今日目标

1.完成参数管理
2.推送代码到码云
3.制作商品列表页面
4.制作商品添加页面

1.参数管理

A.展示动态参数可选项

动态参数可选项展示及操作
在获取动态参数的方法中进行处理。

```
//将获取到的数据中的attr_vals字符串转换为数组
res.data.forEach(item => {
  item.attr_vals = item.attr_vals ? item.attr_vals.split(' ') : []
  //添加一个bool值控制文本框的显示或者隐藏
  item.inputVisible = false
  //添加一个inputValue保存文本框值
  item.inputValue = ''
})

//然后再修改展开行中的代码，生成el-tag和文本框以及添加按钮
<!-- 展开行 -->
<el-table-column type="expand">
  <template slot-scope="scope">
    <!-- 循环生成的el-tag -->
    <el-tag v-for="(item,i) in scope.row.attr_vals" :key="i" closable>{{item}}
</el-tag>
    <!-- 输入框 -->
    <el-input class="input-new-tag" v-if="scope.row.inputVisible" v-
model="scope.row.inputValue" ref="saveTagInput" size="small"
@keyup.enter.native="handleInputConfirm(scope.row)"
@blur="handleInputConfirm(scope.row)">
    </el-input>
    <!-- 添加按钮 -->
    <el-button v-else class="button-new-tag" size="small"
@click="showInput(scope.row)">+ New Tag</el-button>
  </template>
</el-table-column>


//最后对应文本框的事件和按钮的事件添加处理函数
handleInputConfirm(row){
  //当用户在文本框中按下enter键或者焦点离开时都会触发执行
  //判断用户在文本框中输入的内容是否合法
  if(row.inputValue.trim().length===0){
    row.inputValue = ''
    row.inputVisible = false
    return
  }
```

```
  // row.inputVisible = false
  //如果用户输入了真实合法的数据，需要保存起来
},
showInput(row){
  //用户点击添加按钮时触发
  row.inputVisible = true
  //$nextTick:在页面上元素被重新渲染之后，调用回调函数的代码
  this.$nextTick(_=>{
    //让文本框自动获得焦点
    this.$refs.saveTagInput.$refs.input.focus()
  })
}
```

B.添加/删除可选项

添加/删除动态参数可选项

```
给el-tag添加删除事件
<el-tag v-for="(item,i) in scope.row.attr_vals" :key="i" closable
@close="handleClose(i,scope.row)">{{item}}</el-tag>

在methods中添加新增，删除事件处理函数
handleInputConfirm(row){
  //当用户在文本框中按下enter键或者焦点离开时都会触发执行
  //判断用户在文本框中输入的内容是否合法
  if(row.inputValue.trim().length===0){
    row.inputValue = ''
    row.inputVisible = false
    return
  }

  // row.inputVisible = false
  //如果用户输入了真实合法的数据，需要保存起来
  row.attr_vals.push(row.inputValue.trim())
  row.inputValue = ''
  row.inputVisible = false

  this.saveAttrVals(row)
},
handleClose(index,row){
  //删除对应索引的参数可选项
  row.attr_vals.splice(index,1)
  //调用函数，完成保存可选项的操作
  this.saveAttrVals(row)
},
async saveAttrVals(row){
  //封装函数，完成保存可选项的操作
  //发起请求，保存参数细项
  const {data:res} = await
this.$http.put(`categories/${this.cateId}/attributes/${row.attr_id}`,
  {attr_name:row.attr_name,attr_sel:row.attr_sel,attr_vals:row.attr_vals.join('
')})

  if (res.meta.status !== 200) {
    return this.$message.error('修改参数项失败')
  }
```

```
  this.$message.success('修改参数项成功')
}
```

补充：当用户在级联选择框中选中了非三级分类时，需要清空表格中数据

```
async handleChange() {
      //如果用户选择的不是三级分类
      if(this.selectedCateKeys.length !== 3){
        this.selectedCateKeys = []
        this.manyTableData = []
        this.onlyTableData = []
        return
      }
      ......
```

