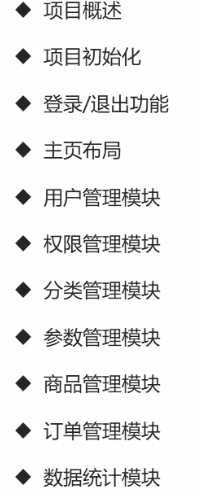
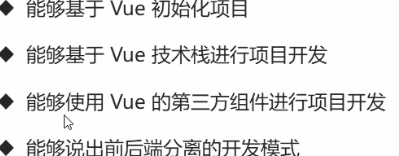
# 电商后台管理系统



# 启动项目

启动项目步骤：

1、启动数据库

打开phpStudy>其他选项菜单>服务管理器>MySQL>启动

2、启动API接口

H:\Project\vue-电商项目\vue\_api\_server\vue\_api\_server文件下

>shift+鼠标右键打开powershell>node app.js

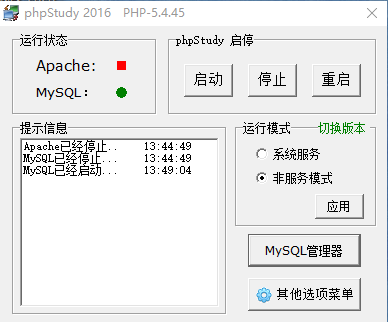
3、启动浏览器

VSCode中H:\Project\vue-电商项目\code\vue\_shop目录下

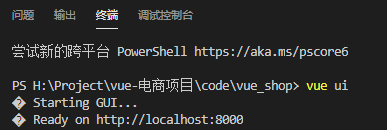
>ctrl+`启动终端>vue ui>弹出页面

4、打开vue\_shop>任务>serve>运行>启动APP

页面：







# 提交本地代码到git仓库

1、查看所处分支git branch

2、添加所有修改过的文件git add .

3、提交代码git commit -m

推送代码到码云

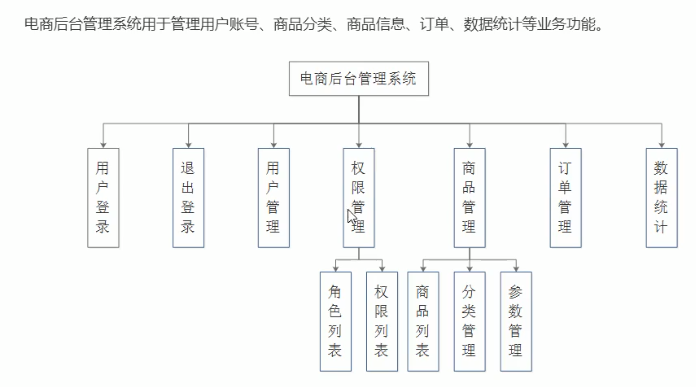
码云的码获取失败……

# 项目概述

## 基本业务

电商项目一般提供了PC端，PC后台管理，移动app，移动web，微信小程序等多种终端访问方式，本项目为PC后台管理终端

## 功能

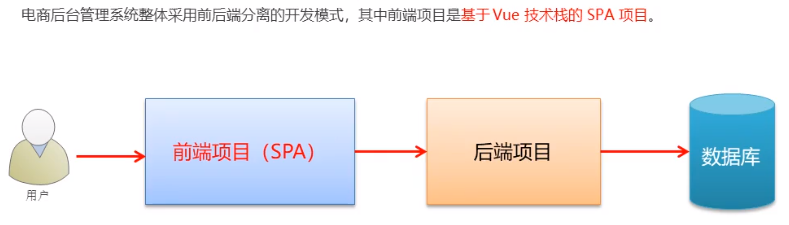


## 开发模式

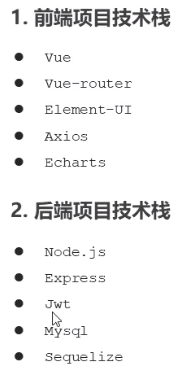
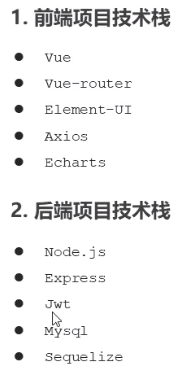
前后端分离：

