第3章 商品发布

学习目标

- SPU与SKU概念理解

```
SPU：某一款商品的公共属性
SKU:某款商品的不同参数对应的商品信息[某个商品]
```

- 新增商品、修改商品

```
增加：增加SPU和SKU
修改：修改SPU和SKU

tb_template    模板表
tb_spec        规格表
tb_para        参数表
tb_spu         某款商品的公共属性
tb_sku         某款商品中某类商品的信息
```

- 商品审核、上架、下架

```
审核：修改审核状态
上架下架：修改上架下架状态
```

- 删除商品

```
逻辑删除：修改了删除状态 0 1
物理删除：真实删除了数据
```

- 找回商品

```
找回商品：一定是属于逻辑删除的商品
```

1 SPU与SKU

目标

- 理解SKU和SPU

路径

- SKU概念讲解
- SPU概念讲解
- 商品表结构设计

讲解

1.1 SPU与SKU概念

SPU = Standard Product Unit （标准产品单位）

- 概念 : SPU 是商品信息聚合的最小单位，是一组可复用、易检索的标准化信息的集合，该集合描述了一个产品的特性。
- 通俗点讲，属性值、特性相同的货品就可以称为一个 SPU

  同款商品的公共属性抽取

  SPU：同款商品的公共属性抽取。

  例如：**华为P30 就是一个 SPU**

SKU=stock keeping unit(库存量单位)

- SKU 即库存进出计量的单位， 可以是以件、盒、托盘等为单位。
- SKU 是物理上不可分割的最小存货单元。在使用时要根据不同业态，不同管理模式来处理。
- 在服装、鞋类商品中使用最多最普遍。

  例如：**华为P30 红色 64G 就是一个 SKU**

  某个库存单位的商品独有属性(某个商品的独有属性)

  SKU：同款商品中，某类或者某个商品的独立特性。

同款商品 华为P30

名字	颜色	价格	内存	网络	OS	分类	厂家	品牌
P30	红色	6000	128G	全网通	鸿蒙	手机	华为	华为
P30 Pro	红色	9000	268G	全网5G	鸿蒙	手机	华为	华为
P30	黑色	4000	64G	联通4G	Android	手机	华为	华为

设计表结构：

方案一：goods

字段名字	类型	是否为空	描述
id	int	No	主键
name	varchar(200)	No	名字
color	varchar(20)	No	颜色
price	bigint	No	价格
neicun	varchar(100)	No	内存
network	varchar(100)	No	网络
os	varchar(100)	No	操作系统
category	varchar(100)	No	分类
factory	varchar(100)	No	厂家
brand	varchar(100)	No	品牌

方案二：

商品款式：P30手机

tb_sku

P30	红色	6000	128G	全网通	鸿蒙	1	No001
P30 Pro	激光蓝	9000	268G	全网5G	鸿蒙	2	No001
P30	黑色	4000	64G	联通4G	Android	3	No001
名字	颜色	价格	内存	网络	OS	id	spuid

tb_spu

id	厂家	品牌	分类
No001	华为	华为	手机

方案二：优点：更节省空间 解决了数据的冗余问题

SPU和SKU

```
SPU:即同款商品的公共属性的抽取称之为SPU。
SKU:即同款商品中不同型号(每个)商品的独有属性。
```

1.2 表结构分析-商品表

tb_spu 表 （SPU表）:公共属性抽取

字段名称	字段含义	字段类型	字段长度	备注
id	主键	VARCHAR		
sn	货号	VARCHAR		
name	SPU名	VARCHAR		
caption	副标题	VARCHAR		
brand_id	品牌ID	INT		
category1_id	一级分类	INT		
category2_id	二级分类	INT		
category3_id	三级分类	INT		
template_id	模板ID	INT		
freight_id	运费模板id	INT		
image	图片	VARCHAR		
images	图片列表	VARCHAR		
sale_service	售后服务	VARCHAR		
introduction	介绍	TEXT		
spec_items	规格列表	VARCHAR		
para_items	参数列表	VARCHAR		
sale_num	销量	INT		
comment_num	评论数	INT		
is_marketable	是否上架	CHAR		
is_enable_spec	是否启用规格	CHAR		
is_delete	是否删除	CHAR		
status	审核状态	CHAR		

tb_sku 表（SKU商品表）:某款商品中某个商品的独有属性

字段名称	字段含义	字段类型	字段长度	备注
id	商品id	BIGINT		
sn	商品条码	VARCHAR		
name	SKU名称	VARCHAR		
price	价格（分）	INT		
num	库存数量	INT		
alert_num	库存预警数量	INT		
image	商品图片	VARCHAR		
images	商品图片列表	VARCHAR		
weight	重量（克）	INT		
create_time	创建时间	DATETIME		
update_time	更新时间	DATETIME		
spu_id	SPUID	BIGINT	(FK)	
category_id	类目ID	INT		
category_name	类目名称	VARCHAR		
brand_name	品牌名称	VARCHAR		
spec	规格	VARCHAR		
sale_num	销量	INT		
comment_num	评论数	INT		
status	商品状态 1-正常，2-下架，3-删除	CHAR		

总结

每一个sku实际上对应就是一个实际的商品

spu对应的是某一类商品的共同的一些内容的集合

2 新增和修改商品

目标

- 实现商品新增和修改

路径

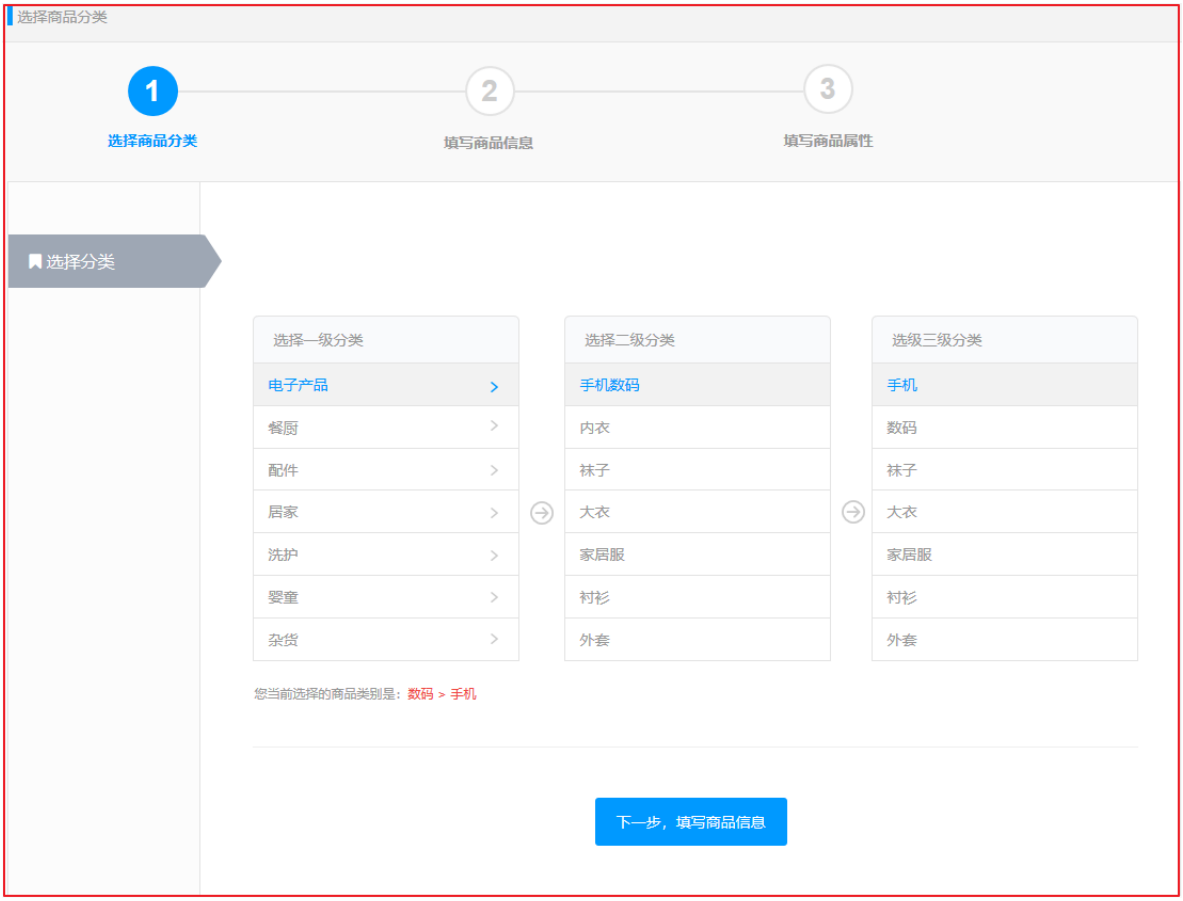
- 商品新增关联信息分析
- 代码生成器(手写dao service controller)
- 商品增加关联数据查询(商品增加过程中,相关商品数据的查询)
- 商品增加
- 商品修改

