**品优购电商系统开发**

**第2章**

**通用Mapper&Nginx&品牌管理**

传智播客.黑马程序员

# 课程目标

目标1：掌握代码生成器的使用

目标2：完成安装Nginx与配置nginx.conf

目标3：实现品牌列表的前端代码

目标4：实现增加品牌的功能

目标5：实现修改品牌的功能

目标6：实现品牌条件查询的功能

目标7：实现删除品牌的功能

# 分页PageHelper与通用Mapper

## 1.1集成分页助手PageHelper

使用分页助手PageHelper的好处？

1. 简化分页查询。select \* from tb\_user limit ?,?
2. 简化统计查询。Select count(1) from tb\_user
3. 不用关心底层数据库(mysql、oracle)，因为分页助手会根据方言自动生成数据库相关分页、统计SQL语句。

### 1.1.1配置部分

修改pinyougou-mapper的pom.xml文件，加入依赖

|  |
| --- |
| *<!-- 配置分页插件 -->* <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**> </**dependency**> |

修改pinyougou-mapper/src/main/resources/mybatis-config.xml

|  |
| --- |
| *<?***xml version="1.0" encoding="UTF-8"***?>* **<!DOCTYPE configuration  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"  "http://mybatis.org/dtd/mybatis-3-config.dtd"*>*** <**configuration**>  *<!-- 全局的设置信息-->* <**settings**>  *<!-- 开启驼峰命名(方便自动映射) dept\_id deptId -->* <**setting name="mapUnderscoreToCamelCase" value="true"**/>  </**settings**>  *<!-- 配置插件 -->* <**plugins**>  *<!-- 配置分页拦截器(拦截SqL语句，在你的sql语句后面加上limit) -->* <**plugin interceptor="com.github.pagehelper.PageInterceptor"**>  *<!-- 配置数据库方言(mysql、oracle、sqlserver等) -->* <**property name="helperDialect" value="mysql"**/>  *<!-- 配置开启统计查询 count -->* <**property name="rowBoundsWithCount" value="true"**/>  </**plugin**>  </**plugins**> </**configuration**> |

### 1.1.2代码部分

pinyougou-sellergoods-service/src/main/java/com/pinyougou/sellergoods/service/impl/BrandServiceImpl.java

|  |
| --- |
| @Override **public** List<Brand> findAll() {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(1, 10)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper**.findAll();  }  });  System.***out***.println(**"总记录数："** + pageInfo.getTotal());  System.***out***.println(**"总页数："** + pageInfo.getPages());  **return** pageInfo.getList(); } |

**测试地址为：[http://localhost:9101/brand/f](http://localhost:9091/rest/item/cat/query/1?rows=5)indAll**

PageInfo对象常用方法：

* pageInfo.getTotal(): 获取总记录数
* pageInfo.getPages(): 获取总页数
* pageInfo.getList(): 获取分页数据集合

## 1.2集成通用Mapper

使用通用Mapper的好处？

1. 简化单表CRUD操作(不需要写SQL语句)，添加、删除、修改、查询是经常需要用到的功能，加快项目开发进度。

### 1.2.1配置部分

修改pinyougou-pojo项目的pom.xml文件，加入依赖

|  |
| --- |
| <**dependencies**>  *<!-- 配置通用mapper -->* <**dependency**>  <**groupId**>tk.mybatis</**groupId**>  <**artifactId**>mapper</**artifactId**>  </**dependency**> </**dependencies**> |

pinyougou-mapper/src/main/resources/applicationContext-mapper.xml添加如下配置信息：

|  |
| --- |
| *<!-- 配置通用Mapper集成Spring4 -->* <**bean class="tk.mybatis.spring.mapper.MapperScannerConfigurer"**>  *<!-- 设置数据访问接口类 -->* <**property name="basePackage" value="com.pinyougou.mapper"**/> </**bean**> |

### 1.2.2代码部分

修改pinyougou-pojo/src/main/java/com/pinyougou/pojo/Brand.java，加入JPA注解，通用Mapper需要该类与数据库表之间的映射关系

|  |
| --- |
| @Table(name=**"tb\_brand"**) **public class** Brand **implements** Serializable{  */\*\* 主键ID \*/* @Id @GeneratedValue(strategy=GenerationType.***IDENTITY***)  @Column(name=**"id"**)  **private** Long **id**;  */\*\* 品牌名称 \*/* @Column(name=**"name"**)  **private** String **name**;  */\*\* 品牌首字母 \*/* @Column(name=**"first\_char"**)  **private** String **firstChar**;  ......  } |

pinyougou-mapper/src/main/java/com/pinyougou/mapper/BrandMapper.java，继承Mapper接口(Mapper接口有CRUD方法)

|  |
| --- |
| */\*\* 品牌数据访问接口 \*/* **public interface** BrandMapper **extends** Mapper<Brand> { } |

