谷粒manage

# 1 pms商品管理系统的介绍

|  |
| --- |
| 1 系统名称  Gmall-Manage  2 数据结构  Pms  Sku + spu  Sku : stock keeping until 库存存储单元(一般只一个具体的库存商品，单位时台、部、件)  Spu : standard product until 标准的商品单元(一般一个商品XXX系列，就是一个spu) |

# 2 pms商品的数据结构的划分

|  |
| --- |
| 1 sku的结构 pms\_sku\_  2 spu的结构 pms\_spu\_  3类目的结构 pms\_catalog\_  4属性的结构 pms\_attr\_ |

# 3 manage系统前后端分离

|  |
| --- |
| 1 准备阶段  渲染、模板技术  2 前后端分离  Jvm spring maven idea  Nodejs vue npm vscode  3 安装nodejs  下一步。。。  Node -v查看安装结果  4 用npm install命令安装npm    5 解压前端项目(gmall-admin)  Conf目录下  配置前端服务的ip和前端访问数据的后端的服务的ip地址  dev.env.js 前端访问后端的数据服务的地址  index.js 前端的服务器端口  6 用npm命令编译和启动前端项目  在gmall-admin目录下运行  Npm run dev    7 前后端请求格式  一般前端会用post想后端发送请求(把参数封装到json中)  请求格式@RequestBody  返回格式@ResponseBody |

JavaEE、前端模板技术

浏览器

前端服务器

JavaEE

浏览器

Json

# 4 商城的商品录入功能(manage)

|  |
| --- |
| 1 三级分类的查询 |
| 2 商品的平台属性列表的增删改查 |
| 3 商品spu的添加  Spu列表查询  spu的销售属性、属性值、Fastdfs图片上传 |
| 4 商品sku的添加  sku信息、sku关联的销售属性、sku关联的平台属性、sku图片 |

|  |
| --- |
| 1 sku的结构 pms\_sku\_  2 spu的结构 pms\_spu\_  3类目的结构 pms\_catalog\_  三级分类，一级二级三级  4属性的结构 pms\_attr\_  平台属性的外键是三级分类id，在使用平台属性功能之前必须选择三级分类 |

# 5 商品分类的查询

|  |
| --- |
| 1 新建gmall-manage-web项目  2 配置gmall-manage-web  3 写一个getCatalog1写给前端项目调用  4 返回一个分类列表的集合(josn)  5 新建一个catalogService的服务    6 实现cagalogService的功能  新建mapper |

# 6 前后端的跨域问题

|  |
| --- |
| 1 前端127.0.0.1:8888  2 后端127.0.0.1:8080  前端和后端因为来自不同的网域，所以在http的安全协议策略下，不信任    3 解决方案，在springmvc的控制层加入@CrossOrigin跨域访问的注解 |

127.0.0.1:8888

127.0.0.1:8080

# 7 商品平台属性的管理功能(增删改查)

|  |
| --- |
| 1 根据三级分类id，catalog3Id查询平台属性列表  A新建一个AttrController，方法attrInfoList  B新建PmsBaseAttrInfo和PmsBaseAttrValue的映射类  C新建service、mapper |

|  |
| --- |
| Ajax  Axios |

# 8 作业

|  |
| --- |
| 1 复习《04 谷粒商品pms.docx》的数据结构  2 完成pms的属性增删改查 |

# 9 数据结构的复习

|  |
| --- |
| 1 什么是spu和sku？  Spu(标准的商品单元) standard product unit  Sku(库存存储单元，单位件、台、部、盒) stock keep unit  2它们之间的关系是什么?  范围上来说Spu包含sku  Spu表和sku表是一对多的关系  3 什么是平台属性，平台属性功能涉及的表由那两张  两张：pms\_base\_attr\_info(平台属性表) pms\_base\_attr\_value(平台属性值表)  4 平台属性和商品分类、商品之间是什么关系 |