讲解

2.1 需求分析

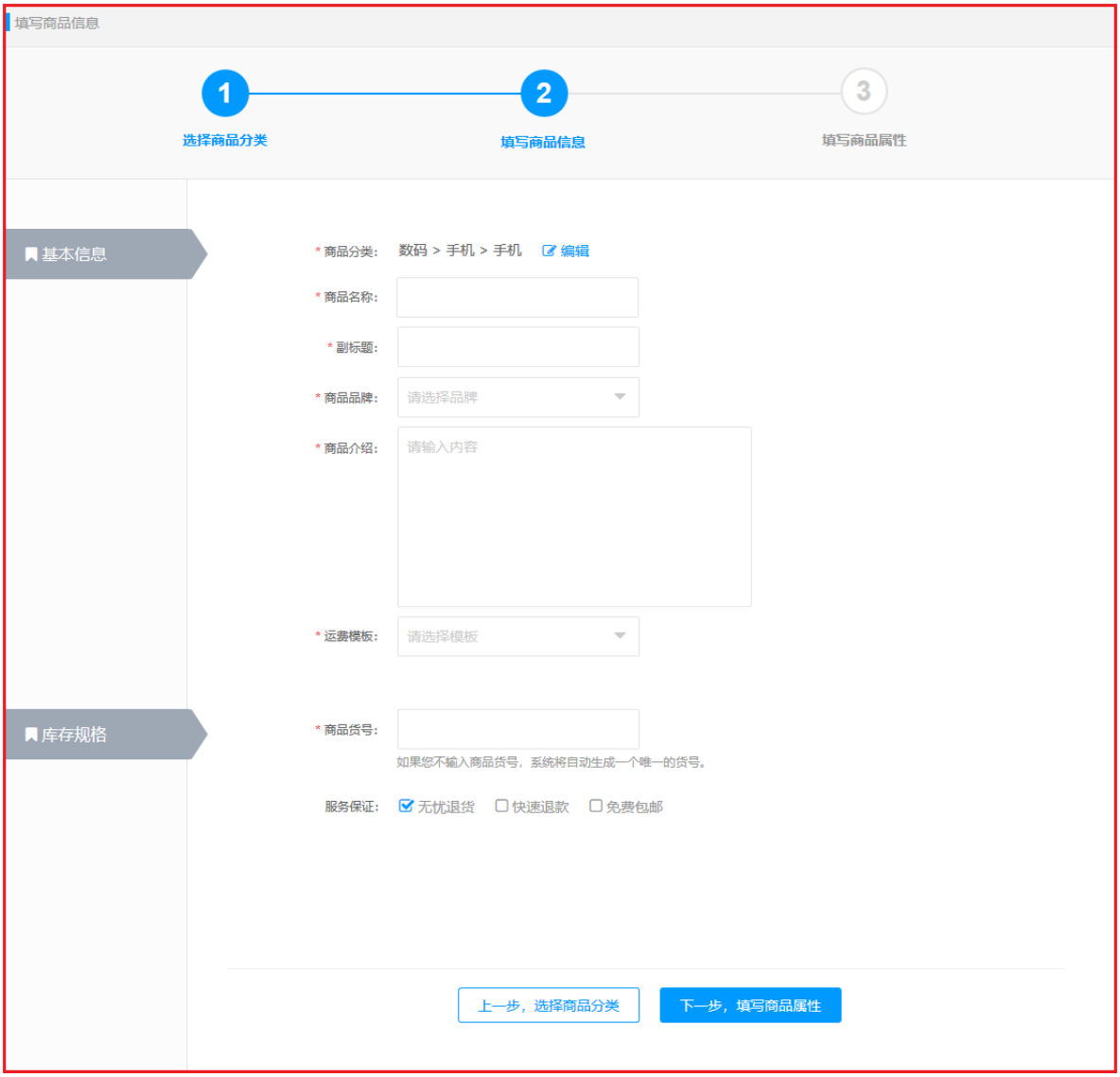
实现商品的新增与修改功能。

(1)第1个步骤，先选择添加的商品所属分类

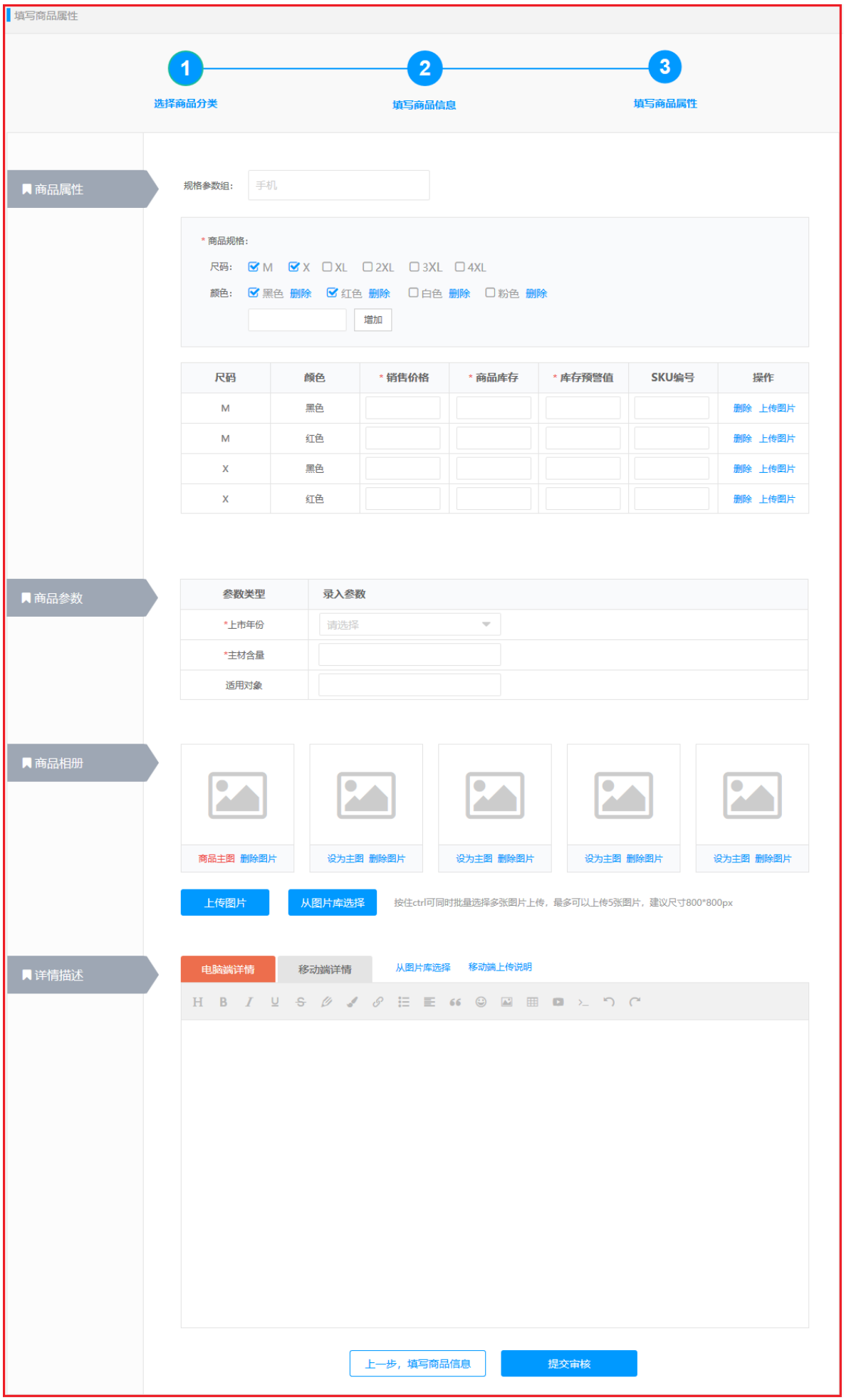


这块在第2天的代码中已经有一个根据父节点ID查询分类信息的方法，参考第2天的4.3.4的findByPrantId方法，首先查询顶级分类，也就是pid=0，然后根据用户选择的分类，将选择的分类作为pid查询子分类。

(2)第2个步骤，填写SPU的信息



(3)第3个步骤，填写SKU信息



先进入选择商品分类 再填写商品的信息 填写商品的属性添加商品。

2.2 实现思路

前端传递给后端的数据格式 是一个spu对象和sku列表组成的对象,如下图:

SPU参数信息

SKU参数信息

```
{
"spu": {
  "name": "这个是商品名称",
  "caption": "这个是副标题",
  "brandId": 12,
  "category1Id": 558,
  "category2Id": 559,
  "category3Id": 560,
  "freightId": 10,
  "image": "http://www.qingcheng.com/image/1.jpg",
  "images": "http://www.qingcheng.com/image/1.jpg,http://www.qingcheng.com/image/2.jpg",
  "introduction": "这个是商品详情，html代码",
  "paraItems": {
    "出厂年份": "2019",
    "赠品": "充电器"
  },
  "saleService": "七天包退,闪电退货",
  "sn": "020102331",
  "specItems": {
    "颜色": [
      "红",
      "绿"
    ],
    "机身内存": [
      "64G",
      "8G"
    ]
  },
  "templateId": 42
},
"skuList": [
  {
    "sn": "10192010292",
    "num": 100,
    "alertNum": 20,
    "price": 900000,
    "spec": {
      "颜色": "红",
      "机身内存": "64G"
    },
    "image": "http://www.qingcheng.com/image/1.jpg",
    "images": "http://www.qingcheng.com/image/1.jpg,http://www.qingcheng.com/image/2.jpg",
    "status": "1",
    "weight": 130
  },
  {
    "sn": "10192010293",
    "num": 100,
    "alertNum": 20,
    "price": 600000,
    "spec": {
      "颜色": "绿",
      "机身内存": "8G"
    },
    "image": "http://www.qingcheng.com/image/1.jpg",
    "images": "http://www.qingcheng.com/image/1.jpg,http://www.qingcheng.com/image/2.jpg",
    "status": "1",
    "weight": 130
  }
]
}
```

上图JSON数据如下:

```
1 {
```

```
  "spu": {
    "name": "这个是商品名称",
    "caption": "这个是副标题",
    "brandId": 12,
    "category1Id": 558,
    "category2Id": 559,
    "category3Id": 560,
    "freightId": 10,
    "image": "http://www.qingcheng.com/image/1.jpg",
    "images":
"http://www.qingcheng.com/image/1.jpg,http://www.qingcheng.com/image/2.jpg",
    "introduction": "这个是商品详情，html代码",
    "paraItems": {
      "出厂年份": "2019",
      "赠品": "充电器"
    },
    "saleService": "七天包退,闪电退货",
    "sn": "020102331",
    "specItems": {
      "颜色": [
        "红",
        "绿"
      ],
      "机身内存": [
        "64G",
        "8G"
      ]
    },
    "templateId": 42
  },
  "skuList": [
    {
      "sn": "10192010292",
      "num": 100,
      "alertNum": 20,
      "price": 900000,
      "spec": {
        "颜色": "红",
        "机身内存": "64G"
      },
      "image": "http://www.qingcheng.com/image/1.jpg",
      "images":
"http://www.qingcheng.com/image/1.jpg,http://www.qingcheng.com/image/2.jpg",
      "status": "1",
      "weight": 130
    },
    {
      "sn": "10192010293",
      "num": 100,
      "alertNum": 20,
      "price": 600000,
      "spec": {
        "颜色": "绿",
        "机身内存": "8G"
      },
      "image": "http://www.qingcheng.com/image/1.jpg",
      "images":
"http://www.qingcheng.com/image/1.jpg,http://www.qingcheng.com/image/2.jpg",
```

```
      "status": "1",
      "weight": 130
    }
  ]
}
```

