后台管理项目

一、创建项目

```
vue init webpack backProject(项目名称)
```

二、启动后端服务器之导入表

首先链接3306，如果有就双击运行
然后，右键新建数据，自定义名字，选择utf-8
然后双击打开，在表上右键选择运行sql,找到sql文件点开始，成功

三、运行后端服务器

一、解压缩
如果没有Node_modules npm isntall 安装一下
二、修改config =>global.js

```
exports.dbConfig = {
    host: 'localhost',
    user: 'root', //你的用户名
    password: '123',//你的密码
    port: 3306, //你的数据库端口号
    database: 'ushop' // 你的数据库名字
}
```

三、注释掉app.js中的token拦截
大概 37-46

```
/* app.use(async (req, res, next) => {
    if (!req.headers.authorization) {
```

```
        res.send(MError("请设置请求头,并携带验证字
符串"));
    } else {
        if (!await checkToken(req)) { // 过期
            res.send(Guest([],"登录已过期或访问权
限受限"));
        } else {
            next();
        }
    }
}); */
```

四、如果你想修改端口
bin文件=>www文件中，大概15行的位置去修改

```
var port = normalizePort(process.env.PORT ||
'3000');
```

修改之后一定要记得重启！！！

五、启动
npm start

四、分析当前项目技术栈下载相关插件

```
element-ui ui框架
axios  http库
vue-router 路由插件
vuex  状态管理插件
stylus css预处理器
stylus-loader@3.0.2

版本号:
+ axios@0.21.0
+ vuex@3.5.1
+ stylus-loader@4.2.0
+ stylus@0.54.8
+ element-ui@2.14.0
```

五、分析项目创建出文件夹以及组件

src目录

```
src目录
assets  静态资源
    js  js文件
    css css文件
        reset.css 用于重置样式
    images
        静态资源图片
common  共通组件
components 或者共通组件放置到这里
filter  全局过滤器的封装
router  路由文件
store   vuex仓库
    index.js
util    工具管理
    axios   http库文件夹
```

```
        axios.js    新的axios实例
        index.js    接口模块
pages   一级路由组件
views   二级路由组件
```

router=>index.js

```
import Vue from 'vue'
import Router from 'vue-router'

Vue.use(Router)

export default new Router({
  routes: [
    {
      path:'/login',
      component:()=>import('@/pages/login')
    },
    {
      path:'/index',
      component:()=>import('@/pages/index')
    },
    {//重定向
      path:'*',
      redirect:'/login'
    }
  ]
})
```

store=>index.js

```
//引入核心库
import Vue from 'vue'
//引入vuex
import Vuex from 'vuex'
//调用vuex插件
Vue.use(Vuex)

//导出实例化的仓库
export default new Vuex.Store({})
```

app.vue

```
<template>
  <div id="app">
    <!-- 一级路由出口 -->
    <router-view/>
  </div>
</template>

<script>
export default {
  name: 'App'
}
</script>

<style>

</style>
```

main.js

```
import Vue from 'vue'
```

```
import App from './App'
import router from './router'
//引入vuex状态管理
import store from './store'
//引入重置样式
import './assets/css/reset.css'
//全局引入element-ui
import ElementUI from 'element-ui'
//引入elementUIcss样式
import 'element-ui/lib/theme-chalk/index.css'
//调用插件
Vue.use(ElementUI)

Vue.config.productionTip = false

/* eslint-disable no-new */
new Vue({
  el: '#app',
  router,
  store,
  components: { App },
  template: '<App/>'
})
```

util=>axios=>axios.js

```
//引入axios核心库
import axios from 'axios'

//重新创建一个实例
const http = axios.create()

//导出http实例
export default htttp
```

util=>axios=>index.js

```
//引入重新封装的axios实例
import http from './axios'

//暴露（导出）接口
```

