芋道源码 2022-04-17

开发规范

0. 实战案例

本小节，提供大家开发管理后台的功能时，最常用的普通列表、树形列表、新增与修改的表单弹窗、详情表单弹窗的实战案例。

0.1 普通列表

可参考 [系统管理 -> 岗位管理] 菜单：

- API 接口： `/src/api/system/post/index.ts`
- 列表界面： `/src/views/system/post/index.vue`
- 表单界面： `/src/views/system/post/PostForm.vue`

> **为什么界面拆成列表和表单两个 Vue 文件?**
> 每个 Vue 文件，只实现一个功能，更简洁，维护性更好，Git 代码冲突概率低。

0.2 树形列表

可参考 [系统管理 -> 部门管理] 菜单：

- API 接口： `/src/api/system/dept/index.ts`
- 列表界面： `/src/views/system/dept/index.vue`
- 表单界面： `/src/views/system/dept/DeptForm.vue`

0.3 高性能列表

可参考 [系统管理 -> 地区管理] 菜单，对应 `/src/views/system/area/index.vue` 列表界面

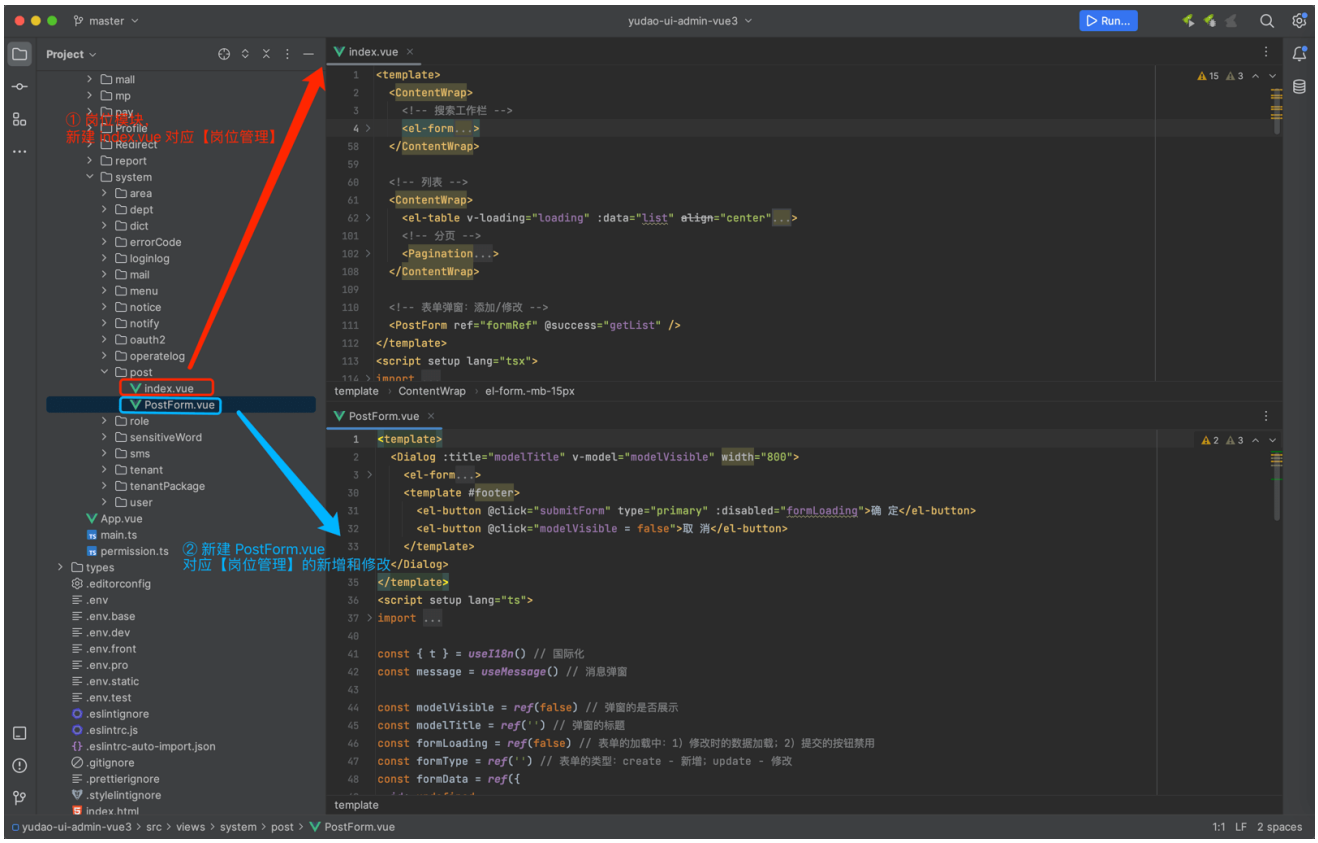
基于 Virtualized Table 虚拟化表格 实现，解决一屏里超过 1000 条数据记录时，就会出现卡顿等性能问题。

