
视频讲解

第5章

实践项目整体布局

在 Vue 3 前端项目开发中，可以利用 Element Plus 组件库设计应用的整体布局。通过安装并配置 vue-router 路由模块，实现在布局页面上 Vue 组件之间的平滑无缝切换。

5.1 集成 Element Plus 组件库

Element Plus 是一个专为 Vue 3 设计的 UI 组件库，旨在简化 Web 应用程序界面的开发。它提供了包括如按钮、输入框、表格等 70 多个前端元素组件，通过使用 Element Plus 组件库，开发者能高效地构建出前端页面布局。

按照以下步骤，可在前端 Vue 3 项目中安装 Element Plus。

5.1.1 安装 Element Plus

安装 Element Plus 非常简单，只需为项目添加 Element Plus 相关依赖即可，过程如下。

用 VSCode 工具打开前端 Vue 3 项目 prj-frontend。单击菜单栏中的 Terminal 选项，并选取 New Terminal 节点来启动一个终端窗口。在终端窗口中，用 cd 命令切换到项目所在目录后，执行以下命令：

```
npm install element-plus --save
```

此命令将自动从 npm 源下载并安装 Element Plus 组件库。此外，--save 参数会将 Element Plus 库的依赖项信息添加到项目的 package.json 文件中，以便于后续管理和依赖跟踪。package.json 文件中的相关依赖信息如下：

```
1.  "dependencies": {
2.    "element-plus": "^2.5.5",
3.    "vue": "^3.3.11"
4.  }
```

代码的第 2 行，指定项目将使用 element-plus 这个依赖库，并且版本号为 ^2.5.5。“^”符号在这里表示允许安装高于 2.5.5 但不改变主版本号的最新版本。也就是说，如果 2.6.0、2.6.1 等版本被发布，npm 在安装时会选择这些版本中的最新版本，但不会选择 3.0.0 或更高版本，以免不兼容问题的发生。

5.1.2　注册 Element Plus

1. 项目中导入和注册 Element Plus 组件库

在 main.js 文件中，添加如下代码：

```
1. import { createApp } from 'vue'
2. import App from './App.vue'
3. import ElementPlus from 'element-plus';                  // 导入 Element Plus
4. import 'element-plus/lib/theme-chalk/index.css';  // 导入 Element Plus 样式表
5.
6. const app = createApp(App)
7. app.use(ElementPlus);                                    // 注册 Element Plus
8. app.mount('#app')
```

其中第 3～4 行代码，分别实现了 Element Plus 库的导入以及对应样式表的加载。第 6～7 行代码，通过调用 createApp()函数创建了 Vue 应用实例，然后使用 app.use()方法注册了 Element Plus 库，使其能够在整个 Vue 应用中被使用。

2. 测试 Element Plus 组件

修改 App.vue 文件，以显示 Element Plus 的组件。在 template 标签中添加以下代码：

```
<el-button type="danger">Element Plus 按钮</el-button>
```

注意：如需了解 Element Plus 的更多组件详情，请访问其官方网站。

然后，重新编译和运行项目，在命令窗口中运行以下命令：

```
npm run dev
```

使用浏览器访问 http://localhost:5173，若看到一个带有 Element Plus 按钮的页面，则说明前端 Vue 3 项目配置 Element Plus 成功，如图 5-1 所示。

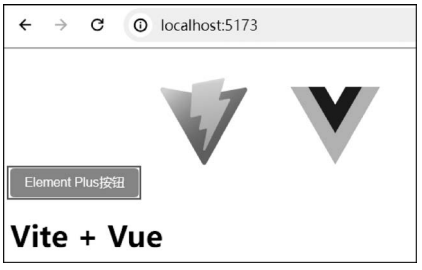

图 5-1　Vue 3 项目配置 Element Plus 成功

5.2 实施路由配置和单页布局

路由机制可视为一种映射体系，其核心在于将多个 URL 地址与相应的视图组件相关联。在 Vue 3 前端开发项目中，通过 vue-router 模块配置路由，App.vue 入口就能够依据当前 URL 来动态加载并渲染相应的 Vue 组件，从而实现单页应用（Single Page Application，SPA）的流畅导航与界面无缝切换。

针对实践项目，可遵循以下步骤实施。

5.2.1 路由配置

1. 安装配置路由模块 vue-router

Vue Router 是 Vue 官方路由管理器。Vue-router 安装到项目中后，构建单页面应用将变得更为容易。

用 VSCode 打开前端 Vue 3 项目 prj-frontend。单击菜单栏中的 Terminal 选项，并选取 New Terminal 节点来启动一个终端窗口。在终端窗口中，用 cd 命令切换到项目所在目录后，执行以下命令：

```
npm install vue-router
```