修改pinyougou-sellergoods-service/src/main/java/

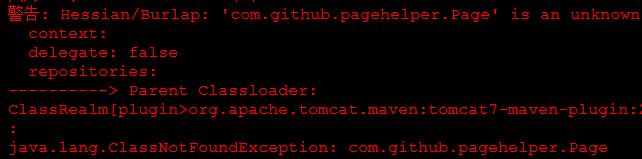
com.pinyougou.sellergoods.service.impl包下的BrandServiceImpl.java：

|  |
| --- |
| @Override **public** List<Brand> findAll() {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(1, 10)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper.selectAll();**  }  });  System.***out***.println(**"总记录数："** + pageInfo.getTotal());  System.***out***.println(**"总页数："** + pageInfo.getPages());  **return** pageInfo.getList(); } |

**测试地址为：[http://localhost:9101/brand/f](http://localhost:9091/rest/item/cat/query/1?rows=5)indAll**

Mapper接口常用到得四批方法：

1. mapper.selectXxx(): 查询方法
2. mapper.insertXxx(): 添加方法
3. mapper.updateXxx(): 修改方法
4. mapper.deleteXxx(): 删除方法



注意：因为pinyougou-manager-web工程也需要用到Page类，所以需要在pinyougou-interface工程的pom.xml文件加入以下依赖：

|  |
| --- |
| *<!-- 配置分页插件 -->* <**dependency**>  <**groupId**>com.github.pagehelper</**groupId**>  <**artifactId**>pagehelper</**artifactId**> </**dependency**> |

# 代码生成器【快速开发】

## 2.1拷贝实体类

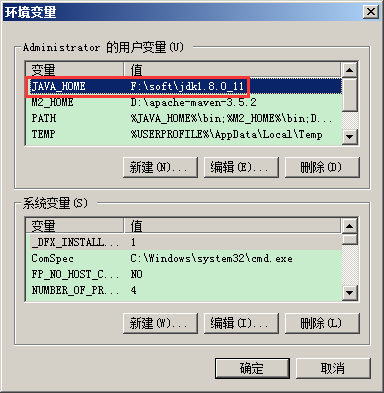
将【资料\实体类】下面所有的类 拷贝到pinyougou-pojo/src/main/java的com.pinyougou.pojo包下。

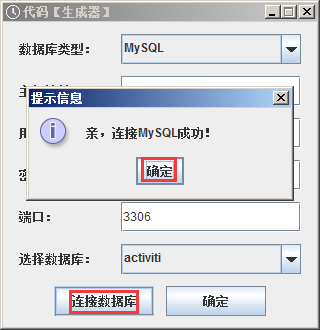
## 2.2代码生成器

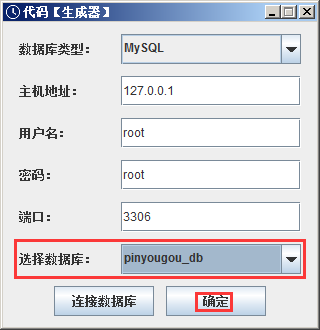
在【资料\生成器】文件夹：



在使用“代码生成器.exe”之前，需配置环境变量：**JAVA\_HOME**







## 2.3生成数据访问层



**注意：**

* **拷贝pinyougou\pinyougou-mapper\src\main\java\com\pinyougou\**

**Mapper目录下全部XxxMapper.java数据访问接口类到pinyougou-mapper/src/main/java/com/pinyougou/mapper/包下。**

* **拷贝pinyougou\pinyougou-mapper\src\main\resources\mappers目录下全部XxxMapper.xml映射文件到pinyougou-mapper/src/main/**

**resources/mappers/包下。**

## 2.4生成服务接口类



**注意：**

**第一步：**拷贝pinyougou\pinyougou-service\src\main\java\com\pinyougou

\service\目录下面全部的XxxService.java服务接口类到pinyougou-interface/src/main/java/com/pinyougou/service/包下。

**第二步：**pinyougou-sellergoods-service/src/main/java/com/pinyougou/

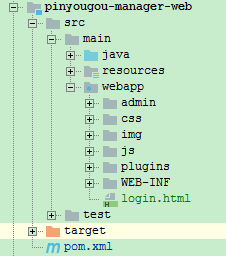
Sellergoods/service/impl/BrandServiceImpl.java实现接口中的方法：

|  |
| --- |
| @Service(interfaceName=**"com.pinyougou.service.BrandService"**) */\*\* 上面指定接口名，产生服务名，不然会用代理类的名称 \*/* @Transactional **public class** BrandServiceImpl **implements** BrandService {  */\*\* 注入数据访问接口代理对象 \*/* @Autowired  **private** BrandMapper **brandMapper**;  @Override  **public void** save(Brand brand) {   }  @Override  **public void** update(Brand brand) {  }  @Override  **public void** delete(Serializable id) {  }  @Override  **public void** deleteAll(Serializable[] ids) {  }  @Override  **public** Brand findOne(Serializable id) {  **return null**;  }  @Override  **public** List<Brand> findAll() {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(1, 10)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper**.selectAll();  }  });  System.***out***.println(**"总记录数："** + pageInfo.getTotal());  System.***out***.println(**"总页数："** + pageInfo.getPages());  **return** pageInfo.getList();  }  @Override  **public** List<Brand> findByPage(Brand brand, **int** page, **int** rows) {  **return null**;  } } |