商品的分类

商品

分类下的平台属性

# 10 平台属性的修改操作

|  |
| --- |
| 1 根据平台属性id判断，有id是修改操作，没有id是添加操作  2 修改操作  A 先修改平台属性  *// 属性修改* Example example = **new** Example(PmsBaseAttrInfo.**class**); example.createCriteria().andEqualTo(**"id"**,pmsBaseAttrInfo.getId()); **pmsBaseAttrInfoMapper**.updateByExampleSelective(pmsBaseAttrInfo,example);  B 修改平台属性值  *// 属性值修改 // 按照属性id删除所有属性值* PmsBaseAttrValue pmsBaseAttrValueDel = **new** PmsBaseAttrValue(); pmsBaseAttrValueDel.setAttrId(pmsBaseAttrInfo.getId()); **pmsBaseAttrValueMapper**.delete(pmsBaseAttrValueDel);  *// 删除后，将新的属性值插入* List<PmsBaseAttrValue> attrValueList = pmsBaseAttrInfo.getAttrValueList(); **for** (PmsBaseAttrValue pmsBaseAttrValue : attrValueList) {  **pmsBaseAttrValueMapper**.insertSelective(pmsBaseAttrValue); }  3 进入修改页面，调用<http://127.0.0.1:8081/getAttrValueList?attrId=43>  实现getAttrValueList的方法，为修改页面查询一个平台属性值的集合 |

# 11 商品spu的功能介绍

商品分类

商品的属性

商品的spu

商品的sku

商品的sku

|  |
| --- |
| 1 根据spu和sku的电商设计模型  2 根据电商网站用户检索和过滤的需求 |

分类

商品spu

平台属性

商品的可购买单元sku

# 12 spu的功能

|  |
| --- |
| 1 <http://127.0.0.1:8081/spuList?catalog3Id=61>  2 spu数据列表的查询  3 spu的添加功能 |

# 13 spu 的添加功能

|  |
| --- |
| 1 spu信息  Spu名称，spu的描述  2 spu图片信息  图片的对象数据保存在分布式的文件存储服务器上(fastdfs)  图片的元数据信息保存在数据库中  用户在选择完图片后  a我们把图片在用户提交时候和其他的商品spu信息一起提交到后台  b我们在用户选择图片是，就将图片上传至服务器  3 spu的销售属性信息  商品的平台属性属于电商网站后台管理（整个商品平台的维度下的）  商品的销售属性属于在电商网站上卖商品的商家管理(属于某一件商品的维度下的)  4销售属性字典表（**baseSaleAttrList**）  商家在添加spu商品信息时，需要添加销售属性(自定义)  在添加spu的页面，商家先选择销售属性(平台后台定义的销售属性字典表)，然后自定义当前商品的销售属性值    \*修改前端的数据属性名和后端的数据属性名不对应的地方  spuSaleAttrList -> **pmsProductSaleAttrList**  spuSaleAttrValueList -> **pmsProductSaleAttrValueList**  spuImageList -> **pmsProductImageList**  **5 图片信息的处理（**fileUpload**）**  <form method = “post” enctype=”multipart/Form-data”>  <input type = “file”/>  </form>  用户点击上传图片后，后台将图片存储到服务器上，然后返回图片的访问路径给前端  6 用户点击保存spu信息（saveSpuInfo）  将spu的基本信息、销售属性列表、图片地址列表的元数据保存到后台 |

