

学习一款基于Vue2的极简后台管理系统模板：vue-admin-template

文章分类：Project； 标签：后台管理系统； 作者：Hackyle；

更新时间：Thu Jan 12 17:54:31 CST 2023

1. 克隆项目

1. 项目结构
2. 封装axios
3. webpack-alias
2. 页面结构Layout
 1. 左侧导航栏Sidebar
 2. Header
3. 多级导航/路由
4. store
5. 登录

本文是一篇我在学习[vue-admin-template](#)项目时的笔记，可以帮助读者快速掌握该项目的核心思想

vue-admin-template是什么？

- 是一款基于Vue2的极简后台管理系统，只含前端。
- 只包含了 Element UI & axios & iconfont & permission control & lint，这些搭建后台必要的东西。
- 项目作者以及地址：<https://github.com/PanJiaChen/vue-admin-template>

背景

- vue-admin-template这个项目**很适合初学者学习**
- 因为它足够的简单，设计清晰，可以帮助初学者快速建立基于Vue的项目概念和一般项目的设计思想。可以基于该项目，快速构建起一个后台管理管理系统（前端）
- 我在学习这个项目的过程中，对于一些我不是很熟悉的技术，需要在**源码上写一些注释**，对于作者**为什么这么设计、代码为什么这样写**，也会写一些**我的思考、设计说明**
- 所以我克隆了该项目，将我的代码注释提交于[Github](#)，我对项目的一些思考与设计说明记录在本文中

强烈建议读者先阅读原作者的ReadMe

- <https://github.com/PanJiaChen/vue-admin-template/blob/master/README-zh.md>
- 其中的系列配套教程也非常优秀

本文内容导览

- [克隆项目](#)
 - [项目结构](#)
 - [封装axios](#)
 - [webpack-alias](#)
- [页面结构Layout](#)
 - [左侧导航栏Sidebar](#)
 - [Header](#)
- [多级导航/路由](#)
- [store](#)
- [登录](#)

克隆项目

Source Code

- 原作者：<https://github.com/PanJiaChen/vue-admin-template/>
- 我在原作者的源码上添加了用于学习的注释：https://github.com/HackyleShawe/vue-admin-template_4_learn

原作者提供的Demo

- <https://panjiachen.github.io/vue-admin-template/>
- <https://panjiachen.gitee.io/vue-admin-template>

检查项目的依赖源

- 获取：npm config get registry
- 配置：npm config set registry <https://registry.npm.taobao.org>

依赖安装：进入项目根目录，执行：npm install

项目结构

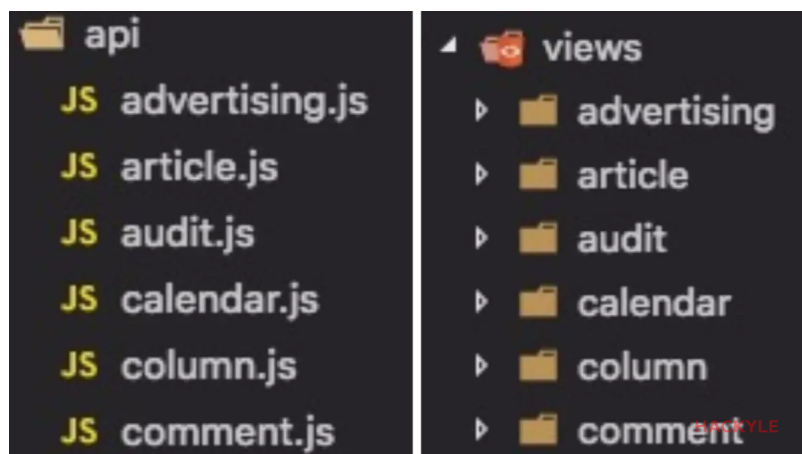
1		└─ build	// 构建相关
2		└─ config	// 配置相关
3		└─ src	// 源代码
4		└─ api	// 所有请求
5		└─ assets	// 主题 字体等静态资源
6		└─ components	// 全局公用组件
7		└─ directive	// 全局指令
8		└─ filter	// 全局 filter
9		└─ icons	// 项目所有 svg icons
10		└─ lang	// 国际化 language
11		└─ mock	// 项目mock 模拟数据
12		└─ router	// 路由
13		└─ store	// 全局 store管理
14		└─ styles	// 全局样式
15		└─ utils	// 全局公用方法
16		└─ views	// 具体页面
17		└─ App.vue	// 入口页面
18		└─ main.js	// 入口 加载组件 初始化等
19		└─ permission.js	// 权限管理
20		└─ static	// 第三方不打包资源
21		└─ package.json	// 依赖管理