后端负责操作数据库，向前端暴露API接口

前端负责绘制页面，基于Ajax技术调用后端提供的API接口



## 技术选型



Element-UI：前端UI组件库

Axios：发起网络数据请求

Echarts:绘制图形报表

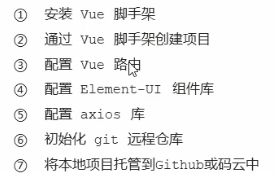
Jwt:状态保持工具，模拟登录记录功能

Mysql：后台数据库

Sequelize:操作数据库的框架

# 项目初始化

## 前端项目初始化步骤



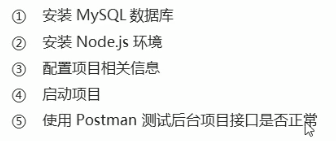
1、Vue全局安装过一次就可使用

2、通过可视化方式创建vue项目，在终端运行：vue ui

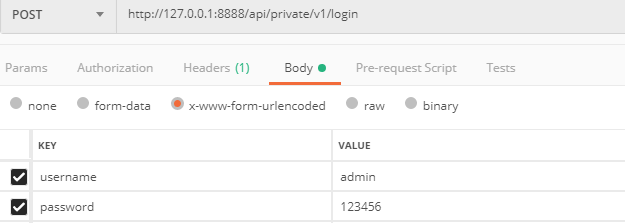
4、添加插件：vue-cli-plugin-element

5、安装依赖：axios运行依赖

## 后台项目的环境安装配置

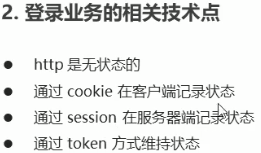
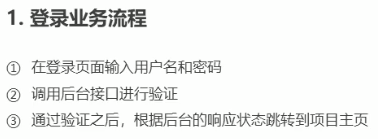


1. vue\_api\_server 就是后台API项目，npm install安装依赖包
2. node app.js启动项目



# 登录/退出功能

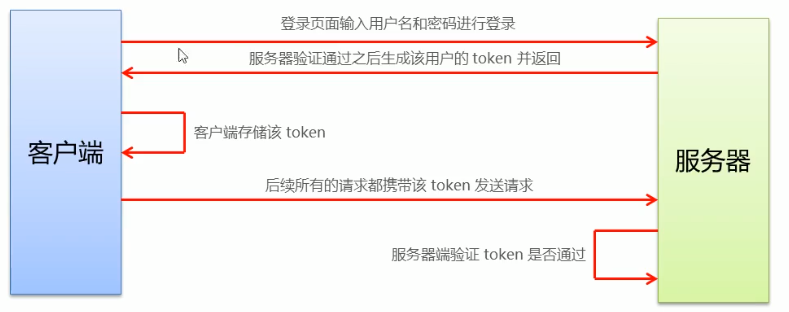
## 登陆概述



若存在跨域问题：选择token

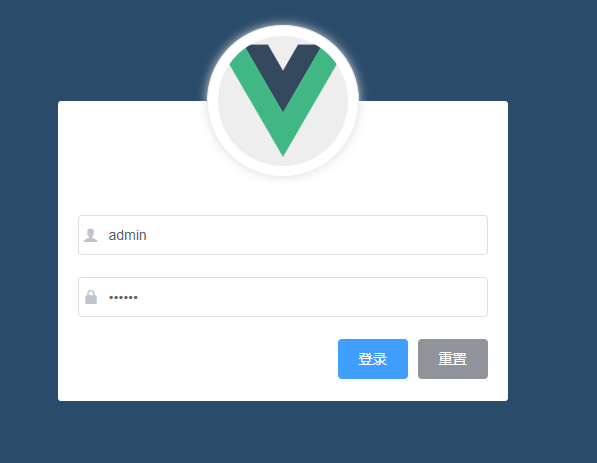
不存在跨域问题：cookie,session

## Token原理分析



token主要是服务器端和客户端身份校验

## 登录功能



**登录业务流程**：

在登录页面输入用户名和密码

调用后台接口进行验证

通过验证之后，根据后台的响应状态跳转到项目主页

**登录业务的技术点：**

http是无状态的

通过cookie在客户端记录

通过session在服务器端记录状态

通过token方式维持状态

**token原理分析：**

登录页面输入用户名和密码登录

服务器验证通过之后生成 该用户的token并返回

客户端存储该toke

后续用户所有的浏览器请求都会携带token进行请

服务器端验证token是否通

**登录页面布局：**

使用Element-UI 的Form表单进行页面布局

**登录功能实现：**

通过axios调用接口验证，为了减少代码重复，我们的axios是挂载到Vue的原型上

登陆成功之后保持用户的token信息

跳转到项目主页

**路由导航守卫控制访问权限**

页面操作是建立在登录之后的操作，在进行操作的时候后台会进行检测是否在登录状态下，主要是根据保存在本地sessionStorage中的token进行判断

**退出功能**

在这里我们的退出功能是基于清除本地sessionStorage中的token令牌，以此来实现退出登录功能

### 页面布局

1、通过Element-UI组件布局，用到：el-form，el-form-item，el-input，el-button，字体图标

2、Git status 查看状态

git checkout -b login 新建子分支login

git branch 查看当前分支

3、先在components中新建login.vue文件：template,script,style

在index.js中 import ，并添加路由{ path: '/login', component: Login }

在app.vue中的div中添加占位符<router-view></router-view>

### **element-ui添加组件**

1、在element-ui中找到代码，粘贴到Login.vue文件中

2、在plugins/element.js文件中import{}from ‘element-ui’并Vue.use()

