今日目标

1.完成商品分类

2.完成参数管理

1.商品分类

A.新建分支goods_cate

新建分支goods_cate并推送到码云
git checkout -b goods_cate
git push -u origin goods_cate

B.创建子级路由

创建categories子级路由组件并设置路由规则

```
import Cate from './components/goods/Cate.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  }
]
```

C.添加组件基本布局

在Cate.vue组件中添加面包屑导航以及卡片视图中的添加分类按钮

```
<template>
    <div>
        <h3>商品分类</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>商品分类</el-breadcrumb-item>
        </el-breadcrumb>
        <!-- 卡片视图区域 -->
        <el-card>
            <!-- 添加分类按钮区域 -->
            <el-row>
                <el-col>
                    <el-button type="primary">添加分类</el-button>
                </el-col>
            </el-row>
            <!-- 分类表格  -->

            <!-- 分页 -->
        </el-card>
```

```
    </div>
</template>
```

D.请求分类数据

请求分类数据并将数据保存在data中

```
<script>
export default {
  data() {
    return {
      // 商品分类数据列表
      cateList: [],
      //查询分类数据的条件
      queryInfo: {
        type: 3,
        pagenum: 1,
        pagesize: 5
      },
      //保存总数据条数
      total: 0
    }
  },
  created() {
    this.getCateList()
  },
  methods: {
    async getCateList() {
      //获取商品分类数据
      const { data: res } = await this.$http.get('categories', {
        params: queryInfo
      })

      if (res.meta.status !== 200) {
        return this.$message.error('获取商品列表数据失败')
      }
      //将数据列表赋值给cateList
      this.cateList = res.data.result
      //保存总数据条数
      this.total = res.data.total
      //   console.log(res.data);
    }
  }
}
</script>
```

E.使用插件展示数据

使用第三方插件vue-table-with-tree-grid展示分类数据

1).在vue 控制台中点击依赖->安装依赖->运行依赖->输入vue-table-with-tree-gird->点击安装

2).打开main.js，导入vue-table-with-tree-grid

import TreeTable from 'vue-table-with-tree-grid'

.....

Vue.config.productionTip = false

```
//全局注册组件
Vue.component('tree-table', TreeTable)
3).使用组件展示分类数据
```

```
<!-- 分类表格
:data(设置数据源) :columns(设置表格中列配置信息) :selection-type(是否有复选框)
:expand-type(是否展开数据) show-index(是否设置索引列) index-text(设置索引列头)
border(是否添加纵向边框) :show-row-hover(是否鼠标悬停高亮) -->
<tree-table :data="cateList" :columns="columns" :selection-type="false"
:expand-type="false" show-index index-text="#" border :show-row-hover="false">

</tree-table>

在数据中添加columns:
columns: [
    {label:'分类名称',prop:'cat_name'}
]
```

F.自定义数据列

使用vue-table-with-tree-grid定义模板列并添加自定义列

```
//先在columns中添加一个列
columns: [
    {label:'分类名称',prop:'cat_name'},
    //type:'template'(将该列设置为模板列)，template:'isok'(设置该列模板的名称为isok)
    {label:'是否有效',prop:'',type:'template',template:'isok'},
    {label:'排序',prop:'',type:'template',template:'order'},
    {label:'操作',prop:'',type:'template',template:'opt'}
]

<!-- 是否有效区域， 设置对应的模板列： slot="isok"(与columns中设置的template一致) -->
<template slot="isok" slot-scope="scope">
  <i class="el-icon-success" v-if="scope.row.cat_deleted === false"
style="color:lightgreen"></i>
  <i class="el-icon-error" v-else style="color:red"></i>
</template>
<!-- 排序 -->
<template slot="order" slot-scope="scope">
  <el-tag size="mini" v-if="scope.row.cat_level===0">一级</el-tag>
  <el-tag size="mini" type="success" v-else-if="scope.row.cat_level===1">二级
</el-tag>
  <el-tag size="mini" type="warning" v-else>三级</el-tag>
</template>

<!-- 操作 -->
<template slot="opt" slot-scope="scope">
  <el-button size="mini" type="primary" icon="el-icon-edit">编辑</el-button>
  <el-button size="mini" type="danger" icon="el-icon-delete">删除</el-button>
</template>
```