2.3 代码生成

准备工作：为了更快的实现代码编写，我们可以采用《黑马代码生成器》来批量生成代码，这些代码就已经实现了我们之前的增删改查功能。

《黑马代码生成器》一款由传智播客教育集团研究院开发的基于Freemarker模板引擎的“代码生成神器”。即便是一个工程几百个表，也可以瞬间完成基础代码的构建！用户只需建立数据库表结构，运行main方法就可快速生成可以运行的一整套代码，可以极大地缩短开发周期，降低人力成本。《黑马代码生成器》的诞生主要用于迅速构建生成微服务工程的Pojo、Dao、Service、Controller、Feign各层、并且可以生成swagger API模板等。 用户通过自己开发模板也可以实现生成php、python、C# 、c++、数据库存储过程等其它编程语言的代码。

《黑马代码生成器》目前已经开源 地址：https://github.com/shenkunlin/code-template.git

后期会继续更新。

代码生成器作用：帮我们生成90%以上的基本操作功能的代码,大大缩短我们的项目开发周期。

2.4 代码实现

一会儿会用到ID生成，我们可以使用IdWorker，在启动类GoodsApplication中添加如下代码,用于创建IdWorker，并将IdWorker交给Spring容器，代码如下：

```
/***
 * IdWorker
 * @return
 */
@Bean
public IdWorker idworker(){
    return new IdWorker(0,0);
}
```

2.4.1 查询分类

2.4.1.1 分析

在实现商品增加之前，需要先选择对应的分类，选择分类的时候，首选选择一级分类，然后根据选中的分类，将选中的分类作为查询的父ID，再查询对应的子分类集合，因此我们可以在后台编写一个方法，根据父类ID查询对应的分类集合即可。

2.4.1.2 代码实现

(1)Service层

修改 `com.changgou.goods.service.CategoryService` 添加根据父类ID查询所有子节点，代码如下：

```
/***
 * 根据分类的父ID查询子分类节点集合
 */
List<Category> findByParentId(Integer pid);
```

修改 `com.changgou.goods.service.impl.CategoryServiceImpl` 添加上面的实现，代码如下：

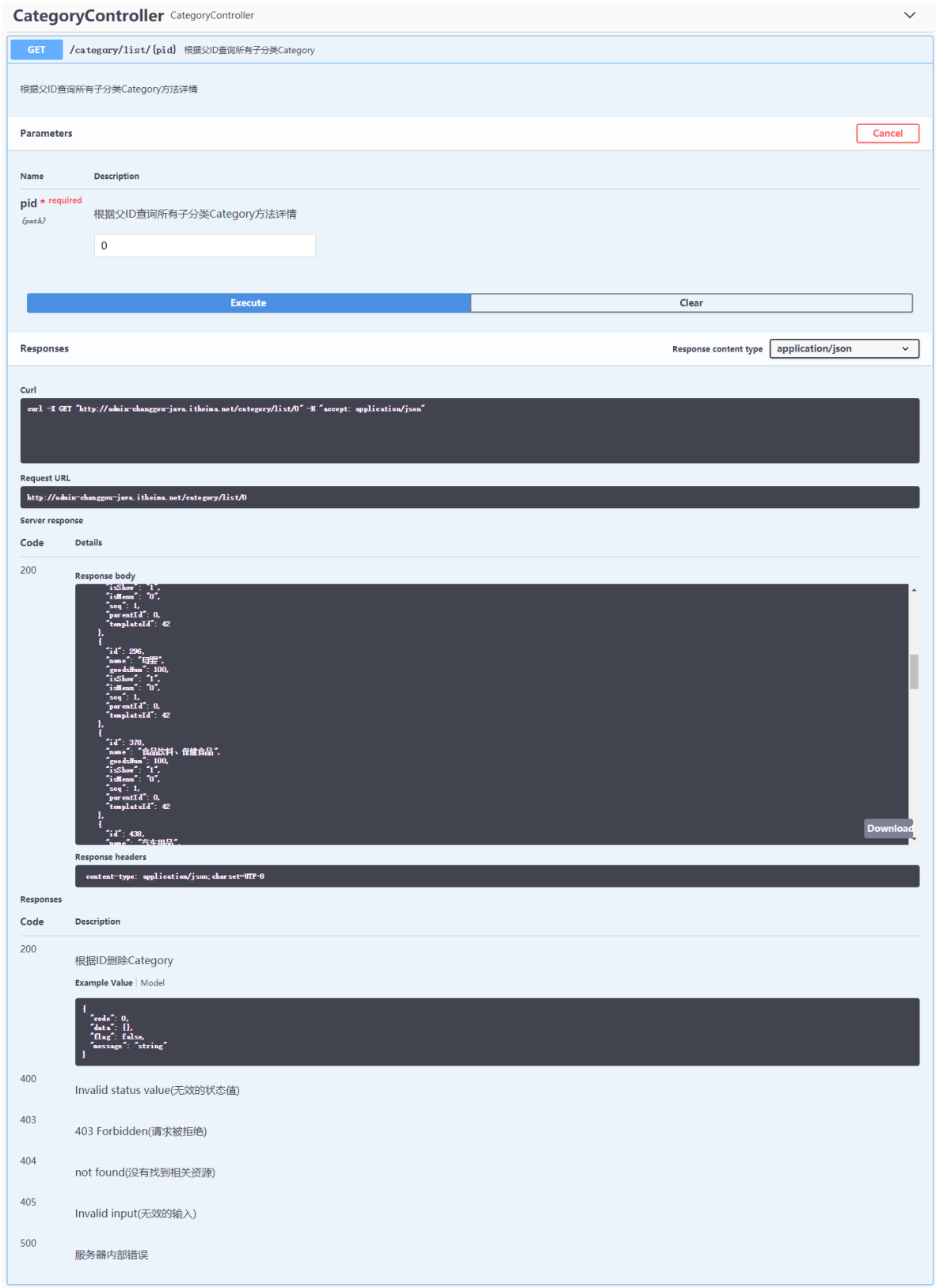
```
/***
 * 根据分类的父节点ID查询所有子节点
 * @param pid
 * @return
 */
@Override
public List<Category> findByParentId(Integer pid) {
    //SELECT * FROM tb_category WHERE parent_id=?
    Category category = new Category();
    category.setParentId(pid);
    return categoryMapper.select(category);
}
```

(2)Controller层

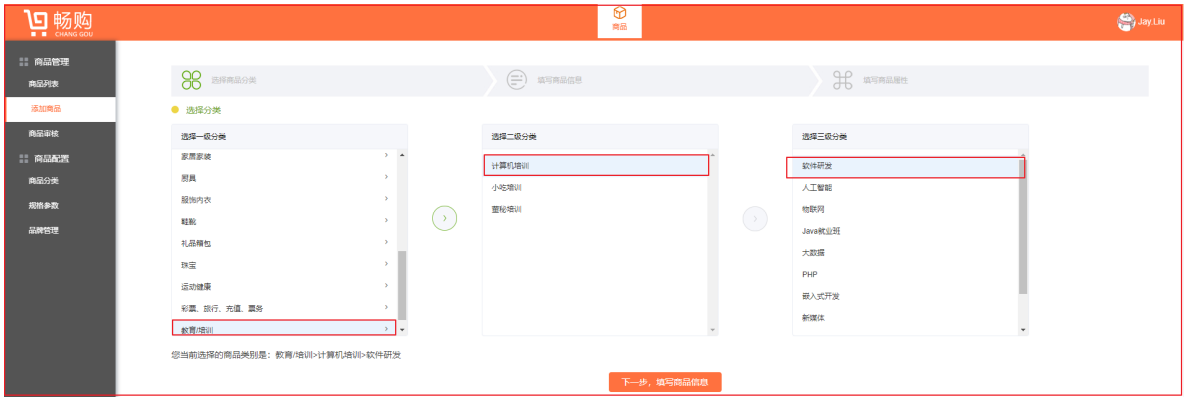
修改 `com.changgou.goods.controller.CategoryController` 添加根据父ID查询所有子类集合，代码如下：

```java
/****
 * 根据节点ID查询所有子节点分类集合
 */
@GetMapping(value = "/list/{pid}")
public Result<List<Category>> findByParentId(@PathVariable(value =
"pid")Integer pid){
    //调用Service实现查询
    List<Category> categories = categoryService.findByParentId(pid);
    return new Result<List<Category>>(true,StatusCode.OK,"查询成
功! ",categories);
}
```

Swagger测试如下:


CategoryController CategoryController
GET /category/list/{pid} 根据父ID查询所有子分类Category
根据父ID查询所有子分类Category方法详情
Parameters
Cancel
Name
Description
pid * required
(path)
根据父ID查询所有子分类Category方法详情
0
Execute
Clear
Responses
Response content type
application/json
Curl
curl -X GET "http://admin-changgou-java.itheima.net/category/list/0" -H "accept: application/json"
Request URL
http://admin-changgou-java.itheima.net/category/list/0
Server response
Code
Details
200
Response body
Download
Response headers
content-type: application/json;charset=UTF-8
Responses
Code
Description
200
根据ID删除Category
Example Value | Model
{
"code": 0,
"data": {},
"flag": false,
"message": "string"
}
400
Invalid status value(无效的状态值)
403
403 Forbidden(请求被拒绝)
404
not found(没有找到相关资源)
405
Invalid input(无效的输入)
500
服务器内部错误

项目测试：



2.4.2 模板查询

2.4.2.1 分析

如上图，当用户选中了分类后，需要根据分类的ID查询出对应的模板数据，并将模板的名字显示在这里，模板表结构如下：

```
CREATE TABLE `tb_template` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'ID',
  `name` varchar(50) DEFAULT NULL COMMENT '模板名称',
  `spec_num` int(11) DEFAULT '0' COMMENT '规格数量',
  `para_num` int(11) DEFAULT '0' COMMENT '参数数量',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=44 DEFAULT CHARSET=utf8;
```