0.4 详情弹窗

可参考 [基础设施 -> API 日志 -> 访问日志] 菜单，对应 `/src/views/infra/apiAccessLog/ApiAccessLogDetail.vue` 详情弹窗

1. view 页面

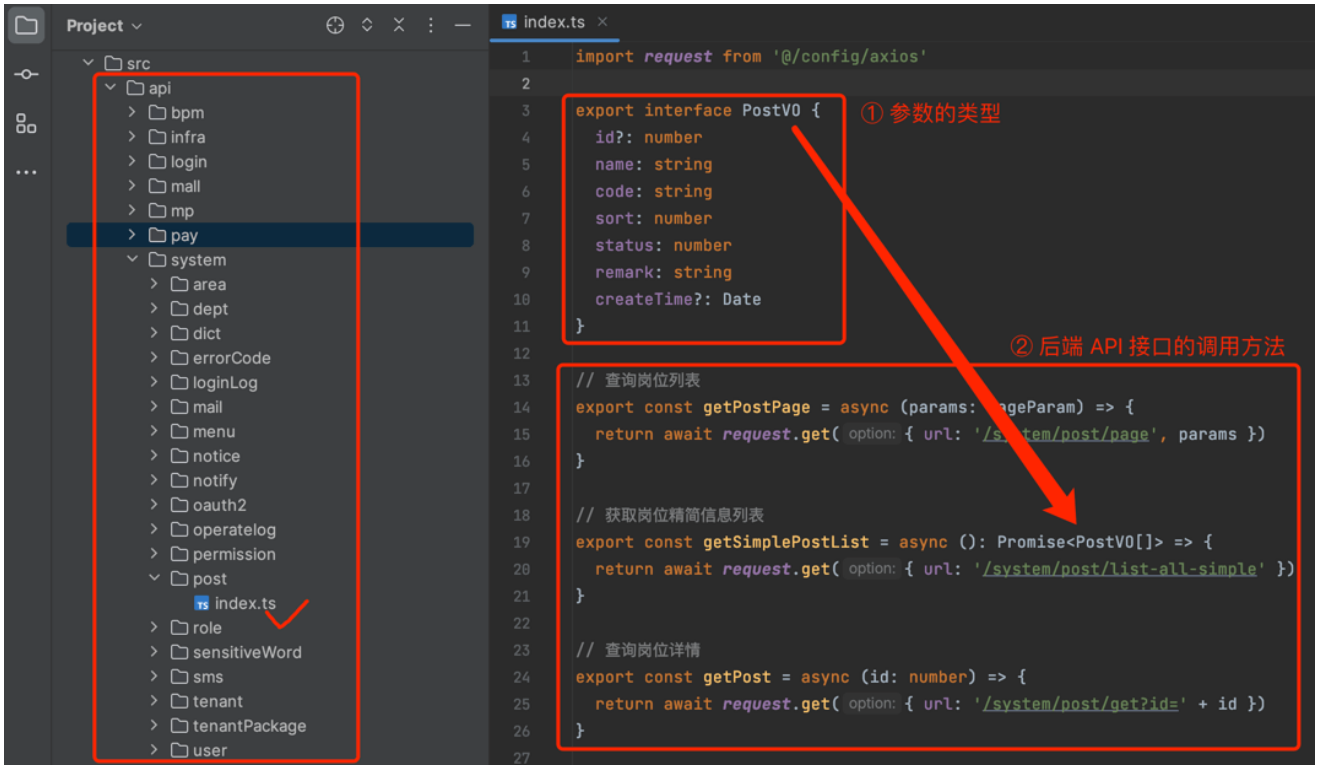
在 `@views` 目录下，每个模块对应一个目录，它的所有功能的 `.vue` 都放在该目录里。