G.完成分页功能

```
<!-- 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>

//添加对应的事件函数
methods:{
  .......
  handleSizeChange(newSize){
    //当pagesize发生改变时触发
    this.queryInfo.pagesize = newSize;
    this.getCateList();
  },
  handleCurrentChange(newPage){
    //当pagenum发生改变时触发
    this.queryInfo.pagenum = newPage;
    this.getCateList();
  }
}
```

H.完成添加分类

```
......
<!-- 添加分类按钮区域 -->
<el-row>
  <el-col>
    <el-button type="primary" @click="showAddCateDialog">添加分类</el-button>
  </el-col>
</el-row>
......
<!-- 添加分类对话框 -->
<el-dialog title="添加分类" :visible.sync="addCateDialogVisible" width="50%"
 @close="addCateDialogClosed">
  <!-- 添加分类表单 -->
  <el-form :model="addCateForm" :rules="addCateFormRules"
ref="addCateFormRuleForm" label-width="100px">
    <el-form-item label="分类名称" prop="cat_name">
      <el-input v-model="addCateForm.cat_name"></el-input>
    </el-form-item>
    <el-form-item label="父级分类" prop="cat_pid">

    </el-form-item>
  </el-form>
  <span slot="footer" class="dialog-footer">
    <el-button @click="addCateDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="addCate">确 定</el-button>
  </span>
</el-dialog>


//用来显示或隐藏添加分类对话框
addCateDialogVisible: false,
```

```
//添加分类的表单数据对象
addCateForm:{
  //分类名称
  cat_name:'',
  //添加分类的父级id，0则表示父级为0.添加一级分类
  cat_pid:0,
  //添加分类的等级，0则表示添加一级分类
  cat_level:0
},
//添加分类校验规则
addCateFormRules:{
  //验证规则
  cat_name:[ {required:true , message:'请输入分类名称',trigger:'blur'} ]
},
//保存1,2级父级分类的列表
parentCateList:[]
.......
showAddCateDialog() {
  //调用getParentCateList获取分类列表
  this.getParentCateList()
  //显示添加分类对话框
  this.addCateDialogVisible = true
},
async getParentCateList(){
  //获取父级分类数据列表
  const { data: res } = await this.$http.get('categories', {
    params: {type:2}
  })

  if (res.meta.status !== 200) {
    return this.$message.error('获取商品分类列表数据失败')
  }
  this.parentCateList = res.data
}
```

添加级联菜单显示父级分类
先导入Cascader组件，并注册
然后添加使用级联菜单组件：

```
<el-form-item label="父级分类" prop="cat_pid">
  <!-- expandTrigger='hover'(鼠标悬停触发级联) v-model(设置级联菜单绑定数据)
:options(指定级联菜单数据源)  :props(用来配置数据显示的规则)
  clearable(提供“X”号完成删除文本功能) change-on-select(是否可以选中任意一级的菜单) -->
  <el-cascader expandTrigger='hover' v-model="selectedKeys"
:options="parentCateList" :props="cascaderProps" @change="parentCateChange"
clearable change-on-select></el-cascader>
</el-form-item>

添加数据
//配置级联菜单中数据如何展示
cascaderProps:{
  value:'cat_id',
  label:'cat_name',
  children:'children',
  expandTrigger:'hover'
},
//绑定用户选择的分类值
```

```
selectedKeys:[]
.....
methods:{
  .....
  parentCateChange(){
    //级联菜单中选择项发生变化时触发
    console.log(this.selectedKeys)
    //如果用户选择了父级分类
    if(this.selectedKeys.length > 0){
      //则将数组中的最后一项设置为父级分类
      this.addCateForm.cat_pid = this.selectedKeys[this.selectedKeys.length - 1]
      //level也要跟着发生变化
      this.addCateForm.cat_level = this.selectedKeys.length
      return
    }else{
      this.addCateForm.cat_pid = 0
      this.addCateForm.cat_level = 0
      return
    }
  },
  addCateDialogClosed(){
    //当关闭添加分类对话框时，重置表单
    this.$refs.addCateFormRef.resetFields()
    this.selectedKeys = [];
    this.addCateForm.cat_pid = 0
    this.addCateForm.cat_level = 0
  },
  addCate() {
    //点击确定，完成添加分类
    console.log(this.addCateForm)
    this.$refs.addCateFormRef.validate(async valid => {
      if (!valid) return
      //发送请求完成添加分类
      const { data: res } = await this.$http.post(
        'categories',
        this.addCateForm
      )

      if (res.meta.status !== 201) {
        return this.$message.error('添加分类失败')
      }

      this.$message.success('添加分类成功')
      this.getCateList()
      this.addCateDialogVisible = false
    })
  }
}
```

