt/nginx/logs'
test -d '/opt/nginx/html' \
    || cp -R html '/opt/nginx'
test -d '/opt/nginx/logs' \
    || mkdir -p '/opt/nginx/logs'
make[1]: Leaving directory `/j1z/fastDSF/nginx-1.10.0'

```

编辑nginx的配置文件

```
vi /opt/nginx/conf/nginx.conf
```

server的配置

```

listen      80;
server_name localhost:8888;#storage的端口号

# 监听域名中带有group的，交给FastDFS模块处理
location ~/group([0-9])/ {
    ngx_fastdfs_module;
}

```

保存并退出

```
cd /opt/nginx #nginx主目录
```

```
[root@VM-0-5-centos nginx]# pwd
/opt/nginx
```

启动nginx

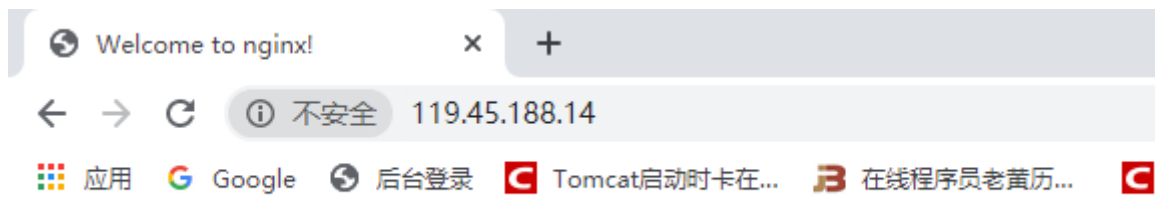
```

nginx
nginx -s stop #停止
nginx -s reload #重新加载配置

```

```
[root@VM-0-5-centos nginx]# nginx
ngx_http_fastdfs_set pid=16352
```

浏览器输入ip地址访问



至此,fastDFS安装成功,尽量不要重启云服务,因为没有设置开机启动.....想设置的话自行百度吧

2.3 java上传文件到fastDFS服务

2.3.1 pom.xml

```
<dependency>
  <groupId>com.github.tobato</groupId>
  <artifactId>fastdfs-client</artifactId>
  <version>1.26.1-RELEASE</version>
</dependency>
```

2.3.2 application.yml

```
fdfs:
  so-timeout: 1501
  connect-timeout: 601
  thumb-image: # 缩略图
    width: 60
    height: 60
  tracker-list: # tracker地址
    - 119.45.191.248:22122

mingrui:
  upload:
    path:
      windows: E:\\upload
      linux: /shenyaqi/upload
  img:
    #图片服务器主机
    host: http://119.45.191.248/
```

2.3.3 config包下新建配置类FastClientImporter

```
import com.github.tobato.fastdfs.FdfsClientConfig;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.annotation.EnableMBeanExport;
import org.springframework.context.annotation.Import;
import org.springframework.jmx.support.RegistrationPolicy;

/**
 * @ClassName FastClientImporter
```

```

    * @Description: TODO
    * @Author shenyaqi
    * @Date 2020/8/24
    * @Version v1.0
    **/
@Configuration
@Import(FdfsClientConfig.class)
// 解决jmx重复注册bean的问题
@EnableMBeanExport(registration = RegistrationPolicy.IGNORE_EXISTING)
public class FastClientImporter {
}

```

2.3.4 上传文件controller

```

import com.baidu.shop.base.Result;
import com.baidu.shop.status.HTTPStatus;
import com.github.tobato.fastdfs.domain.StorePath;
import com.github.tobato.fastdfs.domain.ThumbImageConfig;
import com.github.tobato.fastdfs.service.FastFileStorageClient;
import lombok.extern.slf4j.Slf4j;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.io.InputStream;

/**
 * @ClassName FastDFSUploadController
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/24
 * @Version v1.0
 **/
@RestController
@RequestMapping(value = "upload")
@Slf4j
public class FastDFSUploadController {

    //图片服务器的地址
    @Value(value = "${mingrui.upload.img.host}")
    private String imgHost;

    @Autowired
    private FastFileStorageClient storageClient;

    @Autowired
    private ThumbImageConfig thumbImageConfig;

    @PostMapping

```

```

    public Result<String> uploadImg(@RequestParam MultipartFile file) throws
IOException {
        InputStream inputStream = file.getInputStream();//获取文件输入流

        String filename = file.getOriginalFilename();//文件名

        String ex = filename.substring(filename.lastIndexOf(".") + 1);//文件后缀名
        // 上传并且生成缩略图
        StorePath storePath = this.storageClient.uploadImageAndCrtThumbImage(
            inputStream, file.getSize(), ex, null);//上传
        // 带分组的路径
        log.info("上传图片全路径:{}", storePath.getFullPath());
        // 不带分组的路径
        log.info("上传图片路径:{}", storePath.getPath());
        // 获取缩略图路径
        String path =
thumbImageConfig.getThumbImagePath(storePath.getFullPath());
        log.info("缩略图路径:{}", path);

        return new Result<String>(HTTPStatus.OK, "上传成功", imgHost + path);
    }
}

```

3 SPU和SKU数据结构

规格确定以后，就可以添加商品了,先看下数据库表

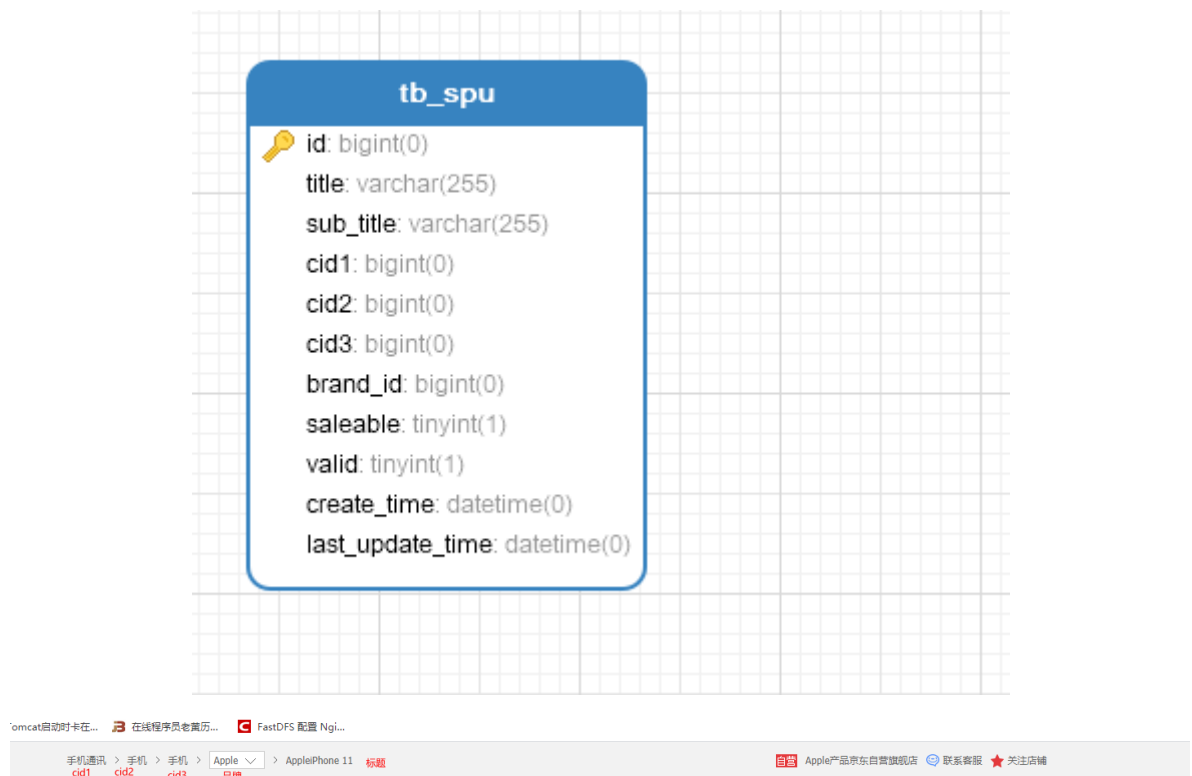
3.1 SPU表

3.1.1 tb_spu

