1 学习目标

- 完成商品品牌的CRUD功能

2 品牌管理（查询）

2.1 vue项目

2.1.1 官网地址:

- 表格:https://v15.vuetifyjs.com/zh-Hans/components/data-tables
- 输入框:https://v15.vuetifyjs.com/zh-Hans/components/snackbars
- 按钮:https://v15.vuetifyjs.com/zh-Hans/components/buttons

2.1.2 在item包下新建MrBrand.vue

```
<template>
  <v-card>
    <v-card-title>
      <v-btn color="primary">新增品牌</v-btn>
      <v-spacer/>
      <v-text-field
        append-icon="search"
        label="搜索"
        single-line
        @keyup.enter="getData()"
        hide-details
        v-model="search"
      />
    </v-card-title>
    <v-divider/>
    <!-- :pagination.sync="pagination" 绑定分页属性,并将分页参数绑定给data.pagination
-->
    <!-- :total-items="total" 设置总条数,获取data.total中的值 -->
    <v-data-table
      :headers="headers"
      :items="desserts"
      :pagination.sync="pagination"
      :total-items="total"
      class="elevation-1"
    >
      <!-- 注意此处不能只能用官网的属性,应当参照Brand.vue的写法 -->
      <template slot="items" slot-scope="props">
        <td class="text-xs-center">{{ props.item.id }}</td>
        <td class="text-xs-center">{{ props.item.name }}</td>
        <!-- :src 给元素绑定src属性 属性的值为props.item.image -->
        <td class="text-xs-center"><img :src="props.item.image"/></td>
        <td class="text-xs-center">{{ props.item.letter }}</td>
      </template>
    </v-data-table>
  </v-card>
```

```
</template>
<script>
export default {
    name:"MrBrand",//组件名称
     data () {
      return {
        search:'',
        total:0,//总条数
        pagination: {},// 分页参数信息
        headers: [
          {
            text: 'id',
            align: 'center',
            sortable: true,
            value: 'id'
          },
          { text: 'name', value: 'name',align: 'center', },
          { text: 'image', value: 'image',align: 'center', },
          { text: 'letter', value: 'letter' ,align: 'center',}
        ],
        desserts: []//数据
      }
    },
    //组件加载完毕之后执行的函数
    mounted () {
        this.getList()//加载数据
    },
    watch:{
        //pagination-->page.pagination
        pagination () {//监控分页属性发生变化,在浏览器调试工具中vue模块可以看见数据发生变
化
            this.getList();
        }
    },
    methods:{
        getList () {
            this.$http.get('brand/getBrandInfo',{
                //查询传递参数-->分页以及排序信息
                params:{
                    page:this.pagination.page,
                    rows:this.pagination.rowsPerPage,
                    sort:this.pagination.sortBy,
                    order:this.pagination.descending
                }
            }).then(resp => {
                //给数据属性赋值
                this.desserts = resp.data.data.list;
                //给总数据条数赋值
                this.total = resp.data.data.total;
            }).catch(error => console.log(error))

        }
    }
}
</script>
```

2.1.3 index.js line : 27

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

2.2 common-core

2.2.1 pom.xml

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

2.2.2 MingruiOperation

```
public interface Search{}
```

2.2.3 在base包下新建BaseDTO

```
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import org.springframework.util.StringUtils;

/**
 * @ClassName BaseDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/18
 * @Version V1.0
 **/
@Data
@ApiModel(value = "BaseDTO用于数据传输,其他dto需要继承此类")
public class BaseDTO {

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

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

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

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


    @ApiModelProperty(hidden = true)
    public String getOrderByClause(){
        if(!StringUtils.isEmpty(sort))return sort + " " +
                order.replace("false","asc").replace("true","desc");
        return "";
    }
}
```

2.3 mingrui-shop-api

2.3.1 pom.xml

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

2.4 mingrui-shop-api-xxx

2.4.1 在com.baidu.shop下新建dto包

2.4.1.1 BrandDTO

```
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;

/**
 * @ClassName BrandDTO
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/18
 * @Version V1.0
 **/
@ApiModel(value = "品牌DTO")
@Data
public class BrandDTO extends BaseDTO {

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

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

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

2.4.2 entity包下新建BrandEntity

```
import lombok.Data;
import javax.persistence.Id;
import javax.persistence.Table;


/**
 * @ClassName BrandEntityy
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/18
 * @Version V1.0
 **/
@Table(name = "tb_brand")
@Data
public class BrandEntity {

    @Id
    private Integer id;

    private String name;

    private String image;

    private Character letter;
}
```

2.4.2 service包下新建BrandService

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

/**
 * @ClassName BrandService
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/18
 * @Version V1.0
 **/
@Api(tags = "品牌接口")
public interface BrandService {
    @ApiOperation(value = "获取品牌信息")
    @GetMapping(value = "brand/getBrandInfo")
    public Result<PageInfo<BrandEntity>> getBrandInfo(BrandDTO brandDTO);
}
```

2.5 mingrui-shop-service-xxx

2.5.1 在mapper包下新建BrandMapper

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

/**
 * @ClassName BrandMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/18
 * @Version V1.0
 **/
public interface BrandMapper extends Mapper<BrandEntity> {
}
```

2.5.2 在impl包下新建BrandServiceImpl

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

import javax.annotation.Resource;
import java.util.List;

/**
 * @ClassName BrandServiceImpl
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/18
 * @Version V1.0
 **/
@RestController
public class BrandServiceImpl extends BaseApiService implements BrandService {

    @Resource
    private BrandMapper brandMapper;

    @Override
    public Result<PageInfo<BrandEntity>> getBrandInfo(BrandDTO brandDTO) {
        //分页
        PageHelper.startPage(brandDTO.getPage(),brandDTO.getRows());

        //排序
        Example example = new Example(BrandEntity.class);

if(!StringUtils.isEmpty(brandDTO.getOrder()))example.setOrderByClause(brandDTO.g
etOrderByClause());

        if(!StringUtils.isEmpty(brandDTO.getName())){
            example.createCriteria().andLike("name","%" + brandDTO.getName() +
"%");
        }
        //查询
        List<BrandEntity> list = brandMapper.selectByExample(example);

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

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

3 品牌管理（新增）

- 模态框地址:https://v15.vuetifyjs.com/zh-Hans/components/dialogs
- form表单地址:https://v15.vuetifyjs.com/zh-Hans/components/forms

3.1 vue项目

3.1.1 新建MrBrandForm.vue