# 使用域名访问后台系统

## 3.1拷贝资源

将“资料\运营商管理后台静态资源”下的全部文件拷贝到pinyougou-manager-web的src/main/webapp/目录下。



其中plugins文件夹中包括了bootstrap、jQuery等常用前端库，我们将在项目中用到。

**品优购-运营商后台请求地址：<http://localhost:9101/admin/index.html>**

## 3.2问题

现在访问的路径：http://127.0.0.1:9101/admin/index.html

存在的问题：

1. 开发环境和测试环境的ip不一样，每次环境变化时，都需要修改访问地址；
2. 页面加载资源文件，有可能使用url的全路径，一旦更换环境（ip变了），资源文件则无法加载；
3. ip地址没有记忆意义，不容易记忆，用户不通过ip访问，一般通过域名访问。

## 3.3域名访问

可以修改windows/system32/drivers/etc/hosts文件，实现域名和ip地址的映射。

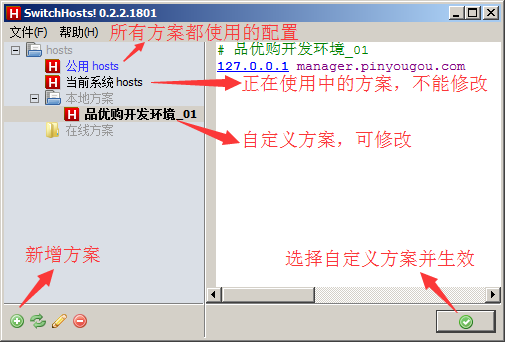
使用域名访问后的流程为：



## 3.4 SwitchHosts工具

手动修改hosts文件，如果以后ip多了，配置和切换都比较麻烦；一般可以使用工具进行管理。而SwitchHosts则可以管理hosts文件；SwitchHosts安装文件在“资料\SwitchHosts.exe”中。

功能说明：



问题：ip不用输入了，但是端口还在。可使用nginx解决。

# 4.Nginx反向代理服务器

## 4.1 Nginx 简介

* Nginx ("engine x") 是一个高性能的HTTP和反向代理服务器；
* 可实现负载均衡；
* Rewrite功能强大；
* 电商架构大部分都采用Nginx + Tomcat的架构。

反向代理（Reverse Proxy）方式是指以代理服务器来接受internet上的连接请求，然后将请求转发给内部网络上的服务器，并将从服务器上得到的结果返回给internet上请求连接的客户端，此时代理服务器对外就表现为一个反向代理服务器。

## 4.2安装与命令

拷贝资料中“nginx-1.11.13.zip”到任意一个磁盘路径解压，解压路径不能包含中文。

用cmd进入nginx解压所在的根目录：

* 启动： start nginx
* 停止：nginx -s stop
* 重新启动：nginx -s reload

注意：启动nginx后在任务管理器中要能看到两个nginx服务启动才是正确的。

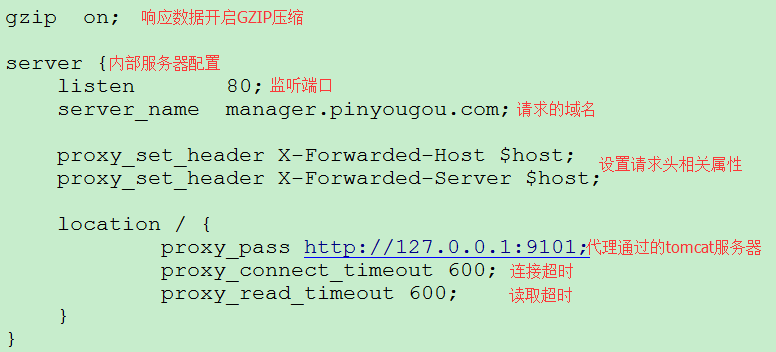


如果启动报错,查看日志文件发现80端口被占用；可以使用命令netstat -ano命令查看端口占用情况。再打开任务管理器，查看pid然后停止程序即可。

netstat -ano|findstr "80"

## 4.3相关配置

修改“nginx-1.11.13\conf\nginx.conf”文件，配置内部服务器：



对于运营商后台管理系统，可以添加如下nginx配置：

|  |
| --- |
| server {  listen 80;  server\_name manager.pinyougou.com;  proxy\_set\_header X-Forwarded-Host $host;  proxy\_set\_header X-Forwarded-Server $host;  location / {  proxy\_pass http://127.0.0.1:9101;  proxy\_connect\_timeout 600;  proxy\_read\_timeout 600;  }  } |

**请求地址：<http://manager.pinyougou.com/admin/index.html>**

**参考：【资料\Nginx反向代理流程图.png】**

# 5.品牌管理【查询全部】

## 5.1需求分析

实现品牌列表的查询（不用分页和条件查询）效果如下：



## 5.2前端代码

### **5.2.1引入JS**

修改pinyougou-manager-web/src/main/webapp/admin/brand.html，引入js：

|  |
| --- |
| *<!-- 引入vue与axios -->* <**script src="/js/vue.min.js"**></**script**> <**script src="/js/axios.min.js"**></**script**> |