一般来说，一个路由对应一个 `index.vue` 文件。

2. api 请求

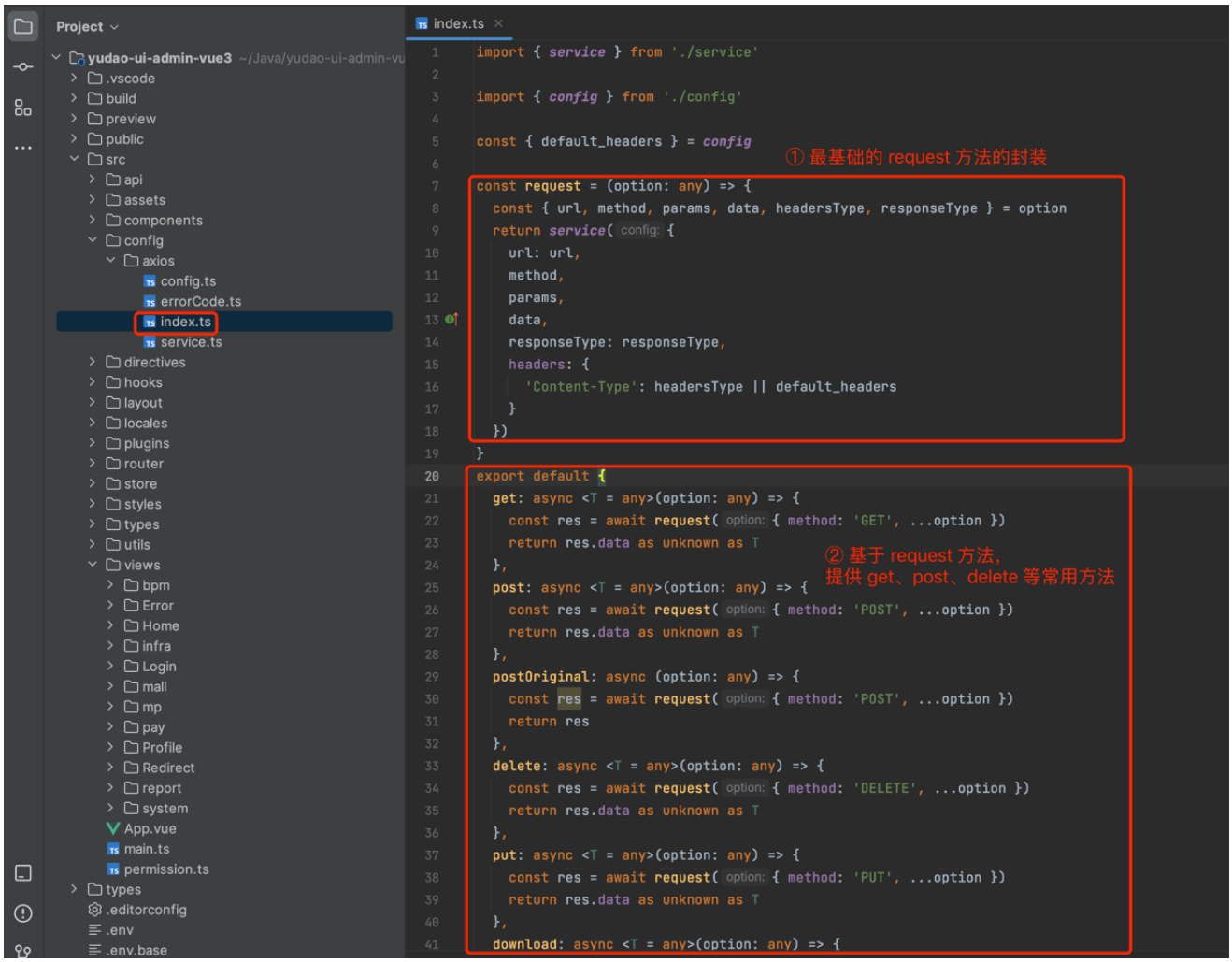
在 `@/api` 目录下，每个模块对应一个 `index.ts` API 文件。



- API 方法：会调用 `request` 方法，发起对后端 RESTful API 的调用。
- `interface` 类型：定义了 API 的请求参数和返回结果的类型，对应后端的 VO 类型。