```
<template>
  <div>
    <v-card-text>
      <v-form v-model="valid">
        <v-text-field label="品牌名称" v-model="brand.name" />
        <v-text-field label="品牌首字母" v-model="brand.letter" />
        <!--选择商品分类,与品牌做绑定 one-to-many-->
        <v-cascader url="/category/list" required v-model="brand.category"
multiple label="商品分类" />
      </v-form>
    </v-card-text>
    <v-card-actions>
      <v-spacer></v-spacer>
      <v-btn color="green darken-1" flat @click="closeModel">取消</v-btn>
      <v-btn color="green darken-1" flat @click="submitForm">确定</v-btn>
    </v-card-actions>
  </div>
</template>
<script>
import Cascader from "./../../components/cascader/Cascader";
export default {
  name: "MrBrandForm",
  data() {
    return {
      valid: false,
      brand: {
        name: "",
        letter: "",
        category: [], //联动组件需要此属性,将获取到的参数绑定到此参数上
      },
    };
  },
  components: {
    Cascader,
  },
  methods: {
    closeModel() {
      //调用父级的自定义事件
      this.$emit("close");
    },
    submitForm() {
      let cates = this.brand.category.map((obj) => obj.id);//map函数可以返回一个新的数组
      let formData = this.brand;
      formData.categories = cates.join();
      this.$http
        .post("brand/addBrandInfo", formData)
        .then((resp) => {
          //关闭模态框
          this.closeModel();
          //提示新增成功
          this.$message.info("新增成功");
          //刷新列表
          this.$emit("refreshTable");
        })
        .catch((error) => console.log(error));
    },
  },
};
</script>
```

3.1.2 Casecader.vue需要修改113行

3.1.3 MrBrand.vue

```
<template>
  <!-- 参照Brand.vue -->
  <v-card>
    <v-card-title>
      <v-btn color="primary" @click="showModel">新增品牌</v-btn>

      <!--模态框-->
      <v-dialog v-model="dialog" persistent max-width="600">
        <v-card>
          <v-card-title class="headline">新增品牌</v-card-title>

          <!--自定义事件,关闭模态框 ,刷新列表-->
          <mr-brand-form @close="dialog = false" @refreshTable="getList"></mr-
brand-form>
        </v-card>
      </v-dialog>
      <!-- 此标签可以使按钮在左,输入框在右 -->
      <v-spacer />
      <!--
        输入框
        v-model="双向绑定"
        hide-details隐藏详情可以让按钮和输入框齐平
      -->
      <v-flex xs12 sm4>
        <v-text-field
          hide-details
          @keyup.enter="getList"
          v-model="brandName"
          label="按品牌名称进行搜索"
          append-icon="search"
        ></v-text-field>
      </v-flex>
    </v-card-title>
    <!-- :pagination.sync="pagination" 绑定分页属性,并将分页参数绑定给data.pagination
-->
    <!-- :total-items="total" 设置总条数,获取data.total中的值 -->
    <v-data-table
      :headers="headers"
      :items="desserts"
      class="elevation-1"
      :pagination.sync="pagination"
      :total-items="total"
    >
      <!-- 注意此处不能只能用官网的属性,应当参照Brand.vue的写法 -->
      <template slot="items" slot-scope="props">
        <td class="text-xs-center">{{ props.item.id }}</td>
        <td class="text-xs-center">{{ props.item.name }}</td>
        <!-- :src 给元素绑定src属性 属性的值为props.item.image -->
        <td class="text-xs-center">
          <img :src="props.item.image" />
        </td>
        <td class="text-xs-center">{{ props.item.letter }}</td>
        <!--操作按钮 删除 编辑-->
        <td class="text-xs-center">
          <v-btn flat icon color="blue">
            <v-icon>edit</v-icon>
          </v-btn>
          <v-btn flat icon color="pink">
            <v-icon>delete</v-icon>
          </v-btn>
        </td>
      </template>
    </v-data-table>
  </v-card>
</template>
<script>
import MrBrandForm from "./MrBrandForm";//导入form表单组件
export default {
  name: "MrBrand", //组件名称
  components: {
    MrBrandForm,//声明组件
  },
  data() {
    return {
      dialog: false,
      brandName: "",
      total: 0, //总条数
      pagination: {}, // 分页参数信息
      headers: [
        {
          text: "id",
          align: "center",
          sortable: true,
          value: "id",
        },
        { text: "name", value: "name", align: "center" },
        { text: "image", value: "image", align: "center" },
        { text: "letter", value: "letter", align: "center" },
        { text: "operation", value: "id", align: "center" },
      ],
      desserts: [], //数据
    };
  },
  //组件加载完毕之后执行的函数
  mounted() {
    this.getList(); //加载数据
  },
  watch: {
    //pagination-->page.pagination
    pagination() {
      //监控分页属性发生变化,在浏览器调试工具中vue模块可以看见数据发生变化
      this.getList();
    },
  },
  methods: {
    showModel() {
      this.dialog = true;
    },
    getList() {
      this.$http
        .get("brand/getBrandInfo", {
          //查询传递参数-->分页以及排序信息
          params: {
            page: this.pagination.page,
            rows: this.pagination.rowsPerPage,
            sort: this.pagination.sortBy,
            order: this.pagination.descending,
            name: this.brandName,
          },
        })
        .then((resp) => {
          //给数据属性赋值
          this.desserts = resp.data.data.list;
          //给总数据条数赋值
          this.total = resp.data.data.total;
        })
        .catch((error) => console.log(error));
    },
  },
};
</script>
```

3.2 common-core

3.2.1 在utils包下新建BaiduBeanUtil

```
import org.springframework.beans.BeanUtils;

/**
 * @ClassName BaiduBeanUtil
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/19
 * @Version V1.0
 **/
public class BaiduBeanUtil<T1,T2> {

    public static <T2> T2 copyProperties(Object source, Class<T2> clazz){

        if(null == source){
            return null;
        }
        if(null == clazz){
            return null;
        }

        try {
            T2 t2 = clazz.newInstance();
            BeanUtils.copyProperties(source,t2);
            return t2;
        } catch (InstantiationException e) {
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }
        return null;
    }

}
```

3.3 mingrui-shop-service-api-xxx

3.3.1 BrandDTO

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

3.3.2 entity包下新建CategoryBrandEntity

```
import lombok.Data;

import javax.persistence.Table;

/**
 * @ClassName CategoryBrandEntity
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/19
 * @Version V1.0
 **/
@Table(name = "tb_category_brand")
@Data
public class CategoryBrandEntity {

    private Integer categoryId;

    private Integer brandId;

}
```

3.3.3 BrandService

```
    @ApiOperation(value = "新增品牌")
    @PostMapping(value = "brand/addBrandInfo")
    public Result<JSONObject> save(@Validated({MingruiOperation.Add.class})
@RequestBody BrandDTO brandDTO);
```

3.4 mingrui-shop-service-xxx

3.4.1 在mapper包下新建CategoryBrandMapper

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

/**
 * @ClassName CategoryBrandMapper
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/19
 * @Version V1.0
 **/
//接口可以多继承,InsertListMapper用于批量新增
public interface CategoryBrandMapper extends
InsertListMapper<CategoryBrandEntity>,Mapper<CategoryBrandEntity> {
}
```

3.4.2 BrandServiceImpl

```
    @Resource
    private CategoryBrandMapper categoryBrandMapper;

        @Override
    public Result<JSONObject> save(BrandDTO brandDTO) {

        try {
            BrandEntity brandEntity = BaiduBeanUtil.copyProperties(brandDTO, 
BrandEntity.class);
            //通用mapper新增返回主键
            brandMapper.insertSelective(brandEntity);
            //绑定关系
            if(StringUtils.isEmpty(brandDTO.getCategories())){

                return this.setResultError("分类数据不能为空");
            }

            if(brandDTO.getCategories().contains(",")){

                List<CategoryBrandEntity> list = new ArrayList<>();

                String[] categoryArr = brandDTO.getCategories().split(",");

                Arrays.asList(categoryArr).stream().forEach(str -> {
                    CategoryBrandEntity categoryBrandEntity = new 
CategoryBrandEntity();
                    categoryBrandEntity.setBrandId(brandEntity.getId());
                    categoryBrandEntity.setCategoryId(Integer.parseInt(str));
                    list.add(categoryBrandEntity);
                });

                categoryBrandMapper.insertList(list);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return this.setResultSuccess();
    }
```

