Vite2 + Vue3 + TypeScript + Element Plus开发后台管理系统实战视频教程

第01讲 课程介绍与项目展示

1、课程适合对象

```
有Vue2、Vue3组合api基础知识，TypeScript基础知识的小伙伴；
```

2、课程涉及技术

```
1、CSS3
2、TypeScript
3、Vue3.2
4、Vuex4.x
5、Vue Router4.x
6、Vite2.x
```

3、课程收获

```
1、掌握Vue3.2语法糖的使用
2、掌握Vue3中组合api的使用
3、掌握组件中业务逻辑抽离的方法；
4、掌握TypeScript在Vue3中的使用；
5、掌握动态菜单、动态路由、按钮权限的实现方式
6、vue3中全局挂载使用方式
7、vue3父子组件的使用
8、vue3中echarts的使用
9、token、权限验证
10、vuex4 + Ts在 commit,getters,dispatch中的代码提示
11、Icons图标动态生成
```

4、在线体验地址

```
http://test.ynitmk.cn/
```

第02讲 项目创建

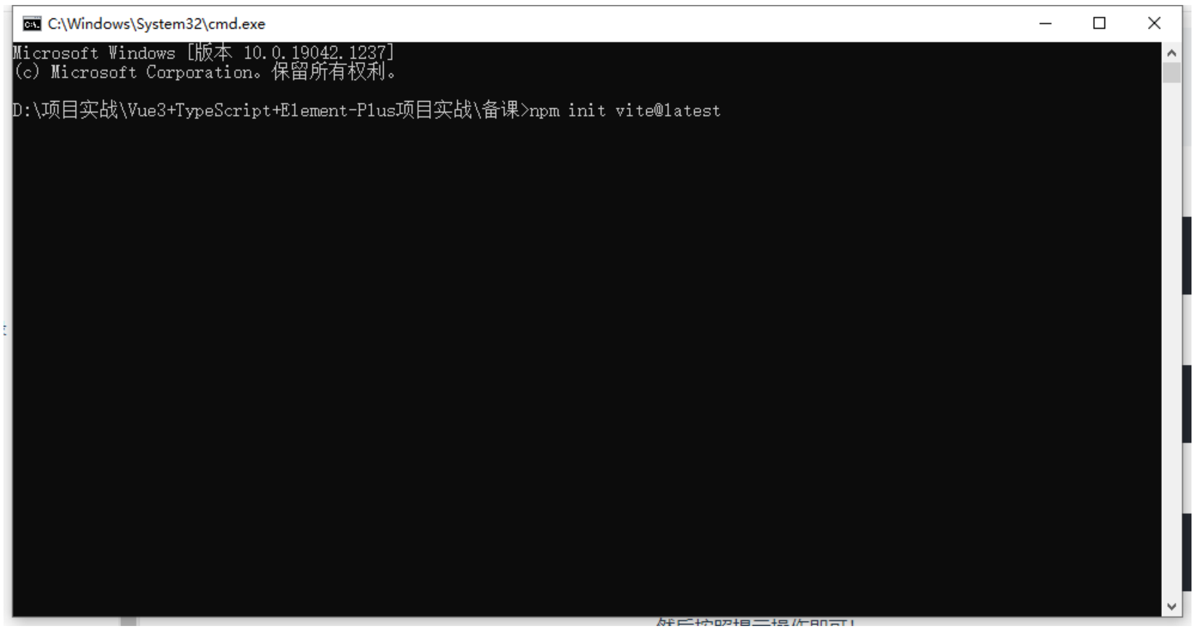
兼容性注意

```
Vite 需要 Node.js 版本 >= 12.0.0

node -v  //查看版本号
```

1、创建项目

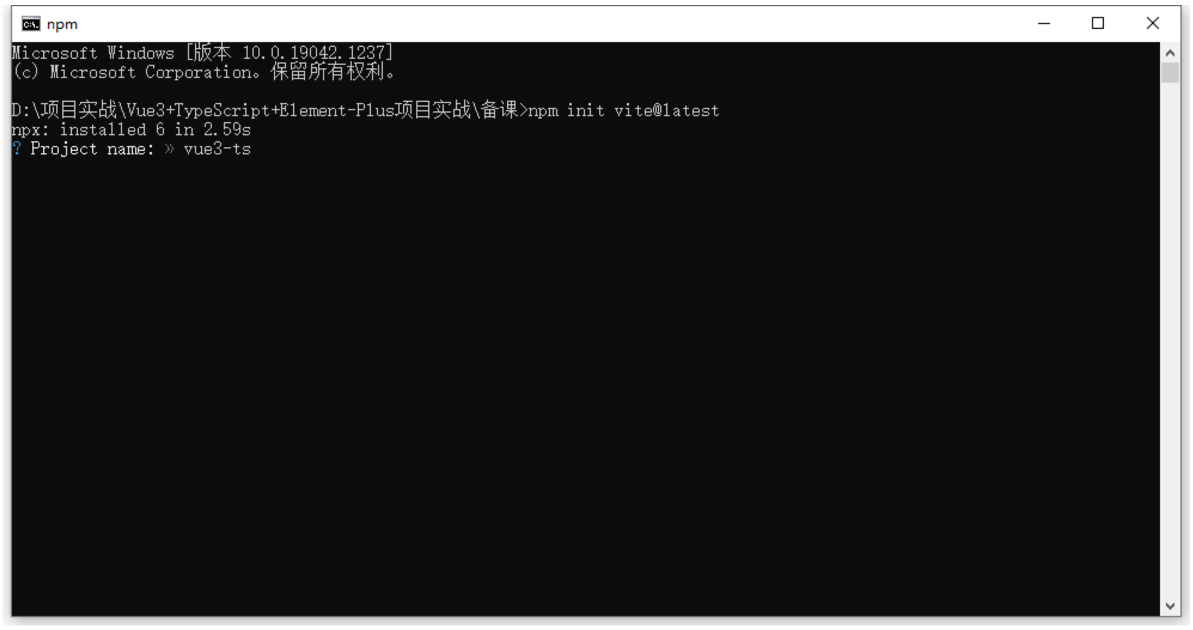
新建项目存放目录，cmd 进入到存放目录，执行如下命令创建项目

```
npm init vite@latest

OR

yarn create vite
```



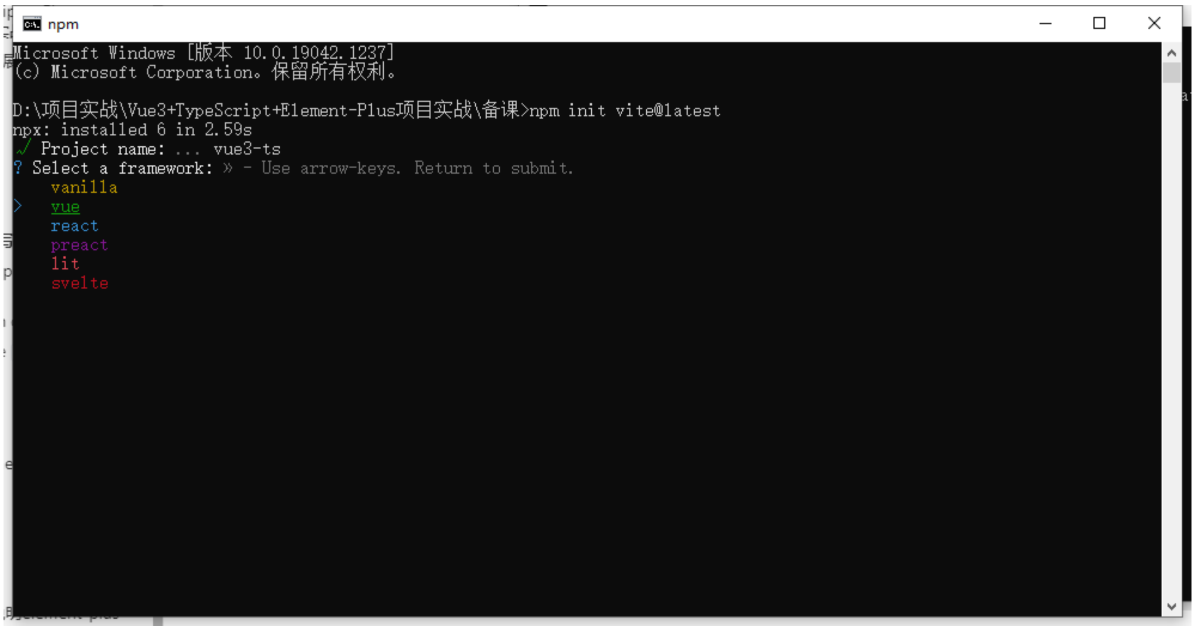
```
Microsoft Windows [版本 10.0.19042.1237]
(c) Microsoft Corporation。保留所有权利。

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课>npm init vite@latest
```

2、 Project name填写项目名称



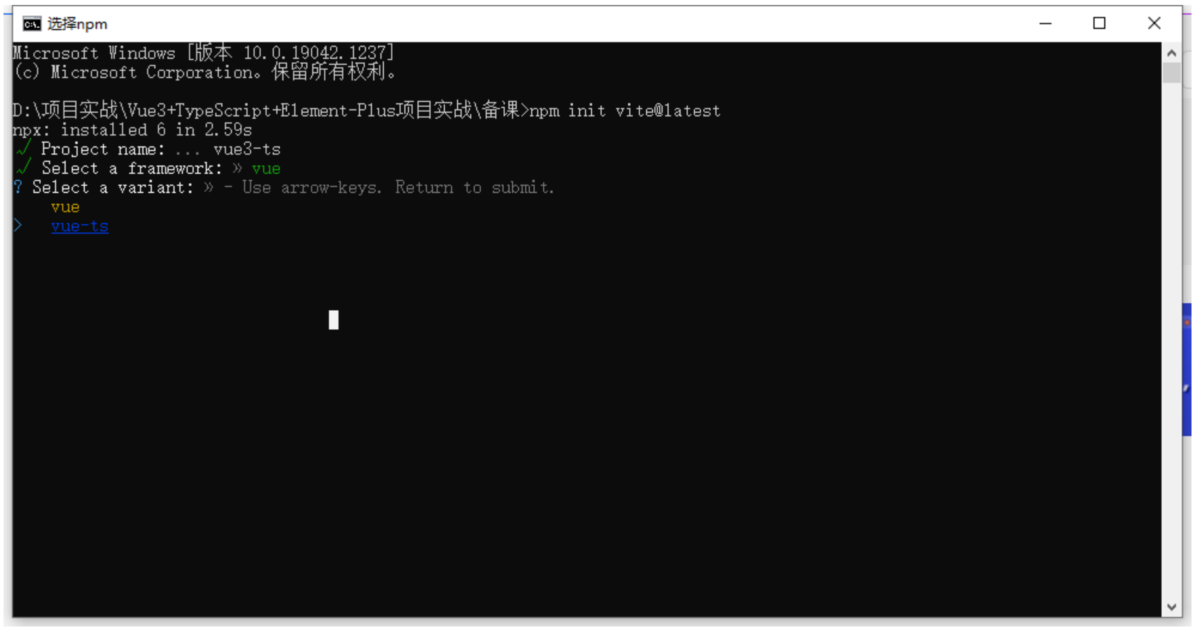
```
Microsoft Windows [版本 10.0.19042.1237]
(c) Microsoft Corporation。保留所有权利。

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课>npm init vite@latest
npx: installed 6 in 2.59s
? Project name: » vue3-ts
```

3、选择框架，此处选择vue



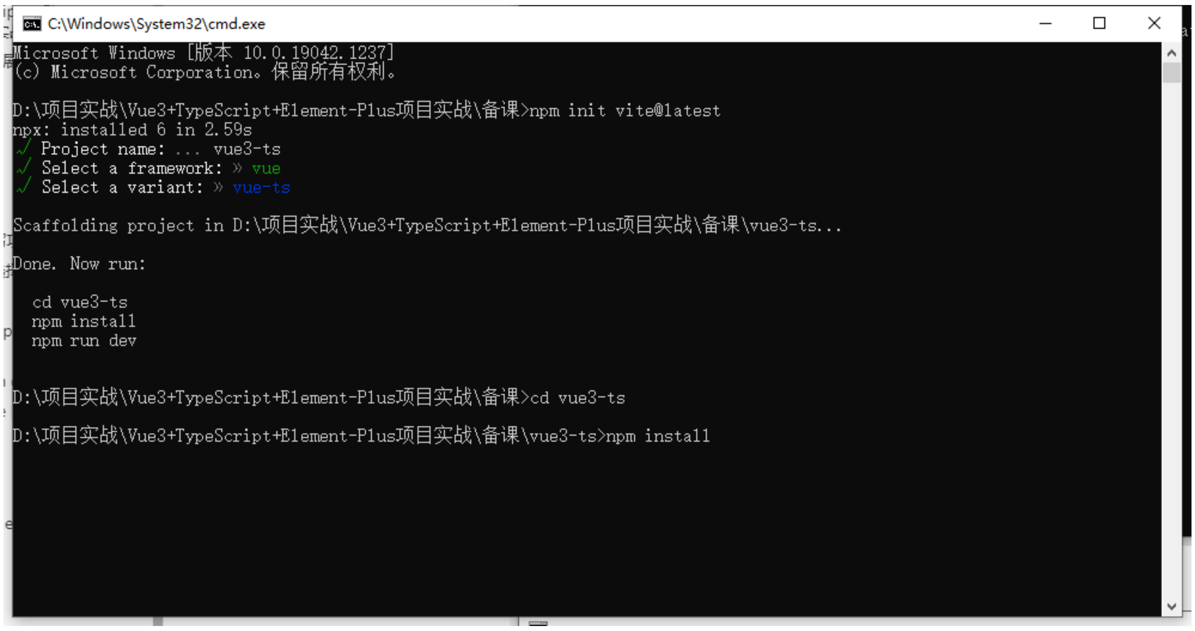
```
Microsoft Windows [版本 10.0.19042.1237]
(c) Microsoft Corporation。保留所有权利。

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课>npm init vite@latest
npx: installed 6 in 2.59s
√ Project name: ... vue3-ts
? Select a framework: » - Use arrow-keys. Return to submit.
    vanilla
>   vue
    react
    preact
    lit
    svelte
```