```

CREATE TABLE `tb_spu` (
  `id` bigint(0) NOT NULL AUTO_INCREMENT COMMENT 'spu id',
  `title` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
DEFAULT '' COMMENT '标题',
  `sub_title` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
DEFAULT '' COMMENT '子标题',
  `cid1` bigint(0) NOT NULL COMMENT '1级类目id',
  `cid2` bigint(0) NOT NULL COMMENT '2级类目id',
  `cid3` bigint(0) NOT NULL COMMENT '3级类目id',
  `brand_id` bigint(0) NOT NULL COMMENT '商品所属品牌id',
  `saleable` tinyint(1) NOT NULL DEFAULT 1 COMMENT '是否上架, 0下架, 1上架',
  `valid` tinyint(1) NOT NULL DEFAULT 1 COMMENT '是否有效, 0已删除, 1有效',
  `create_time` datetime(0) NULL DEFAULT NULL COMMENT '添加时间',
  `last_update_time` datetime(0) NULL DEFAULT NULL COMMENT '最后修改时间',
  PRIMARY KEY (`id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 234 CHARACTER SET = utf8 COLLATE =
utf8_general_ci COMMENT = 'spu表, 该表描述的是一个抽象性的商品, 比如 iphone8'
ROW_FORMAT = Dynamic;

```



与我们前面分析的基本类似，但是似乎少了一些字段，比如商品描述。

我们做了表的垂直拆分，将SPU的详情放到了另一张表：

3.1.2 tb_spu_detail

```
CREATE TABLE `tb_spu_detail` (  
  `spu_id` bigint(0) NOT NULL,  
  `description` text CHARACTER SET utf8 COLLATE utf8_general_ci NULL COMMENT '商品描述信息',  
  `generic_spec` varchar(3000) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL DEFAULT '' COMMENT '通用规格参数数据',  
  `special_spec` varchar(1000) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL COMMENT '特有规格参数及可选值信息，json格式',  
  `packing_list` varchar(1000) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT '' COMMENT '包装清单',  
  `after_service` varchar(1000) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT '' COMMENT '售后服务',  
  PRIMARY KEY (`spu_id`) USING BTREE  
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci ROW_FORMAT = Dynamic;
```

tb_spu_detail


🔑 spu_id: bigint(0)
description: text
generic_spec: varchar(3000)
special_spec: varchar(1000)
packing_list: varchar(1000)
after_service: varchar(1000)

🍏 iPhone 11

一切都刚刚好。

商品描述信息



主体	入网型号 	A2223	通用和特有规格
	品牌	Apple	
	产品名称	iPhone 11	
	上市年份	2019年	
	上市月份	9月	
基本信息	机身长度 (mm)	150.9	
	机身重量 (g)	194	
	机身材质工艺	玻璃搭配铝金属设计	
	机身宽度 (mm)	75.7	
	机身材质分类	玻璃后盖	
	机身厚度 (mm)	8.3	
	运营商标志或内容 	无	
主芯片	CPU品牌	以官网信息为准	
屏幕	屏幕像素密度 (ppi) 	326	
	屏幕材质类型	LCD	
	屏幕刷新率	其他	
	主屏幕尺寸 (英寸)	6.1英寸	
后置摄像头	后摄主摄光圈	f/2.4	
	拍照特点	数码变焦; HDR; 光学防抖	
	闪光灯	其他闪光灯	
	后摄主摄光学防抖	支持光学防抖	
前置摄像头	前摄主摄光圈	f/2.2	
	拍照特点	HDR; 人像模式	
包装清单	装有 iOS 13 的 iPhone,采用闪电接头的 EarPods,闪电转 USB 连接线,USB 电源适配器,资料		包装清单

售后保障

售后



厂家服务

本产品全国联保, 享受三包服务, 质保期为: 一年质保
如因质量问题或故障, 凭厂商维修中心或特约维修点的质量检测证明, 享受7日内退货, 15日内换货, 15日以上在质保期内享受免费保修等三包服务!
(注:如厂家在商品介绍中有售后保障的说明,则此商品按照厂家说明执行售后保障服务。)



京东承诺

京东平台卖家销售并发货的商品, 由平台卖家提供发票和相应的售后服务。请您放心购买!
注: 因厂家会在没有任何提前通知的情况下更改产品包装、产地或者一些附件, 本司不能确保客户收到的货物与商城图片、产地、附件说明完全一致。只能确保为原厂正货! 并且保证与当时市场上同样主流新品一致。若本商城没有及时更新, 请大家谅解!



正品行货

京东商城向您保证所售商品均为正品行货, 京东自营商品开具机打发票或电子发票。



全国联保

凭质保证书及京东商城发票, 可享受全国联保服务 (奢侈品、钟表除外; 奢侈品、钟表由京东联系保修, 享受法定三包售后服务), 与您亲临商场选购的商品享受相同的质量保证。京东商城还为您提供具有竞争力的商品价格和**运费政策**, 请您放心购买!

注: 因厂家会在没有任何提前通知的情况下更改产品包装、产地或者一些附件, 本司不能确保客户收到的货物与商城图片、产地、附件说明完全一致。只能确保为原厂正货! 并且保证与当时市场上同样主流新品一致。若本商城没有及时更新, 请大家谅解!



无忧退货

客户购买京东自营商品7日内 (含7日, 自客户收到商品之日起计算), 在保证商品完好的前提下, 可无理由退货。(部分商品除外, 详情请见各商品细则)

这张表中的数据都比较大，为了不影响主表的查询效率我们拆分出这张表。

需要注意的是这两个字段：generic_spec和special_spec。

0.0]

object {15}

1 : 魅族 (MEIZU)
2 : 魅蓝X
3 : 2016
5 : 165
6 : 其它
7 : Android
8 : 联发科 (MTK)
9 : P20 (MT6757)
10 : 八核
11 : 2.3
14 : 5.5
15 : 1920*1080 (FHD)
16 : 500
17 : 1200
18 : 3200

cid	group_id	name	numeric
1	76	1 品牌	
2	76	1 型号	
3	76	1 上市年份	
4	76	2 机身颜色	
5	76	2 机身重量 (g)	
6	76	2 机身材质工艺	
7	76	3 操作系统	
8	76	4 CPU品牌	
9	76	4 CPU型号	
10	76	4 CPU核数	
11	76	4 CPU频率	
12	76	5 内存	
13	76	5 机身存储	
14	76	11 主屏幕尺寸 (英寸)	
15	76	11 分辨率	
16	76	6 前置摄像头	
17	76	6 后置摄像头	
18	76	7 电池容量 (mAh)	
19	90	12 品牌	
20	90	12 适用机型	
21	90	13 贴膜尺寸	
22	90	13 材质	
23	90	13 类型	

object {3}

4 :
0 : 珠光白
1 : 流光金

12 [1]
0 : 3GB

13 :
0 : 32GB

cid	group_id	name	numeric	unit	generic	search
1	76	1 品牌	0		1	
2	76	1 型号	0		1	
3	76	1 上市年份	1 年		1	
4	76	2 机身颜色	0		0	
5	76	2 机身重量 (g)	1 g		1	
6	76	2 机身材质工艺	0		1	
7	76	3 操作系统	0		1	
8	76	4 CPU品牌	0		1	
9	76	4 CPU型号	0		1	
10	76	4 CPU核数	0		1	
11	76	4 CPU频率	1 GHz		1	
12	76	5 内存	0		0	
13	76	5 机身存储	0		0	
14	76	11 主屏幕尺寸 (英寸)	1 英寸		1	
15	76	11 分辨率	0		1	
16	76	6 前置摄像头	1 万		1	
17	76	6 后置摄像头	1 万		1	
18	76	7 电池容量 (mAh)	1 mAh		1	
19	90	12 品牌	0		1	
20	90	12 适用机型	0		0	
21	90	13 贴膜尺寸	1 英寸		1	
22	90	13 材质	0		1	
23	90	13 类型	0		1	

大家可以想一下:

special_spec字段中就是展示的下图的内容



3.2 SKU

3.2.1 tb_sku