补充2：当完成了动态参数可选项的功能之后，我们也需要一样的方式完成静态属性可选项的功能。

此时我们只需要将动态参数可选项中的展开行复制到静态属性的表格中即可

2.推送代码到码云

添加到暂存求： git add .
提交到本地仓库： git commit -m '完成了分类参数开发'
推送到码云： git push
切换到master： git checkout master
合并到master： git merge goods_params

创建子分支
git checkout -b goods_list
推送至码云 git push -u origin goods_list

3.商品列表

A.制作商品列表基本结构

添加子级路由组件以及对应的规则,并设置组件的基本机构
打开router.js,添加下面的代码

```
import GoodList from './components/goods/List.vue'

path: '/home', component: Home, redirect: '/welcome', children: [
  { path: "/welcome", component: Welcome },
  { path: "/users", component: Users },
  { path: "/rights", component: Rights },
  { path: "/roles", component: Roles  },
  { path: "/categories", component: Cate  },
  { path: "/params", component: Params  },
  { path: "/goods", component: GoodList  }
]
```

打开List.vue组件，添加下列代码

```
<template>
    <div>
        <h3>商品列表</h3>
```

```
        <!-- 面包屑导航 -->
        <el-breadcrumb separator="/">
            <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
            <el-breadcrumb-item>商品管理</el-breadcrumb-item>
            <el-breadcrumb-item>商品列表</el-breadcrumb-item>
        </el-breadcrumb>
        <!-- 卡片视图区域 -->
        <el-card>
            <el-row :gutter="20">
                <el-col :span="8">
                    <el-input placeholder="请输入内容">
                        <el-button slot="append" icon="el-icon-search"></el-
button>
                    </el-input>
                </el-col>
                <el-col :span="4">
                    <el-button type="primary">添加商品</el-button>
                </el-col>
            </el-row>
        </el-card>
    </div>
</template>

<script>
export default {
  data() {
    return {}
  },
  created() {},
  methods: {}
}
</script>

<style lang="less" scoped>
</style>
```

B.数据展示

添加数据表格展示数据以及分页功能的实现,搜索功能的实现
在main.js中添加过滤器:

```
//创建过滤器将秒数过滤为年月日，时分秒
Vue.filter('dateFormat',function(originVal){
  const dt = new Date(originVal)
  const y = dt.getFullYear()
  const m = (dt.getMonth()+1+'').padStart(2,'0')
  const d = (dt.getDate()+'').padStart(2,'0')

  const hh = (dt.getHours()+'').padStart(2,'0')
  const mm = (dt.getMinutes()+'').padStart(2,'0')
  const ss = (dt.getSeconds()+'').padStart(2,'0')

  return `${y}-${m}-${d} ${hh}:${mm}:${ss}`
})
```

```
<!-- 卡片视图区域 -->
```

```
<el-card>
    <!-- 搜索栏 -->
    <el-row :gutter="20">
        <el-col :span="8">
            <el-input placeholder="请输入内容" v-model="queryInfo.query" clearable
@clear="getGoodsList">
                <el-button slot="append" icon="el-icon-search"
@click="getGoodsList"></el-button>
            </el-input>
        </el-col>
        <el-col :span="4">
            <el-button type="primary">添加商品</el-button>
        </el-col>
    </el-row>

    <!-- 表格区域 -->
    <el-table :data="goodsList" border stripe>
        <el-table-column type="index"></el-table-column>
        <el-table-column label="商品名称" prop="goods_name"></el-table-column>
        <el-table-column label="商品价格(元)" prop="goods_price" width="95px">
</el-table-column>
        <el-table-column label="商品重量" prop="goods_weight" width="95px"></el-
table-column>
        <el-table-column label="创建时间" prop="add_time" width="140px">
            <template slot-scope="scope">
                {{scope.row.add_time | dateFormat}}
            </template>
        </el-table-column>
        <el-table-column label="操作" width="125px">
            <template slot-scope="scope">
                <el-button size="mini" type="primary" icon="el-icon-edit"></el-
button>
                <el-button size="mini" type="danger" icon="el-icon-delete"></el-
button>
            </template>
        </el-table-column>
    </el-table>

    <!-- 分页 -->
    <el-pagination @size-change="handleSizeChange" @current-
change="handleCurrentChange" :current-page="queryInfo.pagenum" :page-sizes="[3,
5, 10, 15]" :page-size="queryInfo.pagesize" layout="total, sizes, prev, pager,
next, jumper" :total="total">
    </el-pagination>
</el-card>

//绑定数据以及添加方法
<script>
export default {
  data() {
    return {
      //查询参数
      queryInfo: {
        query: '',
        pagenum: 1,
        pagesize: 10
      },
      //保存商品列表信息
```

```
      goodsList: [],
      //总数据条数
      total: 0
    }
  },
  created() {
    this.getGoodsList()
  },
  methods: {
    async getGoodsList() {
      //   根据分页获取对应的商品列表
      const { data: res } = await this.$http.get('goods', {
        params: this.queryInfo
      })

      if (res.meta.status !== 200) {
        return this.$message.error('获取商品列表失败')
      }
      console.log(res.data)
      this.$message.success('获取商品列表成功')
      this.goodsList = res.data.goods
      this.total = res.data.total
    },
    handleSizeChange(newSize){
        //当页号发生改变时，更改pagesize，重新请求
        this.queryInfo.pagesize = newSize
        this.getGoodsList();
    },
    handleCurrentChange(newPage){
        //当页码发生改变时，更改pagesize，重新请求
        this.queryInfo.pagenum = newPage
        this.getGoodsList();
    }
  }
}
</script>
```