components

- 放置的都是全局公用的一些组件，如上传组件，富文本等等
- 一些页面级的组件建议还是放在各自views文件下，方便管理

views 和 api

- 根据业务模块来划分 views，并且 将views 和 api 两个模块一一对应，从而方便维护



- 一些全区公用的api模块，如七牛upload，remoteSearch等等，这些单独放置就行

封装axios

axios使用步骤

- 配置axios的全局参数，初始化axios
- 请求、响应拦截器
- 写GET/POST/PUT/PATCH/DELETE请求

定义axios

```

1 | import axios from 'axios'
2 | import { Message } from 'element-ui'
3 | import store from '@/store'
4 | import { getToken } from '@/utils/auth'
5 |
6 | // 创建axios实例

```

```

7   const service = axios.create({
8     baseURL: process.env.BASE_API, // api的base_url
9     timeout: 5000 // 请求超时时间
10  })
11
12  // request拦截器: : 从Cookie中获取token, 放在请求头里
13  service.interceptors.request.use(config => {
14    // Do something before request is sent
15    if (store.getters.token) {
16      config.headers['X-Token'] = getToken() // 让每个请求携带token--['X-Token']为自定义
17    }
18    return config
19  }, error => {
20    // Do something with request error
21    console.log(error) // for debug
22    Promise.reject(error)
23  })
24
25  // response拦截器: : 根据后端定义的响应状态码, 判断此次请求是否响应成功
26  service.interceptors.response.use(
27    response => response,
28    /**
29     * 下面的注释为通过response自定义code来标示请求状态, 当code返回如下情况为权限有问题, 3
30     * 如通过xmlhttprequest 状态码标识 逻辑可写在下面error中
31     */
32    // const res = response.data;
33    // if (res.code !== 20000) {
34    //   Message({
35    //     message: res.message,
36    //     type: 'error',
37    //     duration: 5 * 1000
38    //   });
39    //   // 50008:非法token; 50012:其他客户端登录了; 50014:Token 过期了;
40    //   if (res.code === 50008 || res.code === 50012 || res.code === 50014) {
41    //     MessageBox.confirm('你已被登出, 可以取消继续留在该页面, 或者重新登录',
42    //       confirmButtonText: '重新登录',
43    //       cancelButtonText: '取消',
44    //       type: 'warning'
45    //     ).then(() => {
46    //       store.dispatch('FedLogOut').then(() => {
47    //         location.reload(); // 为了重新实例化vue-router对象 避免bug
48    //       });
49    //     })
50    //   }
51    //   return Promise.reject('error');
52    // } else {
53    //   return response.data;
54    // }
55    error => {
56      console.log('err' + error) // for debug
57      Message({
58        message: error.message,
59        type: 'error',
60        duration: 5 * 1000
61      })
62      return Promise.reject(error)
63    }
64  })
65  export default service

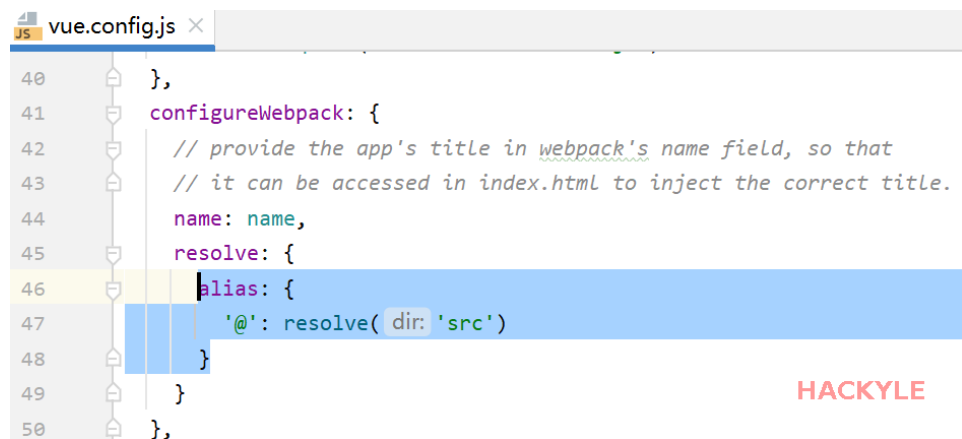
```