### **5.2.2编写JS代码**

|  |
| --- |
| <**script type="text/javascript"**>  *// 窗口加载完* ***window***.onload = **function** () {  *// 创建Vue对象* **var** vue = **new Vue**({  **el** : **'#app'**, *// 元素绑定* **data** : { *// 数据模型* **dataList** : [] *// 定义数组，接收后台响应数据* },  **methods** : { *// 定义操作方法* search : **function**(){ *// 搜索方法  // 发送异步请求* **axios**.get(**"/brand/findAll"**).then(**function**(response){  *// 获取响应数据* **vue.dataList = response.data;**  });  }  },  created : **function** () { *// 创建生命周期(初始化方法)  // 调用搜索方法* **this**.search();  }  });  }; </**script**> |

### **5.2.3循环显示表格数据**

|  |
| --- |
| <**tbody**>  <**tr v-for="entity in dataList"**>  <**td**><**input type="checkbox"** ></**td**>   <**td**>{{entity.**id**}}</**td**>  <**td**>{{entity.**name**}}</**td**>  <**td**>{{entity.**firstChar**}}</**td**>  <**td class="text-center"**>   <**button type="button" class="btn bg-olive btn-xs"**  **data-toggle="modal" data-target="#editModal"**>修改</**button**>   </**td**>  </**tr**> </**tbody**> |

## 5.3后端代码

### **5.3.1控制器层**

pinyougou-manager-web/src/main/java/com/pinyougou/manager/controller/BrandController.java：



### **5.3.2服务实现层**

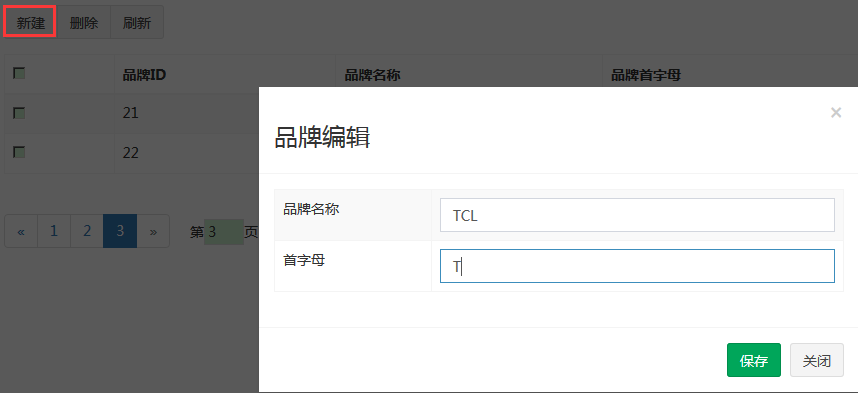
pinyougou-sellergoods-service/src/main/java/com/pinyougou/sellergoods/service/impl/BrandServiceImpl.java：

|  |
| --- |
| @Override **public** List<Brand> findAll() {  **return brandMapper**.selectAll(); } |

# 6.品牌管理【增加品牌】

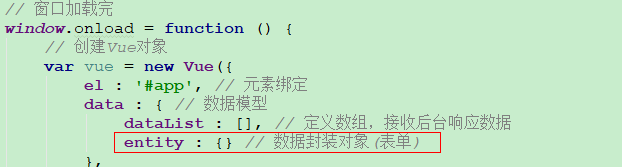
## 6.1需求分析

实现品牌增加功能



## **6.2前端代码**

### **6.2.1数据封装**



### **6.2.2表单&事件**

绑定表单元素，我们用v-model指令，绑定按钮的单击事件我们用@click

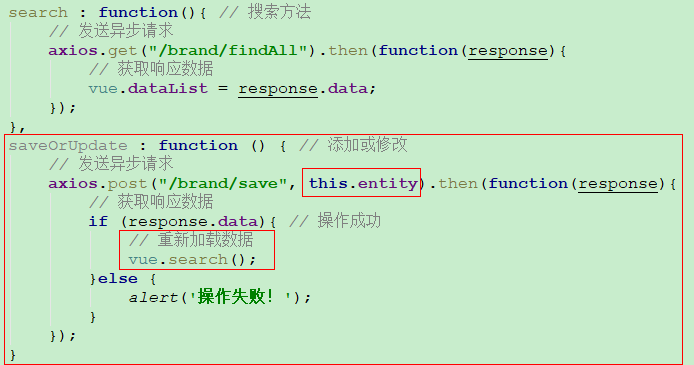
|  |
| --- |
| <**div class="modal-body"**>   <**table class="table table-bordered table-striped" width="800px"**>  <**tr**>  <**td**>品牌名称</**td**>  <**td**>  <**input class="form-control"  v-model="entity.name" placeholder="品牌名称"**/>  </**td**>  </**tr**>   <**tr**>  <**td**>首字母</**td**>  <**td**>  <**input class="form-control"  v-model="entity.firstChar" placeholder="首字母"**/>  </**td**>  </**tr**>   </**table**>  </**div**> <**div class="modal-footer"**>   <**button class="btn btn-success" data-dismiss="modal"  aria-hidden="true" @click="saveOrUpdate();"**>保存</**button**>  <**button class="btn btn-default" data-dismiss="modal"  aria-hidden="true"**>关闭</**button**> </**div**> |

注意：为了每次打开窗口没有遗留上次的数据，我们可以修改新建按钮，对entity变量进行清空操作：

|  |
| --- |
| <**button type="button" class="btn btn-default" title="新建"  data-toggle="modal" data-target="#editModal"  @click="entity={}"**>  <**i class="fa fa-file-o"**></**i**>新建 </**button**> |