```
CREATE TABLE `tb_sku` (
  `id` bigint(0) NOT NULL AUTO_INCREMENT COMMENT 'sku id',
  `spu_id` bigint(0) NOT NULL COMMENT 'spu id',
  `title` varchar(255) CHARACTER SET utf8 COLLATE utf8_general_ci NOT NULL
  COMMENT '商品标题',
  `images` varchar(1000) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  '' COMMENT '商品的图片，多个图片以','分割',
  `price` bigint(0) NOT NULL DEFAULT 0 COMMENT '销售价格，单位为分',
  `indexes` varchar(100) CHARACTER SET utf8 COLLATE utf8_general_ci NULL DEFAULT
  '' COMMENT '特有规格属性在spu属性模板中的对应下标组合',
  `own_spec` varchar(1000) CHARACTER SET utf8 COLLATE utf8_general_ci NULL
  DEFAULT '' COMMENT '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',
  `enable` tinyint(1) NOT NULL DEFAULT 1 COMMENT '是否有效，0无效，1有效',
  `create_time` datetime(0) NOT NULL COMMENT '添加时间',
  `last_update_time` datetime(0) NOT NULL COMMENT '最后修改时间',
  PRIMARY KEY (`id`) USING BTREE,
  INDEX `key_spu_id` (`spu_id`) USING BTREE
) ENGINE = InnoDB AUTO_INCREMENT = 27359021548 CHARACTER SET = utf8 COLLATE =
utf8_general_ci COMMENT = 'sku表,该表表示具体的商品实体,如黑色的 64g的iphone 8'
ROW_FORMAT = Dynamic;
```



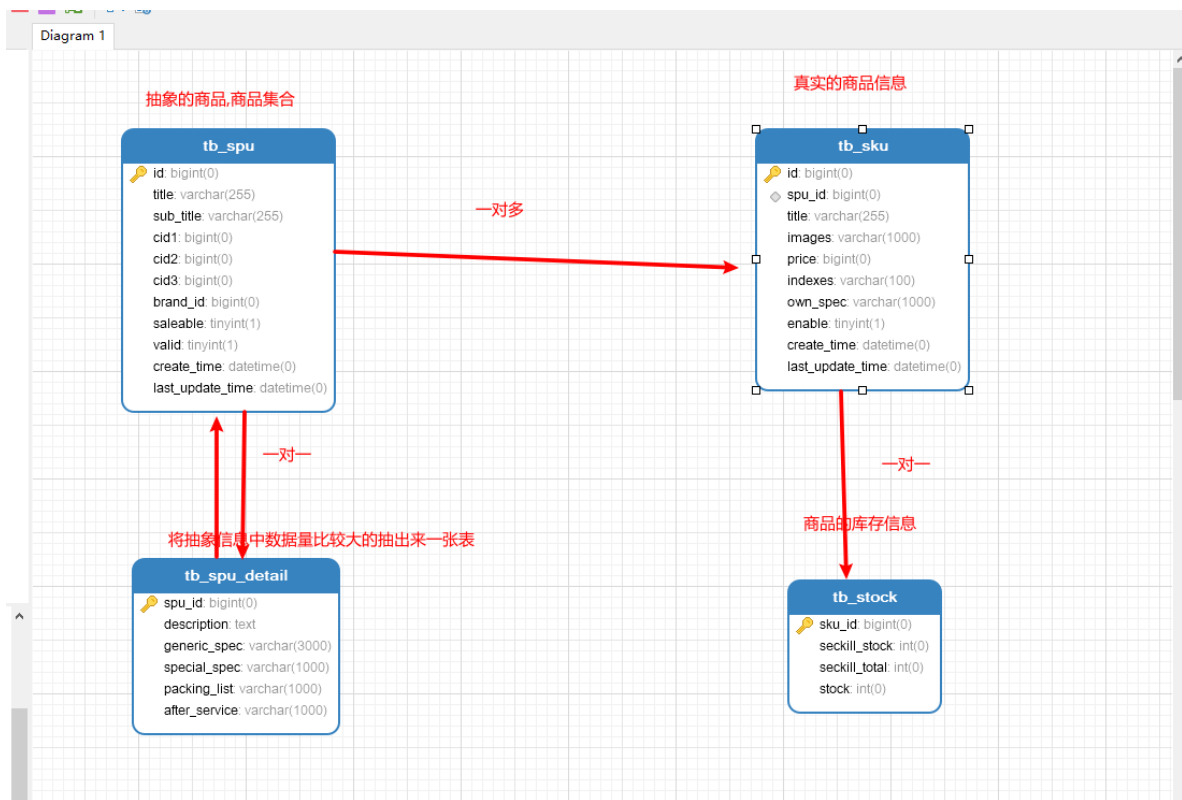


indexes这个字段其实就是特有属性的下标

3.2.2 tb_stock

```
CREATE TABLE `tb_stock` (
  `sku_id` bigint(0) NOT NULL COMMENT '库存对应的商品sku id',
  `seckill_stock` int(0) NULL DEFAULT 0 COMMENT '可秒杀库存',
  `seckill_total` int(0) NULL DEFAULT 0 COMMENT '秒杀总数量',
  `stock` int(0) NOT NULL COMMENT '库存数量',
  PRIMARY KEY (`sku_id`) USING BTREE
) ENGINE = InnoDB CHARACTER SET = utf8 COLLATE = utf8_general_ci COMMENT = '库存表, 代表库存, 秒杀库存等信息' ROW_FORMAT = Dynamic;
```





4 商品管理(查询)

4.1 vue项目

4.1.1 Goods.vue

line:8

```

<v-btn-toggle mandatory v-model="search.saleable">
  <!--将此处改为number类型的值比较好处理-->
  <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>
</v-btn-toggle>
  
```

line: 204

```

getDataFromApi() {
  this.loading = true;
  //第三个参数可以设置axios参数
  //timeout:设置超时时间
  this.$http.get('goods/getSpuInfo',{
    params:{
      page: this.pagination.page,
      rows: this.pagination.rowsPerPage,
    }
  })
}
  
```

```

        sort: this.pagination.sortBy,
        order: this.pagination.descending,
        saleable: this.search.saleable,
        title: this.search.key
    }
}, {timeout: 60000}).then(resp => {
    this.items = resp.data.data;
    this.totalItems = resp.data.message - 0; // 转换成number类型
    this.loading = false;
}).catch(error => console.log(error))
}

```

4.2 mingrui-shop-service-api-xxx

4.2.1 entity包下新建SpuEntity

```

import lombok.Data;

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

/**
 * @ClassName SpuEntity
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/25
 * @Version v1.0
 */
@Table(name = "tb_spu")
@Data
public class SpuEntity {

    @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;
}

```

```
}
```

4.2.2 dto包下新建SpuDTO

```
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;
import java.util.Date;

/**
 * @ClassName SpuDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/25
 * @Version v1.0
 */
@ApiModel(value = "spu数据传输DTO")
@Data
public class SpuDTO 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 = "标题")
    @NotEmpty(message = "标题不能为空", groups = {MingruiOperation.Add.class})
    private String title;

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

    @ApiModelProperty(value = "1级类目id", example = "1")
    @NotNull(message = "1级类目id不能为空", groups = {MingruiOperation.Add.class})
    private Integer cid1;

    @ApiModelProperty(value = "2级类目id", example = "1")
    @NotNull(message = "2级类目id不能为空", groups = {MingruiOperation.Add.class})
    private Integer cid2;

    @ApiModelProperty(value = "3级类目id", example = "1")
    @NotNull(message = "3级类目id不能为空", groups = {MingruiOperation.Add.class})
    private Integer cid3;

    @ApiModelProperty(value = "商品所属品牌id", example = "1")
    @NotNull(message = "商品所属品牌id不能为空", groups = {MingruiOperation.Add.class})
    private Integer brandId;

    //不需要验证,新增时直接设置默认值
    @ApiModelProperty(value = "是否上架, 0下架, 1上架", example = "1")
```

```

private Integer saleable;

//不需要验证,新增时直接设置默认值
@ApiModelProperty(value = "是否有效, 0已删除, 1有效", example = "1")
private Integer valid;

//不需要验证,新增时直接设置默认值
@ApiModelProperty(value = "添加时间")
private Date createTime;

//不需要验证,新增时直接设置默认值,修改时使用java代码赋值
@ApiModelProperty(value = "最后修改时间")
private Date lastUpdateTime;

private String brandName;

private String categoryName;
}

```

4.2.3 service包下新建GoodsService

```

import com.baidu.shop.base.Result;
import com.baidu.shop.dto.SpuDTO;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.GetMapping;

import java.util.Map;