使用axios

```
1 import request from '@/utils/request'
2
3 //使用
4 export function getInfo(params) {
5   return request({
6     url: '/user/info',
7     method: 'get',
8     params
9   });
10 }
```

webpack-alias

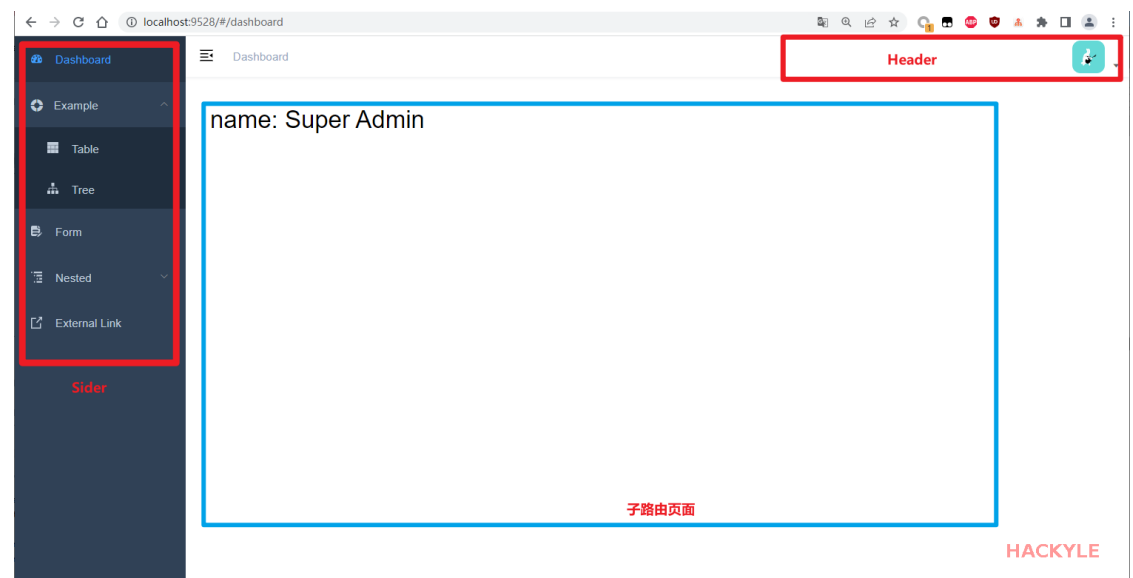
- 当项目逐渐变大之后，文件与文件直接的引用关系会很复杂，这时候就需要使用alias 了。
- 有的人喜欢alias 指向src目录下，再使用相对路径找文件
- @ 是 webpack 的 alias，代表当前项目的src目录
- 在Vue项目配置文件config.js中定义alias