### **6.2.3异步请求**

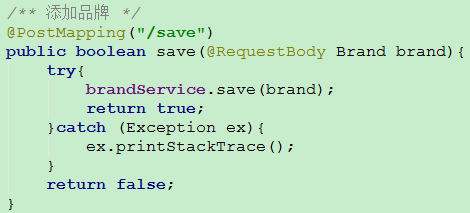
pinyougou-manager-web/src/main/webapp/admin/brand.html添加js代码：



## 6.3后端代码

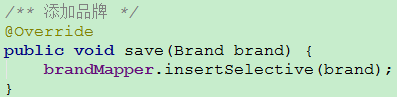
### **6.3.1控制器层**

pinyougou-manager-web/src/main/java/com/pinyougou/manager/controller/BrandController.java新增方法：



### **6.3.2服务实现层**

pinyougou-sellergoods-service/src/main/java/com/pinyougou/sellergoods/service/impl/BrandServiceImpl.java实现方法：



# 7.品牌管理【修改品牌】

## 7.1 需求分析

点击列表的修改按钮，弹出窗口，修改数据后点“保存”执行保存操作



## **7.2 前端代码**

### **7.2.1数据回显**

pinyougou-manager-web/src/main/webapp/admin/brand.html页面，vue的methods中，增加show方法实现数据回显：

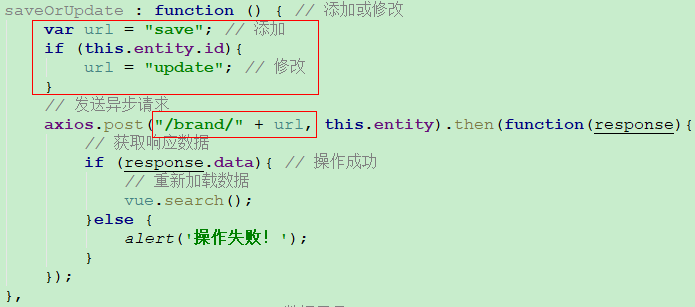
|  |
| --- |
| show : **function**(entity){ *// 数据回显  // 把entity对象转化成json字符串* **var** jsonStr = ***JSON***.stringify(entity);  *// 把json字符串转化成一个新的json对象* **this**.**entity** = ***JSON***.parse(jsonStr); } |

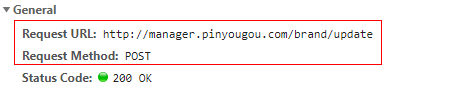
修改列表中的“修改”按钮，调用此方法执行表单数据绑定：

|  |
| --- |
| <**button type="button" class="btn bg-olive btn-xs"  data-toggle="modal"  data-target="#editModal"  @click="show(entity);"**>修改</**button**> |

### **7.2.2异步请求**

修改saveOrUpdate方法：

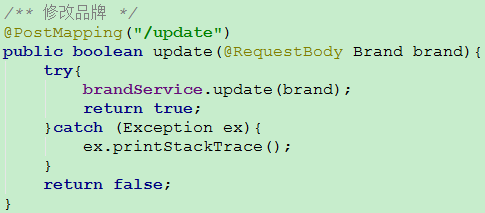




## 7.3 后端代码

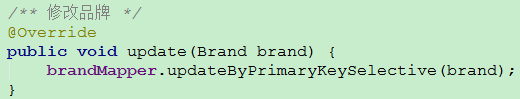
### **7.3.1控制器层**

pinyougou-manager-web/src/main/java/com/pinyougou/manager/controller/BrandController.java新增方法：



### **7.3.2服务实现层**

pinyougou-sellergoods-service/src/main/java/com/pinyougou/sellergoods/service/impl/BrandServiceImpl.java实现方法：



# 8.品牌管理【分页查询】

## 8.1需求分析

在品牌管理下方放置分页栏，实现分页功能：



## **8.2前端代码**

### **1.2.1分页组件**

在pinyougou-manager-web/src/main/webapp/admin/brand.htm引入分页组件

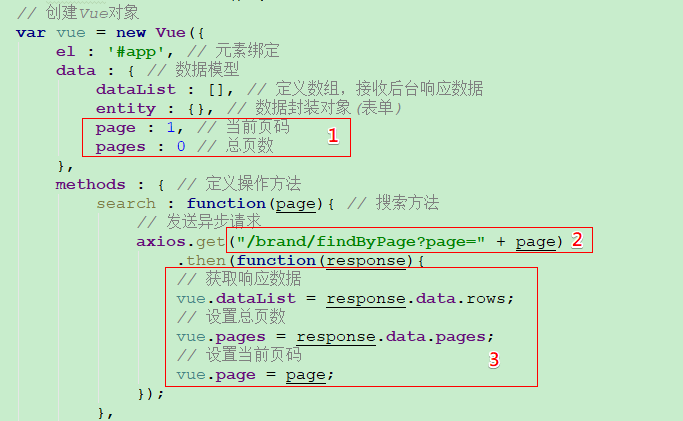
|  |
| --- |
| *<!-- 引入分页组件 -->* <**script src="/js/pagination.js"**></**script**> |