4、选择 vue-ts



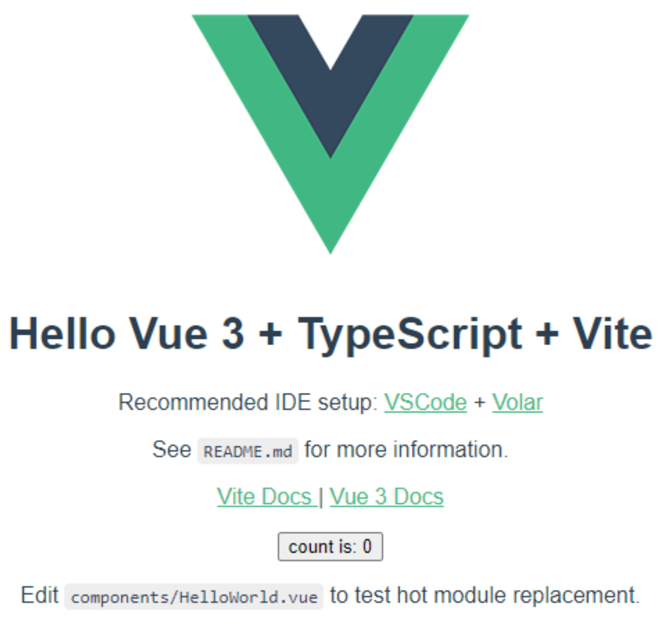
```
Microsoft Windows [版本 10.0.19042.1237]
(c) Microsoft Corporation。保留所有权利。

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课>npm init vite@latest
npx: installed 6 in 2.59s
√ Project name: ... vue3-ts
√ Select a framework: » vue
? Select a variant: » - Use arrow-keys. Return to submit.
    vue
>   vue-ts
```

5、进入项目，执行 npm install 安装依赖



```
Microsoft Windows [版本 10.0.19042.1237]
(c) Microsoft Corporation。保留所有权利。

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课>npm init vite@latest
npx: installed 6 in 2.59s
√ Project name: ... vue3-ts
√ Select a framework: » vue
√ Select a variant: » vue-ts

Scaffolding project in D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课\vue3-ts...

Done. Now run:

  cd vue3-ts
  npm install
  npm run dev

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课>cd vue3-ts

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课\vue3-ts>npm install
```

6、运行项目 npm run dev


Hello Vue 3 + TypeScript + Vite
Recommended IDE setup: VSCode + Volar
See README.md for more information.
Vite Docs | Vue 3 Docs
count is: 0
Edit components/HelloWorld.vue to test hot module replacement.

7、解决 Network: use `--host` to expose

vite.config.js配置文件，添加如下配置

```
server: {
    host: '0.0.0.0',    //解决“vite use `--host` to expose”
    port: 8080,
    open: true
}
```

8、vite 配置别名

```
npm install @types/node --save-dev
```

```
import { defineConfig } from 'vite'
import vue from '@vitejs/plugin-vue'
import { resolve } from 'path'
// https://vitejs.dev/config/
export default defineConfig({
  plugins: [vue()],
  server: {
    host: '0.0.0.0',    //解决“vite use `--host` to expose”
    port: 8080,
    open: true
  },
  resolve:{
    alias:[
      {
        find:'@',
        replacement:resolve(__dirname,'src')
      }
    ]
  }
})
```

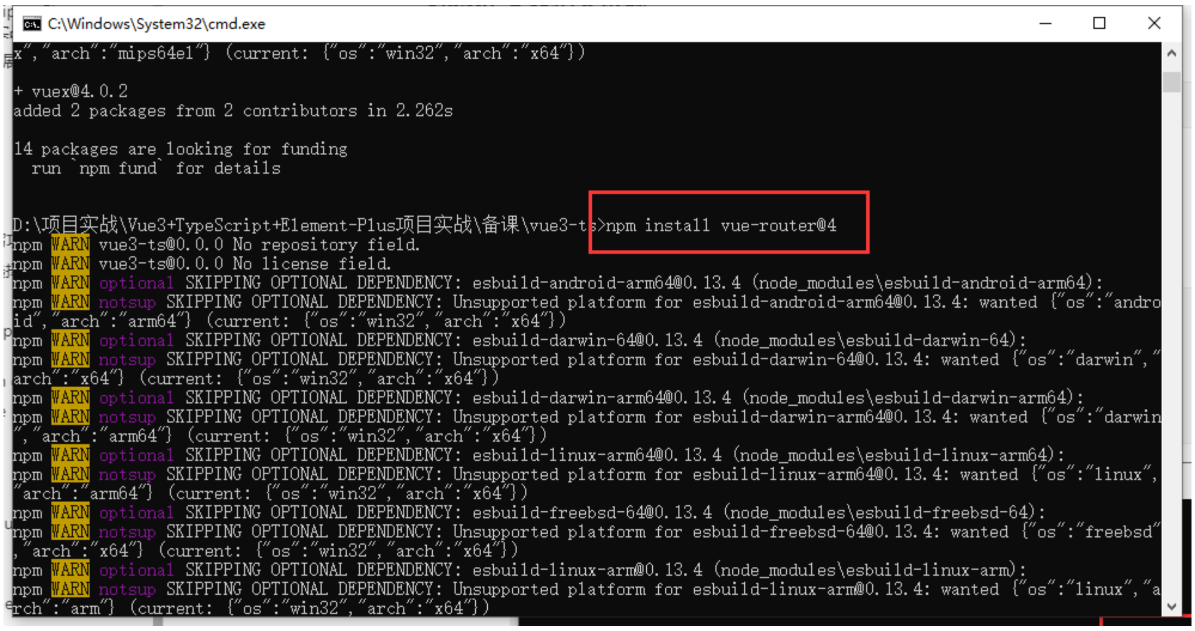
第03讲 安装路由依赖

插件安装

```
1、禁用Vetur
2、安装Vue Language Features (Volar)
3、安装Element UI Snippets
```

1、安装路由

```
npm install vue-router@4
```



```
C:\Windows\System32\cmd.exe
x","arch":"mips64el"} (current: {"os":"win32","arch":"x64"})

+ vuex@4.0.2
added 2 packages from 2 contributors in 2.262s

14 packages are looking for funding
  run `npm fund` for details

D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课\vue3-ts>npm install vue-router@4
npm WARN vue3-ts@0.0.0 No repository field.
npm WARN vue3-ts@0.0.0 No license field.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: esbuild-android-arm64@0.13.4 (node_modules\esbuild-android-arm64):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for esbuild-android-arm64@0.13.4: wanted {"os":"andro
id","arch":"arm64"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: esbuild-darwin-64@0.13.4 (node_modules\esbuild-darwin-64):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for esbuild-darwin-64@0.13.4: wanted {"os":"darwin","
arch":"x64"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: esbuild-darwin-arm64@0.13.4 (node_modules\esbuild-darwin-arm64):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for esbuild-darwin-arm64@0.13.4: wanted {"os":"darwin
","arch":"arm64"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: esbuild-linux-arm64@0.13.4 (node_modules\esbuild-linux-arm64):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for esbuild-linux-arm64@0.13.4: wanted {"os":"linux",
"arch":"arm64"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: esbuild-freebsd-64@0.13.4 (node_modules\esbuild-freebsd-64):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for esbuild-freebsd-64@0.13.4: wanted {"os":"freebsd"
,"arch":"x64"} (current: {"os":"win32","arch":"x64"})
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: esbuild-linux-arm@0.13.4 (node_modules\esbuild-linux-arm):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for esbuild-linux-arm@0.13.4: wanted {"os":"linux","a
rch":"arm"} (current: {"os":"win32","arch":"x64"})
```

2、在src下新建router目录，然后新建index.ts文件

```
// vue2-router
const router = new VueRouter({
  mode:  history ,
  ...
})

// vue-next-router
import { createRouter, createWebHistory } from  'vue-router'
const router = createRouter({
  history: createWebHistory(),
  ...
})
```

```
import { createRouter, createWebHistory, RouteRecordRaw } from 'vue-router'
import Layout from '../components/HelloWorld.vue'

const routes: Array<RouteRecordRaw> = [
  {
    path: '/',
    name: 'Home',
    component: Layout
  }
]

const router = createRouter({
  history: createWebHistory(),
  routes
})

export default router
```

3、修改App.vue为如下所示

```
<template>
  <router-view/>
</template>

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

4、main.ts使用路由

```
import router from './router'
createApp(App).use(router).mount('#app')
```

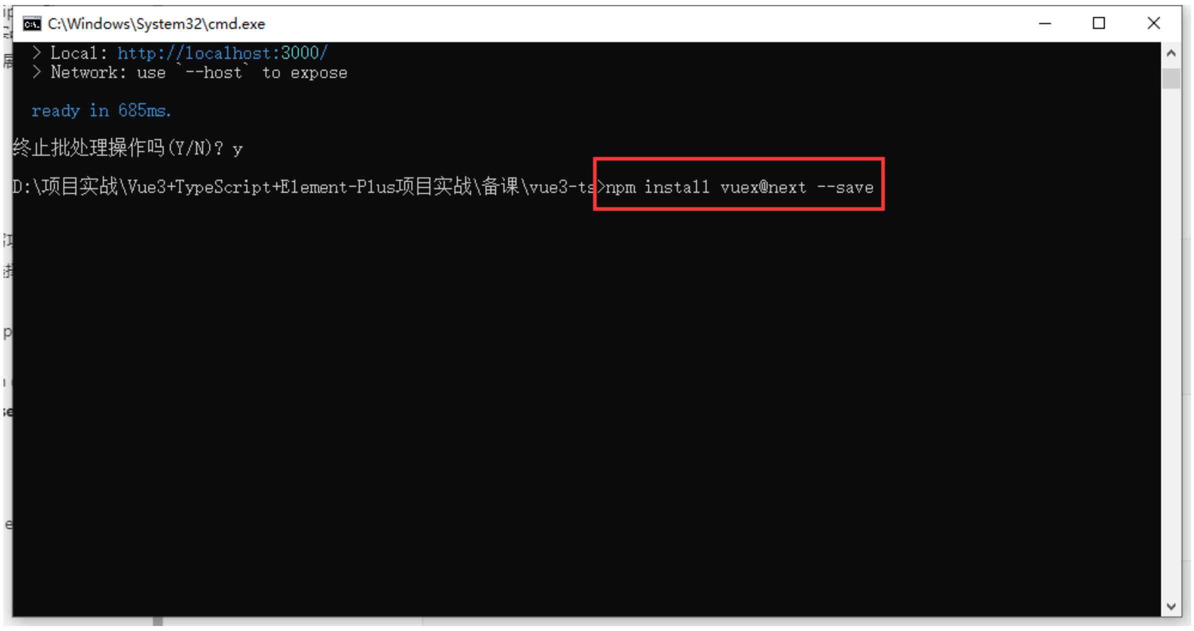
第04讲、安装Vuex

官网

```
https://vuex.vuejs.org/zh/
```

1、安装依赖

```
npm install vuex@next --save

OR

yarn add vuex@next --save
```



```
C:\Windows\System32\cmd.exe
  > Local: http://localhost:3000/
  > Network: use `--host` to expose

  ready in 685ms.
终止批处理操作吗(Y/N)? y
D:\项目实战\Vue3+TypeScript+Element-Plus项目实战\备课\vue3-ts>npm install vuex@next --save
```

2、在src下新建store目录，然后新建index.ts文件

```
import { InjectionKey } from 'vue'
import { createStore, useStore as baseUseStore, Store } from 'vuex'

export interface State {
  count: number
}

export const key: InjectionKey<Store<State>> = Symbol()

export const store = createStore<State>({
  state: {
    count: 0
  },
  mutations:{
    setCount(state:State,count:number){
      state.count = count
    }
  },
  getters:{
    getCount(state:State){
      return state.count
    }
  }
})

// 定义自己的 `useStore` 组合式函数
export function useStore () {
  return baseUseStore(key)
}
```

3、main.ts中使用

```
import { store, key } from './store'
createApp(App).use(store, key).use(router).mount('#app')
```

4、修改HelloWorld.vue组件为如下

```
<script setup lang="ts">
import { ref,computed } from 'vue'
import { useStore } from '../store';
const store = useStore();
const count = ref(0);
const showcount = computed(()=>{
  return store.getters['getCount']
});

const addBtn = ()=>{
  store.commit('setCount',++count.value)
}
</script>

<template>

  <p>{{ showcount }}</p>
  <button @click="addBtn">增加</button>
</template>

<style scoped>

</style>
```

第05讲 eslint、css 预处理器sass安装

1、ts使用@符号引入

tsconfig.json

```
{
  "compilerOptions": {
    "target": "esnext",
    "useDefineForClassFields": true,
    "module": "esnext",
    "moduleResolution": "node",
    "strict": true,
    "jsx": "preserve",
    "sourceMap": true,
    "resolveJsonModule": true,
    "esModuleInterop": true,
    "skipLibCheck": true, //解决打包报`vue-tsc --noEmit && vite build`的错,忽略所有的
声明文件（*.d.ts）的类型检查
    "lib": [
      "esnext",
      "dom"
    ],
    "baseUrl": ".",
    "paths": {
      "@/*": [
        "src/*"
      ]
    }
  },
  "include": [
    "src/**/*.ts",
    "src/**/*.d.ts",
    "src/**/*.tsx",
    "src/**/*.vue"
  ],
   // ts 排除的文件
   "exclude": ["node_modules"]
}
```

2、Eslint安装

```
npm install --save-dev eslint eslint-plugin-vue
```

3、新建 .eslintrc.js 文件

```
module.exports = {
    root: true,
    parserOptions: {
        sourceType: 'module'
    },
    parser: 'vue-eslint-parser',
    extends: ['plugin:vue/vue3-essential', 'plugin:vue/vue3-strongly-
recommended', 'plugin:vue/vue3-recommended'],
    env: {
        browser: true,
        node: true,
        es6: true
    },
    rules: {
        'no-console': 'off',
        'comma-dangle': [2, 'never'] //禁止使用拖尾逗号
    }
}
```

4、添加 css 预处理器 sass