注：.btns {//两个按键靠右对齐

display: flex;

justify-content: flex-end;}

### 在input组件首部引入IconFont 图标

1、将下载好的IconFont 图标文件夹fonts放在src/assets文件夹下

2、在输入框中添加属性prefix-icon="iconfont 图标名字"

### 表单添加验证规则

1、先为el-form添加属性绑定一个rules校验对象，:rules="名字"

2、在data数据中定义校验对象，其中每一个属性都是一个验证规则

3、为不同item项通过prop属性指定验证规则

### 表单重置

1、为表单添加ref引用，ref="loginFormRef"，值就是组件的实例对象

2、为重置按钮添加点击事件，@click="resetLoginForm"

3、在函数中添加：this.$refs.loginFormRef.resetFields();将所有字段值重置为初始值并移除校验结果

### 表单预验证

1、为登录按钮添加点击事件@click="login"

2、函数：this.$refs.loginFormRef.validate(valid=>{console.log(valid); })先获取到表单对象loginFormRef，再调用validate函数进行表单预校验，并接收一个回调函数，第一个形参为接收的结果：布尔值，根据布尔值可以判断表单验证通过还是失败

### 配置axios发起登录请求

1、在main.js中对axios全局配置：

导入包import axios from 'axios'

把包挂载在vue的原型对象上Vue.prototype.$http = axios

配置请求根路径：axios.defaults.baseURL = '根路径 '

2、启动mysql，启动API接口：在vue\_api\_server文件中启动后端node app.js

3、判断valid是否为true，为true则发起请求，false则阻止登录请求

4、用this.$http发起请求，post返回promise，用async和await简化成对象

5、data为服务器返回的真实数据，并通过{}解构赋值，重命名为res对象，根据meta属性判断是否登录成功

### 配置message全局弹框组件

1、element.js进行导入：import导入组件，并全局挂载：

Vue.prototype.$message = Message

2、通过this.$message.success('登陆成功');可弹出成功的提示框

### 完善登录

1、将登录成功之后的 token，保存到客户端的 sessionStorage 中

1.1 项目中出了登录之外的其他API接口，必须在登录之后才能访问

1.2 token 只应在当前网站打开期间生效，所以将 token 保存在 sessionStorage 中

window.sessionStorage.setItem("token", res.data.token);

2、通过编程式导航跳转到后台主页，路由地址是 /home

this.$router.push("/home");

3、新建[Home.vue文件，并在index.js文件中import](http://Home.vue文件，并在index.js文件中import)并添加路由

### 路由导航守卫控制页面访问权限

1、在index.js中，路由对象router挂载beforeEach导航守卫，传递一个回调函数，接收三个形参

2、根据to判断访问的地址和token的值来判断是否发生页面的强制跳转

## 退出功能



在home.vue文件中添加退出按键，为按键添加点击事件，函数：



# 主页布局



## 主页布局

### 基本布局

1、element官网中将符合整体布局的代码复制到home.vue文件中的<template>中

2、在element.js中导入组件：Container,Header,Aside,Main

3、页面美化

## Header区域

1、引入图片img和文字span

2、美化

## Aside导航菜单

在element库中复制代码



### 通过axios拦截器添加token验证

1、除了登录接口，其他所有接口都需要授权才能登录

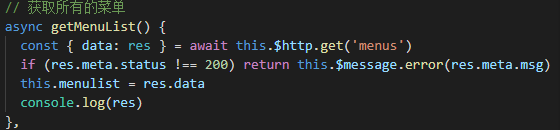
2、通过接口获取菜单数据，需要授权的 API ，必须在请求头中使用 `Authorization` 字段提供 `token` 令牌



### 左侧菜单数据获取

1、API接口文档中，### 1.4.2. 左侧菜单权限，menus为路径

2、页面一加载就立即获取左侧菜单，home.vue中在script区域定义声明周期函数created，res为获取到的数据，如果失败将失败信息赋值给meta.msg，如果成功将数据保存到左侧菜单数据menulist