stylus=>index.styl(css预处理器)

```
@import './bgColor.styl'
```

六、搭建登录页面

```
<template>
  <div class="login">
    <!--
        el-form 属性 model 表单数据对象 rules 表单验证规则
        el-input clearable可清空    show-password 是否显示密码
    -->
    <el-form
      :model="loginForm"
```

```
      :rules="rules"
      ref="loginForm"
      label-width="100px"
      class="loginForm"
    >
      <el-form-item label="用户名"
prop="username">
        <el-input v-model="loginForm.username"
clearable></el-input>
      </el-form-item>
      <el-form-item label="密码"
prop="password">
        <el-input
          v-model="loginForm.password"
          show-password
          clearable
        ></el-input>
      </el-form-item>
      <el-form-item>
        <el-button type="primary"
@click="login('loginForm')">登录</el-button>
      </el-form-item>
    </el-form>
  </div>
</template>

<script>
export default {
  data() {
    return {
      //表单输入参
      loginForm: {
        username: "",
```

```
        password: ""
      },
      //规则验证
      rules: {
        //用户名验证
        username: [
          //必填项验证
          { required: true, message: "请输入用户名", trigger: "blur" },
          //字符验证
          { min: 2, max: 15, message: "长度在 2 到 15 个字符", trigger: "blur" }
        ],
        //密码验证
        password: [
          { required: true, message: "请输入密码", trigger: "blur" },
          { min: 6, max: 18, message: "长度在 6 到 18 个字符", trigger: "blur" }
        ]
      }
    };
  },
  methods: {
    //登录
    login(formName) {

 //console.log(this.$refs[formName],'formDOM')
      //validate 验证
      this.$refs[formName].validate(valid => {
        if (valid) {
          //登录逻辑
          //调取登录接口
```

```
            if (
              this.loginForm.username == "admin"
&&
              this.loginForm.password == "123456"
            ) {
              this.$message.success("登录成功");
              this.$router.push("/index");
            } else {
              this.$message.error("用户名或者密码不
正确");
            }
          } else {
            console.log("error submit!!");
            return false;
          }
        });
      }
    }
};
</script>

<style lang="stylus" scoped>
@import '../stylus/index.styl'
.login
  width 100vw
  height 100vh
  background $fristBgColor
  .loginForm
    position absolute
    left 50%
    top 50%
    margin -150px 0 0 -150px
    width 400px
```

```
    height 220px
    padding 35px 10px
    // background $thirdBgColor
</style>
```

七、拆分导航组件

```
<template>
    <div>
   <!--
            导航菜单
            el-menu属性
          background-color  菜单的背景色（仅支持 hex 格式）
          text-color    菜单的文字颜色（仅支持 hex 格式）
          active-text-color 当前激活菜单的文字颜色（仅支持 hex 格式）
          unique-opened 是否只保持一个子菜单的展开
          default-active    当前激活菜单的 index
          router    是否使用 vue-router 的模式，启用该模式会在激活导航时以 index 作为 path 进行路由跳转
          -->
          <el-menu
            default-active="2"
            class="el-menu-vertical-demo"
            background-color="#585858"
            text-color="#fff"
            active-text-color="#FF8000"
            unique-opened
          >
            <el-menu-item index="2">
```

```
              <i class="el-icon-s-grid"></i>
              <span slot="title">首页</span>
            </el-menu-item>
            <el-submenu index="1">
              <template slot="title">
                <i class="el-icon-s-tools"></i>
                <span>系统管理</span>
              </template>
              <el-menu-item index="3">
                <span slot="title">菜单管理
</span>
              </el-menu-item>
              <el-menu-item index="4">
                <span slot="title">角色管理
</span>
              </el-menu-item>
              <el-menu-item index="5">
                <span slot="title">管理员管理
</span>
              </el-menu-item>
            </el-submenu>
            <el-submenu index="2">
              <template slot="title">
                <i class="el-icon-s-tools"></i>
                <span>商城管理</span>
              </template>
              <el-menu-item index="6">
                <span slot="title">商品分类
</span>
              </el-menu-item>
              <el-menu-item index="7">
                <span slot="title">商品规格
</span>
```

```
            </el-menu-item>
          </el-submenu>
        </el-menu>
    </div>
</template>

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

        };
    },
};
</script>

<style  lang="stylus" scoped>
.el-menu
  height 90vh
</style>
```

八、创建index基本容器布局

```
<template>
  <div>
    <el-container>
      <el-header>xxx大型后台管理项目</el-header>
      <el-container>
        <el-aside width="200px">
          <!-- 引入导航菜单 -->
          <v-nav></v-nav>
        </el-aside>
```

```
        <el-main>
          <!-- 二级路由出口 -->
          <router-view></router-view>
        </el-main>
      </el-container>
    </el-container>
  </div>
</template>

<script>
//引入nav导航组件
import vNav from "../components/nav";
export default {
  data() {
    return {};
  },
  components: {
    vNav
  }
};
</script>

<style lang="stylus" scoped>
@import '../stylus/index.styl'
.el-header
    background $thirdBgColor
</style>
```

九、通过点击左侧菜单导航渲染右侧内容

router属性，可以把index当做path地址进行路由的跳转

```
<el-menu
```

```
    router
>
    <el-menu-item index="/menu">
         <span slot="title">菜单管理</span>
    </el-menu-item>
    <el-menu-item index="/role">
          <span slot="title">角色管理</span>
    </el-menu-item>
    <el-menu-item index="/user">
          <span slot="title">管理员管理</span>
     </el-menu-item>
</el-menu>
```

十、封装面包屑组件

components=>el-bread

```
<template>
    <div>
    <el-breadcrumb separator="/">
      <el-breadcrumb-item :to="{ path: '/home'
}">首页</el-breadcrumb-item>
      <el-breadcrumb-item>{{$route.name}}</el-
breadcrumb-item>
    </el-breadcrumb>
    </div>
</template>

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

```
        };
    },
};
</script>

<style  lang="" scoped>
.el-breadcrumb{
    margin-bottom: 20px;
}
</style>
```

十一、搭建菜单管理的静态视图

1、搭建menu.vue基本视图文件