brand.html页面的表格table结束标签下放置分页组件：

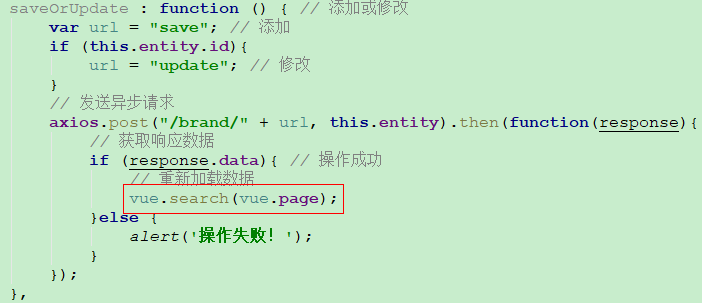
|  |
| --- |
| *<!-- 分页组件*  *pages:总页数*  *current:当前页码*  *navpage:分页方法 -->* <**pagination :pages="pages" :current="page" @navpage="search"**/> |

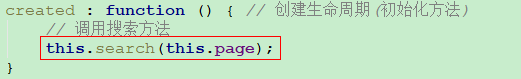
### **1.2.2 JS代码**

在pinyougou-manager-web/src/main/webapp/admin/brand.html中添加分页相关js代码：



**注意：修改saveOrUpdate与created方法**

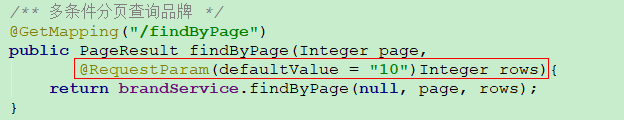




## **8.3后端代码**

### **1.3.1 控制器层**

pinyougou-manager-web/src/main/java/com/pinyougou/manager/controller/BrandController.java新增方法：



### **1.3.2 分页实体**

在pinyougou-common工程中创建com.pinyougou.common.pojo包，用于存放通用实体类，创建PageResult.java

|  |
| --- |
| */\*\* 分页实体 \*/* **public class** PageResult **implements** Serializable{  */\*\* 总页数 \*/* **private long pages**;  */\*\* 分页数据 \*/* **private** List<?> **rows**;  **public** PageResult(){}  **public** PageResult(**long** pages, List<?> rows) {  **this**.**pages** = pages;  **this**.**rows** = rows;  }  */\*\* setter and getter method \*/* **public long** getPages() {  **return pages**;  }  **public void** setPages(**long** pages) {  **this**.**pages** = pages;  }  **public** List<?> getRows() {  **return rows**;  }  **public void** setRows(List<?> rows) {  **this**.**rows** = rows;  } } |

### **1.3.3 服务接口层**

在pinyougou-interface的pom.xml配置依赖：

|  |
| --- |
| *<!-- pinyougou-common -->* <**dependency**>  <**groupId**>com.pinyougou</**groupId**>  <**artifactId**>pinyougou-common</**artifactId**>  <**version**>${project.version}</**version**> </**dependency**> |

pinyougou-interface/src/main/java/com.pinyougou.service/BrandService.java修改findByPage方法：



### **1.3.4 服务实现层**

pinyougou-sellergoods-service/src/main/java/com/pinyougou/sellergoods/service/impl/BrandServiceImpl.java中实现该方法：

|  |
| --- |
| */\*\* 多条件分页查询品牌 \*/* @Override **public** PageResult findByPage(Brand brand, **int** page, **int** rows) {  **try** {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(page, rows)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper**.selectAll();  }  });  **return new** PageResult(pageInfo.getPages(), pageInfo.getList());  }**catch** (Exception ex){  **throw new** RuntimeException(ex);  } } |

注意：PageHelper为MyBatis分页插件。

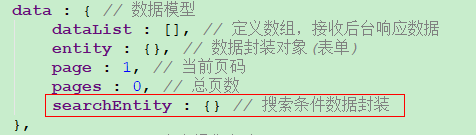
# 9.品牌管理【多条件分页查询】

## 9.1需求分析

实现品牌条件查询功能，输入品牌名称、首字母后查询，并分页。

## 9.2前端代码

brand.html定义搜索条件数据封装对象：



brand.html添加搜索条件数据绑定：

|  |
| --- |
| <**div class="box-tools pull-right"**>  <**div class="has-feedback"**>  *<!-- 条件查询区-->* <**table**>  <**tr**>  <**td**><**input class="form-control"   v-model="searchEntity.name" placeholder="品牌名称"**>  </**td**>  <**td**><**input class="form-control"   v-model="searchEntity.firstChar" placeholder="首字母"**>  </**td**>  <**td**>  <**button type="button" class="btn btn-default"   @click="search(1)"**>  <**i class="fa fa-trash-o"**></**i**> 查询  </**button**>  </**td**>  </**tr**>  </**table**>  </**div**> </**div**> |

在pinyougou-manager-web/src/main/webapp/admin/brand.html页面，修改search方法：

|  |
| --- |
| search : **function**(page){ *// 搜索方法  // 发送异步请求* **axios**.get(**"/brand/findByPage?page="** + page**,  {params : this.searchEntity}**)  .then(**function**(response){  *// 获取响应数据* vue.**dataList** = response.**data**.rows;  *// 设置总页数* vue.**pages** = response.**data**.**pages**;  *// 设置当前页码* vue.**page** = page;  }); } |