### 左侧菜单双层for循环渲染

1、munulist存放所有的一级菜单，每一个一级菜单通过children属性嵌套所有的二级菜单，通过双层for循环渲染，外层for主要渲染一级菜单，内层for主要渲染二级菜单

2、为el-submenu指定唯一index，并通过属性绑定，字符串形式

:index="item.id+''"

### 左侧菜单添加分类图标

1、在data中一个字体对象iconsObj，将id作为key，将”iconfont 图标名称”作为value

2、根据id动态绑定图标:class="iconsObj[item.id]"

### 左侧菜单折叠展开

1、在侧边栏菜单区域上方添加<div class="toggle-button">|||</div>

2、为按钮添加点击事件@click="toggleCollapse"，切换菜单折叠与展开

3、在data中添加是否折叠的布尔值 isCollapse: false,默认不折叠，动态绑定到el-menu中:collapse="isCollapse"

4、在methods中定义toggleCollapse函数，是否折叠的布尔值取反：

this.isCollapse = !this.isCollapse;

5、el-menu中动态绑定属性，关掉动画:collapse-transition="false"

6、折叠时框变小，不折叠不变，属性绑定，判断isCollapse

<el-aside :width="isCollapse?'64px':'200px'">

## 首页

### 首页路由重定向

在登录后，主页显示welcome

定义welcome组件，在main位置放一个占位符，welcome路由设置为home子路由

1、新建welcom.vue文件，在index.js文件中导入welcome组件

在home后面新增children属性，存放子路由路径

children: [{ path: '/welcome', component: Welcome }]

2、页面访问home主页自动显示welcome，index.js中Home重定向

redirect: '/welcome',

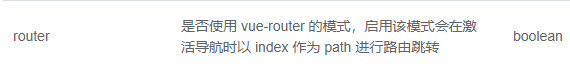
3、在home.vue文件中右侧主体部分添加占位符

<router-view></router-view>

### 侧边栏路由链接

点击不同的二级菜单，右侧主体区域显示不同页面，把每个二级菜单改成路由链接

1、侧边栏菜单区域el-menu，为菜单添加路由属性，改为true



:router="true"或者router

2、跳转地址index，地址要以/开头

<el-menu-item :index="'/'+subItem.path"

## 用户管理—用户列表

### 用户列表组件展示



通过路由展示用户列表页面

1、新建User.vue文件

2、index.js添加路由，添加在home下的子路由中

### 左菜单栏激活状态保存

点击列表，没有高亮：将每一次点击的地址保存在sessionStorage中



1、二级菜单绑定一个单击事件，把地址保存起来

@click="saveNavState('/'+subItem.path)"

2、methods保存链接激活状态

3、动态属性绑定:default-active="activePath"

4、created生命周期函数中赋值

this.activePath = window.sessionStorage.getItem("activePath");

### 用户列表数据获取

1、发起数据请求：users.vue建立基本结构：data，created，methods

2、data数据中return获取的用户列表参数queryInfo

3、methods中函数发起Ajax请求getUserList，get会获取Promise对象，通过async,await优化异步操作，用await会得到一个数据对象，从该对象解构出一个data属性，并重命名为res

### 用户列表数据渲染



1、el-table表格，：data绑定数据源，其中可以嵌套很多模板列el-table-column，每一列中label为列标题，prop为当前列指向的数据

2、border加边框，stripe斑马条纹隔行变色

3、添加索引列type="index"

### 用户列表状态列效果自定义插槽

1、通过作用域插槽，来渲染状态列：在状态列中定义了作用域插槽template，通过slot-scope="scope"接收当前作用域的数据，通过scope.row获取这一行的数据，v-model绑定对应的属性值

### 分页效果实现

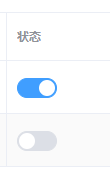


1、element-ui库中赋值代码，el-pagination，在element.js中引入

2、@size-change：切换菜单，@current-change：页码值发生变化，:current-page：当前展示第几页，:page-size：当前每页显示几条数据，layout：页面上展示哪些功能组

3、@size-change，@current-change在事件中监听最新的pagesize和页码值，若发生改变，立即发起新的数据请求this.getUserList();

### 用户状态修改



1、监听switch 开关状态的改变userStateChanged，

2、拿到最新的状态值：@change="userStateChanged(scope.row)"

3、发起Ajax请求，调用API接口，保存最新的状态，投到服务器中那个，put请求表示要修改，请求路径中用户id为${userinfo.id}，用户状态${userinfo.mg\_state}

this.$http.put(`users/${userinfo.id}/state/${userinfo.mg\_state}`

### 搜索功能实现



1、为搜索框的文本框双向绑定数据v-model="queryInfo.query"