```
<template>
  <div>
      <!-- 面包屑 -->
      <el-bread></el-bread>
      <!-- 添加按钮 -->
      <el-button type="primary" size='small'
@click='isAdd'>添加</el-button>
      <!-- 表格列表渲染 -->
      <v-table></v-table>
      <!-- 弹框表单 -->
      <v-dialog @cancel='cancel'
:isShow='isShow'></v-dialog>
  </div>
</template>

<script>
import elBread from '../../components/el-bread'
```

```
import vTable from './list'
import vDialog from './add'
export default {
  data() {
    return {
        isShow:false
    };
  },
  components:{
      elBread,
      vTable,
      vDialog
  },
  methods: {
      //打开弹框事件
      isAdd(){
          this.isShow = true
      },
      //cancel事件去改变弹框的属性
      cancel(e){
          this.isShow = e
      }
  },
};
</script>

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

2、创建一个Table列表页

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

```
    <el-table :data="tableData" style="width:
100%">
      <el-table-column prop="date" label="菜单编
号" width="180"> </el-table-column>
      <el-table-column prop="name" label="菜单名
称" width="180"> </el-table-column>
      <el-table-column prop="address" label="上
级菜单"> </el-table-column>
      <el-table-column prop="date" label="菜单图
标" width="180"> </el-table-column>
      <el-table-column prop="name" label="菜单地
址" width="180"> </el-table-column>
      <el-table-column prop="address" label="状
态"> </el-table-column>
      <el-table-column label="操作">
          <template slot-scope="item">
              <div>
                    <el-button type="primary"
size='small'>编辑</el-button>
                    <el-button type="danger"
size='small'>删除</el-button>
              </div>
          </template>
      </el-table-column>
    </el-table>
  </div>
</template>

<script>
export default {
  data() {
    return {
      tableData: [ ]
```

```
    };
  }
};
</script>

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

3、创建一个对话框表单

```
<template>
  <div>
    <el-dialog title="添加菜单"
:visible.sync="isShow">
      <el-form :model="form">
        <el-form-item label="菜单名称" :label-
width="formLabelWidth">
          <el-input v-model="form.name"
autocomplete="off"></el-input>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click='cancel' >取 消</el-
button>
        <el-button type="primary">确 定</el-
button
        >
      </div>
    </el-dialog>
  </div>
</template>

<script>
```

```
export default {
  data() {
    return {
        form:{
            name:''
        },
        formLabelWidth: '120px'
    };
  },
  props:['isShow'],
  methods: {
      //取消事件，关闭弹框，修改父组件数据
      cancel(){
          this.$emit('cancel',false)
      }
  },
};
</script>

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

十二、解决跨域问题

config=>index.js

```
    proxyTable: {
      '/api':{
        target:'http://localhost:3000',
        changeOrigin:true,
        pathRewrite:{
          '^/api':'http://localhost:3000'
        }
      }
    },
```

十三、调取菜单列表

- menu=>list.vue

```
<template>
  <div>
    <el-table :data="menuList" style="width:
100%">
      <el-table-column prop="id" label="菜单编
号" width="180"> </el-table-column>
      <el-table-column prop="title" label="菜单
名称" width="180"> </el-table-column>
      <el-table-column prop="pid" label="上级菜
单"> </el-table-column>
      <el-table-column prop="icon" label="菜单图
标" width="180"> </el-table-column>
      <el-table-column prop="url" label="菜单地
址" width="180"> </el-table-column>
      <el-table-column prop="status" label="状
态"> </el-table-column>
      <el-table-column label="操作">
          <template slot-scope="item">
              <div>
```

```
                    <el-button type="primary"
size='small'>编辑</el-button>
                    <el-button type="danger"
size='small'>删除</el-button>
                </div>
            </template>
        </el-table-column>
    </el-table>
  </div>
</template>

<script>
//引入辅助性函数
import {mapGetters,mapActions} from 'vuex'
export default {
  data() {
    return {
    };
  },
  computed: {
    ...mapGetters({
      menuList:'menu/getMenuList'
    })
  },
  mounted() {
    //组件一加载就触发调取列表
    this.getMenuList()
  },
  methods: {
    ...mapActions({
      getMenuList:'menu/getMenuListAction'
    })
  },
```

```
};
</script>

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

- store=>modules=>menu.js

```
//引入封装好的接口
import {getMenuList} from '../../util/axios'

//state
const state = {
    menuList:[]
}

//getters
const getters = {
    getMenuList(state){
        return state.menuList
    }
}

//mutations
const mutations = {
    REQ_MENULIST(state,payload){
        state.menuList = payload
    }
}

//actions
const actions = {
    getMenuListAction({commit}){
```

```
        getMenuList()
        .then(res=>{
            console.log(res,'响应')
            if(res.data.code==200){

 commit("REQ_MENULIST",res.data.list)
            }
        })
        .catch(err=>{
            console.log(err,'错误响应')
        })
    }
}