```
npm install -D sass sass-loader
```

第06讲 项目中引入element-plus

1、element-plus官网

```
https://element-plus.gitee.io/zh-CN/guide/installation.html
```

2、安装element-plus

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

3、main.ts中引入

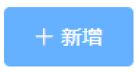
```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import ElementPlus from 'element-plus'
import 'element-plus/dist/index.css'
createApp(App).use(store).use(router).use(ElementPlus).mount('#app')
```

4、测试

在views目录的Home.vue页面加入一个按钮

```
<template>
  <div class="home">
    <el-button type="primary" size="default" icon='el-icon-plus'>新增</el-button>
  </div>
</template>
<script lang="ts">
import { defineComponent } from 'vue';
export default defineComponent({
  name: 'Home',
  components: {

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

5、出现如下所示，说明element-plus引入成功



第07讲 主界面布局

插件安装

```
1、禁用Vetur
2、安装Vue Language Features (Volar)
3、安装Element UI Snippets
```

1、找到index.html添加如下样式,设置高度

```
<style>
  html,body,#app{
    padding: 0px;
    margin: 0px;
    height: 100%;
    box-sizing: border-box;
  }
</style>
```

完整源码

```
<!DOCTYPE html>
<html lang="">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title><%= htmlWebpackPlugin.options.title %></title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but <%= htmlWebpackPlugin.options.title %> doesn't
work properly without JavaScript enabled. Please enable it to continue.</strong>
    </noscript>
    <div id="app"></div>
    <!-- built files will be auto injected -->
  </body>
</html>
<style>
  html,body,#app{
    padding: 0px;
    margin: 0px;
    height: 100%;
    box-sizing: border-box;
  }
</style>
```

2、App.vue修改为如下所示

```
<template>
  <router-view/>
</template>

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

3、src目录下新建layout目录，并新建Index.vue主页面组件

```
<template>
    <el-container class="layout">
        <el-aside class="asside" width="200px">Aside</el-aside>
        <el-container class="layout">
            <el-header class="header">Header</el-header>
            <el-main class="main">Main</el-main>
        </el-container>
    </el-container>
</template>
<script setup lang="ts">
</script>
<style lang="scss">
.layout {
    height: 100%;
    .asside {
        background-color: blueviolet;
    }
    .header {
        background-color: chocolate;
    }
    .main {
        background-color: darkcyan;
    }
}
</style>
```

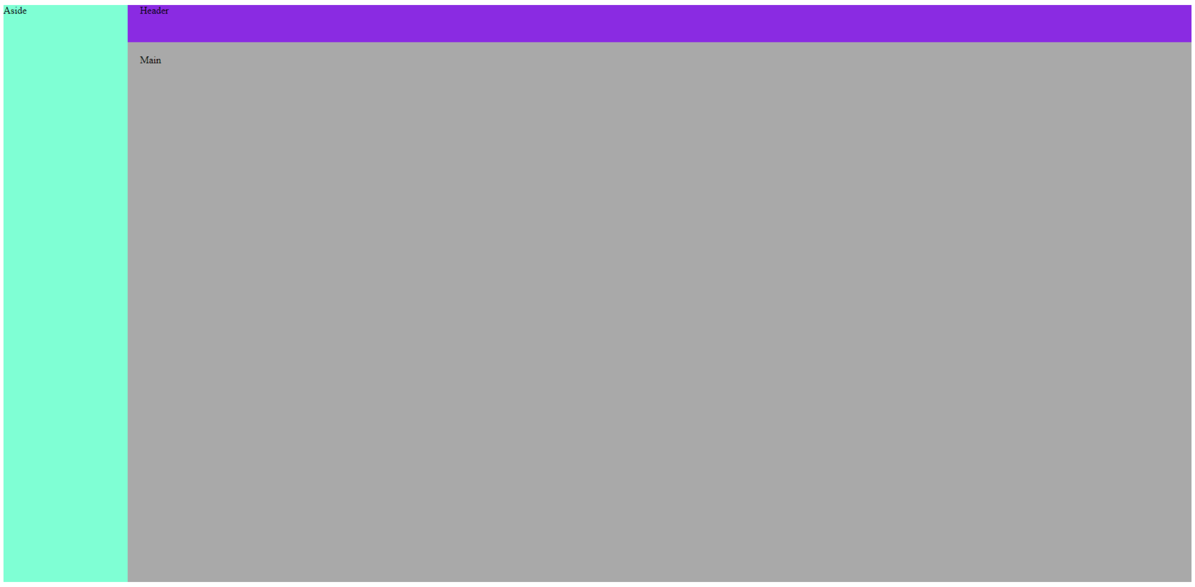
4、在router中引入主页面组件

```
import { createRouter, createWebHistory, RouteRecordRaw } from 'vue-router'
import Layout from '@/layout/Index.vue'

const routes: Array<RouteRecordRaw> = [
  {
    path: '/',
    name: 'Home',
    component: Layout
  }
]

const router = createRouter({
  history: createWebHistory(),
  routes
})

export default router
```

```
```

5、启动项目

```
http://localhost:8080/
```



第08讲 左侧导航菜单制作讲解1

前置知识

```
https://v3.cn.vuejs.org/api/sfc-script-setup.html
https://github.com/vuejs/rfcs/tree/master/active-rfcs

1、setup语法糖中，组件的使用方式：
   a、setup语法糖中，引入的组件可以直接使用，无需再通过components进行注册，并且无法指定当前组件的名字，它会自动以文件名为主，不用再写name属性了；
   b、setup语法糖中 定义的数据和方法，直接可以在模板中使用，无需要 return

2、ref使用：
    a、定义：const xxx = ref(sss);
    b、作用： 定义一个响应式的数据
    c、js中操作，需要使用xxx.value
    d、模板中使用不需要 .value
```

vue3的三种写法

```
1、Option API
这个就不多说了，会写vue的都会，这就是vue2 大家最常用的 选项式API
https://v3.cn.vuejs.org/api/options-api.html

2、Composition API
组合式API，也就是Vue3诞生以来，最为大家喜欢语法更新，也是我们下面script setup 语法的基础

3、script setup（Composition API 的语法糖）
<script setup> 是在单文件组件 (SFC) 中使用组合式 API 的编译时语法糖。相比于之前的代码更简洁。
能够使用纯 Typescript 声明 props 和发出事件。
更好的 IDE 类型推断性能
script setup 已经由实验状态正式毕业，现提供稳定版本
在添加了setup的script标签中，我们不必声明和方法，这种写法会自动将所有顶级变量、函数，均会自动暴露给模板（template）使用
```

```
<script lang="ts">
import { ref, defineComponent } from "vue";
export default defineComponent({
  name: "HelloWorld",
  props: {
    msg: {
      type: String,
      required: true,
    },
  },
  setup: () => {
    const count = ref(0);
    return { count };
  },
});
</script>
```

```
<script lang="ts" setup>
import { ref, defineProps } from "vue";
const count = ref(0);
const props = defineProps({
  msg: {
    type: String,
    required: true,
  },
});
</script>
```

1、抽离头部组件

在layout目录下新建header目录，然后新建Header.vue组件；如下所示

```
<template>
    <el-container class="layout">
        <el-aside class="asside" width="auto">
            <MenuBarVue></MenuBarVue>
        </el-aside>
        <el-container class="layout">
            <el-header class="header">
                <HeaderVue></HeaderVue>
            </el-header>
            <el-main class="main">Main</el-main>
        </el-container>
    </el-container>
</template>
<script setup lang="ts">
import MenuBarVue from './menu/MenuBar.vue';
import HeaderVue from './header/Header.vue';
</script>
<style lang="scss" scoped>
.layout {
    height: 100%;
    .asside {
        // background-color: #304156
    }
    .header {
       height: 50px;
       border-bottom: 1px solid #e5e5e5;
    }
    .main {
        // background-color: darkcyan;
    }
}
</style>
```

2、抽离左侧菜单组件

在layout目录下新建menu目录，然后新建MenuBar.vue组件

```
<template>
    <el-menu
        default-active="2"
        class="el-menu-vertical-demo"
        :collapse="isCollapse"
    >
        <el-sub-menu index="1">
            <template #title>
                <i class="el-icon-location"></i>
                <span>Navigator One</span>
            </template>
            <el-menu-item-group>
                <template #title>
                    <span>Group One</span>
                </template>
                <el-menu-item index="1-1">item one</el-menu-item>
                <el-menu-item index="1-2">item two</el-menu-item>
            </el-menu-item-group>
            <el-menu-item-group title="Group Two">
                <el-menu-item index="1-3">item three</el-menu-item>
            </el-menu-item-group>
            <el-sub-menu index="1-4">
                <template #title>
                    <span>item four</span>
                </template>
                <el-menu-item index="1-4-1">item one</el-menu-item>
            </el-sub-menu>
        </el-sub-menu>
        <el-menu-item index="2">
            <i class="el-icon-menu"></i>
            <template #title>Navigator Two</template>
        </el-menu-item>
        <el-menu-item index="3" disabled>
            <i class="el-icon-document"></i>
            <template #title>Navigator Three</template>
        </el-menu-item>
        <el-menu-item index="4">
            <i class="el-icon-setting"></i>
            <template #title>Navigator Four</template>
        </el-menu-item>
    </el-menu>
</template>
<script lang="ts" setup>
import { ref } from 'vue';
//控制菜单展开和关闭
const isCollapse = ref(false);
</script>
<style scoped>
.el-menu-vertical-demo:not(.el-menu--collapse) {
    width: 230px;
    min-height: 400px;
}
</style>
```

3、Index.vue中引入组件

```
<template>
    <el-container class="layout">
        <el-aside class="asside" width="auto">
            <MenuBarVue></MenuBarVue>
        </el-aside>
        <el-container class="layout">
            <el-header class="header">
                <HeaderVue></HeaderVue>
            </el-header>
            <el-main class="main">Main</el-main>
        </el-container>
    </el-container>
</template>
<script setup lang="ts">
import MenuBarVue from './menu/MenuBar.vue';
import HeaderVue from './header/Header.vue';
</script>
<style lang="scss" scoped>
.layout {
    height: 100%;
    .asside {
        // background-color: #304156
    }
    .header {
        height: 50px;
        border-bottom: 1px solid #e5e5e5;
    }
    .main {
        // background-color: darkcyan;
    }
}
</style>
```

第09讲 左侧导航菜单制作讲解2

1、前置知识

```
1、setup语法糖父子组件传值的方法
   a、父组件传值给子组，通过属性绑定方式
   b、子组件通过 defineProps接收,无需显示的引入;
   c、插槽的使用
2、reactive：响应式数据定义，适用于对象类型
```

2、抽离MenuItem.vue组件

```
<template>
  <template v-for="menu in menuList" :key="menu.path">
    <el-sub-menu
      v-if="menu.children && menu.children.length > 0"
      :index="menu.path"
      :key="menu.path"
    >
      <template #title>
        <i :class="menu.meta.icon"></i>
        <span>{{ menu.meta.title }}</span>
      </template>
      <menu-item :menuList="menu.children"></menu-item>
    </el-sub-menu>
    <el-menu-item
    style="color: #f4f4f5"
      v-else
      :index="menu.path">
      <i :class="menu.meta.icon"></i>
      <template #title>{{ menu.meta.title }}</template>
    </el-menu-item>
  </template>
</template>

<script setup lang="ts">
import {defineProps} from 'vue'
//接收父组件MenuBar传递过来的值
defineProps(['menuList'])
</script>

<style scoped>

</style>
```

3、修改MenuBar.vue组件为如下

```
<!--
    default-active：当前激活菜单的index，此处设置为当前路由的path
    collapse: 是否水平折叠收起菜单（仅在 mode 为 vertical 时可用）
    background-color : 菜单的背景色
    unique-opened: 是否只保持一个子菜单的展开
 -->
<template>
    <el-menu
        default-active="2"
        class="el-menu-vertical-demo"
        :collapse="isCollapse"
        background-color="#304156"
        unique-opened
    >
        <MenuItemVue :menuList='menuList'></MenuItemVue>
    </el-menu>
</template>
<script lang="ts" setup>
import MenuItemVue from './MenuItem.vue';
import { ref,reactive } from 'vue';
//菜单数据
let menuList = reactive([
    {
        path: '/dashboard',
        component: "Layout",
        meta: {
            title: "首页",
            icon: "el-icon-s-home",
            roles: ["sys:manage"]
        },
        children: []
    },
    {
        path: "/system",
        component: "Layout",
        alwaysShow: true,
        name: "system",
        meta: {
            title: "系统管理",
            icon: "el-icon-menu",
            roles: ["sys:manage"],
            parentId: 0,
        },
        children: [
            {
                path: "/department",
                component: "/system/department/department",
                alwaysShow: false,
                name: "department",
                meta: {
                    title: "机构管理",
                    icon: "el-icon-document",
                    roles: ["sys:dept"],
                    parentId: 17,
                },
            },
            {
                path: "/userList",
                component: "/system/User/UserList",
                alwaysShow: false,
                name: "userList",
                meta: {
                    title: "用户管理",
                    icon: "el-icon-s-custom",
                    roles: ["sys:user"],
                    parentId: 17,
                },
            },
            {
                path: "/roleList",
                component: "/system/Role/RoleList",
                alwaysShow: false,
                name: "roleList",
                meta: {
                    title: "角色管理",
                    icon: "el-icon-s-tools",
                    roles: ["sys:role"],
                    parentId: 17,
                },
            },
            {
                path: "/menuList",
                component: "/system/Menu/MenuList",
                alwaysShow: false,
                name: "menuList",
                meta: {
                    title: "权限管理",
                    icon: "el-icon-document",
                    roles: ["sys:menu"],
                    parentId: 17,
                },
            },
        ],
    },
    {
        path: "/goods",
        component: "Layout",
        alwaysShow: true,
        name: "goods",
        meta: {
            title: "商品管理",
            icon: "el-icon-document",
            roles: ["sys:goods"],
            parentId: 0,
        },
        children: [
            {
                path: "/goodCategory",
                component: "/goods/goodsCategory/goodsCategoryList",
                alwaysShow: false,
                name: "goodCategory",
                meta: {
                    title: "商品分类",
                    icon: "el-icon-document",
                    roles: ["sys:goodsCategory"],
                    parentId: 34,
                },
            },
        ],
    },
    {
        path: "/systenConfig",
        component: "Layout",
        alwaysShow: true,
        name: "systenConfig",
        meta: {
            title: "系统工具",
            icon: "el-icon-document",
            roles: ["sys:systenConfig"],
            parentId: 0,
        },
        children: [
            {
                path: "/document",
                component: "/system/config/systemDocument",
                alwaysShow: false,
                name: "http://42.193.158.170:8089/swagger-ui/index.html",
                meta: {
                    title: "接口文档",
                    icon: "el-icon-document",
                    roles: ["sys:document"],
                    parentId: 42,
                },
            },
        ],
    },
]);
//控制菜单展开和关闭
const isCollapse = ref(false);
</script>
<style scoped>
.el-menu-vertical-demo:not(.el-menu--collapse) {
    width: 230px;
    min-height: 400px;
}