2.4.2.2 代码实现

(1)Service层

修改`com.changgou.goods.service.TemplateService`接口，添加如下方法根据分类ID查询模板：

```
/**
 * 根据分类ID查询模板信息
 * @param id
 * @return
 */
Template findByCategoryId(Integer id);
```

修改`com.changgou.goods.service.impl.TemplateServiceImpl`添加上面方法的实现：

```
@Autowired
private CategoryMapper categoryMapper;

/***
 * 根据分类ID查询模板信息
 * @param id
 * @return
```

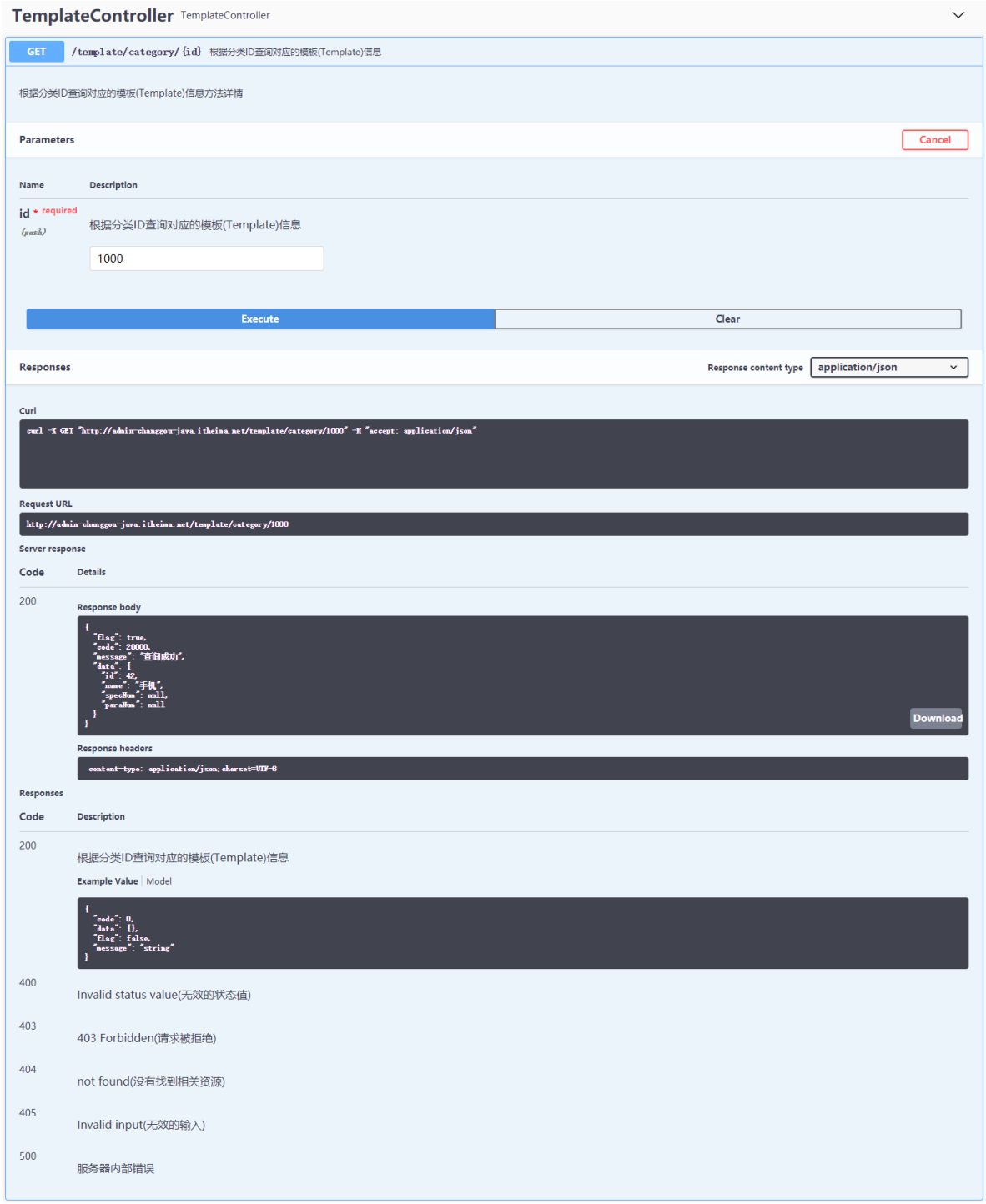
```
 */
@Override
public Template findByCategoryId(Integer id) {
    //查询分类信息
    Category category = categoryMapper.selectByPrimaryKey(id);

    //根据模板Id查询模板信息
    return templateMapper.selectByPrimaryKey(category.getTemplateId());
}
```

(2)Controller层

修改`com.changgou.goods.controller.TemplateController`，添加根据分类ID查询模板数据：

```
/***
 * 根据分类查询模板数据
 * @param id:分类ID
 */
@GetMapping(value = "/category/{id}")
public Result<Template> findByCategoryId(@PathVariable(value = "id")Integer
id){
    //调用Service查询
    Template template = templateService.findByCategoryId(id);
    return new Result<Template>(true, StatusCode.OK,"查询成功",template);
}
```

Swagger测试效果如下：



2.4.3 查询分类品牌数据

2.4.3.1 分析

用户每次选择了分类之后，可以根据用户选择的分类到`tb_category_brand`表中查询指定的品牌集合ID,然后根据品牌集合ID查询对应的品牌集合数据，再将品牌集合数据拿到这里来展示即可实现上述功能。

2.4.3.2 代码实现

(1)Dao实现

修改`com.changgou.goods.dao.BrandMapper`添加根据分类ID查询对应的品牌数据，代码如下：

```
public interface BrandMapper extends Mapper<Brand> {

    /***
     * 查询分类对应的品牌集合
     */
    @Select("SELECT tb.* FROM tb_category_brand tcb,tb_brand tb WHERE tcb.category_id=#{categoryid} AND tb.id=tcb.brand_id")
    List<Brand> findByCategory(Integer categoryid);
}
```

(2)Service层

修改`com.changgou.goods.service.BrandService`，添加根据分类ID查询指定的品牌集合方法，代码如下：

```
/***
 * 根据分类ID查询品牌集合
 * @param categoryid:分类ID
 */
List<Brand> findByCategory(Integer categoryid);
```

修改`com.changgou.goods.service.impl.BrandServiceImpl`添加上面方法的实现，代码如下：

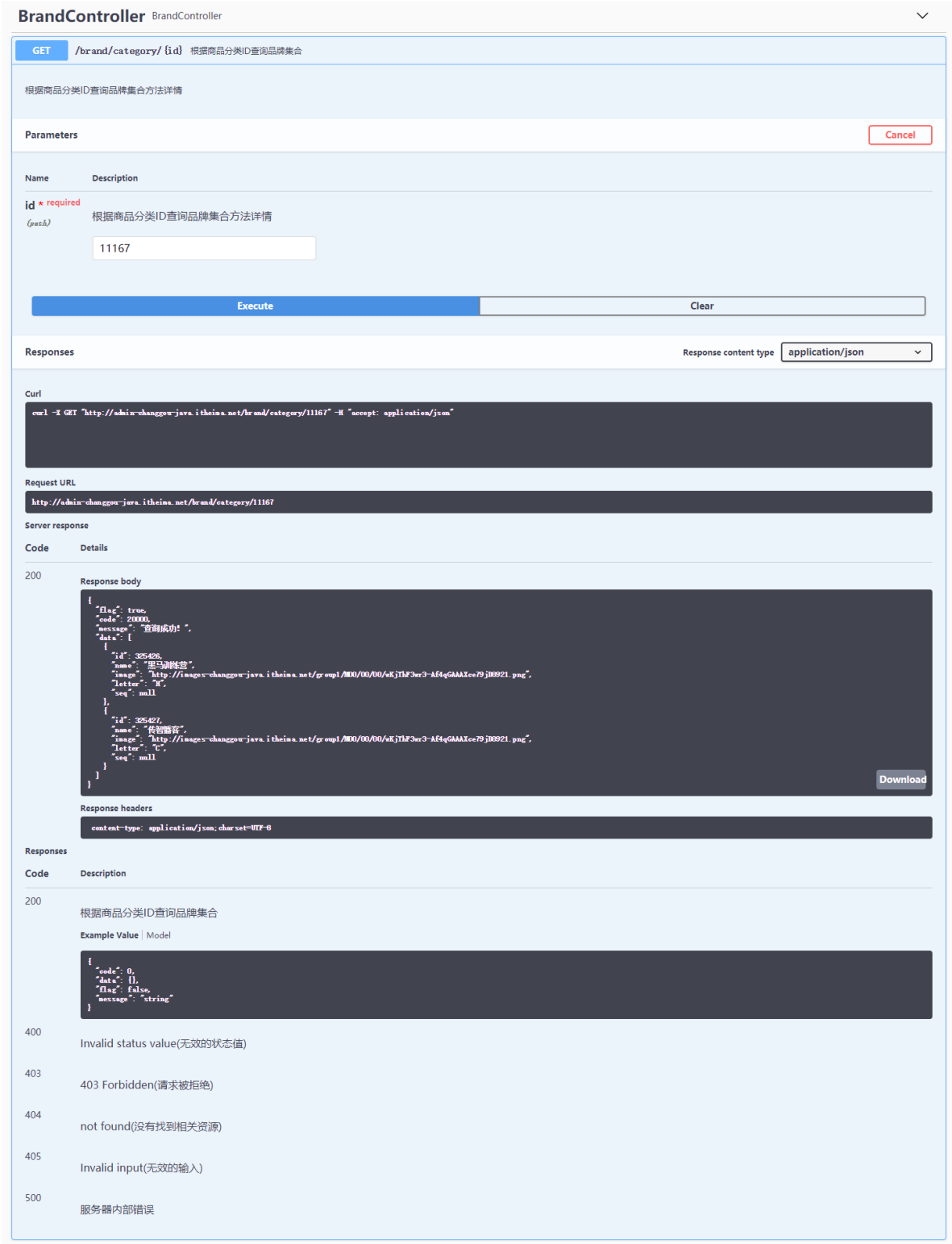
```java
/***
 * 根据分类ID查询品牌集合
 * @param categoryid:分类ID
 * @return
 */
@Override
public List<Brand> findByCategory(Integer categoryid) {
    //1.查询当前分类所对应的所有品牌信息
    //2.根据品牌ID查询对应的品牌集合

    //自己创建DAO实现查询
    return brandMapper.findByCategory(categoryid);
}
```