C.实现删除商品

```
//绑定按钮点击事件
<el-button size="mini" type="danger" icon="el-icon-delete"
@click="removeGoods(scope.row.goods_id)"></el-button>

//事件函数代码编写
async removeGoods(goods_id) {
  //根据id删除对应的参数或属性
  //弹窗提示用户是否要删除
  const confirmResult = await this.$confirm(
    '请问是否要删除该商品',
    '删除提示',
    {
      confirmButtonText: '确认删除',
      cancelButtonText: '取消',
      type: 'warning'
    }
  ).catch(err => err)
  //如果用户点击确认，则confirmResult 为'confirm'
```

```
  //如果用户点击取消，则confirmResult获取的就是catch的错误消息'cancel'
  if (confirmResult != 'confirm') {
    return this.$message.info('已经取消删除')
  }

  //没有取消就是要删除，发送请求完成删除
  const {data:res} = await this.$http.delete(`goods/${goods_id}`)

  if (res.meta.status !== 200) {
    return this.$message.error('删除商品失败')
  }

  this.$message.success('删除商品成功')
  this.getGoodsList()
}
```

4.添加商品

A.添加编程式导航

在List.vue中添加编程式导航，并创建添加商品路由组件及规则

```
//在List.vue中添加编程式导航
<el-col :span="4">
    <el-button type="primary" @click="goAddPage">添加商品</el-button>
</el-col>

goAddPage(){
    this.$router.push('/goods/add')
}
```

在router.js中引入goods/Add.vue,并添加路由规则

```
import GoodAdd from './components/goods/Add.vue'
path: '/home', component: Home, redirect: '/welcome', children: [
  { path: "/welcome", component: Welcome },
  { path: "/users", component: Users },
  { path: "/rights", component: Rights },
  { path: "/roles", component: Roles  },
  { path: "/categories", component: Cate  },
  { path: "/params", component: Params  },
  { path: "/goods", component: GoodList  },
  { path: "/goods/add", component: GoodAdd  }
]
```

B.布局Add.vue组件

布局过程中需要使用Steps组件，在element.js中引入并注册该组件，并在global.css中给组件设置全局样式

```
import {Steps,Step} from 'element-ui'
Vue.use(Step)
Vue.use(Steps)

//global.css
.el-steps{
    margin:15px 0;
}
.el-step__title{
    font-size: 13px;
}
```