.el-menu {
  border-right: none;
}
::v-deep .el-sub-menu .el-sub-menu__title {
  color: #f4f4f5 !important;
}
/* .el-submenu .is-active .el-submenu__title {
  border-bottom-color: #1890ff;
} */
::v-deep .el-menu .el-menu-item {
  color: #bfcbd9;
}

/* 菜单点中文字的颜色 */
::v-deep .el-menu-item.is-active {
  color: #409eff !important;
}
/* 当前打开菜单的所有子菜单颜色 */
::v-deep .is-opened .el-menu-item {
  background-color: #1f2d3d !important;
}
/* 鼠标移动菜单的颜色 */
::v-deep .el-menu-item:hover {
  background-color: #001528 !important;
}
</style>
```

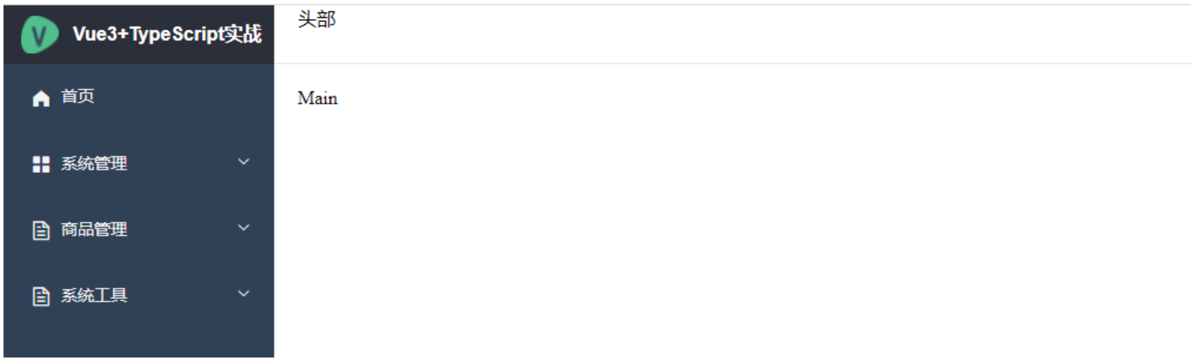
第10讲 导航菜单logo组件制作

1、解决 ::v-deep 警告提示

```
[@vue/compiler-sfc] ::v-deep usage as a combinator has been deprecated. Use
:deep(<inner-selector>) instead
```

```

.el-menu-vertical-demo:not(.el-menu--collapse) {
  width: 230px;
  min-height: 400px;
}
.el-menu {
  border-right: none;
}

:deep(.el-sub-menu .el-sub-menu__title){
     color: #f4f4f5 !important;
}

:deep(.el-menu .el-menu-item){
    color: #bfcbd9;
}
/* 菜单点中文字的颜色 */

:deep(.el-menu-item.is-active){
    color: #409eff !important;
}
/* 当前打开菜单的所有子菜单颜色 */

:deep(.is-opened .el-menu-item){
    background-color: #1f2d3d !important;
}
/* 鼠标移动菜单的颜色 */

:deep(.el-menu-item:hover){
    background-color: #001528 !important;
}
```

2、效果展示



3、在menu目录下，新建MenuLogo.vue组件

```
<template>
    <div class="logo">
        <img src="https://wpimg.wallstcn.com/69a1c46c-eb1c-4b46-8bd4-
e9e686ef5251.png" alt="logo" />
        <span class="title">Vue3+TypeScript实战</span>
    </div>
</template>
<script setup lang="ts">
</script>
<style lang="scss" scoped>
.logo {
    background-color: #2b2f3a;
    height: 50px;
    border: none;
    line-height: 50px;
    display: flex;
    align-items: center;
    padding-left: 15px;
    color: #fff;
    img {
        width: 32px;
        height: 32px;
        margin-right: 12px;
    }
    span {
        font-weight: 600;
        line-height: 50px;
        font-size: 16px;
        font-family: Avenir, Helvetica Neue, Arial, Helvetica, sans-serif;
        vertical-align: middle;
    }
}
</style>
```

4、在MenuBar.vue组件中引入MenuLogo.vue组件，并使用

```
<template>
    <MenuLogo></MenuLogo>
    <el-menu
        default-active="2"
        class="el-menu-vertical-demo"
        :collapse="isCollapse"
        @open="handleOpen"
        @close="handleClose"
        background-color="#304156"
    >
        <MenuItem :menuList='menuList'></MenuItem>
    </el-menu>
</template>
<script setup lang="ts">
import { ref,reactive } from 'vue'
import MenuItem from './MenuItem.vue'
import MenuLogo from './MenuLogo.vue'
// setup语法糖中 定义的数据和方法，直接可以在模板中使用，无需要 return
//菜单数据
let menuList = reactive([
    {
        path: '/dashboard',
        component: "Layout",
        meta: {
            title: "首页",
            icon: "el-icon-s-home",
            roles: ["sys:manage"]
        },
        children: []
    },
    {
        path: "/system",
        component: "Layout",
        alwaysShow: true,
        name: "system",
        meta: {
            title: "系统管理",
            icon: "el-icon-menu",
            roles: ["sys:manage"],
            parentId: 0,
        },
        children: [
            {
                path: "/department",
                component: "/system/department/department",
                alwaysShow: false,
                name: "department",
                meta: {
                    title: "机构管理",
                    icon: "el-icon-document",
                    roles: ["sys:dept"],
                    parentId: 17,
                },
            },
            {
                path: "/userList",
                component: "/system/User/UserList",
                alwaysShow: false,
                name: "userList",
                meta: {
                    title: "用户管理",
                    icon: "el-icon-s-custom",
                    roles: ["sys:user"],
                    parentId: 17,
                },
            },
            {
                path: "/roleList",
                component: "/system/Role/RoleList",
                alwaysShow: false,
                name: "roleList",
                meta: {
                    title: "角色管理",
                    icon: "el-icon-s-tools",
                    roles: ["sys:role"],
                    parentId: 17,
                },
            },
            {
                path: "/menuList",
                component: "/system/Menu/MenuList",
                alwaysShow: false,
                name: "menuList",
                meta: {
                    title: "权限管理",
                    icon: "el-icon-document",
                    roles: ["sys:menu"],
                    parentId: 17,
                },
            },
        ],
    },
    {
        path: "/goods",
        component: "Layout",
        alwaysShow: true,
        name: "goods",
        meta: {
            title: "商品管理",
            icon: "el-icon-document",
            roles: ["sys:goods"],
            parentId: 0,
        },
        children: [
            {
                path: "/goodCategory",
                component: "/goods/goodsCategory/goodsCategoryList",
                alwaysShow: false,
                name: "goodCategory",
                meta: {
                    title: "商品分类",
                    icon: "el-icon-document",
                    roles: ["sys:goodsCategory"],
                    parentId: 34,
                },
            },
        ],
    },
    {
        path: "/systenConfig",
        component: "Layout",
        alwaysShow: true,
        name: "systenConfig",
        meta: {
            title: "系统工具",
            icon: "el-icon-document",
            roles: ["sys:systenConfig"],
            parentId: 0,
        },
        children: [
            {
                path: "/document",
                component: "/system/config/systemDocument",
                alwaysShow: false,
                name: "http://42.193.158.170:8089/swagger-ui/index.html",
                meta: {
                    title: "接口文档",
                    icon: "el-icon-document",
                    roles: ["sys:document"],
                    parentId: 42,
                },
            },
        ],
    },
]);
const isCollapse = ref(false)
const handleOpen = (key: string | number, keyPath: string) => {
    console.log(key, keyPath)
}
const handleClose = (key: string | number, keyPath: string) => {
    console.log(key, keyPath)
}
</script>
<style scoped>
.el-menu-vertical-demo:not(.el-menu--collapse) {
    width: 230px;
    min-height: 400px;
}

.el-menu {
  border-right: none;
}
::v-deep .el-sub-menu .el-sub-menu__title {
  color: #f4f4f5 !important;
}
/* .el-submenu .is-active .el-submenu__title {
  border-bottom-color: #1890ff;
} */
::v-deep .el-menu .el-menu-item {
  color: #bfcbd9;
}
```

```
/* 菜单点中文字的颜色 */
::v-deep .el-menu-item.is-active {
  color: #409eff !important;
}
/* 当前打开菜单的所有子菜单颜色 */
::v-deep .is-opened .el-menu-item {
  background-color: #1f2d3d !important;
}
/* 鼠标移动菜单的颜色 */
::v-deep .el-menu-item:hover {
  background-color: #001528 !important;
}
</style>
```

第11讲 Element Plus的Icon图标使用

vue3 setup语法糖文档地址

```
1、https://v3.cn.vuejs.org/api/sfc-script-setup.html
2、https://github.com/vuejs/rfcs/tree/master/active-rfcs
```

前置知识

```
1、element plus图标使用
https://element-plus.org/zh-CN/component/icon.html

2、在 <script setup>中，动态组件的使用方式
https://v3.cn.vuejs.org/api/sfc-script-
setup.html#%E4%BD%BF%E7%94%A8%E7%BB%84%E4%BB%B6

3、在 <script setup>中组件使用方式： 引入直接使用，无需注册

4、TypeScritp中typeof 和 keyof的使用
```

官网提示

```
WARNING

Element Plus 团队正在将原有组件内的 Font Icon 向 SVG Icon 迁移，请多多留意 ChangeLog，及时获取到更新信息，Font Icon 将会在第一个正式发布被废弃，请尽快迁移
```

1、Element Plus图标的基本使用

```
1、安装
npm install @element-plus/icons

2、引入图标
import { Fold } from '@element-plus/icons'

3、使用方式
<el-icon>
    <Fold />
</el-icon>
```

2、动态生成菜单时使用Element Plus 图标

2.1、在main.ts把图标注册为全局组件

```
import { createApp } from 'vue'
import App from './App.vue'
import router from './router/index'
import { store, key } from '@/store/index'
import ElementPlus from 'element-plus'
import 'element-plus/dist/index.css'
// 统一导入el-icon图标
import * as Icons from '@element-plus/icons'
const app= createApp(App);
app.use(router).use(store,key).use(ElementPlus).mount('#app')
// 方式一
Object.keys(Icons).forEach((key) => {
    app.component(key,Icons[key as keyof typeof Icons])
});
方式二
const Icon = (props: { icon: string }) => {
    const { icon } = props;
    return createVNode(Icons[icon as keyof typeof Icons]);
};
app.component('Icon', Icon);
```

解决type 'string' can't be used to index type 'typeof 字符串不能做下标的错，在tsconfig.json的compilerOptions 中添加如下配置

方式一:

```
"suppressImplicitAnyIndexErrors": true, //解决用字符串做下标报错
```

方式二:

```
key as keyof typeof Icons
```

2.2、在菜单数据的icon中添加Element Plus官网的图标名称

```
icon: "Menu"
```

2.3、修改MenuItem.vue组件

```
<template>
    <template v-for="menu in menuList" :key="menu.path">
        <el-sub-menu v-if="menu.children && menu.children.length > 0"
:index="menu.path">
            <template #title>
                <i v-if="menu.meta.icon && menu.meta.icon.includes('el-icon')"
:class="menu.meta.icon"></i>
                <!-- 动态组件的使用方法 -->
                <!-- <component class="icons" v-else :is="menu.meta.icon" /> -->
                <Icon class="icons" v-else :icon='menu.meta.icon'></Icon>
                <span>{{ menu.meta.title }}</span>
            </template>
            <menu-item :menuList='menu.children'></menu-item>
        </el-sub-menu>
        <el-menu-item  style="color: #f4f4f5" v-else :index="menu.path">
            <i v-if="menu.meta.icon && menu.meta.icon.includes('el-icon')"
:class="menu.meta.icon"></i>
                <!-- 动态组件的使用方法 -->
                <!-- <component class="icons" v-else :is="menu.meta.icon" /> -->
                <Icon class="icons" v-else :icon='menu.meta.icon'></Icon>
            <template #title>{{ menu.meta.title }}</template>
        </el-menu-item>
    </template>
</template>
<script setup lang="ts">
defineProps(['menuList'])
</script>
<style scoped>
.icons{
    width: 24px;
    height: 18px;
    margin-right: 5px;
}
</style>
```

第12讲 路由配置与页面创建

vue3 setup语法糖文档地址

```
1、https://v3.cn.vuejs.org/api/sfc-script-setup.html
2、https://github.com/vuejs/rfcs/tree/master/active-rfcs
3、https://next.router.vuejs.org/
```

vue3基础代码模板快速生成配置

```
1、首先在vscode编辑器中打开，【文件】->【首选项】->【用户片段】->【新代码片段】-> 取名
vue3ts.json -> 回车

2、把下面代码贴进去,其中prefix里面的内容就是你自己的快捷键

{
    "Print to console": {
        "prefix": "vb",
        "body": [
            "<template>",
            "  <div></div>",
            "</template>",
            "<script setup lang='ts'>",
            "import { ref,reactive} from 'vue'",
            "</script>",
            "<style scoped lang='scss'>",
            "</style>",
        ],
        "description": "Log output to console"
    }
}
3、新建.vue结尾的文件， 代码区域输入: vb回车 即可生成定义的模板代码
```

功能分析