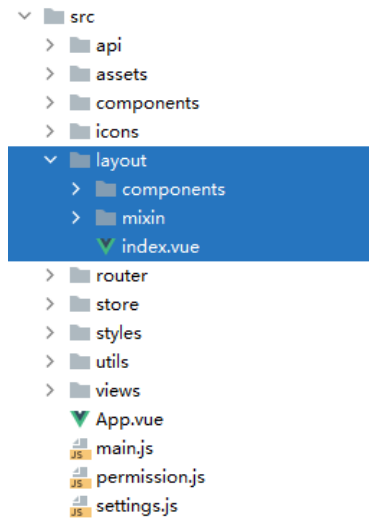
```
40 },
41 configureWebpack: {
42   // provide the app's title in webpack's name field, so that
43   // it can be accessed in index.html to inject the correct title.
44   name: name,
45   resolve: {
46     alias: {
47       '@': resolve(__dirname, 'src')
48     }
49   }
50 },
```

HACKYLE

页面结构Layout



- 在src/layout/目录下定义页面的主体结构



- 在src/router/中的每个路由定义为：
 - 将每个顶级路由的路由页面为：src/layout;
 - 每个子路由的路由页面为：子路由的页面

```

{
  path: '/',
  component: Layout,
  redirect: '/dashboard',
  children: [
    {
      path: 'dashboard',
      component: () => import('@/views/dashboard/index'),
      name: 'Dashboard',
      meta: { title: 'Dashboard', icon: 'dashboard', affix: true }
    }
  ]
},

```

顶级路由页面

子路由才为具体的Vue页面

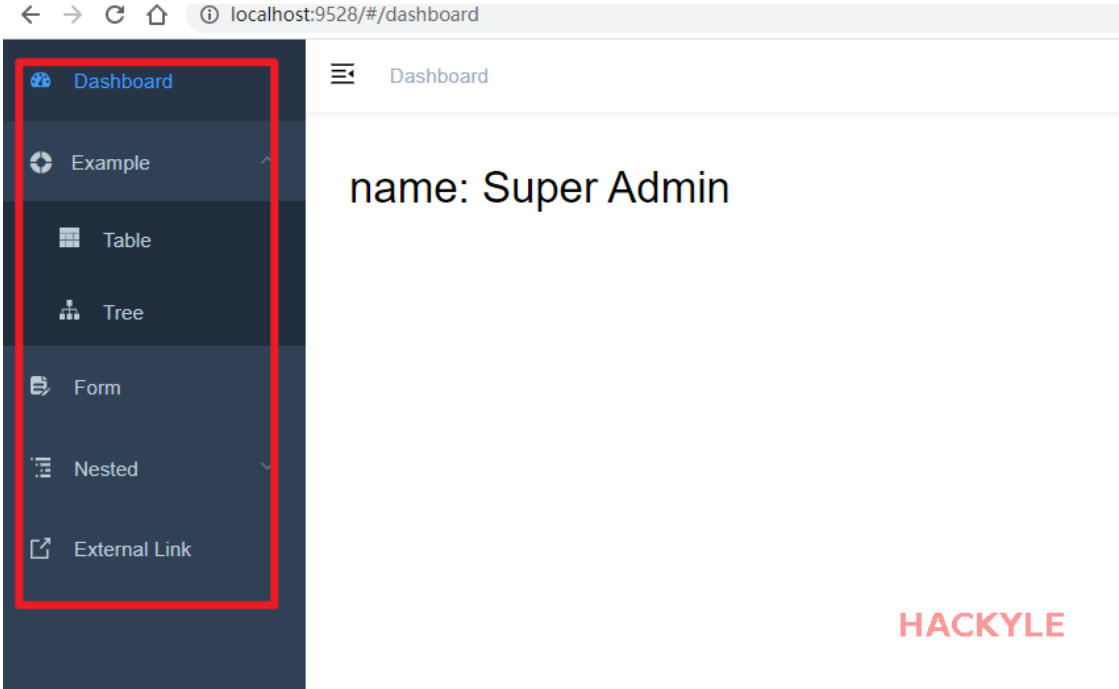
项目主页：src/layout/index.vue

- vue-element-admin 中大部分页面都是基于这个 layout 的，除了个别页面如：login , 404, 401 等页面没有使用该layout。
- 等价于其他项目中的vue组件
- 如果你想在项目中有多种不同的layout也是很方便的，只要在一级路由那里选择不同的layout组件就行。