//默认导出所有的内容
export default {
    state,
    getters,
    mutations,
    actions,
    //命名空间
    namespaced:true
}
```

十四、完成添加功能（为优化）

注意点！！！

点击添加成功之后，数据库表中没有数据？？？原因：提交参数类型不对，

pid和type以及status是Number

menu=>add.vue

```
<template>
  <div>
    <!--
      对话框属性
      visible    是否显示 Dialog，支持 .sync 修饰符
      center     是否对头部和底部采用居中布局
      formLabelWidth label的宽度
     -->
    <el-dialog title="添加菜单"
:visible.sync="isShow" center>
      <el-form :model="form" :rules="rules">
        <el-form-item
          prop="title"
          label="菜单名称"
          :label-width="formLabelWidth"
        >
          <el-input v-model="form.title"
autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item prop="pid" label="上级菜
单" :label-width="formLabelWidth">
          <el-select v-model="form.pid"
placeholder="请选择">
            <el-option label='顶级菜单'
:value='0'></el-option>
            <el-option
              v-for="item in options"
              :key="item.value"
              :label="item.label"
              :value="item.value"
            >
```

```
            </el-option>
          </el-select>
        </el-form-item>
        <el-form-item label="菜单类型" :label-
width="formLabelWidth">
          <el-radio v-model="form.type"
:label="1">目录</el-radio>
          <el-radio v-model="form.type"
:label="2">菜单</el-radio>
        </el-form-item>
        <el-form-item label="菜单图标" :label-
width="formLabelWidth">
          <el-input v-model="form.icon"
autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item label="菜单地址" :label-
width="formLabelWidth">
          <el-input v-model="form.url"
autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item label="状态" :label-
width="formLabelWidth">
          <!-- 可以使用active-color属性与
inactive-color属性来设置开关的背景色。 -->
          <el-switch
            v-model="form.status"
            active-color="#13ce66"
            inactive-color="#ff4949"
            :active-value='1'
            :inactive-value='2'
          >
          </el-switch>
        </el-form-item>
```

```
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="cancel">取 消</el-
button>
        <el-button @click="add" type="primary">
确 定</el-button>
      </div>
    </el-dialog>
  </div>
</template>

<script>
//引入封装好的接口
import {getMenuAdd} from '../../util/axios'
//引入辅助性函数
import {mapActions} from 'vuex'
export default {
  data() {
    return {
      form: {
        title: "", //标题
        pid: 0, //上级分类编号 默认是0 是顶级
        icon: "", //图标
        type: 1, //类型1目录2菜单
        url: "", //菜单地址
        status: 1 //状态1正常2禁用
      },
      //上级菜单
      options: [
        {
          value: "选项1",
          label: "系统管理"
        },
```

```
          {
            value: "选项2",
            label: "商城管理"
          }
        ],
        rules: {
          title: [
            { required: true, message: "请输入菜单名称", trigger: "blur" },
            { min: 2, max: 20, message: "长度在 2 到 20 个字符", trigger: "blur" }
          ],
          pid: [{ required: true, message: "请选择上级菜单", trigger: "blur" }]
        },
        formLabelWidth: "120px"
      };
    },
    props: ["isShow"],
    methods: {
      ...mapActions({
        getMenuList:'menu/getMenuListAction'
      }),
      //取消事件，关闭弹框，修改父组件数据
      cancel() {
        this.$emit("cancel", false);
      },
      //添加事件
      add(){
        getMenuAdd(this.form)
        .then(res=>{
          console.log(res,'添加响应')
          if(res.data.code==200){
```

```
            this.$message.success(res.data.msg)
            //关闭弹框
            this.cancel()
            //重新获取列表
            this.getMenuList()
          }
        })
      }
    }
};
</script>

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

十五、优化侧边栏点击刷新选中状态

视图

```
//default-active     当前激活菜单的 index
<el-menu
   :default-active="defaultActive"
>
```

逻辑代码

```
    data() {
        return {
          defaultActive:'/home'
        };
    },
    mounted() {
      //组件一加载就会触发挂载
      //console.log(this.$route,'路由源信息')
      this.defaultActive = this.$route.path
```

```
    },
```