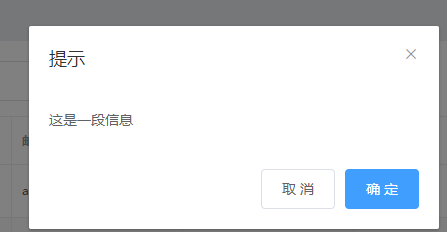
2、为搜索按钮添加点击事件@click="getUserList"

3、添加清空文本框功能clearable属性

4、点击清空按钮立即调用clear事件，在clear中重新发起数据请求

@clear="getUserList"

### 添加用户对话框



1、element.js中按需导入Dialog对话框，在组件库中找到对应结构粘贴到组件中

2、title为标题，:visible.sync显示与隐藏，点击取消或按钮可隐藏对话框@click="addDialogVisible = false"

### 添加用户表单渲染



1、el-form，el-form-item粘贴到组件中，:model="addForm"数据绑定对象，:rules="addFormRules"验证规则对象，ref="addFormRef"引用对象，label-width="70px"前面文本所占宽度，label="用户名"指定文本，prop="username"指定校验规则，v-model="addForm.username"双向数据绑定

2、自定义校验邮箱和手机号：

在data中定义一个校验规则，值指向一个箭头函数，包含三个变量：rule，要校验的值value，回调函数cb，通过validator属性指定校验规则，trigger指定校验时机

### 添加表单的重置操作

1、监听对话框的close事件

2、在close处理函数addDialogClosed中，拿到表单的引用，调用resetFields

添加用户通过调用API接口

1、表单与验证this.$refs.addFormRef.validate

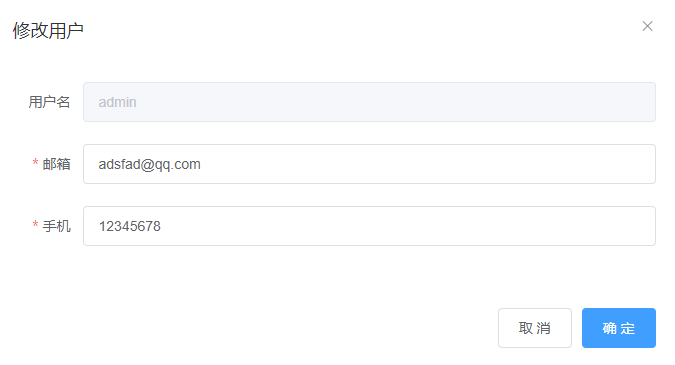
2、发起用户请求：

const { data: res } = await this.$http.post("users", this.addForm);

3、隐藏添加对话框this.addDialogVisible = false;

重新获取用户列表数据this.getUserList();

### 修改表单



1、展示修改用户对话框

为修改按钮绑定事件，只要调用了这个函数就会弹出修改对话框

@click="showEditDialog()"

在methods中添加事件处理函数，展示编辑用户对话框

添加修改用户的对话框组件

默认隐藏对话框this.editDialogVisible = false;

2、根据id查询用户信息

点击对应那行的修改按钮，显示改行用户信息scope.row.id

根据API接口文档，查看请求路径和请求文档，发起数据请求

this.$http.get("users/" + id);根据请求状态判断是否请求成功

在data中定义一个数据对象，专门用来保存用户信息，将查询到的信息保存到editForm

this.editForm = res.data;

3、渲染修改用户的表单

粘贴代码到修改用户对话框的内容区

:model="editForm"为值默认，:rules="editFormRules"验证规则，ref="editFormRef"表单引用对象

用户名中添加属性disabled则可禁用

在data的return中添加修改表单的验证规则对象editFormRules:

4、表单预验证

在确定按钮中添加点击事件@click="editUserInfo"

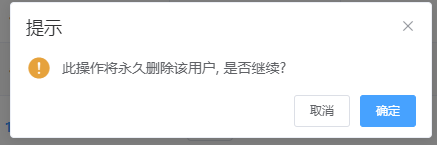
判断预验证是否成功，成功则发起数据请求

5、提交表单

发起请求this.$http.put("users/" + this.editForm.id,

关闭对话框，刷新数据列表，提示修改成功

### 删除用户



1、element.js按需导入，全局挂载，可通过this. $confirm弹框

import MessageBox,

Vue.prototype.$confirm = MessageBox.confirm

2、删除按钮绑定事件<el-tooltip effect="dark" content="分配角色

在metohds中添加方法：根据Id删除对应的用户信息

弹框询问用户是否删除数据$confirm

如果用户确认删除，则返回值为字符串 confirm，

如果用户取消了删除，则返回值为字符串 cancel，弹框已取消删除

3、确认删除时，需发起请求删除对应数据，查看API接口文档的请求路径和请求方法delete和成功状态码

const { data: res } = await this.$http.delete("users/" + id);