(3)Controller层

修改,添加根据分类ID查询对应的品牌数据代码如下:

```java
/***
 * 根据分类实现品牌列表查询
 * /brand/category/{id}  分类ID
 */
@GetMapping(value = "/category/{id}")
public Result<List<Brand>> findBrandByCategory(@PathVariable(value =
"id")Integer categoryId){
    //调用Service查询品牌数据
    List<Brand> categoryList = brandService.findByCategory(categoryId);
    return new Result<List<Brand>>(true,StatusCode.OK,"查询成
功! ",categoryList);
}
```

Swagger测试如下:



2.4.4 规格查询

2.4.4.1 分析

用户选择分类后，需要根据所选分类对应的模板ID查询对应的规格，规格表结构如下：

```
CREATE TABLE `tb_spec` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'ID',
  `name` varchar(50) DEFAULT NULL COMMENT '名称',
  `options` varchar(2000) DEFAULT NULL COMMENT '规格选项',
  `seq` int(11) DEFAULT NULL COMMENT '排序',
  `template_id` int(11) DEFAULT NULL COMMENT '模板ID',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=40 DEFAULT CHARSET=utf8;
```

2.4.4.2 代码实现

(1)Service层

修改`com.changgou.goods.service.SpecService`添加根据分类ID查询规格列表，代码如下：

```
/***
 * 根据分类ID查询规格列表
 * @param categoryid
 * @return
 */
List<Spec> findByCategoryId(Integer categoryid);
```

修改`com.changgou.goods.service.impl.SpecServiceImpl`添加上面方法的实现，代码如下：

```
@Autowired
private CategoryMapper categoryMapper;

/***
 * 根据分类ID查询规格列表
 * @param categoryid
 * @return
 */
@Override
public List<Spec> findByCategoryId(Integer categoryid) {
    //查询分类
    Category category = categoryMapper.selectByPrimaryKey(categoryid);
    //根据分类的模板ID查询规格
```

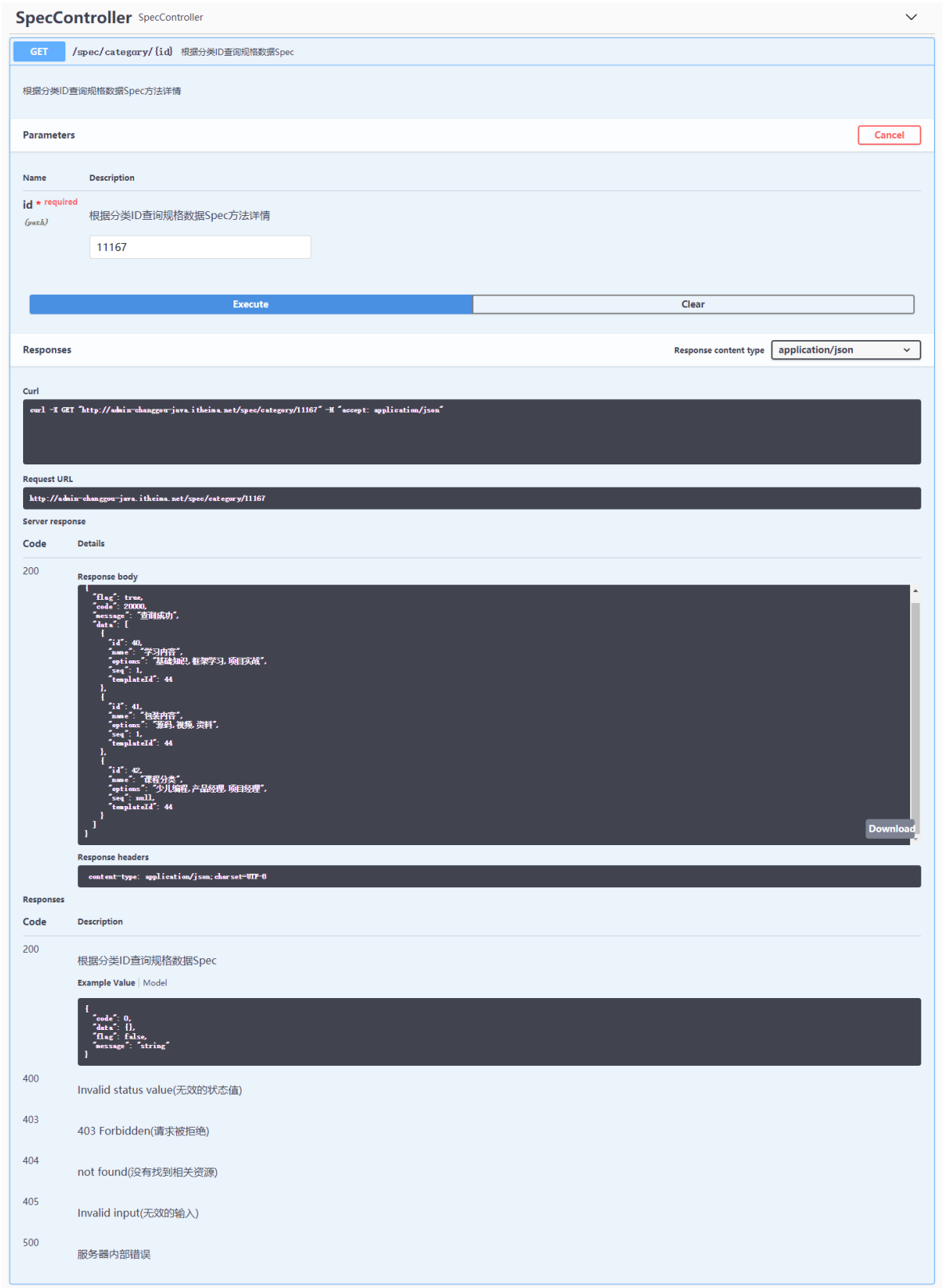
```
    Spec spec = new Spec();
    spec.setTemplateId(category.getTemplateId());
    return specMapper.select(spec);
}
```

(2)Controller层

修改`com.changgou.goods.controller.SpecController`添加根据分类ID查询规格数据，代码如下：

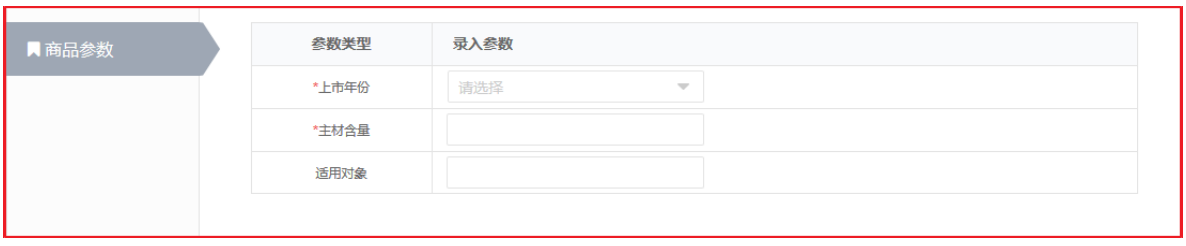
```
/***
 * 根据分类ID查询对应的规格列表
 */
@GetMapping(value = "/category/{id}")
public Result<List<Spec>> findByCategoryId(@PathVariable(value = "id")Integer
 categoryid){
    //调用Service查询
    List<Spec> specs = specService.findByCategoryId(categoryid);
    return new Result<List<Spec>>(true, StatusCode.OK,"查询成功",specs);
}
```

Swagger测试：



2.4.5 参数列表查询

2.4.5.1 分析



当用户选中分类后，需要根据分类的模板ID查询对应的参数列表，参数表结构如下：

```
CREATE TABLE `tb_para` (
  `id` int(11) NOT NULL AUTO_INCREMENT COMMENT 'id',
  `name` varchar(50) DEFAULT NULL COMMENT '名称',
  `options` varchar(2000) DEFAULT NULL COMMENT '选项',
  `seq` int(11) DEFAULT NULL COMMENT '排序',
  `template_id` int(11) DEFAULT NULL COMMENT '模板ID',
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8;
```

2.4.5.2 代码实现

(1)Service层

修改`com.changgou.goods.service.ParaService`添加根据分类ID查询参数列表，代码如下：

```
/***
 * 根据分类ID查询参数列表
 * @param id
 * @return
 */
List<Para> findByCategoryId(Integer id);
```