此命令将自动从 npm 源下载并安装 vue-router 组件。

2. 创建 Vue 组件

在项目的 src/views 目录下，创建 4 个用于切换功能界面的 Vue 组件：Category.vue（分类管理）、Dessert.vue（甜点管理）、Home.vue（默认主界面）和 Register.vue（注册），如图 5-2 所示。

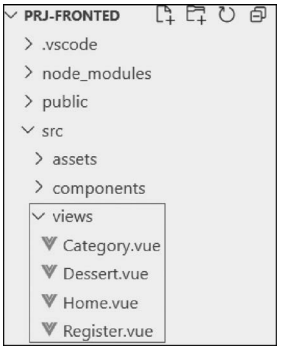

图 5-2 创建 4 个 Vue 组件

Home.vue 组件作为实践项目的默认显示页面，在 Vue 3 应用程序启动时即被渲染显示，其代码实现如下：

```
<template>
  <div class="bg-img"></div>
</template>

<script setup>
</script>

<style scoped>
  .bg-img{
   width:100%; height: 100%;
   background: url('../../public/img/bg.jpeg');
   background-size: 100% 100%;
   opacity: 0.7;
  }
</style>
```

注意：读者可以自制背景图 bg.jpeg，并将其放置到 public/img 目录中。第 9～14 行代码，则设置了该背景图的显示样式。

Category.vue、Dessert.vue、Register.vue 组件内容可暂时做简单处理，代码如下：

```
<template>
   <div></div>
</template>
<script setup>
</script>
<style scoped>
</style>
```

为区别 3 个 Vue 组件，可分别在第 2 行代码 div 标签中写入文字：甜点分类、甜点信息、用户注册。

3. 设置 vue-router 路由

在项目的 src/router 目录下，创建路由文件 index.js，代码如下：

```
import {createRouter,createWebHistory} from 'vue-router'
import Register from "../views/Register.vue"
import Category from "../views/Category.vue"
import Dessertfrom "../views/Dessert.vue"
// 路由规则，定义 URL 地址与组件之间的对应关系
const routes = [
   { path: '/', name:'Home', component: () => import('../views/Home.vue') },
   { path: '/Register', name:'Register', component:Register },
   { path: '/Category', name:'Category', component:Category },
   { path: '/Dessert', name:'Dessert', component:Dessert },
]
const router = createRouter({
```

```
13.   history: createWebHistory(),
14.   routes: routes
15. })
16. export default router
```

对上述代码,具体说明如下。

第1行,导入了路由管理器Vue Router的两个资源:createRouter和createWebHistory,在后续的路由设置中将使用到这两个资源。

第2～4行,分别导入3个先前定义的Vue组件。

第6～11行,设置4个路由规则,每个规则都映射到一个特定的Vue组件。当用户访问指定的path时,将渲染对应的Vue组件。例如,当用户访问"/Register"路径时,系统将动态加载并渲染Register.vue组件。其中,第7行与第8～10行不同,使用了动态import()语法,以实现组件的按需加载。

第13行,设置Vue路由模式。Vue Router支持Hash和History两种模式,这里选择了History模式,以便更贴近传统Web开发中的URL结构。History模式利用浏览器的History API来管理URL,使得用户在导航到不同资源时,实际切换到不同的Vue组件,无须刷新整个页面,从而提供了更为流畅的用户体验。而Hash模式,则是通过URL中的hash(#)部分来模拟一个完整的URL,从而触发前端路由的跳转。然而,Hash模式的一个显著缺点是URL中带有"#"号,这不符合传统Web开发中URL的直观性和美观性,也可能对SEO(搜索引擎优化)产生不利影响。

第16行,用export关键字导出以上所设置的路由。

4. 为应用指定路由

在页面入口文件main.js中指定路由,代码如下:

```
 1. import { createApp } from 'vue'
 2. import App from './App.vue'
 3. import ElementPlus from 'element-plus'              // 导入 Element Plus
 4. import 'element-plus/theme-chalk/index.css'        // 导入 Element Plus 样式表
 5.
 6. import router from "./router/index"                // 导入路由
 7.
 8. const app = createApp(App)
 9. app.use(ElementPlus)                               // 注册 Element Plus
10. app.use(router)                                    // 为应用指定路由
11. app.mount('#app')
```