若状态码为200则请求成功，提示删除用户成功，刷新列表

### 分配角色

对话框渲染并请求数据

1、分配角色对话框<el-dialog title="分配角色"

2、控制分配角色对话框的显示与隐藏

setRoleDialogVisible: false,

分配新角色select下拉菜单el-select，并导入import

3、[分配角色按钮绑定@click="setRole(scope.row)](mailto:分配角色按钮绑定@click=%22setRole(scope.row))"

向服务端发起请求，弹框，更新

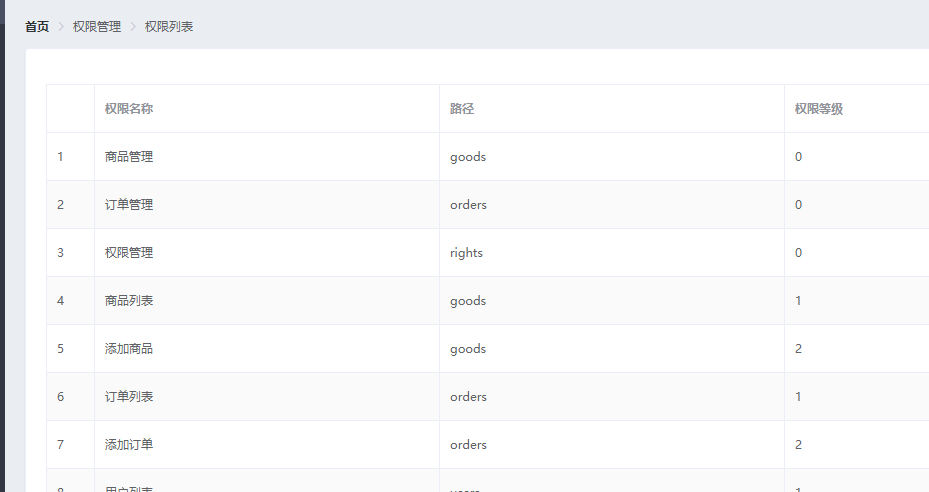
4、对话框关闭重置选项@close="setRoleDialogClosed"

设置为空 this.selectedRoleId = "";

this.userInfo = {};

## 权限管理—权限列表

创建组件，对应的路由规则



### 权限列表展示通过路由

1、新建Rights.vue初始化

2、通过路由加载，在index.js中导入

### 列表数据获取通过API

1、绘制面包屑导航el-breadcrumb和卡片示图el-card

2、请求列表数据

data中return权限列表rightsList:[]

created发起数据请求，获取所有的权限this.getRightsList()

methods定义函数，获取权限列表getRightsList()

若状态为200则获取成功，将获取数据赋值给列表

this.rightsList=res.data;

### 渲染权限列表UI结构

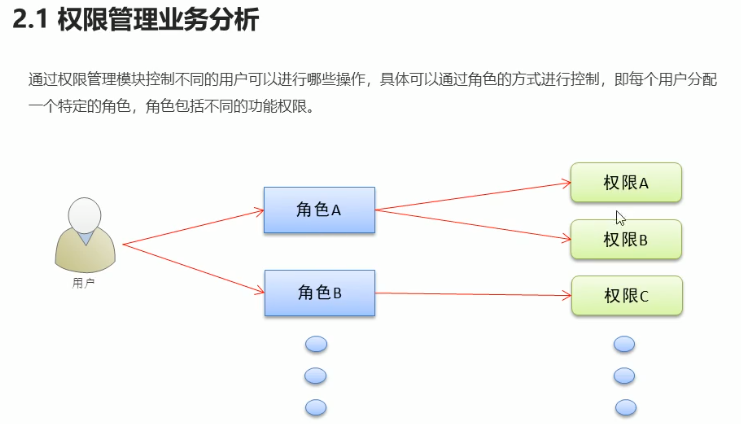
1、表格，引入数据<el-table :data="rightsList" >

2、通过作用域插槽来自定义权限等级格式

<template slot-scope="scope">

3、通过v-if-else决定显示哪个标签

## 用户-角色-权限关系



## 权限管理—角色列表



### 角色列表展示通过路由

1、新建Roles.vue初始化，<template>，<script>，<style>

2、在index.js中导入，导入组件，通过路由加载

### 列表结构布局与数据获取

1、绘制面包屑导航el-breadcrumb和卡片示图el-card

2、请求列表数据

data中return权限列表rolelist:[]

created发起数据请求，获取所有的权限this.getRolesList()

methods定义函数，获取权限列表getRolesList(){}

函数中获取数据：异步简化

const { data: res } = await this.$http.get('roles')