I.推送代码

制作完添加分类之后，将代码提交到仓库，推送到码云,将goods_cate分支合并到master
git add .
git commit -m '完成商品分类'
git push

git checkout master
git merge goods_cate

2.参数管理

只允许给三级分类内容设置参数，参数分为动态参数和静态参数属性
####A.添加子级组件
添加Params.vue子组件，并在router.js中引入该组件并设置路由规则

```
import Params from './components/goods/Params.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  }
]
```

B.完成组件基本布局

完成Params.vue组件的基本布局
其中警告提示信息使用了el-alert，在element.js引入该组件并注册

```
<template>
    <div>
        <h3>分类参数</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>分类参数</el-breadcrumb-item>
        </el-breadcrumb>
        <!-- 卡片视图区域 -->
        <el-card>
            <!-- 警告区域 :closable="false"(是否展示“X”号) show-icon(显示图标) -->
            <el-alert title="注意：只允许为第三级分类设置相关参数" type="warning"
:closable="false" show-icon>
            </el-alert>

            <!-- 选择商品分类区域 -->
            <el-row class="cat_opt">
                <el-col>
                    <span>选择商品分类：</span>
                    <!-- 选择商品分类的级联选择框 -->
                </el-col>
                <el-col></el-col>
            </el-row>
        </el-card>
    </div>
</template>
```

C.完成级联选择框

完成商品分类级联选择框

```
<!-- 选择商品分类区域 -->
<el-row class="cat_opt">
    <el-col>
        <span>选择商品分类：</span>
        <!-- 选择商品分类的级联选择框 -->
        <el-cascader expandTrigger='hover' v-model="selectedCateKeys"
:options="cateList" :props="cateProps" @change="handleChange" clearable></el-
cascader>
    </el-col>
    <el-col></el-col>
</el-row>
......
<script>
export default {
  data() {
    return {
        //分类列表
        cateList:[],
        //用户在级联下拉菜单中选中的分类id
        selectedCateKeys:[],
        //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(){
        //获取所有的商品分类列表
        const { data: res } = await this.$http.get('categories')

        if (res.meta.status !== 200) {
            return this.$message.error('获取分类数据失败')
        }
        //将数据列表赋值给cateList
        this.cateList = res.data
        // //保存总数据条数
        // this.total = res.data.total
        //   console.log(res.data);
      },
      handleChange(){
        //当用户在级联菜单中选择内容改变时触发
        console.log(this.selectedCateKeys);
      }
  }
}
</script>
```