十六、添加弹框中联动效果

效果：上级菜单如果是顶级，联动菜单类型是目录，出现菜单图片。上级菜单如果是非顶级，联动菜单类型是菜单，出现菜单地址

视图

```
        <el-form-item prop="pid" label="上级菜单" :label-width="formLabelWidth">
          <el-select
            v-model="form.pid"
            placeholder="请选择"
            @change="changeMenu"
          >
            <el-option label="顶级菜单" :value="0"></el-option>
            <!-- 下拉框循环我的菜单列表 -->
            <el-option
              v-for="item in menuList"
              :key="item.id"
              :label="item.title"
              :value="item.id"
            >
            </el-option>
          </el-select>
        </el-form-item>
        <el-form-item label="菜单类型" :label-width="formLabelWidth">
```

```
          <el-radio v-model="form.type"
:label="1" disabled>目录</el-radio>
          <el-radio v-model="form.type"
:label="2" disabled>菜单</el-radio>
        </el-form-item>
        <el-form-item
          label="菜单图标"
          :label-width="formLabelWidth"
          v-if="form.type === 1"
        >
          <el-select v-model="form.icon"
placeholder="请选择">
            <!-- 下拉框循环我的菜单列表 -->
            <el-option
              v-for="item in iconList"
              :key="item"
              :label="item"
              :value="item"
            >
            </el-option>
          </el-select>
        </el-form-item>
        <el-form-item label="菜单地址" :label-
width="formLabelWidth" v-else>
          <!-- <el-input v-model="form.url"
autocomplete="off"></el-input> -->

          <el-select v-model="form.url"
placeholder="请选择">
            <!-- 下拉框循环我的菜单列表 -->
            <el-option
              v-for="item in indexRoutes"
              :key="item.path"
```

```
              :label="item.name"
              :value="item.path"
            >
            </el-option>
          </el-select>
        </el-form-item>
```

逻辑代码

```
//引入辅助性函数
import { mapGetters } from "vuex";
//引入封装好的二级路由
import { indexRoutes } from "../../router";

export default {
  data() {
    return {
      indexRoutes: indexRoutes,
      iconList: [
        "el-icon-s-tools",
        "el-icon-setting",
        "el-icon-s-goods",
        "el-icon-goods",
        "el-icon-menu"
      ]
    };
  },

  computed: {
    //获取菜单列表的值
    ...mapGetters({
      menuList: "menu/getMenuList"
    })
```

```
  },
     //下拉菜单修改触发的事件
    changeMenu() {
      console.log(this.form.pid, "pid");
      //通过用户是否选择顶级菜单来渲染你的菜单类型
      if (this.form.pid == 0) {
        this.form.type = 1;
      } else {
        this.form.type = 2;
      }
    }
```

重新封装二级路由

```
export let indexRoutes = [
  {
    path: "/menu",
    component: () =>
import("@/views/menu/menu"),
    name: "菜单管理"
  },
  {
    path: "/role",
    component: () =>
import("@/views/role/role"),
    name: "角色管理"
  },
  {
    path: "/user",
    component: () =>
import("@/views/user/user"),
    name: "管理员管理"
  }
```

```
];


{
      path: "/index",
      component: () => import("@/pages/index"),
      children: [
        {
          path: "/home",
          component: () =>
import("@/views/home")
        },
        ...indexRoutes,
        {
          //二级路由重定向
          path: "",
          redirect: "/menu"
        }
      ]
    },
```

十七、侧边栏菜单动态渲染

视图

```
            <el-submenu
:index="item.id.toString()" v-for="item in
menuList" :key='item.id'>
              <template slot="title">
                <i :class="item.icon"></i>
                <span>{{item.title}}</span>
              </template>
              <el-menu-item :index="menu.url"
v-for="menu in item.children" :key='menu.id'>
                <span slot="title">
{{menu.title}}</span>
              </el-menu-item>
            </el-submenu>
```

逻辑代码

```
import {mapGetters,mapActions} from 'vuex'
mounted() {
      //组件一加载就会触发挂载
      //console.log(this.$route,'路由源信息')
      this.defaultActive = this.$route.path
      //组件加载触发action
      this.getMenuList()
    },
    computed: {
      ...mapGetters({
        menuList:'menu/getMenuList'
      })
    },
    methods: {
      ...mapActions({
        getMenuList:'menu/getMenuListAction'
      })
```

```
    },
```

十八、调取删除接口完善删除功能

list.vue

视图

```
            <el-button type="danger"
@click="del(item.row.id)" size="small"
              >删除</el-button
            >
```

逻辑

```
    //封装一个删除事件
    del(id) {
      this.$confirm("确定要删除这条数据吗?! ! ",
"提示", {
        confirmButtonText: "确定",
        cancelButtonText: "取消",
        type: "warning"
      })
        .then(() => {
          //调取删除接口
          getMenuDelete({ id }).then(res => {
            if (res.data.code == 200) {

 this.$message.success(res.data.msg);
              //重新触发调取列表
              this.getMenuList();
            }
          });
        })
```

```
        .catch(() => {
          this.$message({
            type: "info",
            message: "已取消删除"
          });
        });
    }
```

十九、完善编辑功能并调取接口

1、table表格点击编辑传送id给父组件menu

- 视图

```
<el-button type="primary" size="small"
@click='edit(item.row.id)'>编辑</el-button>
```

- 逻辑

```
    //点击编辑获取id
    edit(id){
      //把id给父组件menu,之后供弹框使用
      this.$emit('edit',id)
    },
```