判断状态，若状态为200则获取成功，将获取数据赋值给列表

this.rightsList=res.data;

若失败则提示获取角色列表失败！

return this.$message.error("获取角色列表失败！");

### 渲染角色列表数据

1、设计表格，引入数据<el-table :data="rightsList" >

为表格添加索引列<el-table-column type="index">

2、通过作用域插槽来自定义权限等级格式

<template slot-scope="scope">

索引列用prop绑定对应的数据值

3、添加展开列

<el-table-column type="expand">

### 角色列表功能模块

1、角色下权限渲染

定义作用域插槽，slot-scope接收这一行数据

<template slot-scope="scope">

2、通过第一层for循环渲染一级权限

scope.row为角色数据对象，.children为一级权限的数组

v-for="(item1, i1) in scope.row.children"

每循环一次往tag放一个标签

<el-tag >{{item1.authName}}</el-tag>

添加右三角图标

<i class="el-icon-caret-right"></i>

3、通过第二层for循环嵌套渲染二级权限

v-for循环v-for="(item2, i2) in item1.children" :key="item2.id"

往tag放一个绿色标签type="success"

添加右三角图标

4、通过第三层for循环嵌套渲染三级权限

同上

### 删除权限按钮弹框

1、可移除标签，为标签添加closable

2、为标签绑定点击事件，根据id删除对应权限

@close="removeRightById

3、弹框提示用户是否要删除$confirm

### 删除角色指定权限功能

1、为函数添加参数async removeRightById(role,rightId)

2、发起请求，请求方式delete，请求地址roles/${role.id}/rights/${rightId}

const{data:res}=await this.$http.delete(`roles/${role.id}/rights/${rightId}`)

3、若状态不为200，请求失败，弹出错误信息$message.error

return this.$message.error('删除权限失败！')

4、若成功，为当前角色信息重新赋值权限role.children=res.data

### 弹出分配权限对话框并请求权限数据

1、为分配权限按钮绑定点击事件

@click="showSetRightDialog(scope.row)"

2、在methods中添加/展示分配权限的对话框函数

3、添加分配权限的对话框<el-dialog>

4、获取所有请求数据，并判断状态，若成功数据赋值

async showSetRightDialog() {

const{data:res} = await this.$http.get('rights/tree')

if(res.meta.status!==200){

return this.$message.error('获取权限数据失败！')

}

// 把获取的权限数据保存到data

this.rightlist=res.data

this.setRightDialogVisible = true

}

### 分配权限配置tree树形控件

1、添加tree树形控件<el-tree>，import tree

:data数据绑定，提供数据源:data="rightslist"

:props指定属性绑定对象:props="treeProps"

2、添加复选框show-checkbox

3、点击后默认选定id值，添加树形node-key="id"

默认展开所有节点default-expand-all

默认勾选的节点数组:default-checked-keys="defKeys"

4、点击分配权限按钮，将所有三级节点的id保存在数组中

递归获取三级节点的Id

this.getLeafKeys(role, this.defKeys)

通过递归的形式，获取角色下所有三级权限的id，并保存到 defKeys 数组中getLeafKeys(node, arr)

如果当前 node 节点不包含 children 属性，则是三级节点

通过forEach递归找到所有三级节点

node.children.forEach(item => this.getLeafKeys(item, arr))

5、关闭分配权限时，清空所有项

为按钮绑定点击事件setRightDialogClosed

将数组重置为空this.defKeys = []

### 调用API完成分配权限

1、确定按钮绑定分配权限函数@click="allotRights

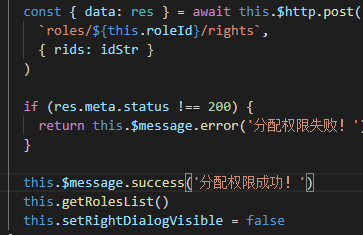
2、拿到所有半选的和全选的数组，通过…合并成一个数组

...this.$refs.treeRef.getCheckedKeys(),

...this.$refs.treeRef.getHalfCheckedKeys()

以逗号分隔const idStr = keys.join(',')