D.展示参数

展示动态参数数据以及静态属性数据

```
<!-- tab页签区域 -->
<el-tabs v-model="activeName" @tab-click="handleTabClick">
  <!-- 添加动态参数的面板 将标签页改为many -->
  <el-tab-pane label="动态参数" name="many">
    <el-button size="mini" type="primary" :disabled="isButtonDisabled">添加参数
</el-button>
    <!-- 动态参数表格 -->
    <el-table :data="manyTableData" border stripe>
      <!-- 展开行 -->
      <el-table-column type="expand"></el-table-column>
      <!-- 索引列 -->
      <el-table-column type="index"></el-table-column>
      <el-table-column label="参数名称" prop="attr_name"></el-table-column>
      <el-table-column label="操作">
        <template slot-scope="scope">
          <el-button size="mini" type="primary" icon="el-icon-edit">编辑</el-
button>
          <el-button size="mini" type="danger" icon="el-icon-delete">删除</el-
button>
        </template>
      </el-table-column>
    </el-table>
  </el-tab-pane>
  <!-- 添加静态属性的面板 将标签页改为only -->
  <el-tab-pane label="静态属性" name="only">
    <el-button size="mini" type="primary" :disabled="isButtonDisabled">添加属性
</el-button>
    <!-- 静态属性表格 -->
    <el-table :data="onlyTableData" border stripe>
      <!-- 展开行 -->
      <el-table-column type="expand"></el-table-column>
      <!-- 索引列 -->
      <el-table-column type="index"></el-table-column>
      <el-table-column label="属性名称" prop="attr_name"></el-table-column>
      <el-table-column label="操作">
        <template slot-scope="scope">
          <el-button size="mini" type="primary" icon="el-icon-edit">编辑</el-
button>
          <el-button size="mini" type="danger" icon="el-icon-delete">删除</el-
button>
        </template>
      </el-table-column>
    </el-table>
  </el-tab-pane>

</el-tabs>


<script>
export default {
  data() {
    return {
      ......
```

```
      //tab页签激活显示的页签项
      activeName: 'many',
      //用来保存动态参数数据
      manyTableData: [],
      //用来保存静态属性数据
      onlyTableData: []
    }
  methods: {
    .......
    async handleChange() {
      //当用户在级联菜单中选择内容改变时触发
      console.log(this.selectedCateKeys)
      //发送请求，根据用户选择的三级分类和面板获取参数数据
      const { data: res } = await this.$http.get(
        `categories/${this.cateId}/attributes`,
        { params: { sel: this.activeName } }
      )
      if (res.meta.status !== 200) {
        return this.$message.error('获取参数列表数据失败')
      }

      console.log(res.data)
      if (this.activeName === 'many') {
        //获取的是动态参数
        this.manyTableData = res.data
      } else if (this.activeName === 'only') {
        //获取的是静态属性
        this.onlyTableData = res.data
      }
    },
    handleTabClick() {
      console.log(this.activeName)
      this.handleChange()
    }
  },
  computed: {
    //添加计算属性用来获取按钮禁用与否
    isButtonDisabled() {
      return this.selectedCateKeys.length !== 3
    },
    //获取选中的三级分类id
    cateId() {
      if (this.selectedCateKeys.length === 3) {
        return this.selectedCateKeys[this.selectedCateKeys.length - 1]
      }
      return null
    }
  }
```

E.添加参数

完成添加参数或属性

```
<!-- 添加参数或属性对话框 -->
<el-dialog :title="'添加'+titleText" :visible.sync="addDialogVisible" width="50%"
@close="addDialogClosed">
  <!-- 添加表单 -->
```

```
  <el-form :model="addForm" :rules="addFormRules" ref="addFormRef" label-
width="100px">
    <el-form-item :label="titleText" prop="attr_name">
      <el-input v-model="addForm.attr_name"></el-input>
    </el-form-item>
  </el-form>
  <span slot="footer" class="dialog-footer">
    <el-button @click="addDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="addParams">确 定</el-button>
  </span>
</el-dialog>

export default {
  data() {
    return {
      .......
      //控制添加参数.属性对话框的显示或隐藏
      addDialogVisible: false,
      //添加参数的表单数据对象
      addForm: {
        attr_name: ''
      },
      //添加表单验证规则
      addFormRules: {
        attr_name: [{ required: true, message: '请输入名称', trigger: 'blur' }]
      }
    }
  },methods: {
    .......
    addParams() {
      //当用户点击对话框中的确定时，校验表单
      this.$refs.addFormRef.validate(async valid => {
        //校验不通过，return
        if (!valid) return
        //校验通过，发送请求完成添加参数或者属性
        const { data: res } =
this.$http.post(`categories/${this.cateId}/attributes`,
          {
            attr_name: this.addForm.attr_name,
            attr_sel: this.activeName,
            attr_vals: "a,b,c"
          }
        )

        console.log(res)
        if (res.meta.status !== 201) {
          return this.$message.error('添加' + this.titleText + '数据失败')
        }
        this.$message.success('添加' + this.titleText + '数据成功')
        this.addDialogVisible = false
        this.getCateList()
      })
    }
  }
```