## 9.3后端代码

### **2.3.1控制器层**

修改pinyougou-manager-web/src/main/java/

com.pinyougou.manager.controller的BrandController.java分页方法：

|  |
| --- |
| */\*\* 多条件分页查询品牌 \*/* @GetMapping(**"/findByPage"**) **public** PageResult findByPage(Brand brand, Integer page,  @RequestParam(defaultValue = **"10"**)Integer rows){  */\*\* GET请求中文转码 \*/* **if** (StringUtils.*isNoneBlank*(brand.getName())) {  **try** {  brand.setName(**new** String(brand.getName()  .getBytes(**"ISO8859-1"**), **"UTF-8"**));  }**catch** (Exception ex){  ex.printStackTrace();  }  }  **return brandService**.findByPage(brand, page, rows); } |

### **2.3.2服务实现层**

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl的BrandServiceImpl.java分页实现方法：

|  |
| --- |
| @Override **public** PageResult findByPage(Brand brand, **int** page, **int** rows) {  **try** {  *// 开始分页* PageInfo<Brand> pageInfo = PageHelper.*startPage*(page, rows)  .doSelectPageInfo(**new** ISelect() {  @Override  **public void** doSelect() {  **brandMapper.findAll(brand);**  }  });  **return new** PageResult(pageInfo.getPages(), pageInfo.getList());  }**catch** (Exception ex){  **throw new** RuntimeException(ex);  } } |

### **2.3.3数据访问层**

pinyougou-mapper/src/main/java/com.pinyougou.mapper/BrandMapper.java增加方法：



修改pinyougou-mapper/src/main/resources/mappers/BrandMapper.xml

|  |
| --- |
| *<!-- 多条件查询品牌 -->* <**select id="findAll" resultType="Brand"**>  select \* from tb\_brand  <**where**>  *<!-- 品牌名称 -->* <**if test="name != null and name !=''"**>  name like CONCAT('%',#{name},'%')  </**if**>  *<!-- 品牌首字母 -->* <**if test="firstChar != null and firstChar != ''"**>  and first\_char = #{firstChar}  </**if**>  </**where**>  order by id asc </**select**> |

最终效果：



# 10.品牌管理【删除品牌】

## 10.1 需求分析

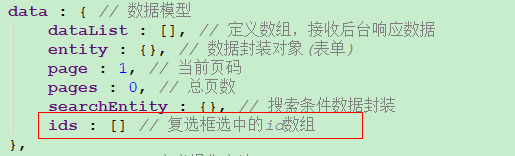
brand.html页面点击列表前的复选框，点击删除按钮，删除选中的品牌。



## 10.2 前端代码

### **10.2.1单选复选框**

在vue的data中定义ids数组，封装用户选中的复选框的值：

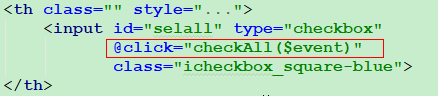


brand.html页面为复选框绑定数据：

|  |
| --- |
| <**td**><**input type="checkbox"  :value="entity.id" v-model="ids"**></**td**> |

### **10.2.2全选复选框**

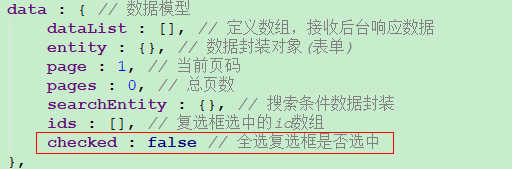
为全选复选框绑定点击事件：



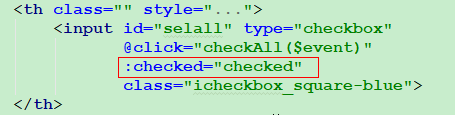
在vue的methods中定义事件方法：

|  |
| --- |
| checkAll : **function** (e) { *// 全选* **this**.**ids** = []; // 先清空数组  **if** (**e.target**.**checked**){ // 判断复选框是否选中  **for** (**var** i = 0; i < **this**.**dataList**.length; i++){  **this**.**ids**.push(**this**.**dataList**[i].**id**);  }  } } |

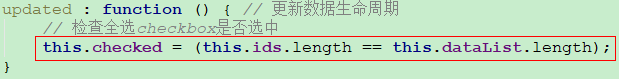
在vue的数据模型data中定义checked变量用来控制复选框是否选中：



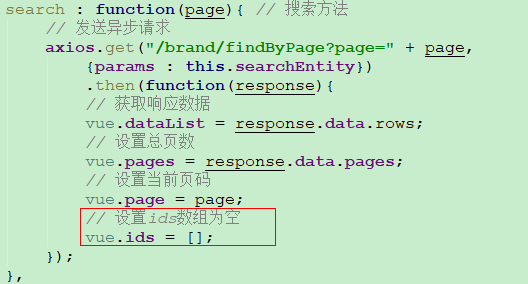
为全选复选框添加选中的属性值：



在vue的更新数据生命周期中判断全选复选框是否选中：



当点击页码、下一页时，上一页的复选框不能选中，需修改search方法：



### **10.2.3 异步请求**

brand.html页面为删除按钮绑定点击事件：

|  |
| --- |
| <**button type="button" class="btn btn-default" title="删除"  @click="del();"**>  <**i class="fa fa-trash-o"**></**i**> 删除 </**button**> |