其中,第6行代码为导入自定义的路由;第10行代码为Vue应用指定了路由。

5.2.2 单页布局

修改Vue 3项目的根组件文件App.vue,代码如下:

```
 1. < script setup >
 2.   document.title = "甜点信息管理系统";
 3. </script >
 4. <! -- 页面布局 -- >
 5. < template >
 6.   < div class = "common - layout">
 7.    < el - container >
 8.      < el - header >
 9.       甜点信息管理系统
10.      < div id = "loginOut">
11.        < el - button type = "text"
12.         @click = "handleLogin">< img src = ". . /public/img/login. png">登录</el - button >
13.        欢迎^^< el - button type = "text"
14.        @click = "handleLogout">< img src = ". . /public/img/logout. png">退出</el - button >
15.      </div >
16.     </el - header >
17.     < div class = "common - layout">
18.      < el - container >
19.        < el - aside width = "200px">
20.         < router - link to = "/Register" class = "nav - link module">用户注册</router -
link >
21.         < router - link to = "/Category" class = "nav - link module">分类管理</router -
link >
22.         < router - link to = "/Dessert" class = "nav - link module">甜点管理</router - link >
23.        </el - aside >
24.        < el - main >< router - view /></el - main >
25.      </el - container >
26.     </div >
27.     < el - footer > Copyright © 2024 </el - footer >
28.    </el - container >
29.   </div >
30. </template >
31.
32. < style scoped >
33. body{margin:0; font - family: 微软雅黑;}
34. header,footer{ width:1000px;height: 40px; background: slateblue;text - align: center;
line - height:40px;
35.    color: azure; position: relative;}
36. # loginOut{position: absolute;right:3px;top:0px;}
37. # loginOut . el - button - - text{background - color: transparent; color: white; font -
size:15px;}
38. # loginOut img{ width: 15px; }
39. aside{width: 100px; height: 640px; background: # a6a6a6;padding - top: 20px;}
40. . module{ text - decoration: none; background: black; color:white;
41.    display: block;   margin: 8px auto;   width: 80 % ;
42.    height: 35px; line - height: 35px; text - align: center; border - radius: 5px;}
43. . module:hover{ box - shadow: 3px 3px 3px}
44. </style >
```

其中代码的第 2 行，设置了页面的 title 信息。

第 5～30 行，参考 Element Plus 官网中常见的页面布局文档(component/container.

html),进行了前端项目整体页面的布局,并在此基础上,增加了相应的功能操作按钮,单击这些按钮将切换到 Vue 组件上,显示相应的功能模块界面。

接下来进行布局效果测试:执行 npm run dev 命令启动 Vue 应用,通过 Chrome 浏览器访问 http://localhost:5173,此时访问的是根路径("/")资源,而在页面主体部分上显示的则是 Home.vue 组件的渲染效果,如图 5-3 所示。

图 5-3 访问根路径("/")显示 Home.vue 组件渲染效果

单击左侧栏“用户注册”按钮后,页面将导航至“/Register”路径,而实际会显示 Register.vue 组件信息“用户注册”,如图 5-4 所示。

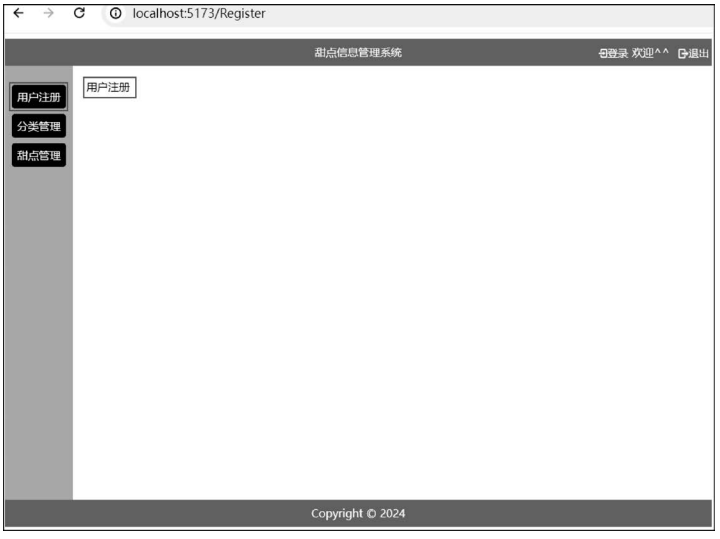

图 5-4 访问路径“/Register”显示 Register.vue 组件信息

继续单击左侧栏“分类管理”，将呈现 Category. vue 组件信息“甜点分类”；单击左侧栏“甜点管理”，将呈现 Dessert. vue 组件的“甜点信息”。

至此，实践项目的整体布局设计已顺利完成。

5.3　练习

试为“员工管理系统”项目实现整体布局。

(1) 集成 Element Plus。

在“员工管理系统”前端 Vue 3 项目中，安装和添加 Element Plus 组件，并在 main. js 文件中导入和注册 Element Plus 组件库。

(2) 实施路由配置。

将练习 3.4 中设计界面转化为 Vue 组件；安装配置路由模块 vue-router；设置 vue-router 路由；为 Vue 应用指定路由。

(3) 设计单页布局。

修改 Vue 3 项目的根组件文件 App. vue，按需求设计单页布局。