src/layout/components/AppMain.vue

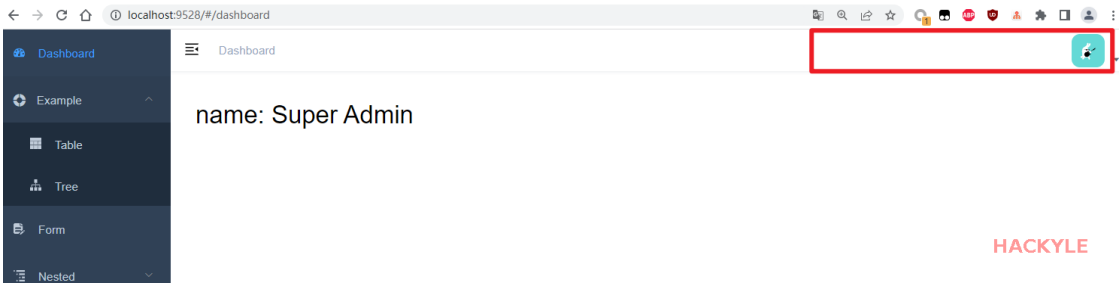
- 功能：在vue组件中展示子路由页面
- <router-view> 为子路由的占位符，<keep-alive> 用于保存切换了Tab页后依然保存了上个页面上的数据，配合页面的 tabs-view 标签导航使用