修改`com.changgou.goods.service.impl.ParaServiceImpl`添加上面方法的实现，代码如下：

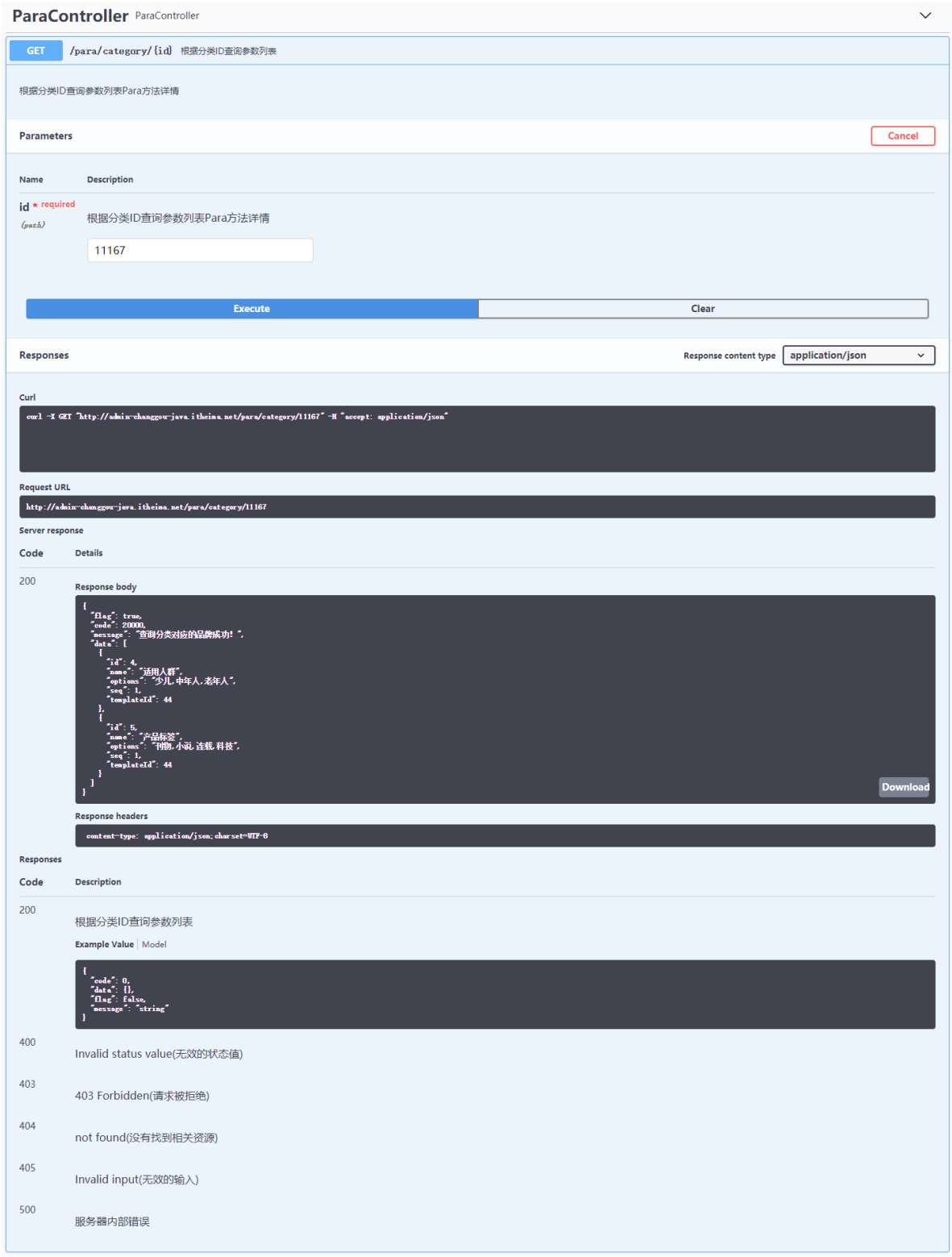
```
@Autowired
private CategoryMapper categoryMapper;

/***
 * 根据分类ID查询参数列表
 * @param id
 * @return
 */
@Override
public List<Para> findByCategoryId(Integer id) {
    //查询分类信息
    Category category = categoryMapper.selectByPrimaryKey(id);
    //根据分类的模板ID查询参数列表
    Para para = new Para();
    para.setTemplateId(category.getTemplateId());
    return paraMapper.select(para);
}
```

(2)Controller层

修改`com.changgou.goods.controller.ParaController`，添加根据分类ID查询参数列表，代码如下：

```java
/**
 * 根据分类ID查询参数列表
 * @param id
 * @return
 */
@GetMapping(value = "/category/{id}")
public Result<List<Para>> getByCategoryId(@PathVariable(value = "id")Integer id){
    //根据分类ID查询对应的参数信息
    List<Para> paras = paraService.findByCategoryId(id);
    Result<List<Para>> result = new Result<List<Para>>(true,StatusCode.OK,"查询分类对应的品牌成功！",paras);
    return result;
}
```

Swagger测试如下：



2.4.6 SPU+SKU保存

2.4.6.1 分析

保存商品数据的时候，需要保存Spu和Sku，一个Spu对应多个Sku，我们可以先构建一个Goods对象，将`Spu`和`List<Sku>`组合到一起,前端将2者数据提交过来，再实现添加操作。

2.4.62 代码实现

(1)Pojo改造

修改changgou-service-goods-api工程创建组合实体类，创建com.changgou.goods.pojo.Goods,代码如下：

```
public class Goods implements Serializable {
    //SPU
    private Spu spu;
    //SKU集合
    private List<Sku> skuList;

    //..get..set..toString
}
```

(2) 业务层

修改com.changgou.goods.service.SpuService接口，添加保存Goods方法，代码如下：

```
/**
 * 保存商品
 * @param goods
 */
void saveGoods(Goods goods);
```

修改com.changgou.goods.service.impl.SpuServiceImpl类，添加保存Goods的方法实现，代码如下：

```
@Autowired
private IdWorker idWorker;

@Autowired
private CategoryMapper categoryMapper;

@Autowired
private BrandMapper brandMapper;

@Autowired
private SkuMapper skuMapper;

/***
 * 保存Goods
 * @param goods
 */
@Override
public void saveGoods(Goods goods) {
    //增加Spu
    Spu spu = goods.getSpu();
    spu.setId("No"+idWorker.nextId());
    spuMapper.insertSelective(spu);

    //增加Sku
    Date date = new Date();
    Category category =
categoryMapper.selectByPrimaryKey(spu.getCategory3Id());
    Brand brand = brandMapper.selectByPrimaryKey(spu.getBrandId());
    //获取Sku集合
```

```
    List<Sku> skus = goods.getSkuList();
    //循环将数据加入到数据库
    for (Sku sku : skus) {
        //构建SKU名称，采用SPU+规格值组装
        if(StringUtils.isEmpty(sku.getSpec())){
            sku.setSpec("{}");
        }
        //获取Spu的名字
        String name = spu.getName();

        //将规格转换成Map
        Map<String,String> specMap = JSON.parseObject(sku.getSpec(),
Map.class);
        //循环组装Sku的名字
        for (Map.Entry<String, String> entry : specMap.entrySet()) {
            name+="  "+entry.getValue();
        }
        sku.setName(name);
        //ID
        sku.setId("No"+idWorker.nextId());
        //SpuId
        sku.setSpuId(spu.getId());
        //创建日期
        sku.setCreateTime(date);
        //修改日期
        sku.setUpdateTime(date);
        //商品分类ID
        sku.setCategoryId(spu.getCategory3Id());
        //分类名字
        sku.setCategoryName(category.getName());
        //品牌名字
        sku.setBrandName(brand.getName());
        //增加
        skuMapper.insertSelective(sku);
    }
}
```

(3)控制层

修改com.changgou.goods.controller.SpuController，增加保存Goods方法，代码如下：

```
/***
 * 添加Goods
 * @param goods
 * @return
 */
@PostMapping("/save")
public Result save(@RequestBody Goods goods){
    spuService.saveGoods(goods);
    return new Result(true,StatusCode.OK,"保存成功");
}
```

测试数据

```
{
  "skuList": [
    {
      "alertNum": 10,
      "brandName": "华为",
      "categoryId": 64,
      "commentNum": 0,
      "image": "http://www.baidu.com",
      "images": "",
      "name": "华为P30手机",
      "num": 5,
      "price": 1000,
      "saleNum": 0,
      "sn": "No1001",
      "spec": "{\"颜色\":\"红色\",\"网络\":\"移动3G\"}",
      "weight": 0
    }
  ],
  "spu": {
    "brandId": 8557,
    "caption": "华为手机大促销",
    "category1Id": 1,
    "category2Id": 59,
    "category3Id": 64,
    "commentNum": 0,
    "freightId": 0,
    "images":
"http://www.qingcheng.com/image/1.jpg,http://www.qingcheng.com/image/2.jpg",
    "introduction": "华为产品世界最强",
    "isEnableSpec": "1",
    "isMarketable": "1",
    "name": "string",
    "specItems": "{\"颜色\":[\"红\",\"绿\"],\"机身内存\":[\"64G\",\"8G\"]}",
    "paraItems": "{\"赠品\":\"充电器\",\"出厂年份\":\"2019\"}",
    "saleNum": 0,
    "saleService": "一年包换",
    "sn": "No10001",
    "status": "1",
    "templateId": 42
  }
}
```

此时也可以使用页面对接了。

2.4.7 根据ID查询商品

2.4.7.1 需求分析

需求：根据id 查询SPU和SKU列表 ， 显示效果如下：

```
{
    "spu": {
        "brandId": 0,
        "caption": "111",
```

```
        "category1Id": 558,
        "category2Id": 559,
        "category3Id": 560,
        "commentNum": null,
        "freightId": null,
        "id": 149187842867993,
        "image": null,
        "images": null,
        "introduction": null,
        "isDelete": null,
        "isEnableSpec": "0",
        "isMarketable": "1",
        "name": "黑马智能手机",
        "paraItems": null,
        "saleNum": null,
        "saleService": null,
        "sn": null,
        "specItems": null,
        "status": null,
        "templateId": 42
    },
    "skuList": [{
        "alertNum": null,
        "brandName": "金立（Gionee）",
        "categoryId": 560,
        "categoryName": "手机",
        "commentNum": null,
        "createTime": "2018-11-06 10:17:08",
        "id": 1369324,
        "image": null,
        "images": "blob:http://localhost:8080/ec04d1a5-d865-4e7f-a313-
2e9a76cfb3f8",
        "name": "黑马智能手机",
        "num": 100,
        "price": 900000,
        "saleNum": null,
        "sn": "",
        "spec": null,
        "spuId": 149187842867993,
        "status": "1",
        "updateTime": "2018-11-06 10:17:08",
        "weight": null
    },{
        "alertNum": null,
        "brandName": "金立（Gionee）",
        "categoryId": 560,
        "categoryName": "手机",
        "commentNum": null,
        "createTime": "2018-11-06 10:17:08",
        "id": 1369325,
        "image": null,
        "images": "blob:http://localhost:8080/ec04d1a5-d865-4e7f-a313-
2e9a76cfb3f8",
        "name": "黑马智能手机",
        "num": 100,
        "price": 900000,
        "saleNum": null,
        "sn": "",
```

```
        "spec": null,
        "spuId": 149187842867993,
        "status": "1",
        "updateTime": "2018-11-06 10:17:08",
        "weight": null
      }
   ]
}
```

2.4.7.2 代码实现

(1)业务层

修改qingcheng-service-goods工程,修改com.changgou.goods.service.SpuService接口,添加根据ID查找方法findGoodsById代码如下:

```
/***
 * 根据SPU的ID查找SPU以及对应的SKU集合
 * @param spuId
 */
Goods findGoodsById(String spuId);
```