然后再在Add.vue中进行页面布局

```
<template>
    <div>
        <h3>添加商品</h3>
        <!-- 面包屑导航 -->
        <el-breadcrumb separator="/">
            <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
            <el-breadcrumb-item>商品管理</el-breadcrumb-item>
            <el-breadcrumb-item>添加商品</el-breadcrumb-item>
        </el-breadcrumb>
        <!-- 卡片视图区域 -->
        <el-card>
            <!-- 消息提示 -->
            <el-alert title="添加商品信息" type="info" center show-icon
:closable="false">
            </el-alert>

            <!-- 步骤条组件 -->
            <!-- align-center(居中效果) -->
            <el-steps :space="200" :active="activeIndex - 0" finish-
status="success" align-center>
                <el-step title="基本信息"></el-step>
                <el-step title="商品参数"></el-step>
                <el-step title="商品属性"></el-step>
                <el-step title="商品图片"></el-step>
                <el-step title="商品内容"></el-step>
                <el-step title="完成"></el-step>
            </el-steps>

            <!-- tab栏区域:el-tab-pane必须是el-tabs的子节点
            :tab-position="'left'"(设置tab栏为左右结构tab栏) -->
            <!-- 表单：label-position="top"(设置label在文本框上方) -->
            <el-form :model="addForm" :rules="addFormRules" ref="addFormRef"
label-width="100px" label-position="top">
                <el-tabs v-model="activeIndex" :tab-position="'left'">
                    <el-tab-pane label="基本信息" name="0">
                        <el-form-item label="商品名称" prop="goods_name">
                            <el-input v-model="addForm.goods_name"></el-input>
                        </el-form-item>
                        <el-form-item label="商品价格" prop="goods_price">
                            <el-input v-model="addForm.goods_price"
type="number"></el-input>
                        </el-form-item>
```

```
                        <el-form-item label="商品重量" prop="goods_weight">
                            <el-input v-model="addForm.goods_weight"
type="number"></el-input>
                        </el-form-item>
                        <el-form-item label="商品数量" prop="goods_number">
                            <el-input v-model="addForm.goods_number"
type="number"></el-input>
                        </el-form-item>
                        <el-form-item label="商品分类" prop="goods_cat">
                            <!-- 选择商品分类的级联选择框 -->
                            <el-cascader expandTrigger='hover' v-
model="addForm.goods_cat" :options="cateList" :props="cateProps"
@change="handleChange" clearable></el-cascader>
                        </el-form-item>
                    </el-tab-pane>
                    <el-tab-pane label="商品参数" name="1">

                    </el-tab-pane>
                    <el-tab-pane label="商品属性" name="2">

                    </el-tab-pane>
                    <el-tab-pane label="商品图片" name="3">

                    </el-tab-pane>
                    <el-tab-pane label="商品内容" name="4">

                    </el-tab-pane>
                </el-tabs>
            </el-form>
        </el-card>
    </div>
</template>

<script>
export default {
  data() {
    return {
      //保存步骤条激活项索引
      activeIndex: '0',
      //添加商品的表单数据对象
      addForm: {
        goods_name: '',
        goods_price: 0,
        goods_weight: 0,
        goods_number: 0,
        goods_cat:[]
      },
      //验证规则
      addFormRules: {
        goods_name: [
          { required: true, message: '请输入商品名称', trigger: 'blur' }
        ],
        goods_price: [
          { required: true, message: '请输入商品价格', trigger: 'blur' }
        ],
        goods_weight: [
          { required: true, message: '请输入商品重量', trigger: 'blur' }
        ],
```

```
        goods_number: [
          { required: true, message: '请输入商品数量', trigger: 'blur' }
        ],
        goods_cat: [
          { required: true, message: '请选择商品分类', trigger: 'blur' }
        ]
      },
      //用来保存分类数据
      cateList: [],
      //配置级联菜单中数据如何展示
      cateProps: {
        value: 'cat_id',
        label: 'cat_name',
        children: 'children'
      }
    }
  },
  created() {
    this.getCateList()
  },
  methods: {
    async getCateList() {
      const { data: res } = await this.$http.get('categories')

      if (res.meta.status !== 200) {
        return this.$message.error('获取商品分类数据失败')
      }
      this.cateList = res.data
    },
    handleChange(){
      //如果用户选择的不是三级分类,该次选择无效，因为必须选择三级分类
      if(this.addForm.goods_cat.length !== 3){
        this.addForm.goods_cat = []
        return
      }
    }
  }
}
</script>

<style lang="less" scoped>
</style>
```