4 新增遗留的问题

4.1 品牌首字母自动识别

4.1.1 vue项目

4.1.1.1 将MrBrandForm.vue组件中所有关于首字母的内容删除掉

```
<template>
  <div>
    <v-card-text>
      <v-form v-model="valid">
        <v-text-field label="品牌名称" v-model="brand.name" />
        <!--选择商品分类,与品牌做绑定 one-to-many-->
        <v-cascader url="/category/list" required v-model="brand.category"
multiple label="商品分类" />
      </v-form>
    </v-card-text>
    <v-card-actions>
      <v-spacer></v-spacer>
      <v-btn color="green darken-1" flat @click="closeModel">取消</v-btn>
      <v-btn color="green darken-1" flat @click="submitForm">确定</v-btn>
    </v-card-actions>
  </div>
</template>
<script>
import Cascader from "./../../components/cascader/Cascader";
export default {
  name: "MrBrandForm",
  data() {
    return {
      valid: false,
      brand: {
        name: "",
        category: [], //联动组件需要此属性,将获取到的参数绑定到此参数上
        categories: [], //新增需要此参数
      },
    };
  },
  components: {
    Cascader,
  },
  methods: {
    closeModel() {
      //调用父级的自定义事件
      this.$emit("close");
    },
    submitForm() {
      let cates = this.brand.category.map((obj) => obj.id);
      this.brand.categories = cates.join(); //map函数可以返回一个新的数组
      this.$http
        .post("brand/addBrandInfo", this.brand)
        .then((resp) => {
          //关闭模态框
          this.closeModel();
          //提示新增成功
          this.$message.info("新增成功");
          //刷新列表
          this.$emit("refreshTable");
        })
        .catch((error) => console.log(error));
    },
  },
};
</script>
```

4.1.2 common-core

4.1.2.1 pom.xml

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

4.1.2.2 在utils包下新建PinyinUtil

```
import net.sourceforge.pinyin4j.PinyinHelper;
import net.sourceforge.pinyin4j.format.HanyuPinyinOutputFormat;
import net.sourceforge.pinyin4j.format.HanyuPinyinToneType;

import java.util.regex.Matcher;
import java.util.regex.Pattern;

/**
 * @ClassName PinyinUtil
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/19
 * @Version V1.0
 **/
public class PinyinUtil {

    public static final Boolean TO_FUUL_PINYIN = true;

    public static final Boolean TO_FIRST_CHAR_PINYIN = false;
    /**
     * 获取汉字首字母或全拼大写字母
     *
     * @param chinese 汉字
     * @param isFull  是否全拼 true:表示全拼 false表示：首字母
     * @return 全拼或者首字母大写字符窜
     */
    public static String getUpperCase(String chinese, boolean isFull) {
        return convertHanzi2Pinyin(chinese, isFull).toUpperCase();
    }

    /**
     * 获取汉字首字母或全拼小写字母
     *
     * @param chinese 汉字
     * @param isFull  是否全拼 true:表示全拼 false表示：首字母
     * @return 全拼或者首字母小写字符窜
     */
    public static String getLowerCase(String chinese, boolean isFull) {
        return convertHanzi2Pinyin(chinese, isFull).toLowerCase();
    }

    /**
     * 将汉字转成拼音
     * <p>
     * 取首字母或全拼
     *
     * @param hanzi  汉字字符串
     * @param isFull 是否全拼 true:表示全拼 false表示：首字母
```

```java
     * @return 拼音
     */
    private static String convertHanzi2Pinyin(String hanzi, boolean isFull) {
        /***
         * ^[\u2E80-\u9FFF]+$ 匹配所有东亚区的语言
         * ^[\u4E00-\u9FFF]+$ 匹配简体和繁体
         * ^[\u4E00-\u9FA5]+$ 匹配简体
         */
        String regExp = "^[\u4E00-\u9FFF]+$";
        StringBuffer sb = new StringBuffer();
        if (hanzi == null || "".equals(hanzi.trim())) {
            return "";
        }
        String pinyin = "";
        for (int i = 0; i < hanzi.length(); i++) {
            char unit = hanzi.charAt(i);
            //是汉字，则转拼音
            if (match(String.valueOf(unit), regExp)) {
                pinyin = convertSingleHanzi2Pinyin(unit);
                if (isFull) {
                    sb.append(pinyin);
                } else {
                    sb.append(pinyin.charAt(0));
                }
            } else {
                sb.append(unit);
            }
        }
        return sb.toString();
    }

    /**
     * 将单个汉字转成拼音
     *
     * @param hanzi 汉字字符
     * @return 拼音
     */
    private static String convertSingleHanzi2Pinyin(char hanzi) {
        HanyuPinyinOutputFormat outputFormat = new HanyuPinyinOutputFormat();
        outputFormat.setToneType(HanyuPinyinToneType.WITHOUT_TONE);
        String[] res;
        StringBuffer sb = new StringBuffer();
        try {
            res = PinyinHelper.toHanyuPinyinStringArray(hanzi, outputFormat);
            sb.append(res[0]);//对于多音字，只用第一个拼音
        } catch (Exception e) {
            e.printStackTrace();
            return "";
        }
        return sb.toString();
    }

    /***
     * 匹配
     * <P>
     * 根据字符和正则表达式进行匹配
     *
     * @param str 源字符串
```

```
      * @param regex 正则表达式
      *
      * @return true：匹配成功  false：匹配失败
      */
    private static boolean match(String str, String regex) {
        Pattern pattern = Pattern.compile(regex);
        Matcher matcher = pattern.matcher(str);
        return matcher.find();
    }

}
```

4.1.3 mingrui-shop-service-xxx

4.1.3.1 BrandServiceImpl

```
    @Override
    public Result<JSONObject> save(BrandDTO brandDTO) {

        //str.charAt(0)获取当前字符串第一个字符
        //String.valueOf()将对象转换为String类型的字符串
        //
//        char c = brandDTO.getName().charAt(0);
//        String s = String.valueOf(c);
//        String upperCase = PinyinUtil.getUpperCase(s,
PinyinUtil.TO_FIRST_CHAR_PINYIN);
//        char c1 = upperCase.charAt(0);
//        brandDTO.setLetter(c1);

 brandDTO.setLetter(PinyinUtil.getUpperCase(String.valueOf(brandDTO.getName().ch
arAt(0)),PinyinUtil.TO_FIRST_CHAR_PINYIN).charAt(0));

        try {
            BrandEntity brandEntity = BaiduBeanUtil.copyProperties(brandDTO,
BrandEntity.class);
            //通用mapper新增返回主键
            brandMapper.insertSelective(brandEntity);
            //绑定关系
            if(StringUtils.isEmpty(brandDTO.getCategories())){

                return this.setResultError("分类数据不能为空");
            }

            if(brandDTO.getCategories().contains(",")){

                List<CategoryBrandEntity> list = new ArrayList<>();

                String[] categoryArr = brandDTO.getCategories().split(",");

                Arrays.asList(categoryArr).stream().forEach(str -> {
                    CategoryBrandEntity categoryBrandEntity = new
CategoryBrandEntity();
                    categoryBrandEntity.setBrandId(brandEntity.getId());
                    categoryBrandEntity.setCategoryId(Integer.parseInt(str));
                    list.add(categoryBrandEntity);
                });
                categoryBrandMapper.insertList(list);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        return this.setResultSuccess();
    }
```