/**
 * @ClassName GoodsService
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/25
 * @Version v1.0
 */
@Api(tags = "商品接口")
public interface GoodsService {

    @ApiOperation(value = "获取spu信息")
    @GetMapping(value = "goods/getSpuInfo")
    public Result<List<SpuDTO>> getSpuInfo(SpuDTO spuDTO);

}

```

4.3 mingrui-shop-service-xxx

4.3.1 mapper包下新建SpuMapper

```

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

/**
 * @ClassName SpuMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/25
 * @Version V1.0
 */
public interface SpuMapper extends Mapper<SpuEntity> {
}

```

4.3.2 修改CategoryMapper

```

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

import java.util.List;

/**
 * @ClassName CategoryMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/17
 * @Version V1.0
 */
//可以通过id集合查询
public interface CategoryMapper extends Mapper<CategoryEntity>,
    SelectByIdListMapper<CategoryEntity,Integer> {
    @Select(value = "select c.id,c.name from tb_category c where c.id in(select t.category_id from tb_category_brand t where t.brand_id=#{brandId})")
    List<CategoryEntity> getCatesByBrand(Integer brandId);
}

```

4.3.3 service.impl包下新建GoodsServiceImpl

```

import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.dto.SpuDTO;
import com.baidu.shop.entity.BrandEntity;
import com.baidu.shop.entity.SpuEntity;
import com.baidu.shop.mapper.BrandMapper;
import com.baidu.shop.mapper.CategoryMapper;
import com.baidu.shop.mapper.SpuMapper;
import com.baidu.shop.service.GoodsService;
import com.baidu.shop.utils.BaiduBeanUtil;
import com.baidu.shop.utils.ObjectUtil;
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.*;
import java.util.stream.Collectors;

/**
 * @ClassName GoodsServiceImpl
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/25
 * @Version v1.0
 */
@RestController
public class GoodsServiceImpl extends BaseApiService implements GoodsService {

    @Resource
    private SpuMapper spuMapper;

    @Resource
    private BrandMapper brandMapper;

    @Resource
    private CategoryMapper categoryMapper;

    @Override
    public Result<List<SpuDTO>> getSpuInfo(SpuDTO spuDTO) {

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

        Example example = new Example(SpuEntity.class);
        Example.Criteria criteria = example.createCriteria();

        if (ObjectUtil.isNotNull(spuDTO)) {
            //按标题模糊匹配
            if(!StringUtils.isEmpty(spuDTO.getTitle()))
                criteria.andLike("title","%" + spuDTO.getTitle() + "%");
            //如果值为2的话不进行拼接查询,默认查询所有
            if (ObjectUtil.isNotNull(spuDTO.getSaleable()) &&
                spuDTO.getSaleable() != 2)
                criteria.andEqualTo("saleable",spuDTO.getSaleable());
            //排序
            if (!StringUtils.isEmpty(spuDTO.getSort()))
                example.setOrderByClause(spuDTO.getOrderByClause());
        }

        List<SpuEntity> list = spuMapper.selectByExample(example);

        List<SpuDTO> dtos = list.stream().map(spu -> {

            //通过品牌id得到品牌名称
            //此处不需要验证参数为空,因为数据库中这个字段是必填的
            BrandEntity brandEntity =
            brandMapper.selectByPrimaryKey(spu.getBrandId());
            //获取分类id-->转成数组-->通过idList查询出数据,得到数据集合-->调用stream函数
            //-->调用map函数返回一个新的函数

```



```

        // -->调用Stream.collect转换集合-->调用Collectors.joining("/")将集合转换成/拼接的字符串
        //select group_concat(name separator '/') as cateroryName from
        tb_category where id in(1,2,3)
        String categoryNames =
        categoryMapper.selectByIdList(Arrays.asList(spu.getCid1(), spu.getCid2(),
        spu.getCid3()))
            .stream()
            .map(category -> category.getName())
            .collect(Collectors.joining("/"));

        //得到id集合
        //List<Integer> idArr = Arrays.asList(spu.getCid1(), spu.getCid2(),
        spu.getCid3());

        //通过Id集合查询数据
        //List<CategoryEntity> categoryEntities =
        categoryMapper.selectByIdList(idArr);

        // String categoryNames = categoryEntities.stream().map(category ->
        category.getName()).collect(joining("/"));

        //map函数返回一个新的数组<String>category.getName()是string类型
        //Stream.collect()集合的转换功能
        //Collectors.joining("/")将集合转换成/拼接的字符串
        /*String categoryNames = categoryEntities.stream().map(category -> {
            return category.getName();
        }).collect(Collectors.joining("/"));*/

        SpuDTO dto = BaiduBeanUtil.copyProperties(spu, SpuDTO.class);
        dto.setBrandName(brandEntity.getName());

        dto.setCategoryName(categoryNames);

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

    PageInfo<SpuEntity> pageInfo = new PageInfo<>(list);

    //要返回的是dto的数据,但是pageinfo中没有总条数
    long total = pageInfo.getTotal();
    //借用一下message属性
    return this.setResult(HttpStatus.OK, total + "", dtos);
}
}

```

5 商品管理(新增[数据准备])

5.1 vue项目

5.1.1 GoodsFrom.vue

5.1.1.1 line:20

```
<!--商品分类-->
<v-cascader
  url="/category/list"
  required
  showAllLevels
  v-model="goods.categories"
  label="请选择商品分类"/>
```

5.1.1.2 line:277

```
handler(val) {
  // 判断商品分类是否存在，存在才查询
  if (val && val.length > 0) {
    // 根据分类查询品牌
    this.$http
      .get("/brand/getBrandByCategory",{
        params:{
          cid:this.goods.categories[2].id
        }
      })
      .then((resp) => {
        this.brandOptions = resp.data.data;
      });
    // 根据分类查询规格参数
    this.$http
      .get("/specparam/getSpecParamInfo",{
        params:{
          cid:this.goods.categories[2].id
        }
      })
      .then(( resp ) => {
        let specs = [];
        let template = [];
        if (this.isEdit){
          specs = JSON.parse(this.goods.spuDetail.genericSpec);
          template = JSON.parse(this.goods.spuDetail.specialSpec);
        }
        // 对特有规格进行筛选
        const arr1 = [];
        const arr2 = [];
        resp.data.data.forEach(({id, name, generic, numeric, unit }) => {
          if(generic){
            const o = { id, name, numeric, unit};
            if(this.isEdit){
              o.v = specs[id];
            }
            arr1.push(o)
          }else{
            const o = {id, name, options:[]};
            if(this.isEdit){
              o.options = template[id];
            }
            arr2.push(o)
          }
        });
        this.specs = arr1;// 通用规格
```

```

        this.specialSpecs = arr2;// 特有规格
    });
}
}

```

5.1.1.3 line:57

```

<v-stepper-content step="2">
  <v-editor v-model="goods.spuDetail.description" upload-url="/upload"/>
</v-stepper-content>

```

5.1.1.4 line:206

```

images: images ? images.map(i => i.data).join(",") : '', // 图片

```

5.1.2 Editor.vue

5.1.3.1 line:93

```

    this.$http.post(this.uploadUrl, data)
      .then(resp => {
        //修改图片回显的路径
        if (resp.data.data) {
          this.editor.insertEmbed(this.editor.getSelection().index, 'image',
            resp.data.data)
        }
      })

```

5.2 mingrui-shop-service-api-xxx

5.2.1 BrandService

```

@ApiOperation(value="通过分类id获取品牌")
@GetMapping(value = "brand/getBrandByCategory")
Result<List<BrandEntity>> getBrandByCategory(Integer cid);

```

5.3 mingrui-shop-service-xxx

5.3.1 BrandMapper

```

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

import java.util.List;

/**
 * @ClassName BrandMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/18
 * @Version V1.0

```

```

/**/
public interface BrandMapper extends Mapper<BrandEntity> {
    //注意子查询和关联查询都可以实现,但是推荐使用子查询
    //select * from tb_brand where id in(select brand_id from tb_category_brand
    where category_id = 76)
    @Select(value = "select b.* from tb_brand b,tb_category_brand cb where b.id
    = cb.brand_id and cb.category_id=#{cid}")
    List<BrandEntity> getBrandByCategory(Integer cid);
}

```

5.3.1 BrandServiceImpl