左侧导航栏Sidebar



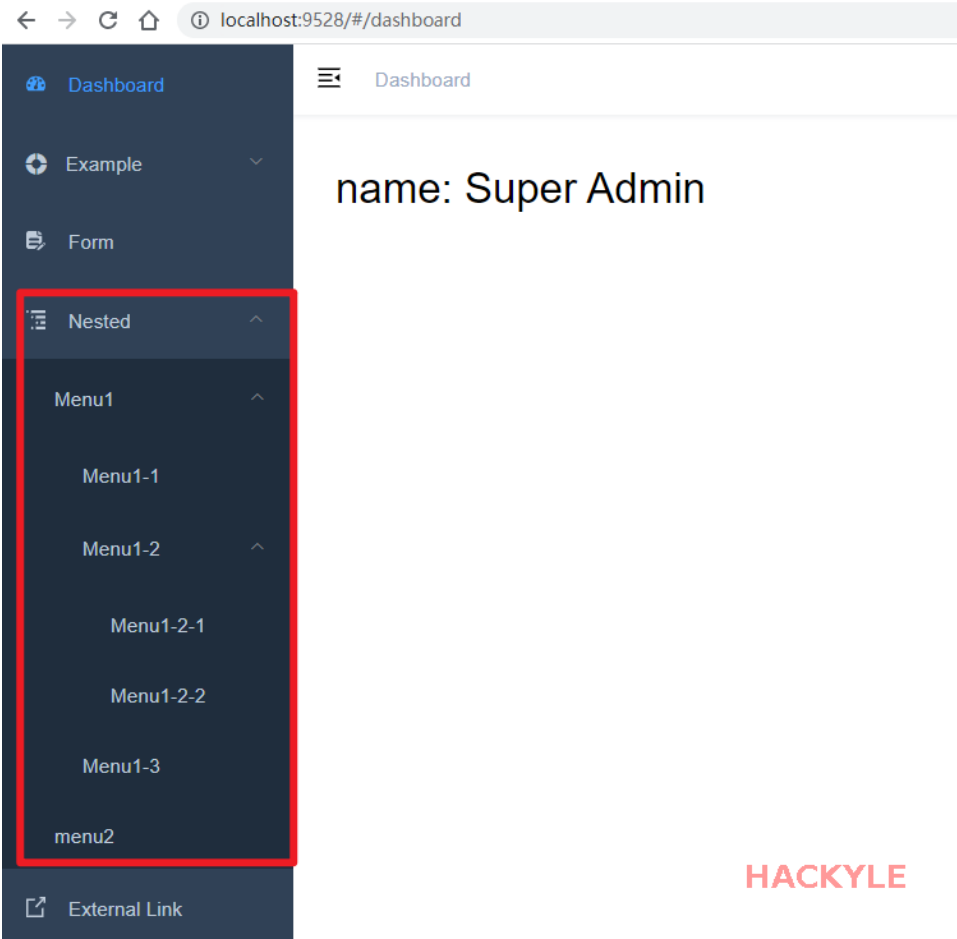
@/layout/components/sidebar/index.vue

Header



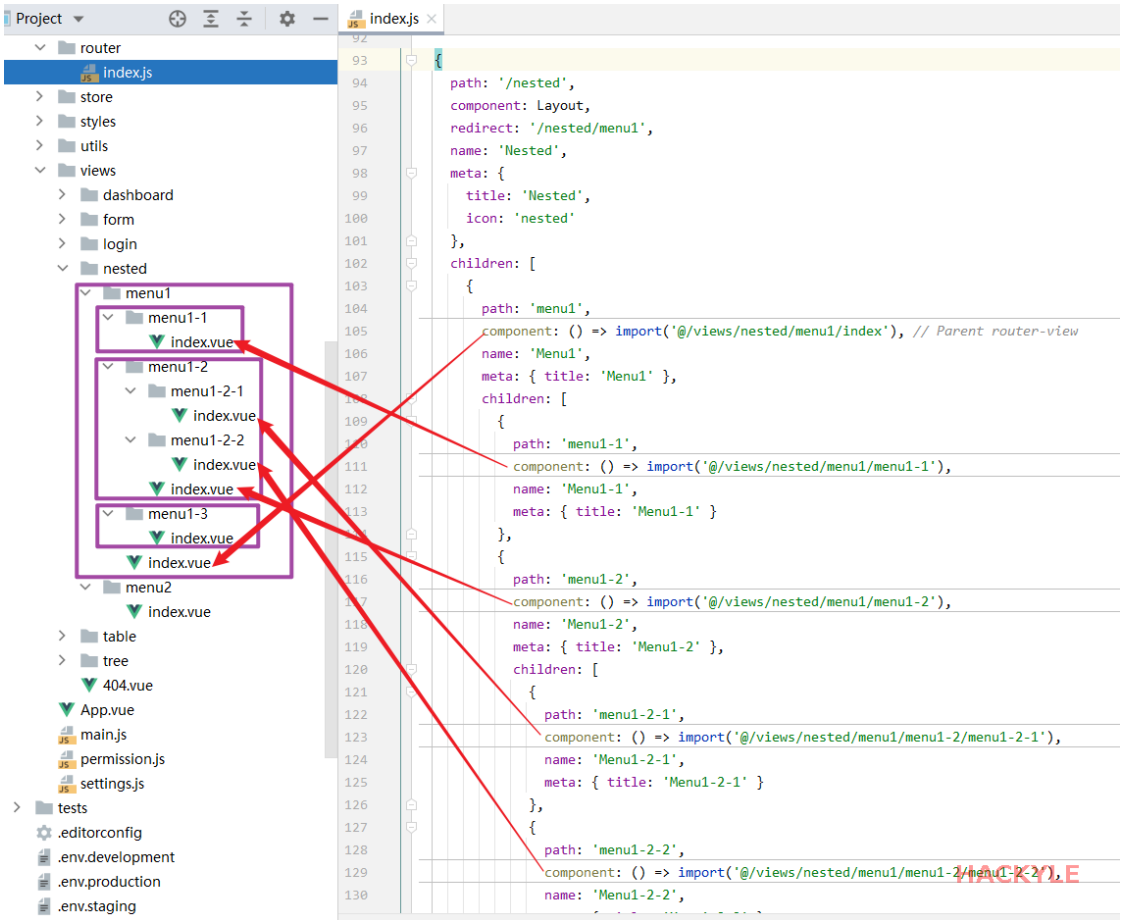
@/layout/components/Navbar.vue

多级导航/路由



HACKYLE

什么是多级路由



HACKYLE

如何实现多级路由：src/router/index.js

store

vuex

- 数据存储、传递工具
- 是专门为 Vue 开发的状态管理方案，我们可以把需要在各个组件中传递使用的变量、方法定义在这里

使用步骤

- 挂载Vuex，定义Store模型：src/store/index.js
- 加载Store，在src/main.js中导入
- 放入数据：\$store.commit(mutations函数名, 参数)
- 读取数据：\$store.state.State中的数据名