4.2 清空form和表单验证的问题

4.2.1 MrBrand.vue

```
          <!--绑定dialog属性,方便子级组件获取模态框的状态-->
          <mr-brand-form @close="dialog = false" :dialog="dialog"
@refreshTable="getList"></mr-brand-form>
```

4.2.2 MrBrandFrom.vue

```
<template>
  <div>
    <v-card-text>
      <!-- ref="form表单名称" 可以在this.$form获取当前表单的属性-->
      <v-form v-model="valid" ref="form">
        <!-- required 必填,只是一个*号的效果而已
         :rules="nameRules" 绑定验证规则-->
        <v-text-field label="品牌名称" required :rules="nameRules" v-
model="brand.name" />
        <!--选择商品分类,与品牌做绑定 one-to-many-->
        <v-cascader url="/category/list" required v-model="brand.category"
multiple label="商品分类" />
      </v-form>
    </v-card-text>
    <v-card-actions>
      <v-spacer></v-spacer>
      <v-btn color="green darken-1" flat @click="closeModel">取消</v-btn>
      <v-btn color="green darken-1" flat @click="submitForm">确定</v-btn>
    </v-card-actions>
  </div>
</template>
<script>
import Cascader from "./../../components/cascader/Cascader";
export default {
  name: "MrBrandForm",
  props: {
    //父组件传递过来的模态框状态
    dialog: Boolean,
  },
  data() {
    return {
      //开启表单验证
      valid: true,
      //品牌名称的验证规则
      nameRules: [
        //v 是当前值
        //!!v-->不是null
        //如果是null的话输出:品牌名称不能为空
        //浏览器控制台false || 'aaaa'查看结果
        v => !!v || "品牌名称不能为空",
        v => (v && v.length > 1 && v.length <= 10) || "品牌名称长度必须大于1且小于10",
      ],
      brand: {
        name: "",
        category: [], //联动组件需要此属性,将获取到的参数绑定到此参数上
      },
    };
  },
  components: {
    Cascader,
  },
  watch: {
    //监控dialog的值
    dialog() {
      if (this.dialog) {
        //将this.brand.id置为null,先修改再新增,之前回显的id值没有清空掉
        if (this.brand.id) this.brand.id = null;
        //模态框开启重置表单
        this.$refs.form.reset();
      }
    },
  },
  methods: {
    closeModel() {
      //调用父级的自定义事件
      this.$emit("close");
    },
    submitForm() {
      if (this.$refs.form.validate()) {//表单验证成功后再提交请求
        let cates = this.brand.category.map((obj) => obj.id);//map函数可以返回一个新的数组
        let formData = this.brand;
        formData.categories = cates.join();
        this.$http
          .post("brand/addBrandInfo", this.brand)
          .then((resp) => {
            //关闭模态框
            this.closeModel();
            //提示新增成功
            this.$message.info("新增成功");
            //刷新列表
            this.$emit("refreshTable");
          })
          .catch((error) => console.log(error));
      }
    },
  },
};
</script>
```

4.2.3 Casedader.vue(bug) line:162

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

            this.selected = val.map(o => {
              return {
                label: o[this.itemText],
                value: o[this.itemValue]
              }
            })
          } else {

            if(typeof val == 'object'){
              this.selected = [val[this.itemText]]
            }

          }
```

4.3 图片上传

图片上传说白了就是文件上传

有可能每个服务都会有涉及到文件上传的操作

所以我们把文件上传抽出来作为一个公共的文件上传服务

4.3.1 vue项目

4.3.1.1 Upload.vue line:88&89

```
          //文件长传成功后回显后台返回的图片
          this.dialogImageUrl = file.response.data;
          //给input表单赋值-->提交表单
          this.$emit("input", file.response.data)
```

4.3.1.2 MrBrandForm.vue

```
<template>
  <div>
    <v-card-text>
      <!-- ref="form表单名称" 可以在this.$form获取当前表单的属性-->
      <v-form v-model="valid" ref="form">
        <!-- required 必填,只是一个*号的效果而已
        :rules="nameRules" 绑定验证规则-->
        <v-text-field label="品牌名称" required :rules="nameRules" v-
model="brand.name" />
        <!--选择商品分类,与品牌做绑定 one-to-many-->
        <v-cascader url="/category/list" required v-model="brand.category"
multiple label="商品分类" />
        <!--文件上传-->
        <!--布局,不加此标签模态框样式会乱,因为联动组件占用了一部分位置-->
        <v-layout row>
          <!--栅格布局 将一行分成12分 xs3:当前组件占3分-->
          <v-flex xs3>
```

```
            <span style="font-size: 16px; color: #444">品牌LOGO：</span>
          </v-flex>
          <v-flex>
            <v-upload
              v-model="brand.image"
              url="/upload"
              :multiple="false"
              :pic-width="250"
              :pic-height="90"
            />
          </v-flex>
        </v-layout>
      </v-form>
    </v-card-text>
    <v-card-actions>
      <v-spacer></v-spacer>
      <v-btn color="green darken-1" flat @click="closeModel">取消</v-btn>
      <v-btn color="green darken-1" flat @click="submitForm">确定</v-btn>
    </v-card-actions>
  </div>
</template>
<script>
import Cascader from "./../../components/cascader/Cascader";
export default {
  name: "MrBrandForm",
  props: {
    //父组件传递过来的模态框状态
    dialog: Boolean,
  },
  data() {
    return {
      //开启表单验证
      valid: true,
      //品牌名称的验证规则
      nameRules: [
        //v 是当前值
        //!!v-->不是null
        //如果是null的话输出:品牌名称不能为空
        //浏览器控制台false || 'aaaa'查看结果
        (v) => !!v || "品牌名称不能为空",
        (v) =>
          (v && v.length > 1 && v.length <= 10) ||
          "品牌名称长度必须大于1且小于10",
      ],
      brand: {
        name: "",
        image:'',
        category: [], //联动组件需要此属性,将获取到的参数绑定到此参数上
        categories: [], //新增需要此参数
      },
    };
  },
  components: {
    Cascader,
  },
  watch: {
    //监控dialog的值
    dialog() {
      if (this.dialog) {
        //模态框开启重置表单
        this.$refs.form.reset();
      }
    },
  },
  methods: {
    closeModel() {
      //调用父级的自定义事件
      this.$emit("close");
    },
    submitForm() {
      if (this.$refs.form.validate()) {
        //表单验证成功后再提交请求
        let cates = this.brand.category.map((obj) => obj.id);
        this.brand.categories = cates.join(); //map函数可以返回一个新的数组
        this.$http
          .post("brand/addBrandInfo", this.brand)
          .then((resp) => {
            //关闭模态框
            this.closeModel();
            //提示新增成功
            this.$message.info("新增成功");
            //刷新列表
            this.$emit("refreshTable");
          })
          .catch((error) => console.log(error));
      }
    },
  },
};
</script>
```

4.3.2 在mingrui-shop-basics下新建mingrui-shop-basic-upload-server

4.3.2.1 pom.xml

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

        <dependency>
            <groupId>com.baidu</groupId>
            <artifactId>mingrui-shop-common-core</artifactId>
            <version>1.0-SNAPSHOT</version>
        </dependency>
    </dependencies>
```