2、点击table编辑按钮联动触发menu中edit事件打开弹框并修改弹框状态为编辑

- 视图

```
    <!-- 表格列表渲染 -->
    <v-table @edit='edit'></v-table>
```

- 逻辑

```
    //编辑事件
    edit(e){
      //e是表格传递的id
      //这个事件一触发，就知道是编辑了，就要修改弹框状
态
      //打开弹框
      this.addInfo.isShow=true
      //告诉弹框你是一个编辑
      this.addInfo.isAdd =false
      //获取弹框的函数
      // console.log(this.$refs.dialog)
      this.$refs.dialog.look(e)
    }
```

3、点击编辑获取当前数据

- 视图

```
<!-- 弹框表单 -->
<v-dialog ref='dialog' ></v-dialog>
```

- 逻辑

```
    //查看一条事件
    look(id) {
      // console.log(id,'形参')
      //调取查看列表
      getMenuInfo({ id }).then(res => {
        // console.log(res,'一条数据')
        if (res.data.code == 200) {
          this.form = res.data.list;
          //当前表单没有id 创建一个id，给确定提交用
          this.form.id = id;
        } else {
```

```
            this.$message.error(res.data.msg);
          }
        });
    },
```

4、点击确定编辑提交表单并调取修改接口

- 视图

```
<el-button v-else @click="update('formDialog')"
type="primary"
           >编 辑</el-button>
```

- 逻辑

```
    //确定修改事件
    update(formName) {
      this.$refs[formName].validate(valid => {
        if (valid) {
          //修改逻辑
          //调取修改接口
          getMenuEdit(this.form).then(res => {
            console.log(res, "修改响应");
            if (res.data.code == 200) {

this.$message.success(res.data.msg);
              //关闭弹框
              this.cancel();
              //重新获取列表
              this.getMenuList();
            } else {

this.$message.error(res.data.msg);
            }
```

```
          });
        } else {
          console.log("error submit!!");
          return false;
        }
      });
    },
```

二十、搭建角色管理静态视图

角色管理的作用：用于区分不同的账号

一般情况：一个后台管理系统一定有一个超集管理员：admin（一般命名为）,它拥有所有的权限，它可以分配账号可以分配权限。其他不同的账号可能含有不同的权限，那么如何分配权限，就在角色管理中做

1、role.vue

```
<template>
    <div>
      <!-- 面包屑 -->
      <el-bread></el-bread>
      <!-- 添加按钮 -->
      <el-button type="primary" plain
size='small' @click='add'>添加</el-button>
      <!-- 列表渲染 -->
      <v-table></v-table>
      <!-- 弹框 -->
      <v-dialog @cancel='cancel'
:addInfo='addInfo'></v-dialog>
    </div>
</template>
```

```
<script>
import elBread from '../../components/el-bread'
import vTable from './list'
import vDialog from './add'
export default {
    data() {
        return {
          addInfo:{
            isShow:false,//弹框打开关闭
            isAdd:true,//添加还是编辑
          }
        };
    },
  components:{
      elBread,
      vTable,
      vDialog
  },
  methods: {
    //点击添加打开弹框
    add(){
      this.addInfo.isShow = true
    },
    //子组件触发的父组件修改弹框状态
    cancel(){
      this.addInfo.isShow =false
    }
  },
};
</script>

<style  lang="stylus" scoped>
```

```
.el-button
  margin-bottom 20px
</style>
```

2、list

```
<template>
  <div>
    <el-table :data="tableData" border
style="width: 100%">
      <el-table-column prop="id" label="角色编
号" width="180"> </el-table-column>
      <el-table-column prop="rolename"
label="角色名称" width="180"> </el-table-column>
      <el-table-column prop="status" label="状
态"> </el-table-column>
      <el-table-column label="操作">
          <template slot-scope="item">
              <div>
                <el-button type="info" plain>编
辑</el-button>
                <el-button type="info" plain>删
除</el-button>
              </div>
          </template> </el-table-column>
    </el-table>
  </div>
</template>

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