```
点击左侧菜单，能够在内容展示区展示对应页面
```

前置知识

```
由于我们在 setup 里面没有访问 this，所以我们不能再直接访问 this.$router 或 this.$route；
我们使用 useRouter 和 useRoute 替代；
const router = useRouter()  ->   this.$router
const route = useRoute()    ->   this.$route
```

1、在router/index.js添加如下路由

```
import { createRouter, createWebHistory, RouteRecordRaw } from 'vue-router'
import Layout from '@/layout/Index.vue'
const routes: Array<RouteRecordRaw> = [
    {
        path: '/',
        component: Layout,
        redirect: '/dashboard',
        children: [
          {
            path: '/dashboard',
            component: () => import('@/layout/dashboard/Index.vue'),
            name: 'dashboard',
            meta: {
              title: '首页',
              icon: '#icondashboard'
            }
          }
        ]
      },
    {
        path: "/system",
        component: Layout,
        name: "system",
        meta: {
            title: "系统管理",
            icon: "el-icon-menu",
            roles: ["sys:manage"],
            parentId: 0,
        },
        children: [
            {
                path: "/department",
                component: () =>
import('@/views/system/department/department.vue'),
                name: "department",
                meta: {
                    title: "机构管理",
                    icon: "el-icon-document",
                    roles: ["sys:dept"]
                },
            },
            {
                path: "/userList",
                component: () => import('@/views/system/User/UserList.vue'),
                name: "userList",
                meta: {
                    title: "用户管理",
                    icon: "el-icon-s-custom",
                    roles: ["sys:user"]
                },
            },
            {
                path: "/roleList",
                component: () => import('@/views/system/Role/RoleList.vue'),
                name: "roleList",
                meta: {
                    title: "角色管理",
                    icon: "el-icon-s-tools",
                    roles: ["sys:role"]
                },
            },
            {
                path: "/menuList",
                component: () => import('@/views/system/Menu/MenuList.vue'),
                name: "menuList",
                meta: {
                    title: "权限管理",
                    icon: "el-icon-document",
                    roles: ["sys:menu"]
                },
            },
        ],
    },
    {
        path: "/goods",
        component: Layout,
        name: "goods",
        meta: {
            title: "商品管理",
            icon: "el-icon-document",
            roles: ["sys:goods"]
        },
        children: [
            {
                path: "/goodCategory",
                component: () =>
import('@/views/goods/goodsCategory/goodsCategoryList.vue'),
                name: "goodCategory",
                meta: {
                    title: "商品分类",
                    icon: "el-icon-document",
                    roles: ["sys:goodsCategory"]
                },
            },
        ],
    },
    {
        path: "/systenConfig",
        component: Layout,
        name: "systenConfig",
        meta: {
            title: "系统工具",
            icon: "el-icon-document",
            roles: ["sys:systenConfig"]
        },
        children: [
            {
                path: "/document",
                component: () =>
import('@/views/system/config/systemDocument.vue'),
                name: "http://42.193.158.170:8089/swagger-ui/index.html",
                meta: {
                    title: "接口文档",
                    icon: "el-icon-document",
                    roles: ["sys:document"]
                },
            },
        ],
    }
]
//创建
const router = createRouter({
    history: createWebHistory(),
    routes
})
export default router
```

2、创建路由对应的页面

3、在MenuBar.vue组件的el-menu添加 router属性

```
router: 是否启用 vue-router 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
```

4、src/layout/Index.vue的添加路由

```
<el-main class="main">
    <router-view></router-view>
</el-main>
```

5、MenuBar.vue设置当前激活的菜单 default-active

在MenuBar.vue添加如下代码

```
import { ref, reactive,computed } from 'vue'
import { useRoute } from 'vue-router';
const route = useRoute();
//获取激活的菜单
const activeIndex = computed(()=>{
    const {path} = route;
    return path;
})
```

完整源码

```
<template>
    <MenuLogo></MenuLogo>
    <el-menu
        :default-active="activeIndex"
        class="el-menu-vertical-demo"
        :collapse="isCollapse"
        @open="handleOpen"
        @close="handleClose"
        background-color="#304156"
        unique-opened
        router
    >
        <MenuItem :menuList="menuList"></MenuItem>
    </el-menu>
</template>
<script setup lang="ts">
import { ref, reactive,computed } from 'vue'
import { useRoute } from 'vue-router';
import MenuItem from './MenuItem.vue'
import MenuLogo from '@/layout/menu/MenuLogo.vue'
// setup语法糖中 定义的数据和方法，直接可以在模板中使用，无需要 return
const route = useRoute();
//获取激活的菜单
const activeIndex = computed(()=>{
    const {path} = route;
    return path;
})
```

```
//菜单数据
let menuList = reactive([
    {
        path: '/dashboard',
        component: "Layout",
        meta: {
            title: "首页",
            icon: "el-icon-s-home",
            roles: ["sys:manage"]
        },
        children: []
    },
    {
        path: "/system",
        component: "Layout",
        alwaysShow: true,
        name: "system",
        meta: {
            title: "系统管理",
            icon: "el-icon-menu",
            roles: ["sys:manage"],
            parentId: 0,
        },
        children: [
            {
                path: "/department",
                component: "/system/department/department",
                alwaysShow: false,
                name: "department",
                meta: {
                    title: "机构管理",
                    icon: "el-icon-document",
                    roles: ["sys:dept"],
                    parentId: 17,
                },
            },
            {
                path: "/userList",
                component: "/system/User/UserList",
                alwaysShow: false,
                name: "userList",
                meta: {
                    title: "用户管理",
                    icon: "el-icon-s-custom",
                    roles: ["sys:user"],
                    parentId: 17,
                },
            },
            {
                path: "/roleList",
                component: "/system/Role/RoleList",
                alwaysShow: false,
                name: "roleList",
                meta: {
                    title: "角色管理",
                    icon: "el-icon-s-tools",
                    roles: ["sys:role"],
                    parentId: 17,
                },
            },
            {
                path: "/menuList",
                component: "/system/Menu/MenuList",
                alwaysShow: false,
                name: "menuList",
                meta: {
                    title: "权限管理",
                    icon: "el-icon-document",
                    roles: ["sys:menu"],
                    parentId: 17,
                },
            },
        ],
    },
    {
        path: "/goods",
        component: "Layout",
        alwaysShow: true,
        name: "goods",
        meta: {
            title: "商品管理",
            icon: "el-icon-document",
            roles: ["sys:goods"],
            parentId: 0,
        },
        children: [
            {
                path: "/goodCategory",
                component: "/goods/goodsCategory/goodsCategoryList",
                alwaysShow: false,
                name: "goodCategory",
                meta: {
                    title: "商品分类",
                    icon: "el-icon-document",
                    roles: ["sys:goodsCategory"],
                    parentId: 34,
                },
            },
        ],
    },
    {
        path: "/systenConfig",
        component: "Layout",
        alwaysShow: true,
        name: "systenConfig",
        meta: {
            title: "系统工具",
            icon: "el-icon-document",
            roles: ["sys:systenConfig"],
            parentId: 0,
        },
        children: [
            {
                path: "/document",
                component: "/system/config/systemDocument",
                alwaysShow: false,
                name: "http://42.193.158.170:8089/swagger-ui/index.html",
                meta: {
                    title: "接口文档",
                    icon: "el-icon-document",
                    roles: ["sys:document"],
                    parentId: 42,
                },
            },
        ],
    },
]);
const isCollapse = ref(false)
const handleOpen = (key: string | number, keyPath: string) => {
    console.log(key, keyPath)
}
const handleClose = (key: string | number, keyPath: string) => {
    console.log(key, keyPath)
}
</script>
<style scoped>
.el-menu-vertical-demo:not(.el-menu--collapse) {
    width: 230px;
    min-height: 400px;
}
.el-menu {
    border-right: none;
}

:deep(.el-sub-menu .el-sub-menu__title) {
    color: #f4f4f5 !important;
}

:deep(.el-menu .el-menu-item) {
    color: #bfcbd9;
}
/* 菜单点中文字的颜色 */

:deep(.el-menu-item.is-active) {
    color: #409eff !important;
}
/* 当前打开菜单的所有子菜单颜色 */

:deep(.is-opened .el-menu-item) {
    background-color: #1f2d3d !important;
}
/* 鼠标移动菜单的颜色 */

:deep(.el-menu-item:hover) {
    background-color: #001528 !important;
}
</style>
```

第13讲 菜单收缩与展开制作

前置知识

```
1、element plus图标使用
https://element-plus.org/zh-CN/component/icon.html

2、在 <script setup>中，动态组件的使用方式
https://v3.cn.vuejs.org/api/sfc-script-
setup.html#%E4%BD%BF%E7%94%A8%E7%BB%84%E4%BB%B6

3、在 <script setup>中组件使用方式： 引入直接使用，无需注册

4、Vuex4的使用
https://next.vuex.vuejs.org/
```

1、切换按钮的实现

1.1、安装element plus图标

```
1、安装
npm install @element-plus/icons

2、引入图标
import { Fold } from '@element-plus/icons'

3、使用方式
<el-icon>
    <Fold />
</el-icon>
```

1.2、在src/layout/header下新建 Collapse.vue组件

```
<template>
  <el-icon @click="changeIcon" class="icons">
   <component :is="status ? Fold : Expand" />
  </el-icon>
</template>
<script setup lang='ts'>
import { Fold,Expand } from '@element-plus/icons'
import { ref } from 'vue';
const status = ref(true);
//搜索按钮切换事件
const changeIcon = ()=>{
  status.value = !status.value;
}
</script>
<style scoped lang='scss'>
.icons{
  display: flex;
  align-items: center;
  font-size: 22px;
  color: #303133;
  cursor: pointer;
}
</style>
```

1.3、在Header.vue组件中引入 Collapse.vue组件，并使用

```
<template>
<div>
    <Collapse></Collapse>
</div>
</template>
<script setup lang="ts">
import Collapse from './Collapse.vue';
</script>
<style scoped>

</style>
```

2、左侧菜单切换实现

2.1、修改store/index.ts

在store/index.ts中State添加collapse属性；mutations添加setCollopse()方法；getters添加getCollapse()方法，如下所示

```
// store.ts
import { InjectionKey } from 'vue'
import { createStore, useStore as baseUseStore, Store } from 'vuex'

export interface State {
    count: number,
    collapse:boolean
}

export const key: InjectionKey<Store<State>> = Symbol()

export const store = createStore<State>({
    state: {
        count: 0,
        collapse:false
    },
    mutations: {
        setCount:(state: State, count: number)=> {
            state.count = count;
        },
        setCollopse:(state: State, collapse: boolean)=>{
            state.collapse = collapse;
        }
    },
    getters: {
        getCount:(state: State) =>{
            return state.count;
        },
        getCollapse:(state:State)=>{
            return state.collapse
        }
    }
})
```

```
// 定义自己的 `useStore` 组合式函数
export function useStore() {
    return baseUseStore(key)
}
```

2.2、修改Collapse.vue组件图标切换事件

changeIcon()添加 store.commit('setCollopse',status.value) 完整源码如下所示

```
<template>
  <el-icon @click="changeIcon" class="icons">
   <component :is="status ? Fold : Expand" />
  </el-icon>
</template>
<script setup lang='ts'>
import { useStore } from '@/store';
import { Fold,Expand } from '@element-plus/icons'
import { ref } from 'vue';
const store = useStore();
const status = ref(true);
//搜索按钮切换事件
const changeIcon = ()=>{
  status.value = !status.value;
  store.commit('setCollopse',status.value)
}
</script>
<style scoped lang='scss'>
.icons{
  display: flex;
  align-items: center;
  font-size: 22px;
  color: #303133;
}
</style>
```

2.3、MenuBar.vue组件添加 :collapse="isCollapse"

从vuex获取 isCollapse值, 给 MenuLogo 和 el-menu 添加 isCollapse

```
<template>
    <MenuLogo v-if="!isCollapse"></MenuLogo>
    <el-menu
        :default-active="activeIndex"
        class="el-menu-vertical-demo"
        :collapse="isCollapse"
        @open="handleOpen"
        @close="handleClose"
        background-color="#304156"
        unique-opened
        router
    >
        <MenuItem :menuList="menuList"></MenuItem>
    </el-menu>
</template>
<script setup lang="ts">
import {  reactive,computed } from 'vue'
import { useRoute } from 'vue-router';
import { useStore } from '@/store';
import MenuItem from './MenuItem.vue'
import MenuLogo from '@/layout/menu/MenuLogo.vue'
// setup语法糖中 定义的数据和方法，直接可以在模板中使用，无需要 return
const route = useRoute();
const store = useStore();
//获取菜单收缩状态
const isCollapse = computed(()=>{
    return store.getters['getCollapse']
})
//获取激活的菜单
const activeIndex = computed(()=>{
    const {path} = route;
    return path;
})
//菜单数据
let menuList = reactive([
    {
        path: '/dashboard',
        component: "Layout",
        meta: {
            title: "首页",
            icon: "el-icon-s-home",
            roles: ["sys:manage"]
        },
        children: []
    },
    {
        path: "/system",
        component: "Layout",
        alwaysShow: true,
        name: "system",
        meta: {
            title: "系统管理",
            icon: "el-icon-menu",
            roles: ["sys:manage"],
            parentId: 0,
        },
        children: [
            {
                path: "/department",
                component: "/system/department/department",
                alwaysShow: false,
                name: "department",
                meta: {
                    title: "机构管理",
                    icon: "el-icon-document",
                    roles: ["sys:dept"],
                    parentId: 17,
                },
            },
            {
                path: "/userList",
                component: "/system/User/UserList",
                alwaysShow: false,
                name: "userList",
                meta: {
                    title: "用户管理",
                    icon: "el-icon-s-custom",
                    roles: ["sys:user"],
                    parentId: 17,
                },
            },
            {
                path: "/roleList",
                component: "/system/Role/RoleList",
                alwaysShow: false,
                name: "roleList",
                meta: {
                    title: "角色管理",
                    icon: "el-icon-s-tools",
                    roles: ["sys:role"],
                    parentId: 17,
                },
            },
            {
                path: "/menuList",
                component: "/system/Menu/MenuList",
                alwaysShow: false,
                name: "menuList",
                meta: {
                    title: "权限管理",
                    icon: "el-icon-document",
                    roles: ["sys:menu"],
                    parentId: 17,
                },
            },
        ],
    },
    {
        path: "/goods",
        component: "Layout",
        alwaysShow: true,
        name: "goods",
        meta: {
            title: "商品管理",
            icon: "el-icon-document",
            roles: ["sys:goods"],
            parentId: 0,
        },
        children: [
            {
                path: "/goodCategory",
                component: "/goods/goodsCategory/goodsCategoryList",
                alwaysShow: false,
                name: "goodCategory",
                meta: {
                    title: "商品分类",
                    icon: "el-icon-document",
                    roles: ["sys:goodsCategory"],
                    parentId: 34,
                },
            },
        ],
    },
    {
        path: "/systenConfig",
        component: "Layout",
        alwaysShow: true,
        name: "systenConfig",
        meta: {
            title: "系统工具",
            icon: "el-icon-document",
            roles: ["sys:systenConfig"],
            parentId: 0,
        },
        children: [
            {
                path: "/document",
                component: "/system/config/systemDocument",
                alwaysShow: false,
                name: "http://42.193.158.170:8089/swagger-ui/index.html",
                meta: {
                    title: "接口文档",
                    icon: "el-icon-document",
                    roles: ["sys:document"],
                    parentId: 42,
                },
            },
        ],
    },
]);
// const isCollapse = ref(false)
const handleOpen = (key: string | number, keyPath: string) => {
    console.log(key, keyPath)
}
const handleClose = (key: string | number, keyPath: string) => {
    console.log(key, keyPath)
}
</script>
<style scoped>
.el-menu-vertical-demo:not(.el-menu--collapse) {
    width: 230px;
    min-height: 400px;
}
.el-menu {
    border-right: none;
}