4.3.2.2 application.yml

```
server:
  port: 8200
spring:
  application:
    name: upload-server
eureka:
  client:
    service-url:
      defaultZone: http://localhost:8761/eureka

mingrui:
  upload:
    path:
      windows: E:\\images
      linux: /shenyaqi/images
    img:
      host: http://image.mrshop.com
```

4.3.2.3 启动类

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

/**
 * @ClassName RunUploadServerApplication
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/19
 * @Version V1.0
 **/
@SpringBootApplication
@EnableEurekaClient
public class RunUploadServerApplication {
    public static void main(String[] args) {
        SpringApplication.run(RunUploadServerApplication.class);
    }
}
```

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

4.3.2.5 在包下新建UploadController

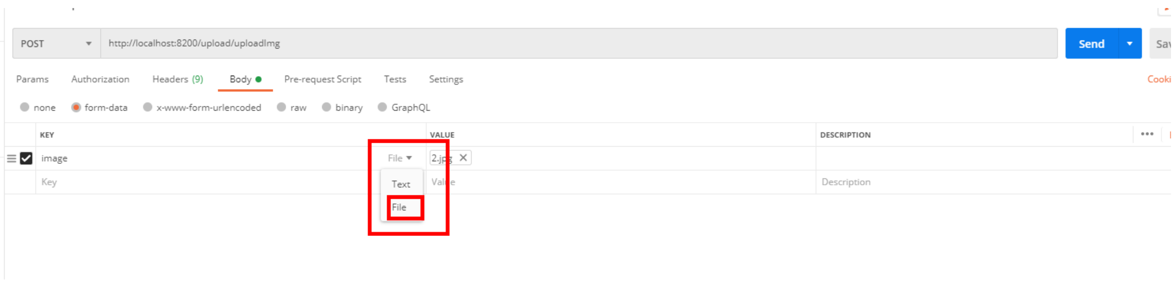
```

import com.baidu.shop.base.BaseApiService;
import com.baidu.shop.base.Result;
import com.baidu.shop.status.HTTPStatus;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.IOException;
import java.util.UUID;

/**
 * @ClassName UploadController
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/19
 * @Version V1.0
 **/
@RestController
@RequestMapping(value = "upload")
public class UploadController extends BaseApiService {

    //linux系统的上传目录
    @Value(value = "${mingrui.upload.path.windows}")
    private String windowsPath;

    //window系统的上传目录
    @Value(value = "${mingrui.upload.path.linux}")
    private String linuxPath;

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

    @PostMapping
    public Result<String> uploadImg(@RequestParam MultipartFile file) {

        if(file.isEmpty()) return this.setResultError("上传的文件为空");//判断上传的
文件是否为空

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

        String path = "";
        String os = System.getProperty("os.name").toLowerCase();
        if(os.indexOf("win") != -1){
            path = windowsPath;
        }else if(os.indexOf("lin") != -1){
            path = linuxPath;
        }

        filename = UUID.randomUUID() + filename;//防止文件名重复

        //创建文件 路径+分隔符(linux和window的目录分隔符不一样)+文件名
        File dest = new File(path + File.separator + filename);

        //判断文件夹是否存在,不存在的话就创建
        if(!dest.getParentFile().exists()) dest.getParentFile().mkdirs();

        try {
            file.transferTo(dest);//上传
        } catch (IllegalStateException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return this.setResult(HTTPStatus.OK,"upload success!!!",imgHost + "/" +
filename);//将文件名返回页面用于页面回显
    }
}
```

4.3.2.6 新建包:com.baidu.global

4.3.2.7 GlobalCorsConfig

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

/**
 * @ClassName GlobalCorsConfig
 * @Description: TODO
 * @Author shenyaqi
 * @Date 2020/8/28
 * @Version V1.0
 **/
@Configuration
public class GlobalCorsConfig {

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

4.3.2.8 使用postman测试上传


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

Body: form-data

KEY		VALUE	DESCRIPTION
image	File	2.jpg	

4.3.3 为什么需要在将图片放在本地目录?

如果将图片放在项目目录的话,前台对项目的访问量就会非常的大,造成了没有必要的服务器压力

4.3.4 将图片放在本地目录浏览器还可以访问吗?

不可以

4.3.5 我们将利用nginx来做代理服务,代理本地目录

4.3.5.1 hosts文件中新增

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

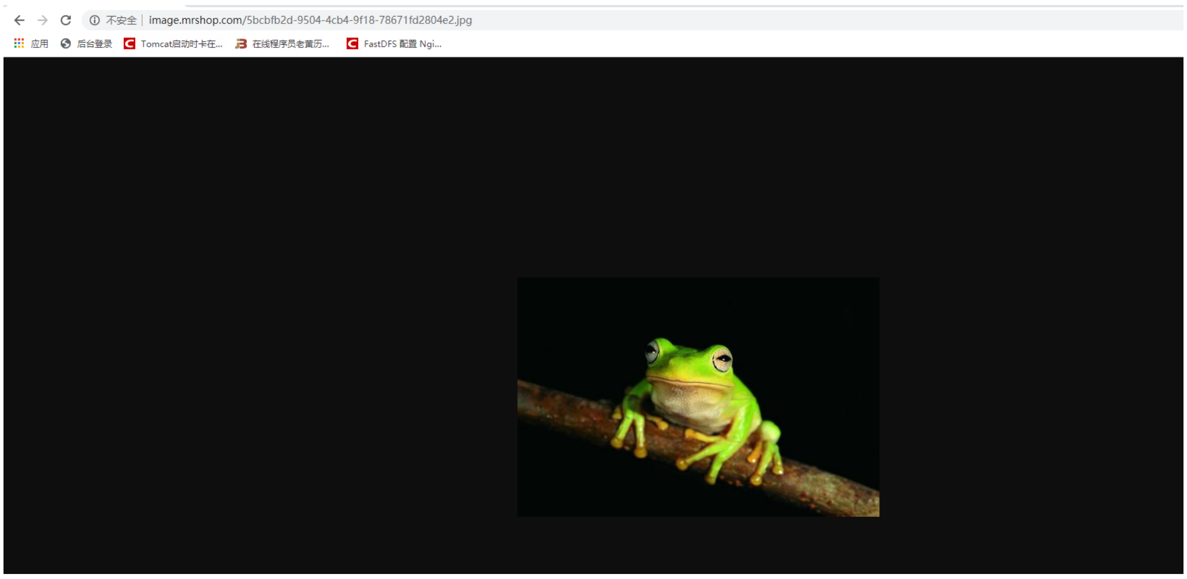
4.3.5.2 nginx-home/conf/nginx.conf新增