store的基本结构

- state：要全局共享的属性，全局只存在一份
- getter：实时监听state属性的变化，并返回
- mutation：提供更改state中的属性的方法
- action：一些业务代码，并且可以提交用于mutation，目的还是为了更改state中的值

store/index.js

- vuex的入口文件
- 模块化管理vuex，引入其他的vuex组件JS文件

store/getters.js

- 将store/modules中的所有JS文件中的state属性值聚合
- 提供类似于getter的操作，实时获取整个vuex中的所有属性

store/modules模块化管理vuex

- app.js：全局存储共享本项目的全局配置信息
- settings.js：全局存储共享本项目的自定义配置项
- user.js：全局存储共享用户的登录信息（用户名称、token、头像URL）

登录

- 登录（认证）的主要思路：
 - 当用户填写完账号和密码后向后端验证是否正确，验证通过之后，后端会返回一个token，拿到token之后（我会将这个token存储到cookie中，保证刷新页面后能记住用户登录状态）
 - 前端会根据token再去后端请求一个 user_info 的接口来获取用户的详细信息（如用户权限，用户名等信息）。
- 权限验证的主要思路：根据后端返回的当前用户的role，动态地算出其对应有权限的路由，通过 **addRoutes 动态挂载** 这些路由。

登录页：src/views/login/index.vue

1. 一个表单：用户名输入框、密码输入框、提交按钮
2. 在向服务端提交之前对账号和密码做一次简单的规则校验
3. 将登录按钮上绑定click事件，点击登录之后向服务端提交账号和密码进行验证

为什么要将调用登录API的动作代码放于store/modules/user.js中？

- 使用vuex来全局共享用户成功的登录信息
- 为了便于将登录成功的用户信息塞入vuex，直接在store的代码中定义调用后端的用户登录、获取登录的用户信息、登出（清空vuex中的信息）的业务代码

store/modules/user.js的action函数

- 登录：请求后端接口；成功后将token信息存储Cookie中
- 获取用户信息：请求后端接口，获取用户的信息，填充到vuex
- 登出：请求后端接口，移除本地Cookie中的Token，清除动态加载的路由，请求vuex中的数据

路由前置拦截器（router.beforeEach）

- 每次路由都判断一下Cookie里有没有token
- 如果没有跳转到登录页面
- 如果有判断一下Vuex有没有用户信息：没有用户信息：请求后端接口获取；有用户信息：路由到想要的页面
- 实现：src/permission.js

为什么不把用户信息也存放于Cookie，而每次都要向后端获取？

- 保证获取到最新的用户信息，最关键的获取的最新的用户权限，通过用户权限动态挂载路由页面。
- 假设这样一种场景，你在公司电脑上修改了权限，然后你家用电脑登录，权限还是没有变化，要等待Cookie失效后，你家用电脑上的权限才会重新请求后端获取。
- 当然如果是做了单点登录得功能的话，用户信息存储在本地也是可以的。当你一台电脑登录时，另一台会被提下线，所以总会重新登录获取最新的内容。

版权声明： 非明确标注皆为原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上本文链接及此声明。
原文链接：<https://blog.hackyle.com/article/project/vue-admin-template-4-learn>

Name:

Email:

Link: Input your phone or website, please

FileEditViewFormatToolsTableHelp

B

I

U

S

A

I_x

{;}

Ω

☺

<

>

=

=

=

=

⋮

Input comment, please

p0 words

SUBMITRESET

Designed and Created by HACKYLE SHAWE

备案号：浙ICP备20001706号-2