:deep(.el-sub-menu .el-sub-menu__title) {
    color: #f4f4f5 !important;
}

:deep(.el-menu .el-menu-item) {
    color: #bfcbd9;
}
/* 菜单点中文字的颜色 */

:deep(.el-menu-item.is-active) {
    color: #409eff !important;
}
/* 当前打开菜单的所有子菜单颜色 */

:deep(.is-opened .el-menu-item) {
    background-color: #1f2d3d !important;
}
/* 鼠标移动菜单的颜色 */

:deep(.el-menu-item:hover) {
    background-color: #001528 !important;
}
</style>
```

2.3、修改Collapse.vue组件

从vuex获取status状态

```
<template>
  <el-icon @click="changeIcon" class="icons">
   <component :is="status ? Fold : Expand" />
  </el-icon>
</template>
<script setup lang='ts'>
import { useStore } from '@/store';
import { Fold,Expand } from '@element-plus/icons'
import { ref,computed } from 'vue';
const store = useStore();
// const status = ref(true);
const status = computed(()=>{
  return store.getters['getCollapse']
})
//搜索按钮切换事件
const changeIcon = ()=>{
  console.log(status.value)
  // status.value = !status.value;
  store.commit('setCollopse',!status.value)
}
</script>
<style scoped lang='scss'>
.icons{
  display: flex;
  align-items: center;
  font-size: 22px;
  color: #303133;
  cursor: pointer;
}
</style>
```

第14讲 面包屑导航制作

前置知识

```
1、在 <script setup>中组件使用方式:  引入直接使用，无需注册

2、ref响应式数据的定义、使用

3、watch的使用
```

1、给MenuBar.vue组件的MenuLogo添加动画

```
@keyframes logoAnimation {
    0% {
        transform: scale(0);
    }
    50% {
        transform: scale(1);
    }
    100% {
        transform: scale(1);
    }
}
.layout-logo {
    animation: logoAnimation 1s ease-out;
}
```

2、在layout/header下新建BredCum.vue组件

```
<template>
    <el-breadcrumb separator="/">
        <el-breadcrumb-item v-for="item in tabs">{{ item.meta.title }}</el-
breadcrumb-item>
    </el-breadcrumb>
</template>
<script setup lang='ts'>
import { ref, watch, Ref } from 'vue';
import { useRoute, RouteLocationMatched } from 'vue-router';
//面包屑数据
const tabs: Ref<RouteLocationMatched[]> = ref([]);
//获取当前路由
const route = useRoute();
const getBredcrumb = () => {
    //从路由里面获取所有有meta和title
    let mached = route.matched.filter(item => item.meta && item.meta.title);
    //判断第一个是否是首页,不是，构造一个
    const first = mached[0];
    if (first.path !== '/dashboard') {
        mached = [{ path: '/dashboard', meta: { title: '首页' } } as
any].concat(mached);
    }
    tabs.value = mached;
}
getBredcrumb();
watch(() => route.path, () => getBredcrumb())
</script>
<style scoped lang='scss'>
</style>
```

3、在Header.vue组件使用BredCum.vue组件

```
<template>
 <Collapse></Collapse>
 <BredCum></BredCum>
</template>
<script setup lang="ts">
import Collapse from './Collapse.vue';
import BredCum from './BredCum.vue';
</script>
```

第15讲 tabs选项卡制作

前置知识

```
1、vuex在组合API中的使用;
    const store = useStore();

2、vue-router在组合API中的使用;
    const route = useRoute();
    const router = useRouter();

3、响应式数据的定义;  ref 、reactive

4、watch、computed的使用;

5、element plus组件Tabs标签的使用;

6、TypeScript中接口 interface的使用;
    接口是一种规范
```

1、功能分析:

```
1、点击左侧菜单，右侧内容展示区显示对应的选项卡;
2、点击右侧选项卡，左侧对应菜单也要相应的选中;
3、解决刷新后，Tabs数据丢失的问题;   window.addEventListener("beforeunload")
```

2、在src/layout目录下新建tabs目录，然后新建Tabs.vue组件

```
<template>
  <el-tabs v-model="activeTab" type="card" @tab-click="tabClick" closable @tab-
remove="removeTab">
    <el-tab-pane
      v-for="item in tabsList"
      :key="item.path"
      :label="item.title"
      :name="item.path"
    ></el-tab-pane>
  </el-tabs>
```

```
</template>
<script setup lang='ts'>
import { computed, watch, onMounted, ref } from 'vue'
import { useRoute,useRouter } from 'vue-router';
import { useStore } from '@/store';
import { ITab } from '@/store/type/Index';
const route = useRoute();
const router = useRouter();
const store = useStore();
//激活的选项卡
const activeTab = ref('');
//获取选项卡数据
const tabsList = computed(() => {
  return store.getters['getTabsList']
})
//添加选项卡
const addTabe = () => {
  const { path, meta } = route;
  const tab: ITab = {
    title: meta.title as string,
    path: path
  }
  store.commit('addTabs', tab);
}
//设置激活的选项卡
const setActive = () => {
  activeTab.value = route.path;
}
//选项卡点击事件
const  tabClick =(tab:any)=> {
  const {props} = tab;
  console.log(props)
  router.push({path:props.name})
}
//关闭选项卡
const removeTab = (targetName: string) => {
  if (targetName === '/dashboard') return;
  const tabs = tabsList.value;
  let activeName = activeTab.value;
  if (activeName === targetName) {
    console.log(activeName)
    console.log(targetName)
    tabs.forEach((tab: ITab, index: number) => {
      if (tab.path === targetName) {
        const nextTab = tabs[index + 1] || tabs[index - 1]
        console.log(nextTab)
        if (nextTab) {
          activeName = nextTab.path
        }
      }
    })
  }
  activeTab.value = activeName
  store.state.tabsList = tabs.filter((tab: ITab) => tab.path !== targetName);
  router.push({path:activeName})
}
//解决刷新数据丢失问题
const beforeUnload = () => {
  window.addEventListener("beforeunload", () => {
    console.log('刷新了')
    sessionStorage.setItem("tabViews", JSON.stringify(tabsList.value));
  })
  let tabSession = sessionStorage.getItem("tabViews");
  if (tabSession) {
    let oldViews = JSON.parse(tabSession);
    if (oldViews.length > 0) {
      store.state.tabsList = oldViews;
    }
  }
}
onMounted(() => {
  beforeUnload();
  setActive();
  addTabe();
})
watch(() => route.path, () => {
  setActive();
  addTabe();
})
</script>
<style scoped lang='scss'>
:deep(.el-tabs__header) {
  margin: 0px;
}
:deep(.el-tabs__item) {
  height: 26px !important;
  line-height: 26px!important;
  text-align: center!important;
  border: 1px solid #d8dce5 !important;
  margin: 0px 3px!important;
  color: #495060;
  font-size: 12px!important;
  padding: 0xp 10px!important;
}
:deep(.el-tabs__nav) {
  border: none !important;
}
:deep(.is-active) {
  border-bottom: 1px solid transparent !important;
  border: 1px solid #42b983 !important;
  background-color: #42b983 !important;
  color: #fff !important;
}
:deep(.el-tabs__item:hover) {
  color: #495060 !important;
}
:deep(.is-active:hover) {
  color: #fff !important;
}
</style>
```

3、在src/store下新建type目录，并建Inex.ts

```
export interface ITab {
    title: string,
    path: string
}
```

4、在src/store下的index.ts添加选项卡相关的操作

```
// store.ts
import { InjectionKey } from 'vue'
import { createStore, useStore as baseUseStore, Store } from 'vuex'
import { ITab } from '@/store/type/Index'
export interface State {
    count: number,
    collapse: boolean,
    tabsList: Array<ITab>
}

export const key: InjectionKey<Store<State>> = Symbol()

export const store = createStore<State>({
    state: {
        count: 0,
        collapse: false,
        tabsList: []
    },
    mutations: {
        setCount(state: State, count: number) {
            state.count = count;
        },
        //设置collapse
        setCollapse: (state: State, collapse: boolean) => {
            state.collapse = collapse;
        },
        //添加选项卡
        addTabs: (state: State, tab: ITab) => {
            console.log(tab)
            if (state.tabsList.some(item => item.path === tab.path)) return;
            state.tabsList.push(tab);
            console.log(state.tabsList)
        }
    },
    getters: {
        getCount(state: State) {
            return state.count;
        },
        //获取collapse
        getCollapse: (state: State) => {
            return state.collapse;
        },
        getTabsList:(state:State)=>{
            return state.tabsList
        }
    }
})
```

```
// 定义自己的 `useStore` 组合式函数
export function useStore() {
    return baseUseStore(key)
}
```

5、在src/layout的Index.vue使用Tabs.vue选项卡组件

```
import Tabs from './tabs/Tabs.vue';


<el-main class="main">
    <Tabs></Tabs>
    <router-view></router-view>
</el-main>
```

第16讲 Tabs选项卡制作总结

1、实现原理

```
1、点击菜单，显示对应的选项卡；
    watch监听路由path的变化，把当前路由的title 和  path 放到 Tabs选项卡对应的数据里面；

2、选项卡的激活设置：
    把当前激活的选项卡v-model绑定项设为当前路由的path

3、点击选项卡，左侧对应菜单激活；
    点击选项卡，跳转到对应的路由；只需要把选项卡的v-model绑定项设为当前路由的path，左侧菜单便
可自动激活；

4、关闭选项卡
    首页不能关闭；关闭时，删除当前选项卡；重新设置vuex里面的选项卡数据；并跳转到新的路由；

5、刷新浏览器时，选项卡数据丢失
    window.addEventListener('beforeunload')
```

2、TypeScript知识复习

```
1、interface和type的基本使用；
2、typeof 和 keyof的使用；
3、泛型、泛型约束的使用；
4、TypeScript模板字符串的基本使用；
5、TypeScript中交叉类型、联合类型的基本使用；
6、Omit、Pick、Parameters、ReturnType的基本使用；
```

第17讲 TypeScript知识补充

1、接口 interface

1.1、什么是接口

```
接口是一系列抽象方法的声明，是一些方法特征的集合，这些方法都应该是抽象的，需要由具体的类去实现，
然后第三方就可以通过这组抽象方法调用，让具体的类执行具体的方法；
```

1.2、接口的作用

```
1、TypeScript的核心原则之一是对值所具有的结构进行类型检查。 在TypeScript里，接口的作用就是为
这些类型命名和为你的代码或第三方代码定义契约。

2、接口是一种规范或约束，通常使用接口（Interfaces）来定义对象的类型；
```

```
//接口是一种规范，通常用于定义对象的类型

//1、定义一个IPerson接口
interface IPerson{
  name:string,
  age:number
}
//2、定义一个tom变量，它的类型是IPerson类型的
//约束tom的形状必须和IPerson的一致
let tom:IPerson = {
  name:'Tom',
  age:18
}

//3、定义的属性比接口多一些属性，或少一些属性是不允许的
let cat : IPerson = {
  name:'Cat',
}
//Property 'age' is missing in type '{ name: string; }' but required in type
'IPerson'

//4、可选属性  ?   表示该属性不是必须的，可有可无
interface IPerson{
  name:string,
  age?:number
}
```

1.3、类型别名 type

```
类型别名用来给一个类型起个新名字。 类型别名type和接口interface都可以描述一个对象或函数；不同点
是类型别名type还可以声明基本类型别名，联合类型，元组等类型

一般来说，如果不清楚什么时候用interface/type，能用 interface 实现，就用 interface ， 如果
不能就用 type
```

```
type User = {
  name:string,
  age:number
}
//定义User类型的变量toms
let toms:User = {
  name:'Toms',
  age:20,
}
console.log(toms.name)
console.log(toms.age)
```

2、typeof和keyof的使用

2.1、typeof

```
在 TypeScript 中， typeof 操作符可以用来获取一个变量或对象的类型
```

```
type User = {
  name:string,
  age:number
}
let toms:User = {
  name:'Toms',
  age:20,
}
console.log(toms.name)
console.log(toms.age)

//typeof使用
//通过typeof获取变量toms的类型，赋值给Toms
type Toms = typeof toms;
//使用Toms
let tm : Toms = {
  name:'Tm',
  age:18
}

//获取对象类型
const infos = {
  name:'张三',
  age:18,
  adress:{
    city:'北京'
  }
}
type infoType = typeof infos;
```

2.2、keyof