```
      server {
        listen       80;
        server_name  image.mrshop.com;

        proxy_set_header X-Forwarded-Host $host;
        proxy_set_header X-Forwarded-Server $host;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

          location ~ .*\.(gif|jpg|pdf|jpeg|png)$
        {
           #root D:/nginx-1.15.5/temp/images/;#指定图片存放路径(可以放在nginx文件夹路
径里也可以放其他p盘)
            root E:\images;
        }

        location / {
             root   html;
             index  index.html index.htm;
        }
    }
```

4.3.5.3 重启nginx

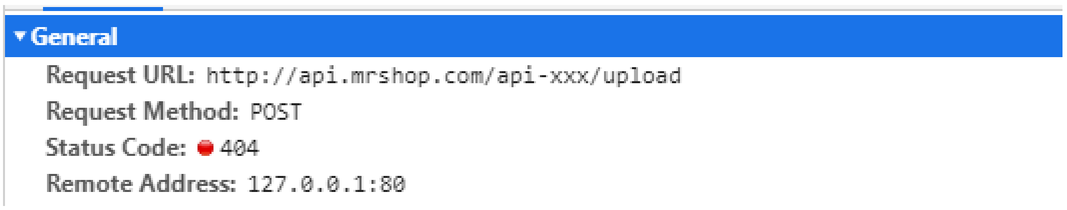
```
nginx.exe -s reload
```

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



4.3.5.4 启动vue项目测试能否成功

不能成功,还是请求了网关,网关会吧api-xxx的请求转发到service-xxx的服务中,而service-xxx并没有upload服务



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

所以我想只要包含/upload请求的我都不进行路由

5.3.5.5 zuul的application.yml

```
zuul:
  # 声明路由
  routes:
    # 路由名称
    api-xxx:
      # 声明将所有以/api-ribbon/的请求都转发到eureka-ribbon的服务中
      path: /api-xxx/**
      serviceId: xxx-service
  # 启用重试
  retryable: true
  # 包含此路径的不进行路由
  ignored-patterns: /upload/**
  # 忽略上传服务
  ignored-services:
    -upload-server
```

重启zuul服务测试,还是不行,为什么?因为这个配置没有从根本上去解决问题,虽然忽略了/upload的请求,但是/上传的请求中依然包含/api-xxx

所以我们需要借助nginx来帮助我们做一些事:

只要包含/api-xxx/upload的请求都将/api-xxx去掉,

为什么是在nginx里面处理?

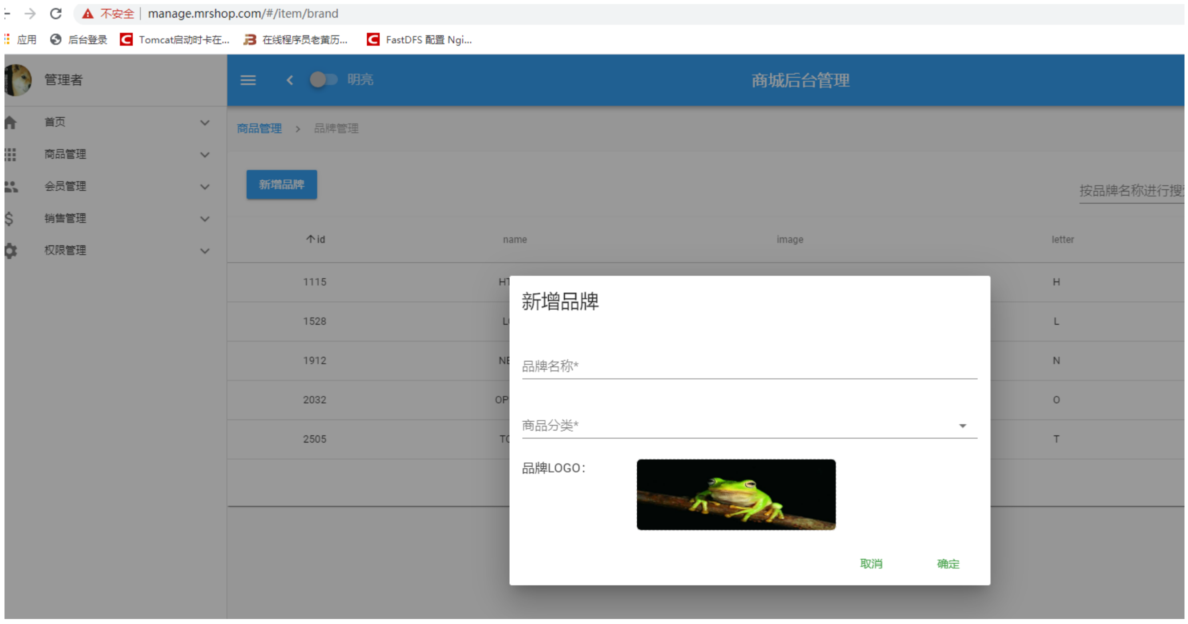
因为浏览器发起请求的时候第一个经过nginx服务

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

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

            rewrite "^/api-xxx/(.*)$" /$1 break;
        }
```

5.3.5.7 重启nginx

重新测试



4 品牌管理(修改)