3、向服务器端发起请求，判断状态，弹框提示，刷新数据列表，隐藏对话框



# 分类管理—商品分类



### 商品分类展示通过路由

1、新建Cate.vue初始化，<template>，<script>，<style>

2、在index.js中导入，导入组件，通过路由加载

### 商品分类基本页面布局

1、绘制面包屑导航el-breadcrumb和卡片示图el-card

放一个行，行里放个列，放个按钮

<el-row> <el-col><el-button></el-button></el-col></el-row>

### 商品分类列表数据获取API

1、查询API文档，获取参数

data中return 查询数据querInfo商品分类的数据列表catelist:[]

2、声明周期函数created() {this.getCateList()}

3、定义函数，获取商品分类数据getCateList()，发起请求

### 商品分类树形表格



1、在vue可视化面板中安装依赖vue-table-with-tree-grid

在main.js中将插件导入项目，并注册为全局可用的组件

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

Vue.component('tree-table', TreeTable)

2、布局表格<tree-table，商品分类的数据列表catelist:[],为table指定列的定义columns,通过各种属性决定表格样式

### 自定义模板渲染表格数据模板



1、定义模板

{ label: "是否有效",

// 当前列定义为模板列

type: "template",

// 当前列使用模板名称

template: "isok"}

2、作用域插槽引用模板<template slot="isok" slot-scope="scope">

slot-scope="scope"接收当前行数据

v-if-else来决定显示的图标

排序，操作模块同上

### 商品分类实现分页



1、element库中完整功能的代码，<el-pagination

2、@size-change="handleSizeChange"，监听pagesize改变函数

@current-change="handleCurrentChange"，监听pagenum改变函数

:current-page="querInfo.pagenum"，当前渲染的页码

:page-size="querInfo.pagesize"，当前每页显示多少条

layout当前分页去有哪些部分组成

:total="total"查询的总数据

定义函数handleSizeChange(newSize)，handleCurrentChange(newPage)

### 添加分类的对话框和表单



1、在element库中找到dialog对话框代码粘贴到添加分类的对话框

:visible.sync="addCateDialogVisible"对话框的显示与隐藏

2、点击添加分类按钮对话框弹出，因此为按钮绑定事件

showAddCateDialog，并定义函数显示对话框

3、在对话框内完善表单

将组件库里的表单验证对应代码粘贴到对话框内部

:model="addCateForm"添加分类的表单数据对象

:rules="addCateFormRules"规则

ref="addCateFormRef"引用

4、分类名称输入框双向数据绑定，绑定对象看API接口文档中1.6.2. 添加分类的分类名称

v-model="addCateForm.cat\_name"

验证规则prop="cat\_name"

5、父级分类

发起请求，获取父级分类的数据列表

### 父级分类级联选择器cascader渲染

1、<el-cascader级联选择器，element.js按需导入

2、expand-trigger="hover"通过鼠标覆盖展开子菜单

:options="parentCateList"数据源

:props="cascaderProps"用来指定配置对象

v-model="selectedKeys"选中的父级分类数据

@change="parentCateChanged"// 选择项发生变化触发这个函数

3、设置级联选择器高度在global.css

.el-cascader\_\_dropdown {height: 200px;}

.el-cascader-panel {height: 200px;}

4、如果 selectedKeys 数组中的 length 大于0，证明选中的父级分类

反之，就说明没有选中任何父级分类

5、// 监听对话框的关闭事件，重置表单数据

@close="addCateDialogClosed"

### 添加分类

1、拿到表单的引用，进行表单预验证，回调函数拿到预校验结果

this.$refs.addCateFormRef.validate

2、发起请求

# 分类管理—分类参数

商品参数用于展示商品固定的特征信息，有动态参数和静态属性

只允许为第三级分类设置相关的参数

动态参数是用户购买商品时可进行选择的

静态属性供用户查看商品信息

### 路由加载分类参数组件页面

1、新建Params.vue初始化，<template>，<script>，<style>

2、在index.js中导入，导入组件，通过路由加载

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

<script>export default {

data(){

return{}

},

created(){},

methods:{}}

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

### 渲染分类参数页面的基本UI结构

1、绘制面包屑导航el-breadcrumb和卡片示图el-card

2、头部警告区域<el-alert>，没有用过需要在element.js按需导入组件

3、title显示文字，show-icon显示默认图标

### 调用API获取数据

1、查询API文档，获取参数

data中return 商品分类的数据列表catelist:[]

2、声明周期函数created() {this.getCateList()}

3、定义函数，获取商品分类数据getCateList()，发起请求

### 级联选择框渲染

1、组件库中<el-cascader

:options="catelist"数据源

:props="cateProps"配置级联选择框

v-model="selectedCateKeys"双向数据绑定

@change="handleChange"级联选择框选中项变化，会触发这个函数