```

@Override
public Result<BrandEntity> getBrandByCategory(Integer cid) {

    if(ObjectUtil.isNotNull(cid)){

        List<BrandEntity> list = brandMapper.getBrandByCategory(cid);

        return this.setResultSuccess(list);
    }

    return null;
}

```

5.3.2 SpecificationServiceImpl

```

@Override
public Result<List<SpecParamEntity>> getSpecParamInfo(SpecParamDTO
specParamDTO) {

    //自定义异常

    //if(ObjectUtil.isNull(specParamDTO.getGroupId())) return
    this.setResultError("规格组id不能为空");
    Example example = new Example(SpecParamEntity.class);
    Example.Criteria criteria = example.createCriteria();

    if(ObjectUtil.isNotNull(specParamDTO.getGroupId())){
        criteria.andEqualTo("groupId", specParamDTO.getGroupId());
    }

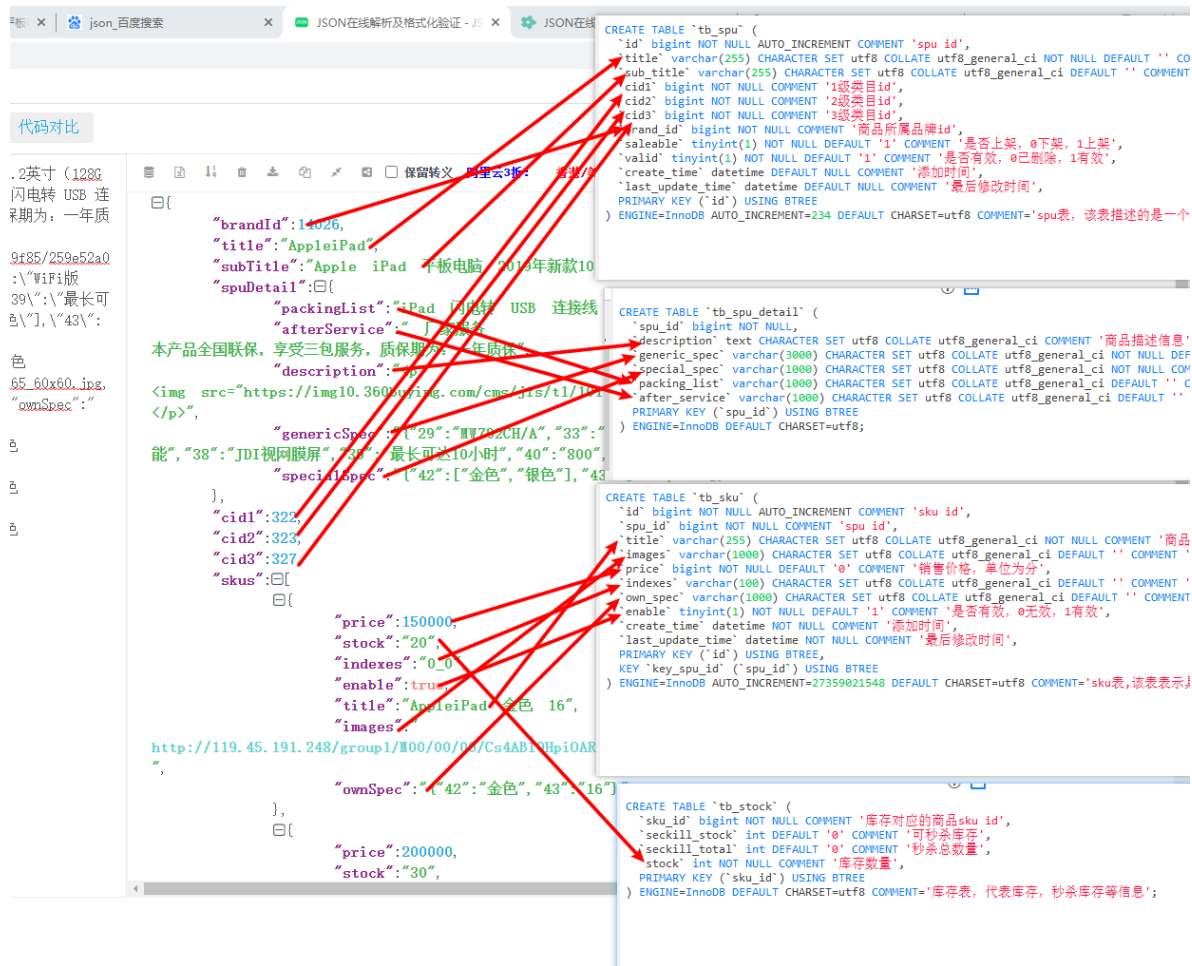
    if(ObjectUtil.isNotNull(specParamDTO.getCid())){
        criteria.andEqualTo("cid", specParamDTO.getCid());
    }

    List<SpecParamEntity> list = specParamMapper.selectByExample(example);

    return this.setResultSuccess(list);
}

```

6 商品管理(新增[传值接值])



6.1 mingrui-shop-service-api-xxx

6.1.1 entity包下实体类

6.1.1.1 SpuDetailEntity

```
import Lombok.Data;

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

/**
 * @ClassName SpuDetail
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/27
 * @Version V1.0
 */
@Table(name = "tb_spu_detail")
@Data
public class SpuDetailEntity {

    @Id
    private Integer spuId;

    private String description;

    private String genericSpec;
```

```

        private String specialSpec;

        private String packingList;

        private String afterService;
    }

```

6.1.1.2 SkuEntity

```

import Lombok.Data;

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

/**
 * @ClassName SkuEntity
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/27
 * @Version V1.0
 */
@Table(name = "tb_sku")
@Data
public class SkuEntity {

    @Id//此处必须写long类型,因为现在新增的id已经超过int的范围了
    private Long id;

    private Integer spuId;

    private String title;

    private String images;

    private Integer price;

    private String indexes;

    private String ownSpec;

    private Integer enable;

    private Date createTime;

    private Date lastUpdateTime;
}

```

6.1.1.3 StockEntity

```

import Lombok.Data;

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

```

```

/**
 * @ClassName StockEntity
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/27
 * @Version v1.0
 **/
@Table(name = "tb_stock")
@Data
public class StockEntity {

    @Id
    private Long skuId;

    private Integer seckillStock;

    private Integer seckillTotal;

    private Integer stock;
}

```

6.1.2 dto包下新建dto

6.1.2.1 SpuDetailDTO

```

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

/**
 * @ClassName SpuDetailDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/27
 * @Version v1.0
 **/
@ApiModel(value = "spu大字段数据传输类")
@Data
public class SpuDetailDTO {

    @ApiModelProperty(value = "spu主键",example = "1")
    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;
}

```

6.1.2.2 SkuDTO

```

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

import java.util.Date;

/**
 * @ClassName SkuDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/27
 * @Version v1.0
 */
@ApiModel(value = "SKU属性数据传输类")
@Data
public class SkuDTO {

    @ApiModelProperty(value = "主键", example = "1")
    private Long id;

    @ApiModelProperty(value = "spu主键", example = "1")
    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", example = "1")
    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;

    //注意此处使用boolean值来接,在service中处理一下就可以了
    @ApiModelProperty(value = "是否有效，0无效，1有效", example = "1")
    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")

```



```

        private Integer stock;

    }

```

6.1.2.3 StockDTO

```

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

/**
 * @ClassName StockDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/27
 * @Version V1.0
 */
@ApiModel(value = "库存数据传输类")
@Data
public class StockDTO {

    @ApiModelProperty(value = "sku主键", example = "1")
    private Long skuId;

    @ApiModelProperty(value = "可秒杀库存", example = "1")
    private Integer seckillStock;

    @ApiModelProperty(value = "秒杀总数量", example = "1")
    private Integer seckillTotal;

    @ApiModelProperty(value = "库存数量", example = "1")
    private Integer stock;
}

```

6.1.3 修改SpuDTO

```

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

/**
 * @ClassName SpuDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/25
 * @Version V1.0
 */

```

```

@ApiModel(value = "spu数据传输DTO")
@Data
public class SpuDTO 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 = "标题")
    @NotEmpty(message = "标题不能为空", groups = {MingruiOperation.Add.class})
    private String title;