4.1 回显

4.1.1 vue项目

4.1.1.1 MrBrand.vue

```
<template>
  <v-card>
    <v-card-title>
      <!--新增按钮-->
      <v-btn depressed color="primary" @click="openModel">新增</v-btn>

      <!--模态框-->
      <v-layout row justify-center>
        <v-dialog v-model="dialog" persistent max-width="500">
          <v-card>
```

```
            <v-card-title class="headline">{{ isEdit?'修改':'新增' }}品牌</v-card-title>

            <!--form表单+操作按钮-->
            <!--将模态框的状态传递到子组件
            @closeModel="closeModel" 关闭模态框的函数中需要重置this.brandDetail,所以需要新建closeModel方法
            :isEdit="isEdit" 将新增还是修改的状态传递给子级组件
            :brandDetail="brandDetail" 修改回显的数据
            -->
            <mr-brand-form @RefreshList="getBrandInfo" @closeModel="closeModel" :isEdit="isEdit" :brandDetail="brandDetail" :dialog="dialog"></mr-brand-form>

          </v-card>
        </v-dialog>
      </v-layout>
      <!--分隔组件-->
      <v-spacer />
      <!--@keyup.enter="searchByBrandName"-->
      <!--输入框-->
      <v-text-field
        v-model="brandName"
        label="通过品牌名称搜索"
        type="text"
        hide-details
        @keyup.enter="getBrandInfo"
        append-icon="search"
      ></v-text-field>
    </v-card-title>

    <!--
        :headers 表格头信息
        :items 数据
        :pagination.sync 分页
        :total-items 总条数
        :loading 加载数据效果
    -->
    <v-data-table
      :headers="headers"
      :items="desserts"
      :pagination.sync="pagination"
      :total-items="total"
      :loading="loading"
      class="elevation-1"
    >
      <!-- <template v-slot:items="props"> -->
      <!-- slot插槽 -->
      <template slot="items" slot-scope="props">
        <!--text-xs-cente 居中[设置列的位置]-->
        <td class="text-xs-center">{{ props.item.id }}</td>
        <td class="text-xs-center">{{ props.item.name }}</td>
        <td class="text-xs-center">
          <img width="100" :src="props.item.image" />
        </td>
        <td class="text-xs-center">{{ props.item.letter }}</td>
         <td class="text-xs-center">
             <v-btn flat icon @click="editBrand(props.item)" color="blue">
              <v-icon>edit</v-icon>
            </v-btn>
            <v-btn flat icon color="red">
              <v-icon>delete</v-icon>
            </v-btn>
          </td>
        </template>
      </v-data-table>
    </v-card>
</template>
<script>
import MrBrandForm from "./MrBrandForm"; //导入组件
export default {
  name: "MrBrand", //组件名
  components: {
    MrBrandForm,
  },
  watch: {
    //监控分页数据是否发生变化
    //如果发生变化应该去请求后台
    pagination() {
      //查询列表
      this.getBrandInfo();
    },
  },
  mounted() {
    //声明周期函数(勾子函数)-->组件加载完成之后执行,并且只会执行一次
    //查询列表
    this.getBrandInfo();
  },
  methods: {
    closeModel () {
      this.dialog = false;
      this.brandDetail = {};//关闭模态框重置回显数据为空
    },
    editBrand (obj) {
      //回显
      this.dialog = true;//打开模态框
      this.brandDetail = obj;//赋值 obj-->需要回显的数据
      this.isEdit = true;//修改状态为true

    },
    openModel() {

      this.isEdit = false;//修改状态为false
      this.dialog = true;
    },

    //多余的操作
    // searchByBrandName(){
    //   this.getBrandInfo();
    // },
    getBrandInfo() {
      //查询列表的方法
      this.$http
        .get("brand/getBrandInfo", {
          //传递的参数
          params: {
            page: this.pagination.page, //当前页
            rows: this.pagination.rowsPerPage, //每页显示多少条数据
            sort: this.pagination.sortBy, //通过那个字段排序
            desc: this.pagination.descending, //排序的方式(true:desc fase:asc)
            name: this.brandName, //条件查询
          },
        })
        .then((resp) => {
          //成功回调
          if (resp.data.code == 200) {
            this.desserts = resp.data.data.list; //!!!!!!!!!!
            this.total = resp.data.data.total;
            this.loading = false; //数据在家完成之后需需要将loding的值设置为false
          }
        })
        .catch((error) => this.$message.error(error));
    },
  },
  data() {
    return {
      isEdit:false,
      brandDetail:{},
      dialog: false,
      brandName: "",
      loading: true,
      pagination: {},
      total: 0,
      headers: [
        {
          text: "id",
          align: "center",
          sortable: true,
          value: "id",
        },
        {
          text: "name",
          align: "center",
          value: "name",
        },
        {
          text: "image",
          align: "center",
          sortable: false,
          value: "image",
        },
        {
          text: "letter",
          value: "letter",
          align: "center",
        },
         {
          text: "操作",
          value: "id",
          sortable: false,
          align: "center",
        }
      ],
      desserts: [],
    };
  },
};
</script>
<style scoped>
</style>
```

4.1.1.2 MrBrandForm.vue

```
<template>
  <div>
    <!--ref="form表单名称" 可以在this.$form获取当前表单的属性-->
    <v-form v-model="valid" ref="form">
      <v-card-text>
        <v-container>
          <v-text-field v-model="brand.name" label="品牌名称" :rules="nameRules"
required></v-text-field>
          <!-- <v-text-field v-model="brand.letter" label="品牌首字母" required>
</v-text-field> -->
          <!--选择分类组件,联动-->
          <v-cascader url="/category/list" required v-model="brand.category"
multiple label="商品分类" />

          <!--上传插件-->
          <!--布局,不加此标签模态框样式会乱,因为联动组件占用了一部分位置-->
          <v-layout row>
            <!--栅格布局 将一行分成12分 xs3:当前组件占3分-->
            <v-flex xs3>
              <span style="font-size: 16px; color: #444">品牌LOGO: </span>
            </v-flex>
            <v-flex>
              <v-upload
                v-model="brand.image"
                url="/upload"
                :multiple="false"
                :pic-width="250"
                :pic-height="90"
              />
            </v-flex>
          </v-layout>
        </v-container>
      </v-card-text>
    </v-form>
    <v-card-actions>
      <v-spacer></v-spacer>
      <v-btn color="green darken-1" flat @click="cancel">取消</v-btn>
      <v-btn color="green darken-1" flat @click="submitBrand">确定</v-btn>
    </v-card-actions>
  </div>
</template>
<script>
export default {
  name: "MrBrandForm",
  props: {
    dialog: Boolean, //接受父级组件传递过来的状态
    brandDetail: Object, //回显的数据
    isEdit: Boolean, //是否为修改
  },
  watch: {
    dialog() {
      //监控模态框
      if (this.dialog) {
        //清空form表单
        this.$refs.form.reset();
      }
    },
    brandDetail() {
      //监控回显数据变化
      if (this.isEdit) {
        //判断当前是否执行修改
        //通过品牌id查询分类信息
        this.$http
          .get("category/getByBrand", {
            params: {
              brandId: this.brandDetail.id,
            },
          })
          .then((resp) => {
            //注意:此处不能直接写this.brandDetail.category = resp.data.data;
            //如果使用上述方式写的话,会直接修改brandDetail,
            //当前监听函数再次监听到发生改变,会死循环
            let brand = this.brandDetail;
            brand.category = resp.data.data;
            this.brand = brand; //回显
          })
          .catch((error) => this.$message.error(error));
      }
    },
  },
  methods: {
    cancel() {
      this.$emit("closeModel");
    },
    submitBrand() {
      //入库
      if (this.$refs.form.validate()) {
        //表单验证成功后入库,submit*****
        const formData = this.brand;
        formData.category = this.brand.category.map((c) => c.id).join();
        this.$http({
          method: this.isEdit ? "put" : "post", //判断当前是否为修改操作
          url: "brand/save",
          data: formData, //传递的数据
        })
          .then((resp) => {
            if (resp.data.code == 200) {
              //刷新列表
              this.$message.success("保存成功");
              this.$emit("RefreshList");
              this.cancel();
            }
          })
          .catch((error) => this.$message.error(error));

        // if(this.isEdit){
        //   //修改
```

```
        //   this.$http.put('brand/save',formData).then(resp => {
        //     if(resp.data.code == 200){
        //         //刷新列表
        //         this.$message.success('修改成功');
        //         this.$emit('RefreshList');
        //         this.cancel();
        //     }
        // }).catch(error => this.$message.error(error))
        // }else{//新增
        // this.$http.post('brand/save',formData).then(resp => {
        //     if(resp.data.code == 200){
        //         //刷新列表
        //         this.$message.success('新增成功');
        //         this.$emit('RefreshList');
        //         this.cancel();
        //     }
        // }).catch(error => this.$message.error(error))
        // }
      }
    },
  },
  data() {
    return {
      valid: true,
      nameRules: [
        (v) => {
          if (v) {
            return v.length <= 10 || "品牌名称长度超过10位";
          }
          return !!v || "品牌名称不能为空";
        },
      ],
      brand: {
        name: "",
        // letter: "",
        image: "",
        category: [],
      },
    };
  },
};
</script>
```