```
    return {
      tableData: []
    };
  }
};
</script>

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

3、add.vue

```
<template>
  <div>
    <el-dialog
      title="添加角色"
      :visible.sync="addInfo.isShow"
      :before-close="cancel"
      center
    >
      <el-form :model="form" :rules="rules">
        <el-form-item prop='rolename' label="角色名称:" :label-width="formLabelWidth">
          <el-input v-model="form.rolename" autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item label="角色权限:" :label-width="formLabelWidth">
            <!-- el-tree 属性
            data     展示数据     array
            default-expand-all   是否默认展开所有节点
```

```
            node-key    每个树节点用来作为唯一标识的
属性，整棵树应该是唯一的
            props    配置选项，具体看下表   object
            show-checkbox    节点是否可被选择 -->
          <el-tree
          v-model="form.menus"
            :data="data"
            default-expand-all
            node-key="id"
            ref="tree"
            :props="defaultProps"
            show-checkbox
          >
          </el-tree>
        </el-form-item>
        <el-form-item label="状态:" :label-
width="formLabelWidth">
          <el-switch
            v-model="form.status"
            active-color="#13ce66"
            inactive-color="#ff4949"
            :active-value="1"
            :inactive-value="2"
          >
          </el-switch>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="cancel">取 消</el-
button>
        <el-button type="primary">确 定</el-
button>
      </div>
```

```
    </el-dialog>
  </div>
</template>

<script>
export default {
  data() {
    return {
      //表单对象
      form: {
        rolename: "", // 角色名称
        menus: "", //角色权限
        status: 1 //状态1正常2禁用
      },
      //label宽度
      formLabelWidth: "120px",
      //规则验证
      rules:{
          rolename: [
            { required: true, message: '请输入角色名称', trigger: 'blur' },
            { min: 2, max: 8, message: '长度在 2 到 8 个字符', trigger: 'blur' }
          ],
      },
      data: [{
          id: 1,
          label: '一级 1',
          children: [{
            id: 4,
            label: '二级 1-1',
            children: [{
              id: 9,
```

```
              label: '三级 1-1-1'
            }, {
              id: 10,
              label: '三级 1-1-2'
            }]
          }]
        }, {
          id: 2,
          label: '一级 2',
          children: [{
            id: 5,
            label: '二级 2-1'
          }]
        }],
        defaultProps: {
          children: 'children',
          label: 'label'
        }
    };
  },
  props: ["addInfo"],
  methods: {
    //点击取消修改父组件弹框事件
    cancel() {
      this.$emit("cancel", false);
    }
  }
};
</script>

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

二十一、角色管理逻辑交互

1、role.vue

```
<template>
    <div>
      <!-- 面包屑 -->
      <el-bread></el-bread>
      <!-- 添加按钮 -->
      <el-button type="primary" plain
size='small' @click='add'>添加</el-button>
      <!-- 列表渲染 -->
      <v-table @edit='edit'></v-table>
      <!-- 弹框 -->
      <v-dialog ref='dialog' @cancel='cancel'
:addInfo='addInfo'></v-dialog>
    </div>
</template>

<script>
import elBread from '../../components/el-bread'
import vTable from './list'
import vDialog from './add'
export default {
    data() {
        return {
          addInfo:{
            isShow:false,//弹框打开关闭
            isAdd:true,//添加还是编辑
          }
        };
    },
  components:{
```

```
        elBread,
        vTable,
        vDialog
    },
    methods: {
      //点击添加打开弹框
      add(){
        this.addInfo.isShow = true
      },
      //子组件触发的父组件修改弹框状态
      cancel(){
        this.addInfo.isShow =false
      },
      //子组件触发父组件联动编辑
      edit(e){
        this.addInfo.isShow =true
        this.addInfo.isAdd = false
        this.$refs.dialog.look(e)
      }
    },
};
</script>

<style  lang="stylus" scoped>
.el-button
  margin-bottom 20px
</style>
```

2、list.vue

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

```
    <el-table :data="roleList" border
style="width: 100%">
      <el-table-column prop="id" label="角色编
号" width="180">
      </el-table-column>
      <el-table-column prop="rolename"
label="角色名称" width="180">
      </el-table-column>
      <el-table-column prop="status" label="状
态">
        <template slot-scope="item">
          <div>
            <el-tag v-if="item.row.status == 1"
type="success">启用</el-tag>
            <el-tag v-else type="danger">禁用
</el-tag>
          </div>
        </template>
      </el-table-column>
      <el-table-column label="操作">
        <template slot-scope="item">
          <div>
            <el-button type="info" plain
@click='edit(item.row.id)'>编辑</el-button>
            <el-button
@click='del(item.row.id)' type="danger" plain>
删除</el-button>
          </div>
        </template>
      </el-table-column>
    </el-table>
  </div>
</template>
```

```
<script>
//引入辅助性函数
import {mapActions,mapGetters} from 'vuex'
//引入封装好的接口
import {getRoleDelete} from '../../util/axios'
export default {
  data() {
    return {
    };
  },
  computed: {
    ...mapGetters({
      roleList:'role/getRoleList'
    })
  },
  mounted() {
    //组件一加载触发行动
    this.getRoleAction()
  },
  methods: {
    //获取行动
    ...mapActions({
      getRoleAction:'role/getRoleAction'
    }),
    //封装一个删除事件
    del(id){
        this.$confirm('确定要删除这一条数据
吗？？？！！！', '提示', {
          confirmButtonText: '确定',
          cancelButtonText: '取消',
          type: 'warning'
        }).then(() => {
```

```
            //调取删除接口
            getRoleDelete({id})
            .then(res=>{
              if(res.data.code==200){

 this.$message.success(res.data.msg)
                //重新调取列表
                this.getRoleAction()
              }
            })
          }).catch(() => {
            this.$message({
              type: 'info',
              message: '已取消删除'
            });
          });
    },
    //封装一个编辑传id事件
    edit(id){
      this.$emit('edit',id)
    }
  },
};
</script>