F.编辑参数

完成编辑参数或属性

```
<!-- 修改参数或属性对话框 -->
<el-dialog :title="'修改'+titleText" :visible.sync="editDialogVisible"
width="50%" @close="editDialogClosed">
  <!-- 添加表单 -->
  <el-form :model="editForm" :rules="editFormRules" ref="editFormRef" label-
width="100px">
    <el-form-item :label="titleText" prop="attr_name">
      <el-input v-model="editForm.attr_name"></el-input>
    </el-form-item>
  </el-form>
  <span slot="footer" class="dialog-footer">
    <el-button @click="editDialogVisible = false">取 消</el-button>
    <el-button type="primary" @click="editParams">确 定</el-button>
  </span>
</el-dialog>

export default {
  data() {
    return {
      .......
      //控制修改参数.属性对话框的显示或隐藏
      editDialogVisible:false,
      //修改参数.属性对话框中的表单
      editForm:{
        attr_name:''
      },
      //修改表单的验证规则
      editFormRules:{
        attr_name:[
          { required: true, message: '请输入名称', trigger: 'blur' }
        ]
      }
    }
  },methods: {
    .......
    async showEditDialog(attr_id){
      //发起请求获取需要修改的那个参数数据
      const {data:res} = await
this.$http.get(`categories/${this.cateId}/attributes/${attr_id}`,
      {params:{ attr_sel:this.activeName }})
      if (res.meta.status !== 200) {
        return this.$message.error('获取参数数据失败')
      }
      this.editForm = res.data;
      //显示修改参数.属性对话框
      this.editDialogVisible = true;
    },
    editDialogClosed(){
      //当关闭修改参数.属性对话框时
      this.$refs.editFormRef.resetFields()
    },
    editParams(){
      //验证表单
```

```
      this.$refs.editFormRef.validate(async valid => {
        if(!valid) return;

        //发送请求完成修改
        const {data:res} = await
this.$http.put(`categories/${this.cateId}/attributes/${this.editForm.attr_id}`,
        {attr_name:this.editForm.attr_name,attr_sel:this.activeName})

        if (res.meta.status !== 200) {
          return this.$message.error('获取参数数据失败')
        }
        this.$message.success('修改' + this.titleText + '数据成功')
        this.editDialogVisible = false
        this.handleChange();
      })
    }
  }
```

G.删除参数

删除参数或属性

```
给两个删除按钮添加事件
<el-button size="mini" type="danger" icon="el-icon-delete"
@click="removeParams(scope.row.attr_id)">删除</el-button>
<el-button size="mini" type="danger" icon="el-icon-delete"
@click="removeParams(scope.row.attr_id)">删除</el-button>

添加对应的事件处理函数
async removeParams(attr_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.$confirm(
    '请问是否要删除该'+this.titleText,
    '删除提示',
    {
      confirmButtonText: '确认删除',
      cancelButtonText: '取消',
      type: 'warning'
    }
  ).catch(err => err)
  //如果用户点击确认，则confirmResult 为'confirm'
  //如果用户点击取消，则confirmResult获取的就是catch的错误消息'cancel'
  if (confirmResult != 'confirm') {
    return this.$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.$http.delete(`categories/${this.cateId}/attributes/${attr_id}`)

  if (res.meta.status !== 200) {
    return this.$message.error('删除参数数据失败')
  }
  this.$message.success('删除' + this.titleText + '数据成功')
  this.handleChange()
}
```