2.1 请求封装

`/src/config/axios/index.ts` 基于 axios 封装，统一处理 GET、POST 方法的请求参数、请求头，以及错误提示信息等。



2.1.1 创建 axios 实例

- `baseURL` 基础路径
- `timeout` 超时时间，默认为 30000 毫秒

▶ 实现代码 /src/config/axios/service.ts

2.1.2 Request 拦截器

- 【重点】 `Authorization` 、 `tenant-id` 请求头
- GET 请求参数的拼接

▶ 实现代码 /src/config/axios/service.ts

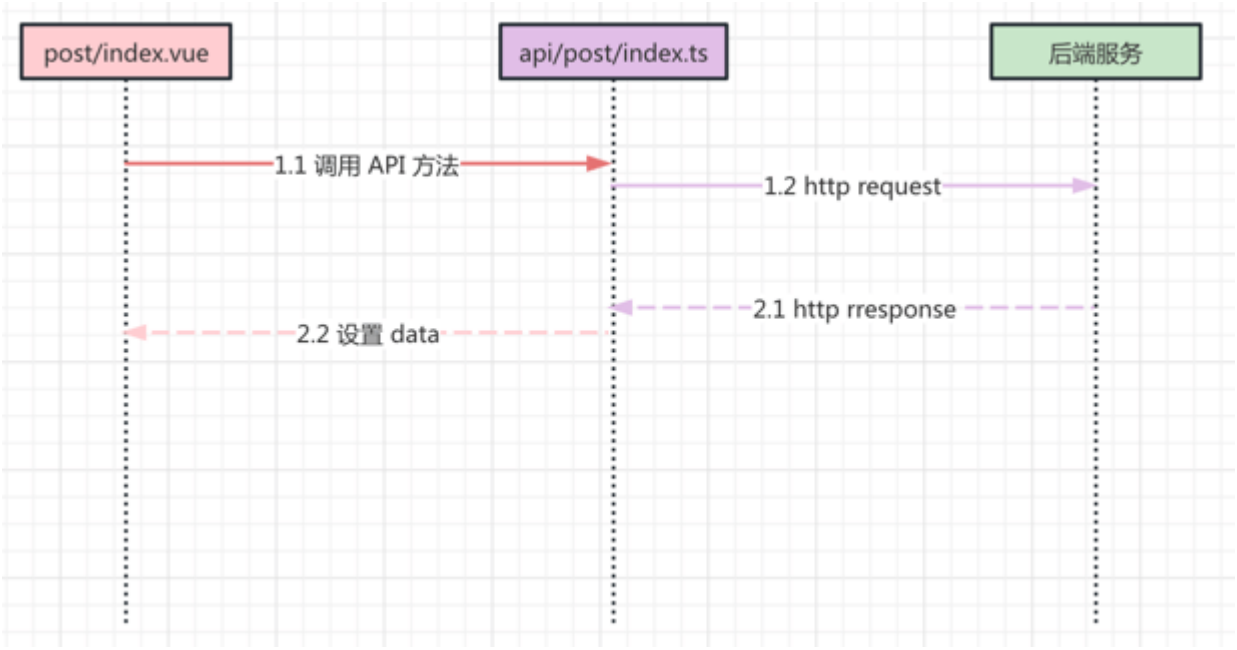
2.1.3 Response 拦截器

- 访问令牌 AccessToken 过期时，使用刷新令牌 RefreshToken 刷新，获得新的访问令牌
- 刷新令牌失败（过期）时，跳回首页进行登录
- 请求失败，Message 错误提示

▶ 实现代码 /src/config/axios/service.ts

2.2 交互流程

一个完整的前端 UI 交互到服务端处理流程，如下图所示：



继续以 [系统管理 -> 岗位管理] 菜单为例，查看它是如何读取岗位列表的。代码如下：

```
// ① api/system/post/index.ts
import request from '@/config/axios'

// 查询岗位列表
export const getPostPage = async (params: PageParam) => {
  return await request.get({ url: '/system/post/page', params })
}

// ② views/system/post/index.vue
<script setup lang="tsx">
const loading = ref(true) // 列表的加载中
```