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

    @ApiModelProperty(value = "1级类目id", example = "1")
    @NotNull(message = "1级类目id不能为空", groups = {MingruiOperation.Add.class})
    private Integer cid1;

    @ApiModelProperty(value = "2级类目id", example = "1")
    @NotNull(message = "2级类目id不能为空", groups = {MingruiOperation.Add.class})
    private Integer cid2;

    @ApiModelProperty(value = "3级类目id", example = "1")
    @NotNull(message = "3级类目id不能为空", groups = {MingruiOperation.Add.class})
    private Integer cid3;

    @ApiModelProperty(value = "商品所属品牌id", example = "1")
    @NotNull(message = "商品所属品牌id不能为空", groups = {MingruiOperation.Add.class})
    private Integer brandId;

    //不需要验证,新增时直接设置默认值
    @ApiModelProperty(value = "是否上架, 0下架, 1上架", example = "1")
    private Integer saleable;

    //不需要验证,新增时直接设置默认值
    @ApiModelProperty(value = "是否有效, 0已删除, 1有效", example = "1")
    private Integer valid;

    //不需要验证,新增时直接设置默认值
    @ApiModelProperty(value = "添加时间")
    private Date createTime;

    //不需要验证,新增时直接设置默认值,修改时使用java代码赋值
    @ApiModelProperty(value = "最后修改时间")
    private Date lastUpdateTime;

    private String brandName;

    private String categoryName;

    @ApiModelProperty(value = "大字段数据")
    private SpuDetailDTO spuDetail;

    @ApiModelProperty(value = "sku属性数据集合")
    private List<SkuDTO> skus;
}

```

6.1.4 GoodsService

```
@ApiOperation(value = "新建商品")
@PostMapping(value = "goods/add")
Result<JSONObject> saveGoods(@RequestBody SpuDTO spuDTO);
```

6.2 GoodsServiceImpl

```
@Override
public Result<JSONObject> saveGoods(SpuDTO spuDTO) {

    System.out.println(spuDTO);
    return this.setResultSuccess();
}
```

能正常接收到所有页面数据即可

7 商品管理(新增[入库])

7.1 vue项目

7.1.1 GoodsForm.vue line:277

```
this.$emit("closeForm");//bug.....
```

7.2 mingrui-shop-service-xxx

7.2.1 mapper包下新建mapper

7.2.1.1 SpuDetailMapper

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

/**
 * @ClassName SpuDetailMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/28
 * @Version v1.0
 */
public interface SpuDetailMapper extends Mapper<SpuDetailEntity> {
}
```

7.2.1.2 SkuMapper

```

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

/**
 * @ClassName SkuMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/28
 * @Version v1.0
 */
public interface SkuMapper extends Mapper<SkuEntity> {
}

```

7.2.1.3 StockMapper

```

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

/**
 * @ClassName StockMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/28
 * @Version v1.0
 */
public interface StockMapper extends Mapper<StockEntity> {
}

```

7.2.2 GoodsServiceImpl

```

@Resource
private CategoryMapper categoryMapper;

@Resource
private SpuDetailMapper spuDetailMapper;

@Resource
private SkuMapper skuMapper;

@Resource
private StockMapper stockMapper;

@Transactional
@Override
public Result<JSONObject> saveGoods(SpuDTO spuDTO) {

    System.out.println(spuDTO);
    //新增spu
    SpuEntity spuEntity = BaiduBeanUtil.copyProperties(spuDTO,
SpuEntity.class);
    spuEntity.setSaleable(1);
    spuEntity.setValid(1);
    final Date date = new Date();//保持两个时间一致
    spuEntity.setCreateTime(date);
}

```

```

        spuEntity.setLastUpdateTime(date);
        spuMapper.insertSelective(spuEntity);

        //新增spuDetail
        SpuDetailEntity spuDetailEntity =
        BaiduBeanUtil.copyProperties(spuDTO.getSpuDetail(), SpuDetailEntity.class);
        spuDetailEntity.setSpuId(spuEntity.getId());
        spuDetailMapper.insertSelective(spuDetailEntity);

        //新增sku
        /*List<SkuEntity> skuEntities = spuDTO.getSkus().stream().map(sku -> {

            SkuEntity skuEntity = BaiduBeanUtil.copyProperties(sku,
            SkuEntity.class);
            skuEntity.setSpuId(spuEntity.getId());
            skuEntity.setCreateTime(date);
            skuEntity.setLastUpdateTime(date);
            return skuEntity;
        }).collect(Collectors.toList());
        //注意此处不能使用批量新增,因为库存数据也需要入库.....
        skuMapper.insertList(skuEntities);*/

        spuDTO.getSkus().stream().forEach(skuDto -> {
            SkuEntity skuEntity = BaiduBeanUtil.copyProperties(skuDto,
            SkuEntity.class);
            skuEntity.setSpuId(spuEntity.getId());
            skuEntity.setCreateTime(date);
            skuEntity.setLastUpdateTime(date);
            skuMapper.insertSelective(skuEntity);

            StockEntity stockEntity = new StockEntity();
            stockEntity.setSkuId(skuEntity.getId());
            stockEntity.setStock(skuDto.getStock());
            stockMapper.insertSelective(stockEntity);
        });

        return this.setResultSuccess();
    }

```

8 修改(回显数据)

8.1 vue项目

注意:源码中bug太多.....

将editItem函数中的内容全部删除掉

editItem函数中需要将spuDetail和skus全部查询出来再赋值给this.selectedGoods赋值

demo中的回显只能回显spu的相关信息

spuDetail和sku stock需要我们自己从数据库中查询

8.1.1 Goods.vue

```
},
editItem(item) {
  //这两行代码必须写在上面,下面代码会给oldGoods重新赋值,
  //于组件监控到oldGoods发生变化,并且是修改状态的时候
  //才会走修改的回显流程
  this.isEdit = true; //是修改的状态
  this.show = true; //打开模态框

  let obj = item;
  this.$http.get("/goods/getSpuDetailInfo",{
    params:{
      spuId:item.id
    }
  }).then(resp => {
    obj.categories = [];
    obj.spuDetail = resp.data.data;
    // this.selectedGoods.spuDetail.specTemplate = JSON.parse(r
    // this.selectedGoods.spuDetail.specifications = JSON.parse(r
    obj.spuDetail.specTemplate = resp.data.data.specialSpec;
    obj.spuDetail.specifications = resp.data.data.genericSpec;
    //通过spuId查询skus

    this.$http.get('/goods/getSkusBySpuId',{
      params:{
        spuId:item.id
      }
    }).then(resp => {
      obj.skus = resp.data.data;
      //this.selectedGoods就是oldGoods
      this.selectedGoods = obj;
    }).catch(error => console.log(error));
  });
},
watch: {
  oldGoods: {
    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 = [];
}
```

```
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206

this.show = true; //打开模态框

let obj = item;
this.$http.get("/goods/getSpuDetailInfo",{
  params:{
    spuId:item.id
  }
}).then(resp => {
  obj.categories = [];
  obj.spuDetail = resp.data.data;
  // this.selectedGoods.spuDetail.specTemplate = JSON.parse(resp
  // this.selectedGoods.spuDetail.specifications = JSON.parse(r
  obj.spuDetail.specTemplate = resp.data.data.specialSpec;
  obj.spuDetail.specifications = resp.data.data.genericSpec;
  //通过spuId查询skus

  this.$http.get('/goods/getSkusBySpuId',{
    params:{
      spuId:item.id
    }
  }).then(resp => {
    obj.skus = resp.data.data;
    //this.selectedGoods就是oldGoods
    this.selectedGoods = obj;
  }).catch(error => console.log(error));
});
},
deleteItem(id) {
  this.$message.confirm('此操作将永久删除该商品, 是否继续?')
},
},
"goods.categories": {
  deep: true,
  handler(val) {}
},
// 判断商品分类是否存在, 存在才查询
if (val && val.length > 0) {
  // 根据分类查询品牌
  this.$http
    .get("/brand/getBrandByCategory",{
      params:{
        categoryId:this.goods.categories[2].id
      }
    })
    .then((resp) => {
      this.brandOptions = resp.data.data;
      //this.brandOptions = data;
    });
  // 根据分类查询规格参数
  this.$http
    .get("/specparam/getSpecParamInfo",{
      params:{
        cid:this.goods.categories[2].id
      }
    })
  })
}
```