4.1.2 mingrui-shop-service-api-xxx

4.1.2.1 CategoryService

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

4.1.3 mingrui-shop-service-xxx

4.1.3.1 CategoryMapper

```
    @Select(value = "select c.id,c.name from tb_category c where c.id in (select
cb.category_id from tb_category_brand cb where cb.brand_id=#{brandId})")
    List<CategoryEntity> getByBrandId(Integer brandId);
```

4.1.3.2 CategoryServiceImpl

```
    @Override
    public Result<List<CategoryEntity>> getByBrand(Integer brandId) {

        List<CategoryEntity> byBrandId = categoryMapper.getByBrandId(brandId);

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

4.2 修改

4.2.1 vue项目

4.2.1.1 MrBrandForm.vue

```
    submitBrand() {
      //入库
      if (this.$refs.form.validate()) {
        //表单验证成功后入库,submit*****
        const formData = this.brand;
        formData.category = this.brand.category.map((c) => c.id).join();
        this.$http({
          method: this.isEdit ? "put" : "post", //判断当前是否为修改操作
          url: "brand/save",
          data: formData, //传递的数据
        })
          .then((resp) => {
            if (resp.data.code == 200) {
              //刷新列表
              this.$message.success("保存成功");
              this.$emit("RefreshList");
              this.cancel();
            }
          })
          .catch((error) => this.$message.error(error));

        // if(this.isEdit){
        //   //修改
        //   this.$http.put('brand/save',formData).then(resp => {
        //     if(resp.data.code == 200){
        //         //刷新列表
        //         this.$message.success('修改成功');
        //         this.$emit('RefreshList');
        //         this.cancel();
        //     }
        // }).catch(error => this.$message.error(error))
        // }else{//新增
        // this.$http.post('brand/save',formData).then(resp => {
        //     if(resp.data.code == 200){
        //         //刷新列表
        //         this.$message.success('新增成功');
        //         this.$emit('RefreshList');
        //         this.cancel();
        //     }
        // }).catch(error => this.$message.error(error))
        // }
      }
    },
```

4.2.2 mingrui-shop-service-api-xxx

4.2.2.1 BrandService

```
    @PutMapping(value = "brand/save")
    @ApiOperation(value = "修改品牌信息")
    Result<JsonObject> editBrand(@Validated({MingruiOperation.Update.class})
@RequestBody BrandDTO brandDTO);
```

4.2.3 mingrui-shop-service-xxx

4.3.3.1 BrandServiceImpl

```
    @Transactional
    @Override
    public Result<JsonObject> saveBrand(BrandDTO brandDTO) {

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

brandEntity.setLetter(PinyinUtil.getUpperCase(String.valueOf(brandEntity.getName
().charAt(0))
                , PinyinUtil.TO_FIRST_CHAR_PINYIN).charAt(0));

        brandMapper.insertSelective(brandEntity);
        //代码优化,将公共的代码抽取出来
        this.insertCategoryAndBrand(brandDTO, brandEntity);

        return this.setResultSuccess();
    }

    @Transactional
    @Override
    public Result<JsonObject> editBrand(BrandDTO brandDTO) {

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

brandEntity.setLetter(PinyinUtil.getUpperCase(String.valueOf(brandEntity.getName
().charAt(0))
                , PinyinUtil.TO_FIRST_CHAR_PINYIN).charAt(0));

        //执行修改操作
        brandMapper.updateByPrimaryKeySelective(brandEntity);

        //通过brandID删除中间表的数据
        Example example = new Example(CategoryBrandEntity.class);
        example.createCriteria().andEqualTo("brandId",brandEntity.getId());
        categoryBrandMapper.deleteByExample(example);

        //新增新的数据
        //代码优化,将公共的代码抽取出来
        this.insertCategoryAndBrand(brandDTO,brandEntity);

        return this.setResultSuccess();
    }
    //新增关系数据
    private void insertCategoryAndBrand(BrandDTO brandDTO,BrandEntity
brandEntity){

        if(brandDTO.getCategory().contains(",")){

            List<CategoryBrandEntity> categoryBrandEntities =
Arrays.asList(brandDTO.getCategory().split(","))
                    .stream().map(cid -> {

                        CategoryBrandEntity entity = new CategoryBrandEntity();
                        entity.setCategoryId(StringUtil.toInteger(cid));
                        entity.setBrandId(brandEntity.getId());

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

            categoryBrandMapper.insertList(categoryBrandEntities);
        }else{

            CategoryBrandEntity entity = new CategoryBrandEntity();
            entity.setCategoryId(StringUtil.toInteger(brandDTO.getCategory()));
            entity.setBrandId(brandEntity.getId());

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

5 删除

5.1 vue项目

5.1.1 Mrbrand.vue

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

```
    deleteBrand (id) {

      this.$message.confirm('此操作将永久删除该品牌, 是否继续?').then(() => {
         this.$http.delete('brand/delete?id=' + id).then(resp => {
        if(resp.data.code == 200){
          this.getBrandInfo();
          this.$message.success("删除成功");
        }else{
          this.$message.error(resp.data.msg);
        }
      }).catch(error => this.$message.error(error));
        }).catch(() => {
          this.$message.info("删除已取消！");
        });

    },
```

5.2 mingrui-shop-api-xxx

5.2.1 BrandService

```
    @DeleteMapping(value = "brand/delete")
    @ApiOperation(value = "通过id删除品牌信息")
    Result<JsonObject> deleteBrand(Integer id);
```

5.3 mingrui-shop-service-xxx

5.3.1 BrandServiceImpl

```
    @Transactional
    @Override
    public Result<JsonObject> editBrand(BrandDTO brandDTO) {

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

brandEntity.setLetter(PinyinUtil.getUpperCase(String.valueOf(brandEntity.getName
().charAt(0))
                , PinyinUtil.TO_FIRST_CHAR_PINYIN).charAt(0));

        //执行修改操作
        brandMapper.updateByPrimaryKeySelective(brandEntity);

        //通过brandID删除中间表的数据
        this.deleteCategoryAndBrand(brandEntity.getId());

        //新增新的数据
        this.insertCategoryAndBrand(brandDTO,brandEntity);

        return this.setResultSuccess();
    }

    @Transactional
    @Override
    public Result<JsonObject> deleteBrand(Integer id) {

        //删除品牌
        brandMapper.deleteByPrimaryKey(id);
        //关系?????
        this.deleteCategoryAndBrand(id);

        return this.setResultSuccess();
    }

    private void deleteCategoryAndBrand(Integer id){

        Example example = new Example(CategoryBrandEntity.class);
        example.createCriteria().andEqualTo("brandId",id);
        categoryBrandMapper.deleteByExample(example);
    }
```