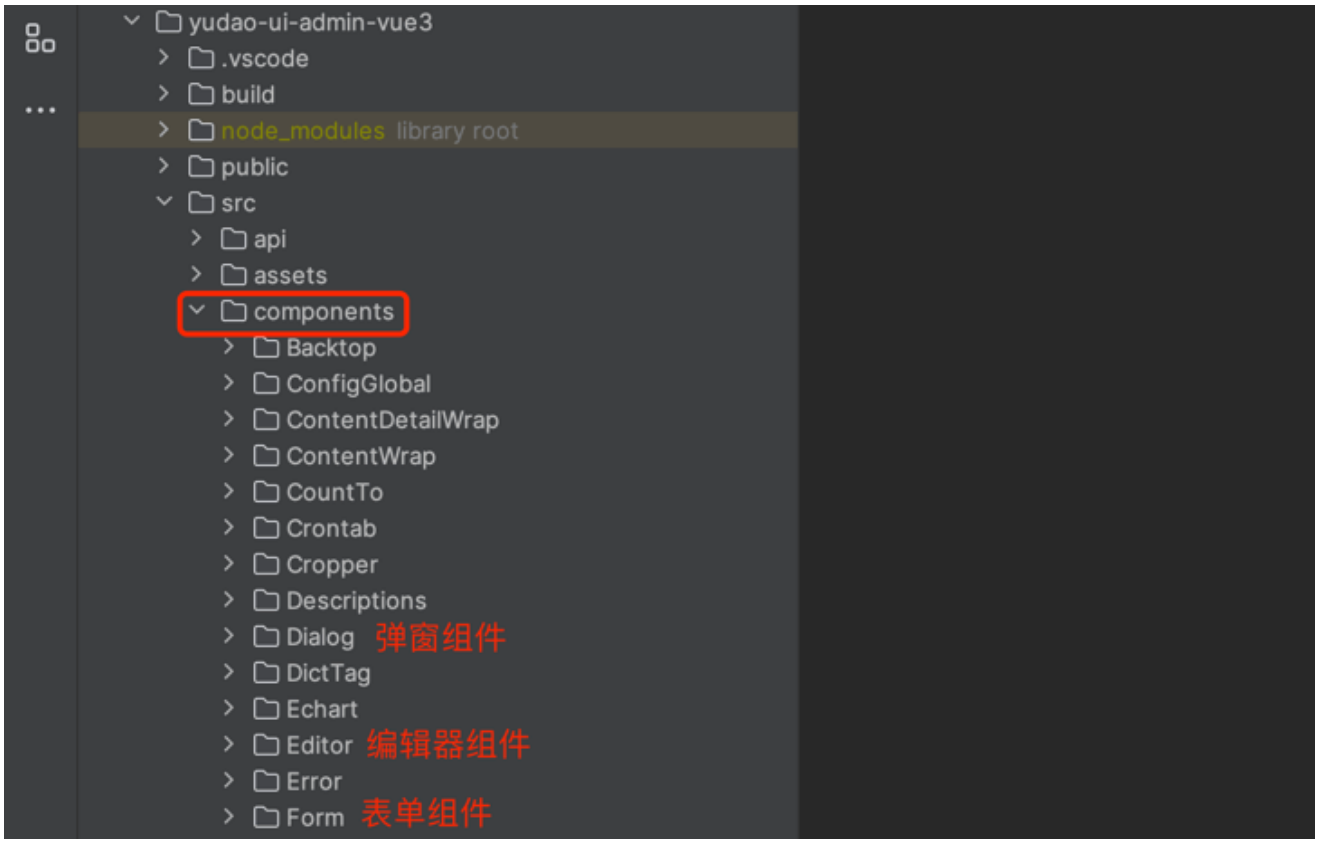
```
const total = ref(0) // 列表的总页数
const list = ref([]) // 列表的数据
const queryParams = reactive({
    pageNo: 1,
    pageSize: 10,
    code: '',
    name: '',
    status: undefined
})

/** 查询岗位列表 */
const getList = async () => {
    loading.value = true
    try {
        const data = await PostApi.getPostPage(queryParams)
        list.value = data.list
        total.value = data.total
    } finally {
        loading.value = false
    }
}
</script>
```