需要给分类集合复制为空集合
因为子组件用的是深度监控,
第一次检测到这个值为 未定义,
之后的监控函数就不会再起作用了

image-20210106151526784

```
setTimeout(() => {
  this.goods.categories = [
    { id: val.cid1, name: names[0] },
    { id: val.cid2, name: names[1] },
    { id: val.cid3, name: names[2] }
  ];
},50);
```

延迟回显的意义是因为在做回显之前,必须得先有全部数据
才能回显 参考之前学过的 下拉框回显

```
editItem(item) {
  //弹出模态框,页面bug,理论上不应该这么做
  this.isEdit = true;
  this.show = true;

  //不能直接给this.selectedGoods赋值,在这赋值以后GoodsForm会监控到oldGoods发生变化
  // this.selectedGoods = item;
  // const names = item.categoryName.split("/");
  // this.selectedGoods.categories = [
  //   { id: item.cid1, name: names[0] },
  //   { id: item.cid2, name: names[1] },
  //   { id: item.cid3, name: names[2] },
  // ];
```

```

//列表中传输过来的数据
let obj = item;

// 查询商品详情
this.$http
  .get("/goods/getSpuDetailBydSpu", {
    params: {
      spuId: item.id,
    },
  })
  .then((resp) => {
    //准备分类需要回显的数据
    obj.categories = []; //解决子级组件监控undefiend问题
    //商品详情
    obj.spuDetail = resp.data.data;
    //特殊规格数据
    obj.spuDetail.specTemplate = resp.data.specialSpec;
    //通用规格数据
    obj.spuDetail.specifications = resp.data.genericSpec;
    //查询sku
    this.$http
      .get("goods/getSkuBySpuId", {
        params: {
          spuId: item.id,
        },
      })
      .then((resp) => {

        obj.skus = resp.data.data;
        //需要回显的数据
        this.selectedGoods = obj; //最后再给selectedGoods赋值
      })
      .catch((error) => console.log(error));
  });
},

```

8.1.2 GoodsForm.vue

```

//监控需要回显的数据
oldGoods: {
  deep: true,
  handler(val) {
    //新增,清空数据
    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 { //修改
        //深拷贝?????深拷贝与浅拷贝的区别
        this.goods = Object.deepCopy(val);
        // 先得到分类名称 bug我们获取到的是categoryName
        const names = val.categoryName.split("/");
        // 组织商品分类数据

        setTimeout(() => {
            this.goods.categories = [
                { id: val.cid1, name: names[0] },
                { id: val.cid2, name: names[1] },
                { id: val.cid3, name: names[2] }
            ];
        }, 100);

        // 将skus处理成map
        const skuMap = new Map();
        this.goods.skus.forEach(s => {
            skuMap.set(s.indexes, s);
        });
        this.goods.skus = skuMap;
    }
},
"goods.categories": {
    deep: true,
    handler(val) {
        // 判断商品分类是否存在，存在才查询
        if (val && val.length > 0) {
            // 根据分类查询品牌
            this.$http
                .get("brand/getBrandByCategory",{
                    params:{
                        cid:this.goods.categories[2].id
                    }
                })
                .then((resp) => {
                    this.brandOptions = resp.data.data;
                });
            // 根据分类查询规格参数
            this.$http
                .get("/specparam/getSpecParamInfo",{
                    params:{
                        cid:this.goods.categories[2].id
                    }
                })
                .then(( resp ) => {
                    let specs = [];
                    let template = [];
                    if (this.isEdit){
                        specs = JSON.parse(this.goods.spuDetail.genericSpec);
                        template = JSON.parse(this.goods.spuDetail.specialSpec);
                    }
                    // 对特有规格进行筛选
                    const arr1 = [];
                    const arr2 = [];

```



```

        resp.data.data.forEach(({id, name, generic, numeric, unit }) => {
            if(generic){
                const o = { id, name, numeric, unit};
                if(this.isEdit){
                    o.v = specs[id];
                }
                arr1.push(o)
            }else{
                const o = {id, name, options:[]};
                if(this.isEdit){
                    o.options = template[id];
                }
                arr2.push(o)
            }
        });
        this.specs = arr1;// 通用规格
        this.specialSpecs = arr2;// 特有规格
    });
    }
}
},

```

8.2 mingrui-shop-service-api-xxx

8.2.1 GoodsService

```

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

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

```

8.3 mingrui-shop-service-xxx

8.3.1 SkuMapper

```

import com.baidu.shop.dto.SkuDTO;
import com.baidu.shop.entity.SkuEntity;
import org.apache.ibatis.annotations.Select;
import tk.mybatis.mapper.common.Mapper;

import java.util.List;

/**
 * @ClassName SkuMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/28
 * @Version v1.0
 */

```

```
public interface SkuMapper extends Mapper<SkuEntity> {

    @Select(value = "select k.*,stock from tb_sku k , tb_stock t where k.id = t.sku_id and k.spu_id=#{spuId}")
    List<SkuDTO> selectSkuAndStockBySpuId(Integer spuId);

}
```

8.3.2 GoodsServiceImpl

```
@Override
public Result<SpuDetailEntity> getSpuDetailBydSpu(Integer spuId) {

    SpuDetailEntity spuDetailEntity =
    spuDetailMapper.selectByPrimaryKey(spuId);
    return this.setResultSuccess(spuDetailEntity);
}

@Override
public Result<List<SkuDTO>> getSkuBySpuId(Integer spuId) {

    List<SkuDTO> list = skuMapper.selectSkuAndStockBySpuId(spuId);

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

9 修改(后台)

9.1 vue项目

9.1.1 GoodsForm.vue

```
this.$http({
  method: this.isEdit ? "put" : "post",
  url: "/goods/save",
  data: goodsParams
})
```

9.2 mingrui-shop-service-api-xxx

9.2.1 GoodsService

```
@ApiOperation(value = "新建商品")
@PostMapping(value = "goods/save")
Result<JSONObject> saveGoods(@RequestBody SpuDTO spuDTO);

@ApiOperation(value = "修改商品")
@PutMapping(value = "goods/save")
Result<JSONObject> editGoods(@RequestBody SpuDTO spuDTO);
```

9.3 mingrui-shop-service-xxx

9.3.1 修改的步骤

1. 修改spu
2. 修改spuDetail
3. 通过spuld查询出来将要被删除的sku
4. 获取所有将要被删除skuld(如果直接删除的话stock没有办法删除)
5. 批量删除sku
6. 批量删除stock
7. 将新的数据新增到数据库

9.3.2 修改SkuMapper和StockMapper

9.3.2.1 SkuMapper

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

import java.util.List;

/**
 * @ClassName SkuMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/28
 * @Version V1.0
 */
public interface SkuMapper extends Mapper<SkuEntity>,
DeleteByIdListMapper<SkuEntity, Long> {

    @Select(value = "select k.*,stock from tb_sku k , tb_stock t where k.id = t.sku_id and k.spu_id=#{spuId}")
    List<SkuDTO> selectSkuAndStockBySpuId(Integer spuId);
}
```

9.3.2.2 StockMapper

```
import com.baidu.shop.entity.StockEntity;
import tk.mybatis.mapper.additional.idlist.DeleteByIdListMapper;
import tk.mybatis.mapper.common.Mapper;

/**
 * @ClassName StockMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/28
 * @Version V1.0
 */
public interface StockMapper extends Mapper<StockEntity>,
DeleteByIdListMapper<StockEntity, Long> {
}
```