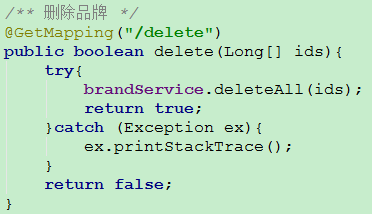
brand.html页面添加JS代码：

|  |
| --- |
| del : **function** () { *// 删除* **if** (**this**.**ids**.length > 0){  **axios**.get(**"/brand/delete?ids="** + **this**.**ids**).then(**function**(response){  **if** (response.**data**){  *// 计算当前页码(如果删除为最后一页查询上一页)* **var** page = vue.**page** == vue.**pages** && vue.**checked** ? vue.**page** - 1 : vue.**page**;  *// 重新加载数据* vue.search(page);  }**else**{  *alert*(**"删除失败！"**);  }  });  }**else** {  *alert*(**"请选择要删除的记录！"**);  } } |

## 10.3 后端代码

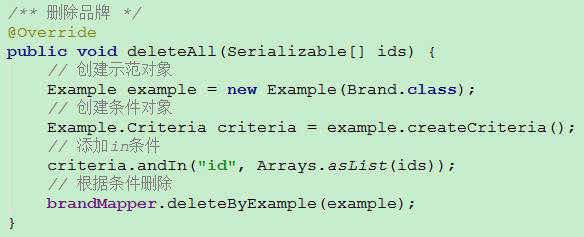
### **10.3.1控制层**

pinyougou-manager-web/src/main/java/com.pinyougou.manager.controller/BrandController.java：



### **10.3.2服务实现层**

pinyougou-sellergoods-service/src/main/java/com.pinyougou.sellergoods.service.impl/BrandServiceImpl.java：



## 10.4 单独js文件

最后，我们可以将js脚本的代码全部放到一个js文件里面，引入即可:

|  |
| --- |
| *<!-- 引入品牌vuejs -->* <**script src="/js/controller/brandController.js"**></**script**> |

# 附录：JS完整代码参考

|  |
| --- |
| *// 窗口加载完* ***window***.onload = **function** () {  *// 创建Vue对象* **var** vue = **new Vue**({  **el** : **'#app'**, *// 元素绑定* **data** : { *// 数据模型* **dataList** : [], *// 定义数组，接收后台响应数据* **entity** : {}, *// 数据封装对象(表单)* **page** : 1, *// 当前页码* **pages** : 0, *// 总页数* **searchEntity** : {}, *// 搜索条件数据封装* **ids** : [], *// 复选框选中的id数组* **checked** : **false** *// 全选复选框是否选中* },  **methods** : { *// 定义操作方法* search : **function**(page){ *// 搜索方法  // 发送异步请求* **axios**.get(**"/brand/findByPage?page="** + page,  {**params** : **this**.**searchEntity**})  .then(**function**(response){  *// 获取响应数据* vue.**dataList** = response.**data**.rows;  *// 设置总页数* vue.**pages** = response.**data**.**pages**;  *// 设置当前页码* vue.**page** = page;  *// 设置ids数组为空* vue.**ids** = [];  });  },  saveOrUpdate : **function** () { *// 添加或修改* **var** url = **"save"**; *// 添加* **if** (**this**.**entity**.**id**){  url = **"update"**; *// 修改* }  *// 发送异步请求* **axios**.post(**"/brand/"** + url, **this**.**entity**)  .then(**function**(response){  *// 获取响应数据* **if** (response.**data**){ *// 操作成功  // 重新加载数据* vue.search(vue.**page**);  }**else** {  *alert*(**'操作失败！'**);  }  });  },  show : **function**(entity){ *// 数据回显  // 把entity对象转化成json字符串* **var** jsonStr = ***JSON***.stringify(entity);  *// 把json字符串转化成一个新的json对象* **this**.**entity** = ***JSON***.parse(jsonStr);  },  checkAll : **function** (e) { *// 全选复选框* **this**.**ids** = []; *// 先清空数组* **if** (e.**target**.**checked**){ *// 判断复选框是否选中* **for** (**var** i = 0; i < **this**.**dataList**.length; i++){  **this**.**ids**.push(**this**.**dataList**[i].**id**);  }  }  },  del : **function** () { *// 删除* **if** (**this**.**ids**.length > 0){  **axios**.get(**"/brand/delete?ids="** + **this**.**ids**).then(**function**(response){  **if** (response.**data**){  *// 计算当前页码(如果删除为最后一页查询上一页)* **var** page = vue.**page** == vue.**pages**  && vue.**checked** ? vue.**page** - 1 : vue.**page**;  *// 重新加载数据* vue.search(page);  }**else**{  *alert*(**"删除失败！"**);  }  });  }**else** {  *alert*(**"请选择要删除的记录！"**);  }  }  },  created : **function** () { *// 创建生命周期(初始化方法)  // 调用搜索方法* **this**.search(**this**.**page**);  },  updated : **function** () { *// 更新数据生命周期  // 检查全选checkbox是否选中* **this**.**checked** = (**this**.**ids**.length == **this**.**dataList**.length);  }  }); }; |