3. component 组件

3.1 全局组件

在 `@/components` 目录下，实现**全局**组件，被所有模块所公用。

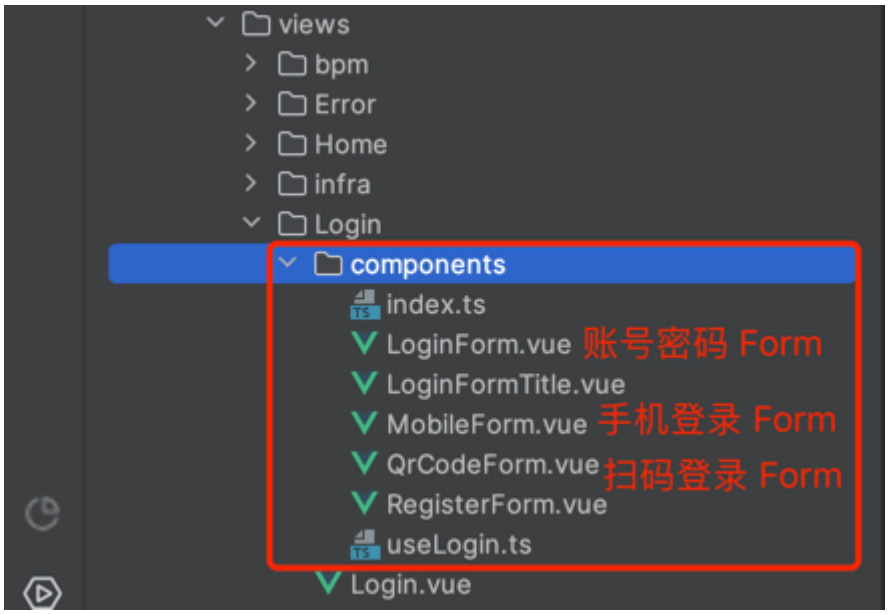
例如说，富文本编辑器、各种各搜索组件、封装的分页组件等等。



3.2 模块内组件

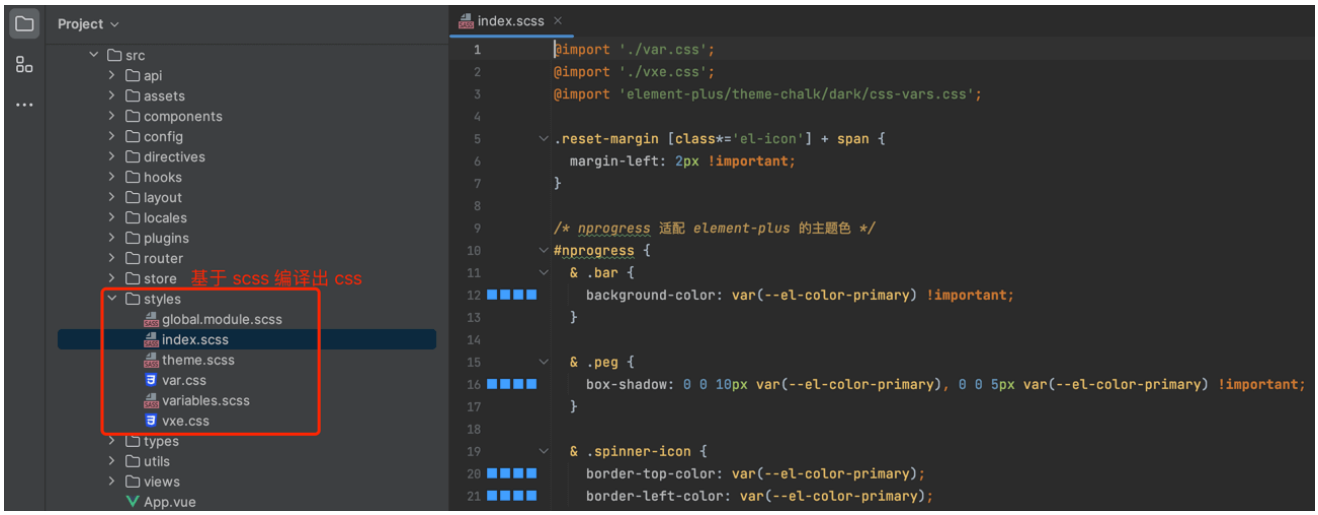
每个模块的业务组件，可实现在 `views` 目录下，自己模块的目录的 `components` 目录下，避免单个 `.vue` 文件过大，降低维护成功。