C.添加tab栏切换验证

也就是说不输入某些内容，无法切换到别的tab栏

```
//首先给tabs添加tab切换前事件
<el-tabs v-model="activeIndex" :tab-position="'left'" :before-
leave="beforeTabLeave">
......
</el-tabs>

//再到methods编写事件函数beforeTabLeave
beforeTabLeave(activeName,oldActiveName){
  //在tab栏切换之前触发，两个形参为切换前，后的tab栏name
  if(oldActiveName === '0'){
```

```
      //在第一个标签页的时候
      if(this.addForm.goods_cat.length !== 3){
          this.$message.error('请选择商品的分类')
          return false
      }else if(this.addForm.goods_name.trim() === ''){
          this.$message.error('请输入商品名称')
          return false
      }else if(this.addForm.goods_price.trim() === '0'){
          this.$message.error('请输入商品价格')
          return false
      }else if(this.addForm.goods_weight.trim() === '0'){
          this.$message.error('请输入商品重量')
          return false
      }else if(this.addForm.goods_number.trim() === '0'){
          this.$message.error('请输入商品数量')
          return false
      }
  }
}
```

D.展示信息

展示商品参数信息,商品属性信息
在商品参数信息展示中使用的el-checkbox,el-checkbox-group组件，打开element.js引入组件并注册组件

```
//在用户点击tab栏时触发事件
<el-tabs v-model="activeIndex" :tab-position="'left'" :before-
leave="beforeTabLeave" @tab-click="tabClicked">
........

//在参数信息，商品属性面板中添加循环生成结构的代码
<el-tab-pane label="商品参数" name="1">
  <!-- 渲染表单item项 -->
  <el-form-item :label="item.attr_name" :key="item.attr_id" v-for="item in
manyTableData">
      <!-- 使用数组渲染复选框组 -->
      <el-checkbox-group v-model="item.attr_vals">
          <el-checkbox border :label="val" v-for="(val,i) in item.attr_vals"
:key="i"></el-checkbox>
      </el-checkbox-group>
  </el-form-item>
</el-tab-pane>
<el-tab-pane label="商品属性" name="2">
  <!-- 循环生成静态属性 -->
  <el-form-item :label="item.attr_name" v-for="item in onlyTableData"
:key="item.attr_id">
      <el-input v-model="item.attr_vals"></el-input>
  </el-form-item>
</el-tab-pane>

//在data数据中添加保存动态参数和静态属性的数组
export default {
  data() {
    return {
      ......
      //动态参数列表
```

```
      manyTableData: [],
      //静态属性列表
      onlyTableData:[]
      }
  },methods: {
    .......
    async tabClicked() {
      //当用户点击切换tab栏时触发
      if (this.activeIndex === '1') {
        //发送请求获取动态参数
        const { data: res } = await this.$http.get(
          `categories/${this.cateId}/attributes`,
          { params: { sel: 'many' } }
        )

        if (res.meta.status !== 200) {
          return this.$message.error('获取动态参数列表失败')
        }
        //将attr_vals字符串转换为数组
        res.data.forEach(item => {
          item.attr_vals =
            item.attr_vals.length === 0 ? [] : item.attr_vals.split(' ')
        })
        this.manyTableData = res.data
      } else if (this.activeIndex === '2') {
        //发送请求获取静态属性
        const { data: res } = await this.$http.get(
          `categories/${this.cateId}/attributes`,
          { params: { sel: 'only' } }
        )

        if (res.meta.status !== 200) {
          return this.$message.error('获取静态属性列表失败')
        }

        this.onlyTableData = res.data
      }
    }
  },
  //添加 计算属性获取三级分类
  computed: {
    cateId() {
      if (this.addForm.goods_cat.length === 3) {
        return this.addForm.goods_cat[2]
      }
      return null
    }
  }
}
```