修改qingcheng-service-goods工程，修改com.changgou.goods.service.impl.SpuServiceImpl类，添加根据ID查找findGoodsById方法，代码如下:

```
/***
 * 根据SpuID查询goods信息
 * @param spuId
 * @return
 */
@Override
public Goods findGoodsById(String spuId) {
    //查询Spu
    Spu spu = spuMapper.selectByPrimaryKey(spuId);

    //查询List<Sku>
    Sku sku = new Sku();
    sku.setSpuId(spuId);
    List<Sku> skus = skuMapper.select(sku);
    //封装Goods
    Goods goods = new Goods();
    goods.setSkuList(skus);
    goods.setSpu(spu);
    return goods;
}
```

(2)控制层

修改com.changgou.goods.controller.SpuController，修改findById方法，代码如下:

```java
/***
 * 根据ID查询Goods
 * @param id
 * @return
 */
@GetMapping("/goods/{id}")
public Result<Goods> findGoodsById(@PathVariable Long id){
    //根据ID查询Goods(SPU+SKU)信息
    Goods goods = spuService.findGoodsById(id);
    return new Result<Goods>(true,StatusCode.OK,"查询成功",goods);
}
```

测试：`http://admin-changgou-java.itheima.net/spu/goods/No1207055688174403584`

2.4.8 保存修改

修改changgou-service-goods的SpuServiceImpl的saveGoods方法，修改添加SPU部分代码：

```java
/***
 * 保存Goods
 * @param goods
 */
@Override
public void saveGoods(Goods goods) {
    //增加Spu
    Spu spu = goods.getSpu();
    if(spu.getId()==null){
        //增加
        spu.setId("No"+idWorker.nextId());
        spuMapper.insertSelective(spu);
    }else{
        //修改数据
        spuMapper.updateByPrimaryKeySelective(spu);
        //删除该Spu的Sku
        Sku sku = new Sku();
        sku.setSpuId(spu.getId());
        skuMapper.delete(sku);
    }

    //增加Sku
    Date date = new Date();
    Category category = categoryMapper.selectByPrimaryKey(spu.getCategory3Id());
    Brand brand = brandMapper.selectByPrimaryKey(spu.getBrandId());
    //获取Sku集合
    List<Sku> skus = goods.getSkuList();
    //循环将数据加入到数据库
    for (Sku sku : skus) {
        //构建SKU名称，采用SPU+规格值组装
        if(StringUtils.isEmpty(sku.getSpec())){
            sku.setSpec("{}");
        }
        //获取Spu的名字
        String name = spu.getName();

        //将规格转换成Map
        Map<String,String> specMap = JSON.parseObject(sku.getSpec(), Map.class);
        //循环组装Sku的名字
        for (Map.Entry<String, String> entry : specMap.entrySet()) {
            name+="  "+entry.getValue();
        }
        sku.setName(name);
        //ID
        sku.setId("No"+idWorker.nextId());
        //SpuId
        sku.setSpuId(spu.getId());
        //创建日期
        sku.setCreateTime(date);
        //修改日期
        sku.setUpdateTime(date);
        //商品分类ID
        sku.setCategoryId(spu.getCategory3Id());
        //分类名字
        sku.setCategoryName(category.getName());
        //品牌名字
        sku.setBrandName(brand.getName());
        //增加
        skuMapper.insertSelective(sku);
    }
}
```

如果传入了ID，表示修改

上图代码如下：

```java
//增加Spu
Spu spu = goods.getSpu();
if(spu.getId()==null){
    //增加
```

```
    spu.setId("No"+idWorker.nextId());
    spuMapper.insertSelective(spu);
}else{
    //修改数据
    spuMapper.updateByPrimaryKeySelective(spu);
    //删除该Spu的Sku
    Sku sku = new Sku();
    sku.setSpuId(spu.getId());
    skuMapper.delete(sku);
}
```

2.4.9 修改SKU库存

（学员实现）

总结

3 商品审核与上下架

目标

- 商品状态操作

路径

- 商品审核
- 商品上架
- 商品下架
- 商品批量上架
- 商品批量下架

讲解

3.1 需求分析

商品新增后，审核状态为0（未审核），默认为下架状态。

审核商品，需要校验是否是被删除的商品，如果未删除则修改审核状态为1，并自动上架

下架商品，需要校验是否是被删除的商品，如果未删除则修改上架状态为0

上架商品，需要审核通过的商品,未删除的商品

3.2 实现思路

（1）按照ID查询SPU信息

（2）判断修改审核、上架和下架状态

（3）保存SPU

3.3 代码实现

3.3.1 商品审核

实现审核通过，自动上架。

(1)业务层

修改修改changgou-service-goods工程的com.changgou.goods.service.SpuService接口，添加审核方法，代码如下：

```
/***
 * 商品审核
 * @param spuId
 */
void audit(String spuId);
```

修改changgou-service-goods工程的com.changgou.goods.service.impl.SpuServiceImpl类，添加audit方法，代码如下：

```
/***
 * 商品审核
 * @param spuId
 */
@Override
public void audit(String spuId) {
    //查询商品
    Spu spu = spuMapper.selectByPrimaryKey(spuId);
    //判断商品是否已经删除
    if(spu.getIsDelete().equalsIgnoreCase("1")){
        throw new RuntimeException("该商品已经删除！");
    }
    //实现上架和审核
    spu.setStatus("1"); //审核通过
    spu.setIsMarketable("1"); //上架
    spuMapper.updateByPrimaryKeySelective(spu);
}
```

(2)控制层

修改com.changgou.goods.controller.SpuController，新增audit方法，代码如下：

```
/**
 * 审核
 * @param id
 * @return
 */
@PutMapping("/audit/{id}")
public Result audit(@PathVariable String id){
    spuService.audit(id);
    return new Result(true,StatusCode.OK,"审核成功");
}
```

3.3.2 下架商品

(1)业务层

修改com.changgou.goods.service.SpuService接口，添加pull方法，用于商品下架，代码如下：

```
/***
 * 商品下架
 * @param spuId
 */
void pull(String spuId);
```

修改com.changgou.goods.service.impl.SpuServiceImpl，添加如下方法：

```
/**
 * 商品下架
 * @param spuId
 */
@Override
public void pull(String spuId) {
    Spu spu = spuMapper.selectByPrimaryKey(spuId);
    if(spu.getIsDelete().equals("1")){
        throw new RuntimeException("此商品已删除！");
    }
    spu.setIsMarketable("0");//下架状态
    spuMapper.updateByPrimaryKeySelective(spu);
}
```

(2)控制层

修改com.changgou.goods.controller.SpuController，添加pull方法，代码如下：

```
/**
 * 下架
 * @param id
 * @return
 */
@PutMapping("/pull/{id}")
public Result pull(@PathVariable String id){
    spuService.pull(id);
    return new Result(true,StatusCode.OK,"下架成功");
}
```

3.3.3 上架商品

必须是通过审核的商品才能上架

(1)业务层

修改com.changgou.goods.service.SpuService，添加put方法，代码如下：

```
/***
 * 商品上架
 * @param spuId
 */
void put(String spuId);
```

修改com.changgou.goods.service.impl.SpuServiceImpl，添加put方法实现，代码如下：

```
/***
 * 商品上架
 * @param spuId
 */
@Override
public void put(String spuId) {
    Spu spu = spuMapper.selectByPrimaryKey(spuId);
    //检查是否删除的商品
    if(spu.getIsDelete().equals("1")){
        throw new RuntimeException("此商品已删除！");
    }
    if(!spu.getStatus().equals("1")){
        throw new RuntimeException("未通过审核的商品不能！");
    }
    //上架状态
    spu.setIsMarketable("1");
    spuMapper.updateByPrimaryKeySelective(spu);
}
```

(2)控制层

修改com.changgou.goods.controller.SpuController，添加put方法代码如下：

```java
/**
 * 商品上架
 * @param id
 * @return
 */
@PutMapping("/put/{id}")
public Result put(@PathVariable String id){
    spuService.put(id);
    return new Result(true,StatusCode.OK,"上架成功");
}
```

3.3.4 批量上架

前端传递一组商品ID，后端进行批量上下架处理

(1)业务层

修改com.changgou.goods.service.SpuService接口，代码如下：

```java
int putMany(String[] ids);
```

修改com.changgou.goods.service.impl.SpuServiceImpl，添加批量上架方法实现，代码如下：

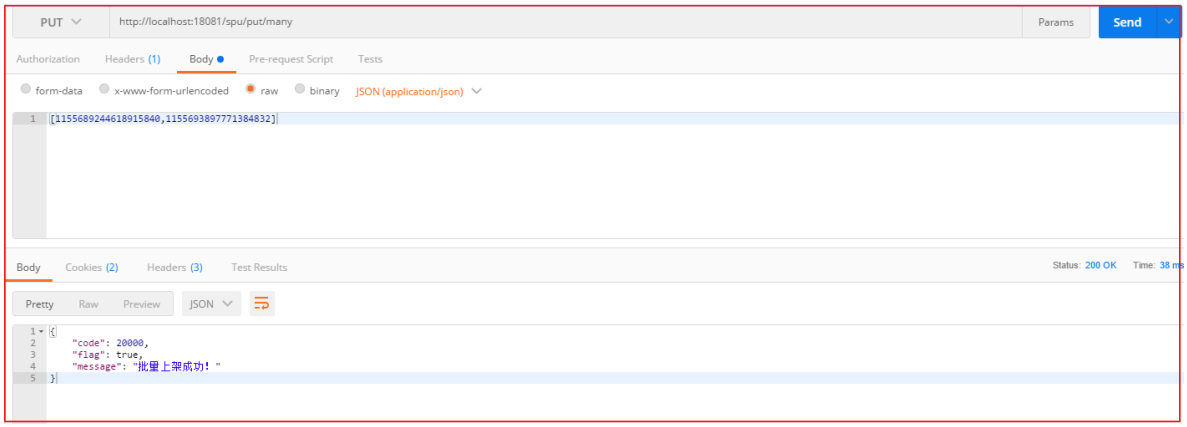
```java
/***
 * 批量上架
 * @param ids:需要上架的商品ID集合
 * @return
 */
@Override
public int putMany(String[] ids) {
    Spu spu=new Spu();
    spu.setIsMarketable("1");//上架
    //批量修改
    Example example=new Example(Spu.class);
    Example.Criteria criteria = example.createCriteria();
    criteria.andIn("id", Arrays.asList(ids));//id
    //下架
    criteria.andEqualTo("isMarketable","0");
    //审核通过的
    criteria.andEqualTo("status","1");
    //非删除的
    criteria.andEqualTo("isDelete","0");
    return spuMapper.updateByExampleSelective(spu, example);
}
```