# 14 分布式文件存储(fastdfs)

图片存储服务器

图片存储服务器

图片存储服务器

|  |
| --- |
| 0 安装依赖软件和类库(安装前的准备)  yum install gcc-c++ -y  yum -y install zlib zlib-devel pcre pcre-devel gcc gcc-c++ openssl openssl-devel libevent libevent-devel perl unzip net-tools wget  yum -y install libevent  yum install perl\*  1 fdfs的依赖库  Libfastcommon  A解压  b./make.sh  c./make.sh install  d用cp /usr/lib64/libfastcommon.so /usr/lib/将类库拷贝到/usr/lib下    2 fastdfs软件(tracker、storage)  配置tracker  配置storage  (依赖于：Gcc、libevent、perl)  A新建目录mkdir /opt/fastdfs  B解压FastDFS\_v5.05.tar.gz到/usr/local  C进入解压目录cd FastDFS  D./make.sh  E./make.sh install  F进入conf配置目录将文件都拷贝到/etc/fdfs下cp \* /etc/fdfs/（安装时自动生成）  G进入/etc/fdfs/，配置tracker.conf  vim /etc/fdfs/tracker.conf ，设置软件数据和日志目录    H storage的配置(storage不需要安装，因为安装tracker时已经同时安装)  vim /etc/fdfs/storage.conf  软件目录    Storage存储文件的目录（新建mkdir /opt/fastdfs/fdfs\_storage）    Storage的trackerip    3 配置tracker和storage的启动服务  进入/etc/init.d启动脚本目录，默认fastdfs已经生成    Vi fdfs\_trackerd脚本文件      mkdir /usr/local/fdfs（因为启动脚本还在安装目录下，所以我们新建/usr/local/fdfs目录，并且将启动脚本cp到该目录）  进入安装目录/opt/FastDFs  cp restart.sh /usr/local/fdfs/  cp stop.sh /usr/local/fdfs/  配置storage启动服务(restart和stop脚本已经拷贝到/usr/local/fdfs下，所以storage只需要配置/etc/init.d/fdfs\_storage脚本就可以了)      将启动脚本加入linux服务    启动服务    测试上传  4 FastDFS-nginx-module  Fdfs整合nginx的插件  A 解压FastDFS-nginx-module插件  B 修改插件读取fdfs的目录（插件自己的配置文件）  Vi fastdfs-nginx-module/src/config    C 将FastDFS-nginx-module插件整合fdfs的配置文件拷贝到fdfs的配置目录下(整合fdfs的配置文件)  FastDFS-nginx-module/src下的mod\_fastdfs.conf拷贝至/etc/fdfs/下(这里面是两个路径)  软件安装目录    Tracker\_server地址    Web的url是否包含group的路径名    上传文件存储目录    5 nginx  做web服务器，提供http请求服务  (依赖于：pcre-devel、zlib-devel)  ./configure \  --prefix=/usr/local/nginx \  --pid-path=/var/run/nginx/nginx.pid \  --lock-path=/var/lock/nginx.lock \  --error-log-path=/var/log/nginx/error.log \  --http-log-path=/var/log/nginx/access.log \  --with-http\_gzip\_static\_module \  --http-client-body-temp-path=/var/temp/nginx/client \  --http-proxy-temp-path=/var/temp/nginx/proxy \  --http-fastcgi-temp-path=/var/temp/nginx/fastcgi \  --http-uwsgi-temp-path=/var/temp/nginx/uwsgi \  --http-scgi-temp-path=/var/temp/nginx/scgi \  --add-module=/opt/fastdfs-nginx-module/src    修改nginx配置文件  Vi /nginx/conf/nginx.conf    别忘了创建临时的目录/var/temp/nginx/client |

# 15 fdfs的集群

Tracker

nginx

Tracker

Tracker

Tracker

nginx

# 16 fdfs和springboot的整合

|  |
| --- |
| 1 通过git下载fdfs的客户端  <https://github.com/happyfish100/fastdfs-client-java>    2 将fdfs的客户端打包到本地maven仓库中  导入maven      3 将fdfs整合到pom中  <**dependency**>  <**groupId**>org.csource</**groupId**>  <**artifactId**>fastdfs-client-java</**artifactId**>  <**version**>1.27-SNAPSHOT</**version**> </**dependency**>  4 实现spu图片的上传工具类    5 spu保存涉及到的数据结构  Pms\_product\_info  Pms\_product\_image  Pms\_product\_sale\_attr  Pms\_procuct\_sale\_attr\_value  Pms\_base\_sale\_attr |

# 17 sku管理功能

|  |
| --- |
| Pms\_Sku\_info  Pms\_sku\_image(从已经添加的spu的图片库中选择，当前sku涉及的图片)  Pms\_Sku\_attr\_value  Pms\_Sku\_sale\_attr\_value\_  当前sku所关联的平台属性中间表    Sku\_info    Sku\_attr\_value    Base\_attr\_info    Base\_attr\_value    平台属性和销售属性的区别：  平台属性属于所有商品，由电商网站维护管理  销售属性属于商家，由商家来维护管理 |

# 18 添加sku

|  |
| --- |
| 1 平台属性列表(平台属性+平台属性值双层集合)  <http://127.0.0.1:8081/attrInfoList?catalog3Id=61>  2 销售属性列表  <http://127.0.0.1:8081/spuSaleAttrList?spuId=24>  3 spu的图片列表  <http://127.0.0.1:8081/spuImageList?spuId=24>  4 保存spu信息  <http://127.0.0.1:8081/saveSkuInfo>  5 保存skuInfo  新建controller、service、mapper，增删改查。。。 |