```
TypeScript 通过 keyof 操作符提取其属性的名称
```

3、泛型和泛型约束

3.1、泛型

```
1、什么是泛型： 我们可以理解为泛型就是在编译期间不确定方法的类型(广泛之意思)，在方法调用时，由程序员指定泛型具体指向什么类型；

2、泛型作用： 通常解决类，接口，方法的复用性，以及对非特定数据类型的支持；
```

```

//泛型的使用
function getMsg<T>(msg:T):T{
  return msg;
}
let msg = getMsg<string>('新增成功');
console.log(msg)

//定义一个对象
const userInfo = {
  name:'张三',
  age:18
}
//获取userInfo对象里面属性的值
//'string' can't be used to index type '{}'.
// function getVal(o:object,name:string){
//   return o[name]
// }
function getVal(o:any,name:string){
  return o[name]
}

const name = getVal(userInfo,'name')
console.log(name)
const age = getVal(userInfo,'age')
console.log(age)

//泛型约束
//上面也可以获取数据，但是获取到的值也变为 any了
// 通过 K extends keyof T 确保参数 key 一定是对象中含有的键
function getProperty<T,K extends keyof T>(obj:T,key:K){
  return obj[key]
}
```

3.2、泛型条件

```
T extends U? X: Y
```

```
//
type Fas<T> = T extends {name: infer Test} ? Test : unknown;
type FasTest = Fas<{name:string}>
```

4、模板字符串

```
//模板字符串
type world = "World";
type Greeting = `Hello ${world}`;
```

5、交叉类型 &

```
1、交叉类型:把几个类型的成员合并，形成一个拥有这几个类型所有成员的新类型，可以理解为合并

2、注意：交叉类型不是取交集，是取并

3、基本类型是不会存在交叉的,一个类型不肯能既是string又是number类型
```

```

//交叉类型:把几个类型的成员合并，形成一个拥有这几个类型所有成员的新类型
//注意：交叉类型不是取交集，是取并
//基本类型是不会存在交叉的,一个类型不肯能既是string又是number类型
type t1 = string & number;

interface Iname{
  name:string
}
interface Iage{
  age:number
}
type users = Iname & Iage;

type ss = keyof users

let d:users = {
  name:'张三',
  age:18
}

//列表查询
interface IPage{
  pageSize:number;
  currentPage:number;
}
interface IParms {
  searchKey:string;
}
type Tparms = IPage & IParms;

let parms:Tparms = {
  pageSize:10,
  currentPage:1,
  searchKey:''
}
```

6、联合类型 |

```
联合类型： 表示取值可以为多种类型中的一种
```

```
//联合类型 ： 取值是多种类型中的一种
function add(parm1:string | number,parm2:string | number){
  if(typeof parm1 === 'string' || typeof parm2 === 'string'){
    return `${parm1}${parm2}`
  }
  return parm1 + parm2
}
let adds = add('Hello','World');
console.log(adds)
let adds1 = add(1,1)
console.log(adds1)
```

7、Omit使用

```
Omit<K,T> ： 用于从指定类型中，排除指定的属性，返回一个新的类型
K：是对象类型名称，T：是剔除K类型中的属性名称
```

```
// Omit<K,T> ：  用于从指定类型中，排除指定的属性，返回一个新的类型
//K：是对象类型名称，T：是剔除K类型中的属性名称
type OmitUser = {
  name:string;
  age:number;
  sex:string;
}

type NewOmitUser = Omit<OmitUser,'sex'>
//等价于
type OmitUser1 = {
  name:string;
  age:number;
}
```

8、Pick使用

```
Pick ： 从一个类型中，取出几个想要的属性
```

```
//Pick使用：从一个类型中取出想要的属性

interface ITest{
  type:string,
  text:string
}
type TTest = Pick<ITest,'text'>;
```

第18讲 Vuex4 + TypeScript实现类型推测方案一

1、创建项目

```
参照第02讲
```

2、安装vuex

官网地址

```
https://next.vuex.vuejs.org/zh/index.html
```

```
npm install vuex@next --save
```

3、在src下新建store目录，然后新建index.ts

```
import { InjectionKey } from 'vue'
import { createStore, useStore as baseUseStore, Store } from 'vuex'

export interface State {
  count: number
}

export const key: InjectionKey<Store<State>> = Symbol()

export const store = createStore<State>({
  state: {
    count: 0
  },
  mutations: {
    addCount: (state: State, count: number) => {
      state.count = count
    }
  },
  getters: {
    getCount: (state: State) => {
      return state.count
    }
  }
})

// 定义自己的 `useStore` 组合式函数
export function useStore() {
  return baseUseStore(key)
}
```

4、HelloWorld.vue页面确定vuex是否能正常使用

```
此处注意： useStore 的引入是引入我们自己的 index.ts里面的useStore , 不是vuex给我们提供的;

import {useStore} from '../store/index'
```

```
<script setup lang="ts">
import { computed, ref } from 'vue'

import {useStore} from '../store/index'
const newcount = computed(()=>{
  return store.getters['getCount']
})
const count = ref(0);
const store = useStore();
const addBtn = ()=>{
  store.commit('addCount',++count.value)
}
</script>

<template>
  <h3>{{newcount}}</h3>
  <button @click="addBtn">新增</button>
</template>

<style scoped>
a {
  color: #42b983;
}

label {
  margin: 0 0.5em;
  font-weight: bold;
}

code {
  background-color: #eee;
  padding: 2px 4px;
  border-radius: 4px;
  color: #304455;
}
</style>
```

第19讲 Vuex4 + TypeScript实现类型推测方案二

第20讲 axios的安装使用一

1、安装axios 和 qs

```
npm install axios
npm install --save @types/qs
npm i qs -S
```

2、新建request.ts

```
import axios, { AxiosInstance, AxiosRequestConfig, AxiosPromise, AxiosResponse }
from "axios";
import { ElMessage } from "element-plus";
import { getToken } from "@/utils/auth";
import qs from 'qs'
export interface Result<T = any> {
    code: number;
    msg: string;
    data: T;
}
enum StatusCode{
    NoAuth = 600, //token失效
    Success = 200 //返回成功
}
class request {
    private instance: AxiosInstance;
    constructor(config: AxiosRequestConfig) {
        //创建axios实例
        this.instance = axios.create(config)
        this.interceptors()
    }
    //axios拦截器
    private interceptors() {
        //请求发送之前拦截
        this.instance.interceptors.request.use((config: AxiosRequestConfig) => {
            let token = getToken() || '';
            if (token) {
                //把token添加到请求头部
                config.headers = {
                    token: token
                };
            }
            //这里可以设置头部token携带到后端进行token的验证
            return config
        }, (error) => {
            // 错误抛到业务代码
            error.data = {}
            error.data.msg = '服务器异常，请联系管理员！'
            return error
        })

        /**
         * 请求返回之后拦截
         * res的类型是AxiosResponse<any>
         */
        this.instance.interceptors.response.use((res: AxiosResponse) => {
            if (res && res.data) {
                const data = res.data as any;
                if (data.code == StatusCode.NoAuth) {
                    //清空sessionStorage数据
                    sessionStorage.clear();
                    //跳到登录
                    window.location.href = "/login";
                    // return res;
                } else if (data.code == StatusCode.Success ||
res.config.responseType === "arraybuffer") {
                    // return Promise.resolve(res);
                    return res;
                } else { // 这里我们在服务端将正确返回的状态码标为200
                    console.log('88889999')
                    ElMessage.error(data.msg || '服务器出错!')
                    return res || null // 返回数据
                }
            }

        }, (error) => { // 这里是遇到报错的回调
            console.log('进入错误')
            error.data = {};
            if (error && error.response) {
                switch (error.response.status) {
                    case 400:
                        error.data.msg = '错误请求';
                        ElMessage.error(error.data.msg)
                        break
                    case 401:
                        error.data.msg = '未授权，请重新登录';
                        ElMessage.error(error.data.msg)
                        break
                    case 403:
                        error.data.msg = '拒绝访问';
                        ElMessage.error(error.data.msg)
                        break
                    case 404:
                        error.data.msg = '请求错误,未找到该资源';
                        ElMessage.error(error.data.msg)
                        break
                    case 405:
                        error.data.msg = '请求方法未允许';
                        ElMessage.error(error.data.msg)
                        break
                    case 408:
                        error.data.msg = '请求超时';
                        ElMessage.error(error.data.msg)
                        break
                    case 500:
                        error.data.msg = '服务器端出错';
                        ElMessage.error(error.data.msg)
                        break
                    case 501:
                        error.data.msg = '网络未实现';
                        ElMessage.error(error.data.msg)
                        break
                    case 502:
                        error.data.msg = '网络错误';
                        ElMessage.error(error.data.msg)
                        break
                    case 503:
                        error.data.msg = '服务不可用';
                        ElMessage.error(error.data.msg)
                        break
                    case 504:
                        error.data.msg = '网络超时';
                        ElMessage.error(error.data.msg)
                        break
                    case 505:
                        error.data.msg = 'http版本不支持该请求';
                        ElMessage.error(error.data.msg)
                        break
                    default:
                        error.data.msg = `连接错误${error.response.status}`;
                        ElMessage.error(error.data.msg)
                }
            } else {
                error.data.msg = "连接到服务器失败";
                ElMessage.error(error.data.msg)
            }
            // return Promise.reject(error)
            return error
        })
    }
}
export default request;
```

第21讲 axios对restful api的支持

1、request.ts

```
import axios, { AxiosInstance, AxiosRequestConfig, AxiosPromise, AxiosResponse }
from "axios";
import { ElMessage } from "element-plus";
import { getToken } from "@/utils/auth";
import qs from 'qs'
export interface Result<T = any> {
    code: number;
    msg: string;
    data: T;
}
enum StatusCode{
    NoAuth = 600, //token失效
    Success = 200 //返回成功
}
class request {
    private instance: AxiosInstance;
    constructor(config: AxiosRequestConfig) {
        //创建axios实例
        this.instance = axios.create(config)
        this.interceptors()
    }
    //axios拦截器
    private interceptors() {
        //请求发送之前拦截
        this.instance.interceptors.request.use((config: AxiosRequestConfig) => {
            let token = getToken() || '';
            if (token) {
                //把token添加到请求头部
                config.headers = {
                    token: token
                };
            }
            //这里可以设置头部token携带到后端进行token的验证
            return config
        }, (error) => {
            // 错误抛到业务代码
            error.data = {}
            error.data.msg = '服务器异常，请联系管理员！'
            return error
        })

        /**
         * 请求返回之后拦截
         * res的类型是AxiosResponse<any>
         */
        this.instance.interceptors.response.use((res: AxiosResponse) => {
            if (res && res.data) {
                const data = res.data as any;
                if (data.code == StatusCode.NoAuth) {
                    //清空sessionStorage数据
                    sessionStorage.clear();
                    //跳到登录
                    window.location.href = "/login";
                    // return res;
                } else if (data.code == StatusCode.Success ||
res.config.responseType === "arraybuffer") {
                    // return Promise.resolve(res);
                    return res;
                } else { // 这里我们在服务端将正确返回的状态码标为200
                    console.log('88889999')
                    ElMessage.error(data.msg || '服务器出错!')
                    return res || null // 返回数据
                }
            }

        }, (error) => { // 这里是遇到报错的回调
```

```
            console.log('进入错误')
            error.data = {};
            if (error && error.response) {
                switch (error.response.status) {
                    case 400:
                        error.data.msg = '错误请求';
                        ElMessage.error(error.data.msg)
                        break
                    case 401:
                        error.data.msg = '未授权，请重新登录';
                        ElMessage.error(error.data.msg)
                        break
                    case 403:
                        error.data.msg = '拒绝访问';
                        ElMessage.error(error.data.msg)
                        break
                    case 404:
                        error.data.msg = '请求错误,未找到该资源';
                        ElMessage.error(error.data.msg)
                        break
                    case 405:
                        error.data.msg = '请求方法未允许';
                        ElMessage.error(error.data.msg)
                        break
                    case 408:
                        error.data.msg = '请求超时';
                        ElMessage.error(error.data.msg)
                        break
                    case 500:
                        error.data.msg = '服务器端出错';
                        ElMessage.error(error.data.msg)
                        break
                    case 501:
                        error.data.msg = '网络未实现';
                        ElMessage.error(error.data.msg)
                        break
                    case 502:
                        error.data.msg = '网络错误';
                        ElMessage.error(error.data.msg)
                        break
                    case 503:
                        error.data.msg = '服务不可用';
                        ElMessage.error(error.data.msg)
                        break
                    case 504:
                        error.data.msg = '网络超时';
                        ElMessage.error(error.data.msg)
                        break
                    case 505:
                        error.data.msg = 'http版本不支持该请求';
                        ElMessage.error(error.data.msg)
                        break
                    default:
                        error.data.msg = `连接错误${error.response.status}`;
                        ElMessage.error(error.data.msg)
                }
            } else {
                error.data.msg = "连接到服务器失败";
                ElMessage.error(error.data.msg)
            }
            // return Promise.reject(error)
            return error
        })
    }
    getParms(parms: any): string {
        let _params = "";
        if (Object.is(parms, undefined || null)) {
            _params = ''
        } else {
            for (const key in parms) {
                if (parms.hasOwnProperty(key) && parms[key]) {
                    _params += `${parms[key]}/`
                }
            }
        }
        //去掉参数最后一位/
        if (_params) _params = _params.substr(0, _params.length - 1);
        return _params;
    }
    /**
     *  http://localhost:8089/api/user/getUserById?userId=10
     * @param url   /api/user/getUserById
     * @param parms  {userId:10}
     * @returns
     */
    get<T = any>(url: string, parms?: any): Promise<Result<T>> {
        return new Promise((resolve, reject) => {
            this.instance.get<T>(url, {
                params: parms,
                paramsSerializer: (parms) => {
                    return qs.stringify(parms)
                }
            }).then((res) => {
                resolve(res.data as any)
            }).catch((error) => {
                reject(error)
            })
        })
    }
    getRestApi<T = any>(url: string, parms?: any): Promise<Result<T>> {
        return new Promise((resolve, reject) => {
            this.instance.get<T>(this.getParms(parms) ?
`${url}/${this.getParms(parms)}` : url)
                .then((res) => {
                    resolve(res.data as any)
                }).catch((error) => {
                    reject(error)
                })
        })
    }

    post<T = any>(url: string, parms: any): Promise<Result<T>> {
        return new Promise((resolve, reject) => {
            this.instance.post<T>(url, parms, {
                transformRequest: [(params) => {
                    return JSON.stringify(params)
                }],
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then((res) => {
                resolve(res.data as any)
            }).catch((error) => {
                reject(error)
            })
        })
    }
    put<T = any>(url: string, parms: any): Promise<Result<T>> {
        return new Promise((resolve, reject) => {
            this.instance.put<T>(url, parms, {
                transformRequest: [(params) => {
                    return JSON.stringify(params)
                }],
                headers: {
                    'Content-Type': 'application/json'
                }
            }).then((res) => {
                resolve(res.data as any)
            }).catch((error) => {
                reject(error)
            })
        })
    }
    delete<T = any>(url: string, parms: any): Promise<Result<T>> {
        return new Promise((resolve, reject) => {
            this.instance.delete<T>(this.getParms(parms) ?
`${url}/${this.getParms(parms)}` : url)
                .then((res) => {
                    resolve(res.data as any)
                }).catch((error) => {
                    reject(error)
                })
        })
    }

    login<T>(url: string, params: any): Promise<Result<T>> {
        return new Promise((resolve, reject) => {
            this.instance.post<T>(url, params, {
                transformRequest: [(params) => {
                    return qs.stringify(params)
                }],
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded'
                }
            }).then((res) => {
                resolve(res as any)
            }).catch((error) => {
                reject(error)
            })
        })
    }
    getImage(url: string) {
        return this.instance.post(url, null, {
            responseType: 'arraybuffer'
        })
    }
}
export default request;
```

2、在http目录下新建http.ts文件