(2)控制层

修改com.changgou.goods.controller.SpuController，天啊及批量上架方法，代码如下：

```java
/**
 *  批量上架
 * @param ids
 * @return
 */
@PutMapping("/put/many")
public Result putMany(@RequestBody String[] ids){
    int count = spuService.putMany(ids);
    return new Result(true,StatusCode.OK,"上架"+count+"个商品");
}
```

使用Postman测试：



3.3.5 批量下架(作业)

学员实现

总结

4 删除与还原商品

目标

- 商品删除

路径

- 商品逻辑删除
- 商品还原
- 商品物理删除

讲解

4.1 需求分析

请看管理后台的静态原型

商品列表中的删除商品功能，并非真正的删除，而是将删除标记的字段设置为1，

在回收站中有恢复商品的功能，将删除标记的字段设置为0

在回收站中有删除商品的功能，是真正的物理删除。

4.2 实现思路(作业)

逻辑删除商品，下架商品,状态为未审核,修改spu表is_delete字段为1

商品回收站显示spu表is_delete字段为1的记录

回收商品，修改spu表is_delete字段为0

4.3 代码实现

4.3.1 逻辑删除商品

(1)业务层

修改com.changgou.goods.service.SpuService接口，增加logicDelete方法，代码如下：

```
/***
 * 逻辑删除
 * @param spuId
 */
void logicDelete(String spuId);
```

修改com.changgou.goods.service.impl.SpuServiceImpl，添加logicDelete方法实现，代码如下：

```
/***
 * 逻辑删除
 * @param spuId
 */
@Override
@Transactional
public void logicDelete(String spuId) {
    Spu spu = spuMapper.selectByPrimaryKey(spuId);
    //检查是否下架的商品
    if(!spu.getIsMarketable().equals("0")){
        throw new RuntimeException("必须先下架再删除！");
    }
    //删除
    spu.setIsDelete("1");
    //未审核
    spu.setStatus("0");
    spuMapper.updateByPrimaryKeySelective(spu);
}
```

(2)控制层

修改com.changgou.goods.controller.SpuController，添加logicDelete方法，如下：

```
/**
 * 逻辑删除
 * @param id
 * @return
 */
@DeleteMapping("/logic/delete/{id}")
public Result logicDelete(@PathVariable String id){
    spuService.logicDelete(id);
    return new Result(true,StatusCode.OK,"逻辑删除成功！");
}
```

4.3.2 还原被删除的商品

(1)业务层

修改com.changgou.goods.service.SpuService接口，添加restore方法代码如下：

```
/***
 * 还原被删除商品
 * @param spuId
 */
void restore(String spuId);
```

修改com.changgou.goods.service.impl.SpuServiceImpl类，添加restore方法，代码如下：

```
/**
 * 恢复数据
 * @param spuId
 */
@Override
public void restore(String spuId) {
    Spu spu = spuMapper.selectByPrimaryKey(spuId);
    //检查是否删除的商品
    if(!spu.getIsDelete().equals("1")){
        throw new RuntimeException("此商品未删除！");
    }
    //未删除
    spu.setIsDelete("0");
    //未审核
    spu.setStatus("0");
    spuMapper.updateByPrimaryKeySelective(spu);
}
```

(2)控制层

修改com.changgou.goods.controller.SpuController，添加restore方法，代码如下：

```java
/**
 * 恢复数据
 * @param id
 * @return
 */
@PutMapping("/restore/{id}")
public Result restore(@PathVariable String id){
    spuService.restore(id);
    return new Result(true,StatusCode.OK,"数据恢复成功！");
}
```

4.3.3 物理删除商品

修改com.changgou.goods.service.impl.SpuServiceImpl的delete方法,代码如下：

```java
/**
 * 删除
 * @param id
 */
@Override
public void delete(String id){
    Spu spu = spuMapper.selectByPrimaryKey(id);
    //检查是否被逻辑删除  ,必须先逻辑删除后才能物理删除
    if(!spu.getIsDelete().equals("1")){
        throw new RuntimeException("此商品不能删除！");
    }
    spuMapper.deleteByPrimaryKey(id);
}
```

总结

5 商品列表

目标

- 商品列表搜索

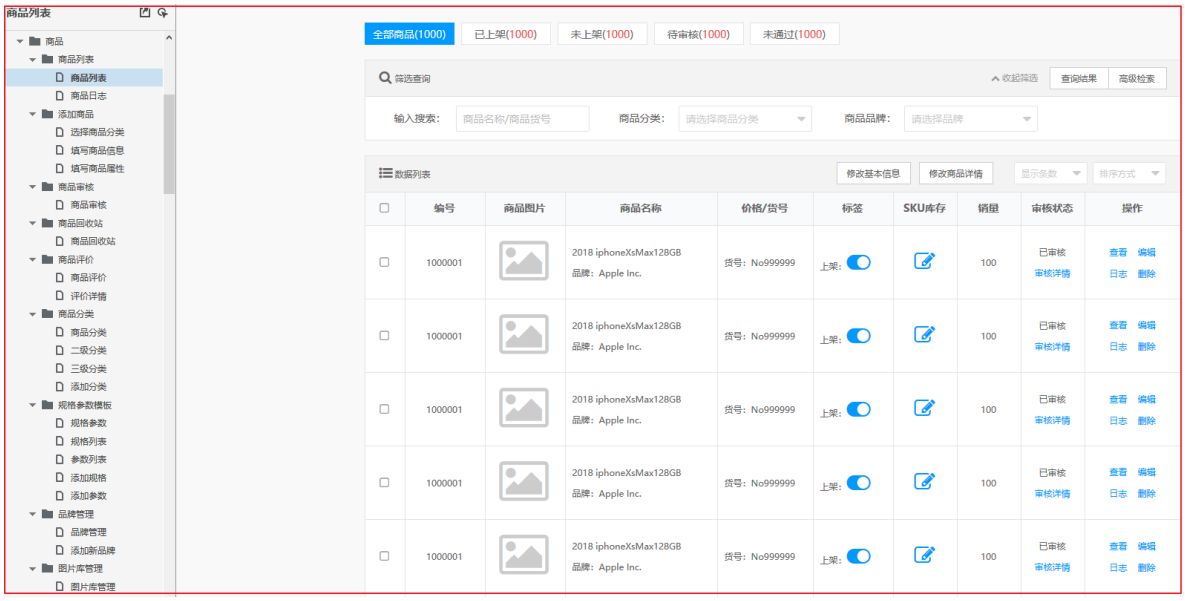
路径

- 商品列表搜索

讲解

5.1 需求分析

如图所示 展示商品的列表。并实现分页。



思路：

```
根据查询的条件 分页查询 并返回分页结果即可。
分页查询 采用 pagehelper ，条件查询  通过map进行封装传递给后台即可。
```

5.2 代码实现

在代码生成器生成的代码中已经包含了该实现，这里就省略了。

控制层（SpuController）：

```
/***
 * Spu分页条件搜索实现
 * @param spu
 * @param page
 * @param size
 * @return
 */
@PostMapping(value = "/search/{page}/{size}" )
public Result<PageInfo> findPage(@RequestBody(required = false) Spu spu,
@PathVariable  int page, @PathVariable  int size){
    //执行搜索
    PageInfo<Spu> pageInfo = spuService.findPage(spu, page, size);
    return new Result(true,StatusCode.OK,"查询成功",pageInfo);
}
```

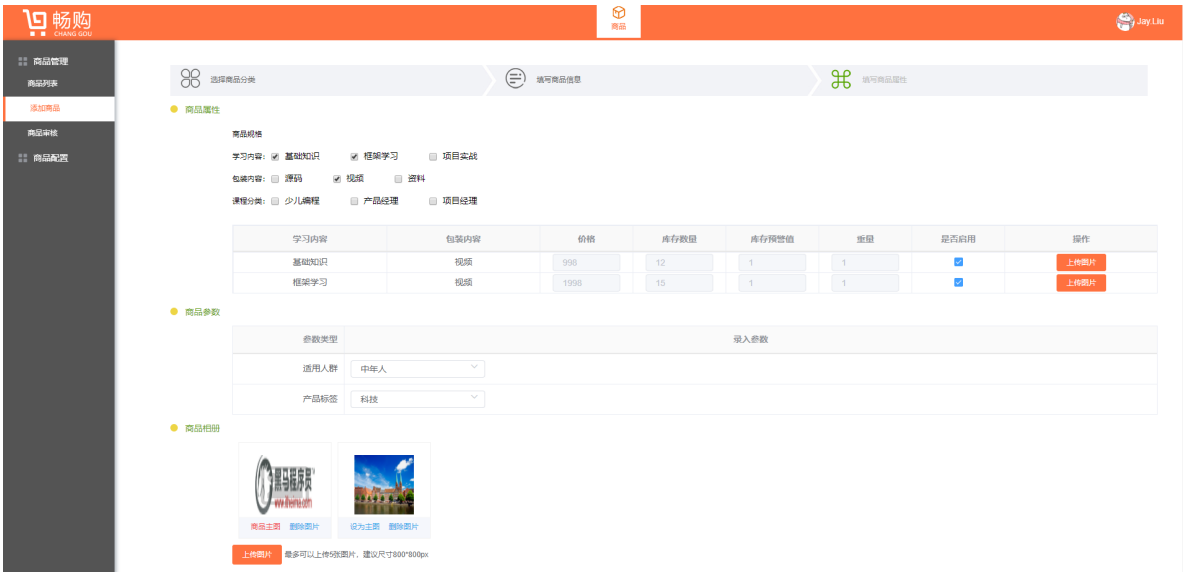
其他每层代码，代码生成器已经生成，这里就不再列出来了。

总结

6 页面对接

Swagger测试接口

页面对接



课后要求:

1.2.4节中的1-8的功能,完成商品的新增 修改 以及新增修改商品的相关的附加的查询

使用postman调用接口能查到数据,能新增数据就OK

2.审核 上下架 批量的上下架功能(晚上的11点以后)

3.商品的删除与还原功能