例如说， `@/views/pay/app/components/xxx.vue` ：

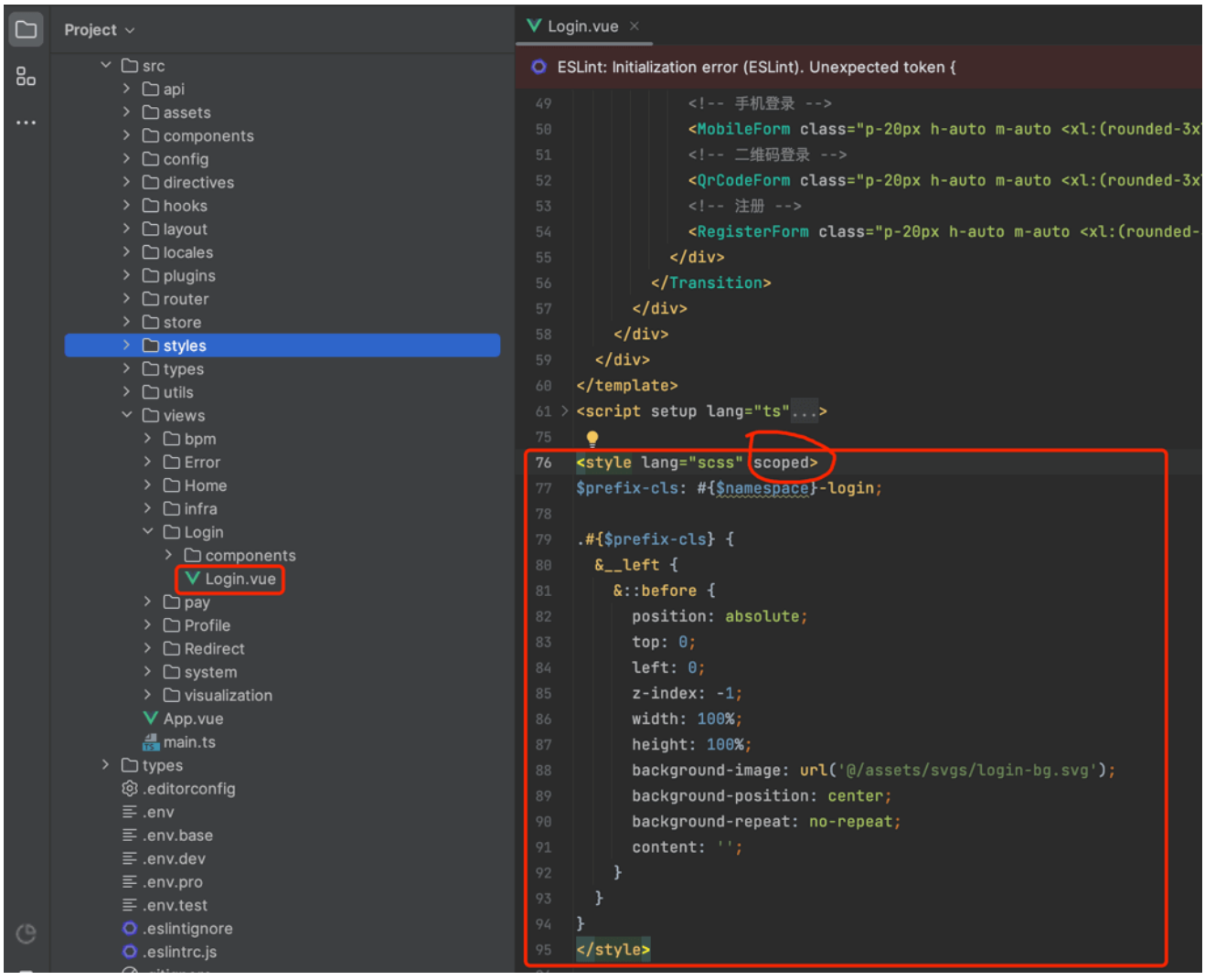


4. style 样式

① 在 `@/styles` 目录下，实现**全局**样式，被所有页面所公用。



② 每个 `.vue` 页面，可在 `<style />` 标签中添加样式，注意需要添加 `scoped` 表示只作用在当前页面里，避免造成全局的样式污染。



更多也可以看看如下两篇文档：

- 《vue-element-plus-admin —— 项目配置「样式配置」》
- 《vue-element-plus-admin —— 样式》

5. 项目规范

可参考 《vue-element-plus-admin —— 项目规范》 文档。

← 配置读取 菜单路由→