9.3.3 GoodsServiceImpl

```
@Override
public Result<JSONObject> editGoods(SpuDTO spuDTO) {
    System.out.println(spuDTO);
    //修改spu
    Date date = new Date();
    SpuEntity spuEntity = BaiduBeanUtil.copyProperties(spuDTO,
SpuEntity.class);
    spuEntity.setLastUpdateTime(date);
    spuMapper.updateByPrimaryKeySelective(spuEntity);

    //修改spuDetail

    spuDetailMapper.updateByPrimaryKeySelective(BaiduBeanUtil.copyProperties(spuDTO
.getSpuDetail(), SpuDetailEntity.class));

    //修改sku
    //先通过spuid删除sku
    //然后新增数据
    Example example = new Example(SkuEntity.class);
    example.createCriteria().andEqualTo("spuId", spuDTO.getId());
    List<SkuEntity> skuEntities = skuMapper.selectByExample(example);
    List<Long> skuIdArr = skuEntities.stream().map(sku ->
sku.getId()).collect(Collectors.toList());

    skuMapper.deleteByIdList(skuIdArr);

    //修改stock
    //删除stock
    //但是sku在上面已经被删除掉了
    //所以应该先查询出被删除的skuid
    //新增stock
    stockMapper.deleteByIdList(skuIdArr);
    List<SkuDTO> skus = spuDTO.getSkus();
    //将新数据新增到数据库
    this.saveSkusAndStocks(spuDTO.getSkus(), spuDTO.getId(), date);
    /* spuDTO.getSkus().stream().forEach(skuDto -> {
        SkuEntity skuEntity = BaiduBeanUtil.copyProperties(skuDto,
SkuEntity.class);
        skuEntity.setSpuId(spuDTO.getId());
        skuEntity.setCreateTime(date);
        skuEntity.setLastUpdateTime(date);
        skuMapper.insertSelective(skuEntity);

        StockEntity stockEntity = new StockEntity();
        stockEntity.setSkuId(skuEntity.getId());
        stockEntity.setStock(skuDto.getStock());
        stockMapper.insertSelective(stockEntity);
    });*/

    return this.setResultSuccess();
}

//与新增的部分代码重复, 所以将重复代码抽取出来, 记得修改新增方法!!!
private void saveSkusAndStocks(List<SkuDTO> skus, Integer spuId, Date date){
```

```

        skus.stream().forEach(skuDto -> {
            SkuEntity skuEntity = BaiduBeanUtil.copyProperties(skuDto,
            SkuEntity.class);
            skuEntity.setSpuId(spuId);
            skuEntity.setCreateTime(date);
            skuEntity.setLastUpdateTime(date);
            skuMapper.insertSelective(skuEntity);

            StockEntity stockEntity = new StockEntity();
            stockEntity.setSkuId(skuEntity.getId());
            stockEntity.setStock(skuDto.getStock());
            stockMapper.insertSelective(stockEntity);
        });
    }
}

```

9.4 导入图片到服务器

之前我们已经在数据库导入了数百条品牌及商品数据，不过这些数据中所需要的图片是无法访问的，需要我们把图片放到服务器对应的nginx 图片服务器。

新建目录static

```
mkdir static
```

```

[root@VM-0-6-centos shenyaqi]# pwd
/shenyaqi
[root@VM-0-6-centos shenyaqi]# ls
erlang es fastDFS java mysql rabbitmq spring-boot spring-cloud static
[root@VM-0-6-centos shenyaqi]#

```

```
cd static #进入目录
```

上传图片压缩包

解压压缩包

```
unzip images.zip
```

修改nginx配置文件,带images的请求映射到本地文件夹

```
vi /opt/nginx/conf/nginx.conf
```

```

location /images {
    root    /shenyaqi/static;
}

```

```

server {
    listen      80;
    server_name localhost:8888;

    #charset koi8-r;

    #access_log  logs/host.access.log  main;
    location ~/group([0-9])/ {
        ngx_fastdfs_module;
    }
    location /images {
        root    /shenyaqi/static;
    }
    location / {
        root    html;
        index   index.html index.htm;
    }

    #error_page  404              /404.html;

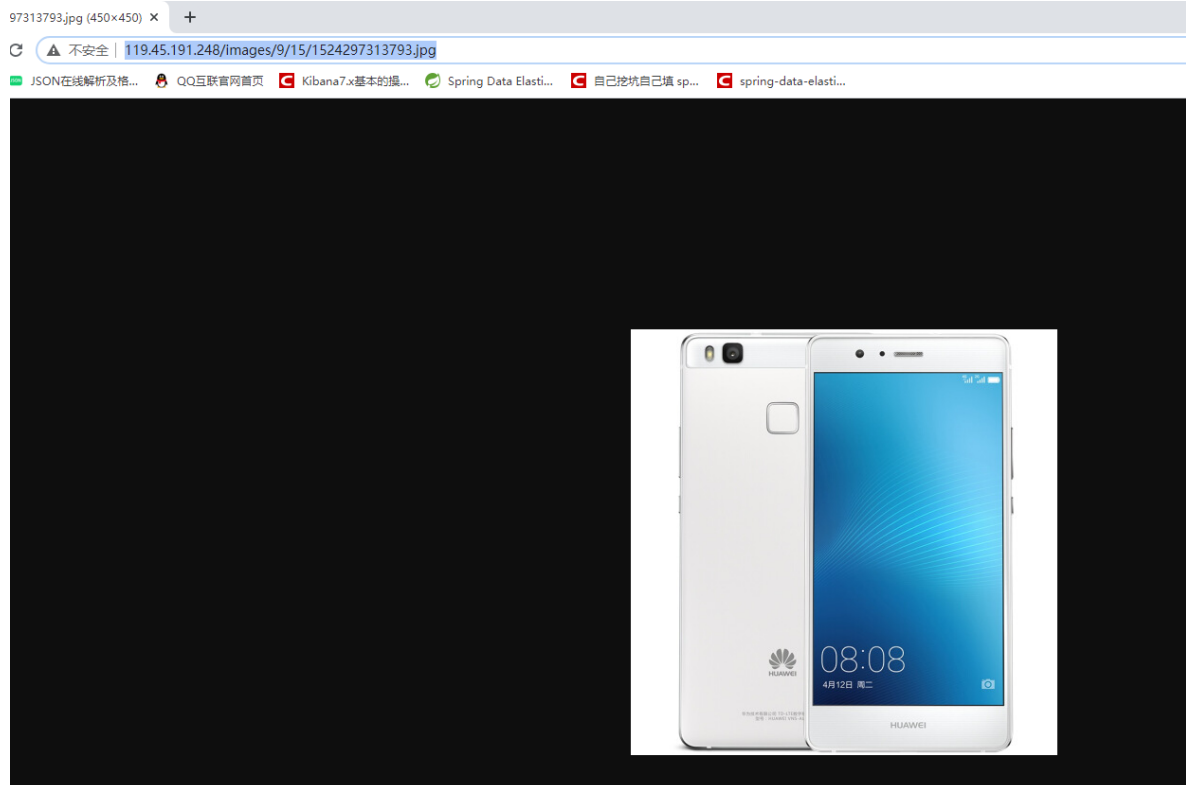
    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root    html;
    }
}

```

重启nginx

```
nginx -s reload
```

浏览器输入<http://ip/images/9/15/1524297313793.jpg>



打开数据库执行

```
update tb_sku set images=REPLACE(images,'http://image.b2c.com','http://自己的ip地址')
```

10 删除

10.1 vue项目

10.1.1 Goods.vue

```
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/del?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");
    });
},
```

10.2 mingrui-shop-service-api-xxx

10.2.1 GoodsService

```
@ApiOperation(value = "删除商品")
@DeleteMapping(value = "goods/del")
Result<JSONObject> delGoods(Integer spuId);
```

10.3 mingrui-shop-service-xxx

10.3.1 GoodsServiceImpl

```
@Override
public Result<JSONObject> delGoods(Integer spuId) {

  //删除spu
  spuMapper.deleteByPrimaryKey(spuId);
  //删除spuDetail
  spuDetailMapper.deleteByPrimaryKey(spuId);

  //查询
  List<Long> skuIdArr = this.getSkuIdArrBySpuId(spuId);
  if(skuIdArr.size() > 0){//尽量加上判断避免全表数据被删除!!!!!!!!!!!!!!
    //删除skus
    skuMapper.deleteByIdList(skuIdArr);
    //删除stock,与修改时的逻辑一样,先查询出所有将要修改skuId然后批量删除
  }
```

```
        stockMapper.deleteByIdList(skuIdArr);
    }

    return this.setResultSuccess();
}

//重复代码抽取出来
private List<Long> getSkuIdArrBySpuId(Integer spuId){
    Example example = new Example(SkuEntity.class);
    example.createCriteria().andEqualTo("spuId", spuId);
    List<SkuEntity> skuEntities = skuMapper.selectByExample(example);
    return skuEntities.stream().map(sku ->
sku.getId()).collect(Collectors.toList());
}
```