

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 install 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 http
```

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）

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>
```