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

3、add.vue

```
<template>
  <div>
    <el-dialog
```

```
      :title="addInfo.isAdd ? '添加角色' : '修改角色'"
      :visible.sync="addInfo.isShow"
      :before-close="cancel"
      center
    >
      <el-form :model="form" :rules="rules">
        <el-form-item
          prop="rolename"
          label="角色名称:"
          :label-width="formLabelWidth"
        >
          <el-input v-model="form.rolename" autocomplete="off"></el-input>
        </el-form-item>
        <el-form-item label="角色权限:" :label-width="formLabelWidth">
          <!-- el-tree 属性
            data    展示数据    array
            default-expand-all   是否默认展开所有节点
            node-key    每个树节点用来作为唯一标识的属性，整棵树应该是唯一的
            props    配置选项，具体看下表   object
            show-checkbox    节点是否可被选择
            default-checked-keys    默认勾选的节点的 key 的数组
             -->
          <el-tree
            :data="menuList"
            default-expand-all
            node-key="id"
            ref="tree"
```

```
            :props="{
              children: 'children',
              label: 'title'
            }"
            show-checkbox
          >
          </el-tree>
        </el-form-item>
        <el-form-item label="状态:" :label-
width="formLabelWidth">
          <el-switch
            v-model="form.status"
            active-color="#13ce66"
            inactive-color="#ff4949"
            :active-value="1"
            :inactive-value="2"
          >
          </el-switch>
        </el-form-item>
      </el-form>
      <div slot="footer" class="dialog-footer">
        <el-button @click="cancel">取 消</el-
button>
        <el-button v-if="addInfo.isAdd"
@click="add" type="primary"
          >添 加</el-button
        >
        <el-button v-else @click="update"
type="primary">修 改</el-button>
      </div>
    </el-dialog>
  </div>
</template>
```

```
<script>
//引入辅助性函数
import { mapGetters, mapActions } from "vuex";
import { getRoleAdd, getRoleInfo, getRoleEdit }
from "../../util/axios";
export default {
  data() {
    return {
      //表单对象
      form: {
        rolename: "", // 角色名称
        menus: "", //角色权限
        status: 1 //状态1正常2禁用
      },
      //label宽度
      formLabelWidth: "120px",
      //规则验证
      rules: {
        rolename: [
          { required: true, message: "请输入角色
名称", trigger: "blur" },
          { min: 2, max: 8, message: "长度在 2
到 8 个字符", trigger: "blur" }
        ]
      }
    };
  },
  computed: {
    //获取菜单列表
    ...mapGetters({
      menuList: "menu/getMenuList"
    })
```

```
  },
  mounted() {
    //减少对服务器的交互
    if (this.menuList.length == 0) {
      this.getMenuList();
    }
  },
  props: ["addInfo"],
  methods: {
    //封装重置事件
    reset() {
      this.form = {
        rolename: "", // 角色名称
        status: 1 //状态1正常2禁用
      };
      //清空tree结构框
      this.$refs.tree.setCheckedKeys([]);
    },
    //点击取消修改父组件弹框事件
    cancel() {
      this.$emit("cancel", false);
      //调用重置
      this.reset();
    },
    //调取行动
    ...mapActions({
      getMenuList: "menu/getMenuListAction",
      getRoleList: "role/getRoleAction"
    }),
    //添加事件
    add() {
      //转化menu的数据类型
```

```
        this.form.menus =
JSON.stringify(this.$refs.tree.getCheckedKeys()
);
        // console.log(this.form,'表单信息')
        //调取添加接口
        getRoleAdd(this.form).then(res => {
          if (res.data.code == 200) {
            this.$message.success(res.data.msg);
            //重新调取接口
            this.getRoleList();
            //关闭弹框
            this.cancel();
          } else {
            this.$message.error(res.data.msg);
          }
        });
      },
      //封装一个查看一条数据的方法
      look(id) {
        //调取查看接口
        getRoleInfo({ id }).then(res => {
          console.log(res, "获取一条数据");
          if (res.data.code == 200) {
            this.form = res.data.list;
            //setCheckedKeys  通过 keys 设置目前勾选
的节点，使用此方法必须设置 node-key 属性
            this.form.menus =
this.$refs.tree.setCheckedKeys(
              JSON.parse(this.form.menus)
            );
            this.form.id = id;
          }
        });
```

```
    },
    //编辑确定事件
    update() {
      //转化menu的数据类型
      this.form.menus =
JSON.stringify(this.$refs.tree.getCheckedKeys()
);
      // console.log(this.form,'表单信息')
      //调取添加接口
      getRoleEdit(this.form).then(res => {
        if (res.data.code == 200) {
          this.$message.success(res.data.msg);
          //重新调取接口
          this.getRoleList();
          //关闭弹框
          this.cancel();
        } else {
          this.$message.error(res.data.msg);
        }
      });
    }
  }
};
</script>

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