```
import request from "./request"
const http = new request({
    baseURL:'http://42.193.158.170:8098',
    timeout:10000
})
export default http;
```

第22讲 登录页面制作

```
<template>
    <div class="logincontainer">
        <el-form
            class="loginForm"
            :model="loginModel"
            ref="loginForm"
            :rules="rules"
            :inline="false"
        >
            <el-form-item>
                <div class="loginTitle">系统登录</div>
            </el-form-item>
            <el-form-item>
                <el-input placeholder="请输入账户" v-model="loginModel.username">
</el-input>
            </el-form-item>
            <el-form-item>
                <el-input placeholder="请输入密码" v-model="loginModel.password">
</el-input>
            </el-form-item>
            <el-form-item>
                <el-row :gutter="20">
                    <el-col :span="16">
                        <el-input placeholder="请输入验证码" v-
model="loginModel.code"></el-input>
                    </el-col>
                    <el-col :span="8">
                        <el-input placeholder="请输入验证码" v-
model="loginModel.code"></el-input>
                    </el-col>
                </el-row>
            </el-form-item>
            <el-form-item>
                <el-row :gutter="20" style="margin-top: 10px;">
                    <el-col :span="12">
                       <el-button class="mybtn" type="primary" size="default">登
录</el-button>
```

```
                    </el-col>
                    <el-col :span="12">
                        <el-button class="mybtn"  size="default">取消</el-button>
                    </el-col>
                </el-row>
            </el-form-item>
        </el-form>
    </div>
</template>
<script setup lang='ts'>
import { ref, reactive } from 'vue'
const loginModel = reactive({
    username: '',
    password: '',
    code: ''
})
const rules = reactive([])
</script>
<style scoped lang='scss'>
.logincontainer {
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    .loginForm {
        height: 320px;
        width: 400px;
        border-radius: 10px;
        padding: 20px 35px;
        box-shadow: 0 0 25px #cac6c6;
        .loginTitle {
            font-size: 24px;
            font-weight: 600;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .mybtn{
            width: 100%;
        }
    }
}
</style>
```

第23讲 验证码接口对接测试

1、验证码请求接口

```
{baseURL}/api/sysUser/image
```

2、请求方式

```
POST
```

3、responseType

```
responseType: 'arraybuffer'
```

4、在request.ts中新增getImage()方法

```
getImage(url: string) {
        return this.instance.post(url, null, {
            responseType: 'arraybuffer'
        })
}
```

4、新建user的api

在src目录下新建user目录，然后新建user.ts

```
import http from '@/http/http'
enum Api {
    getImg = '/api/sysUser/image'
}
//获取验证码
export async function getImageApi() {
    return await http.getImage(Api.getImg)
}
```

5、在src目录下新建composables/login文件，然后新建useImage.ts文件

```
import { ref,onMounted } from 'vue'
import { getImagApi } from '@/api/user/user';
export default function useImage() {

    //定义图片src
    const imgSrc = ref('');

    // 获取图片
    //btoa 创建一个base64编码的字符串
    const getImage = async () => {
        await getImagApi().then(res => {
            return 'data:image/png;base64,' + btoa(
                new Uint8Array(res.data as any).reduce((data, byte) => data +
String.fromCharCode(byte), '')
            )
        }).then(data => {
            imgSrc.value = data;
        })
    }
    onMounted(()=>{
        getImage()
    })
    return {
        imgSrc,
        getImage
    }
}
```

6、Login.vue页面

```
<template>
    <div class="logincontainer">
        <el-form
            class="loginForm"
            :model="loginModel"
            ref="loginFormRef"
            :rules="rules"
            :inline="false"
        >
            <el-form-item>
                <div class="loginTitle">系统登录</div>
            </el-form-item>
            <el-form-item prop='username'>
                <el-input placeholder="请输入账户" v-model="loginModel.username">
</el-input>
            </el-form-item>
            <el-form-item prop='password'>
                <el-input placeholder="请输入密码" v-model="loginModel.password">
</el-input>
            </el-form-item>
            <el-form-item prop='code'>
                <el-row :gutter="20">
                    <el-col :span="16">
                        <el-input placeholder="请输入验证码" v-
model="loginModel.code"></el-input>
                    </el-col>
                    <el-col :span="8">
                        <!-- <el-input placeholder="请输入验证码" v-
model="loginModel.code"></el-input> -->
                        <img :src='imgSrc' @click="getImage" />
                    </el-col>
                </el-row>
            </el-form-item>
            <el-form-item>
                <el-row :gutter="20" style="margin-top: 10px;">
                    <el-col :span="12">
                       <el-button class="mybtn" type="primary" size="default">登
录</el-button>

                    </el-col>
                    <el-col :span="12">
                        <el-button class="mybtn"  size="default">取消</el-button>
                    </el-col>
                </el-row>
            </el-form-item>
        </el-form>
    </div>
</template>
<script setup lang='ts'>
import {reactive} from 'vue'
import useImage from '@/composables/login/useImage';

//验证码
const {imgSrc,getImage} = useImage();

const loginModel = reactive({
    username:'',
    password:'',
    code:''
})

const rules = reactive([])
</script>
<style scoped lang='scss'>
.logincontainer {
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    .loginForm {
        height: 320px;
        width: 400px;
        border-radius: 10px;
        padding: 20px 35px;
        box-shadow: 0 0 25px #cac6c6;
        .loginTitle {
            font-size: 24px;
            font-weight: 600;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .mybtn{
            width: 100%;
        }
    }
}
</style>
```

第24讲 登录接口对接

1、登录请求接口

```
{baseURL}/api/user/login
```

2、请求方式

```
POST
```

注意事项： 'Content-Type': 'application/x-www-form-urlencoded'

3、请求参数

参数名称	类型	必填	描述
username	string	是	登录账户
password	string	是	登录密码
code	string	是	验证码

请求参数示例

```
{
    username:'admin';
    password:'1234';
    code:'3036'
}
```

4、响应参数说明

参数名称	类型	描述	
code	int	状态码	
token	string	token值	
id	int	用户id	
expireTime	int	token过期时间	

响应示例

```
{
    code:200;
    token:XXXXXXXXXXXXXXXX;
    id:1;
    expireTime:1346546541231
}
```

5、在request.ts添加登录方法

```
login<T = any>(url: string, params: any): Promise<Result<T>> {
        return new Promise((resolve, reject) => {
            this.instance.post<T>(url, params, {
                transformRequest: [(params) => {
                    return qs.stringify(params)
                }],
                headers: {
                    'Content-Type': 'application/x-www-form-urlencoded'
                }
            }).then((res) => {
                resolve(res as any)
            }).catch((error) => {
                reject(error)
            })
        })
    }
```

6、api/user模块，新建UserModel.ts

```
/**
 * 登录请求参数
 */
export interface LoginParm{
    username:string;
    password:string;
    code:string
}
/**
 * 登录返回值
 */
export interface LoginResult{
    code:number;
    token:number;
    id:number;
    expireTime:number
}
```

7、api/user/user.ts新增如下方法

```
//登录
export async function loginApi(params: LoginParm) {
    return await http.login<LoginResult>(Api.login, params)
}
```

8、composables/login目录下新建useBaseLogin.ts 和 useLogin.ts文件

useBaseLogin.ts

```
import {reactive,ref} from 'vue'
import { ElForm } from 'element-plus';
import { LoginParm } from '@/api/user/UserModel'
export default function useBaseLogin(){
    //登录表单ref
    const loginFormRef = ref<InstanceType<typeof ElForm>>();
    //表单绑定的数据域
    const loginModel = reactive<LoginParm>({
        username:'',
        password:'',
        code:''
    })
    //表单验证规则
    const rules = reactive({
        username:[{
            required:true,
            trigger:'change',
            message:'请填写登录账户'
        }],
        password:[{
            required:true,
            trigger:'change',
            message:'请填写登录密码'
        }],
        code:[{
            required:true,
            trigger:'change',
            message:'请填写验证码'
        }]
    })
    return {
        loginModel,
        rules,
        loginFormRef
    }
}
```

useLogin.ts

```
import { getCurrentInstance } from 'vue'
import { loginApi } from "@/api/user/user";
import { LoginParm } from "@/api/user/UserModel";
export default function useLogin(loginModel: LoginParm) {
    const { proxy } = getCurrentInstance() as any;
    //登录提交
    const login = () => {
        proxy.$refs.loginFormRef.validate(async (valid: boolean) => {
            console.log(valid)
            if (valid) {
                await loginApi(loginModel).then(res => {
                    if (res.code == 200) {
                        console.log(res)
                    }
                })
            }
        })
    }
    return {
        login
    }
}
```

9、Login.vue页面

```
<template>
    <div class="logincontainer">
        <el-form
            class="loginForm"
            :model="loginModel"
            ref="loginFormRef"
            :rules="rules"
            :inline="false"
        >
            <el-form-item>
                <div class="loginTitle">系统登录</div>
            </el-form-item>
            <el-form-item prop='username'>
                <el-input placeholder="请输入账户" v-model="loginModel.username">
</el-input>
            </el-form-item>
            <el-form-item prop='password'>
                <el-input placeholder="请输入密码" v-model="loginModel.password">
</el-input>
            </el-form-item>
            <el-form-item prop='code'>
                <el-row :gutter="20">
                    <el-col :span="16">
                        <el-input placeholder="请输入验证码" v-
model="loginModel.code"></el-input>
                    </el-col>
                    <el-col :span="8">
                        <!-- <el-input placeholder="请输入验证码" v-
model="loginModel.code"></el-input> -->
                        <img :src='imgSrc' @click="getImage" />
                    </el-col>
                </el-row>
            </el-form-item>
            <el-form-item>
                <el-row :gutter="20" style="margin-top: 10px;">
                    <el-col :span="12">
                       <el-button class="mybtn" @click="login" type="primary"
size="default">登录</el-button>

                    </el-col>
                    <el-col :span="12">
                        <el-button class="mybtn"  size="default">取消</el-button>
                    </el-col>
                </el-row>
            </el-form-item>
        </el-form>
    </div>
</template>
<script setup lang='ts'>
import useImage from '@/composables/login/useImage';
import useBaseLogin from '@/composables/login/useBaseLogin';
import useLogin from '@/composables/login/useLogin';

//验证码
const {imgSrc,getImage} = useImage();

//登录基础数据
const {loginModel,rules,loginFormRef} = useBaseLogin();

//登录提交
const {login} = useLogin(loginModel);

</script>
<style scoped lang='scss'>
.logincontainer {
    height: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    .loginForm {
        height: 320px;
        width: 400px;
        border-radius: 10px;
        padding: 20px 35px;
        box-shadow: 0 0 25px #cac6c6;
        .loginTitle {
            font-size: 24px;
            font-weight: 600;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .mybtn{
            width: 100%;
        }
    }
}
</style>
```

第25讲 登录接口优化

1、在store目录modules下新建user.ts

```
import { loginApi } from "@/api/user/user";
import { LoginParm, LoginResult } from "@/api/user/UserModel";
import { Result } from "@/http/request";
import { ActionContext } from "vuex";
import { RootState } from "../index";
import { setToken,setUserId,setExpireTime } from '@/utils/auth';
//定义state类型
export type UserState = {
    token:string,
    userId:string
}
```

```
//定义state
export const state : UserState = {
    token:'',
    userId:''
}
//定义mutations
export const mutations = {
    setToken(state:UserState,token:string){
        state.token = token;
    },
    setUserId(state:UserState,userId:string){
        state.userId = userId;
    }
}
//定义actios
export const actions = {
    login({commit}:ActionContext<UserState,RootState>,loginParm:LoginParm){
        return new Promise<Result>((resolve,reject)=>{
            loginApi(loginParm).then(res=>{
                if(res.data.code == 200){
                    //设置到vuex中
                    commit('setToken',res.data.token)
                    commit('setUserId',res.data.id)
                    //存储到sessionStorage
                    setToken(res.data.token)
                    setUserId(res.data.id)
                    setExpireTime(res.data.expireTime)
                }
                resolve(res)
            }).catch(error=>{
                reject(error)
            })
        })
    }
}
//定义getters
export const getters = {
    getToken(state:UserState){
        return state.token
    }
}
export default{
    namespaced:true,
    state,
    mutations,
    actions,
    getters
}
```

2、在store/index中加入user模块

```
// store.ts
import { InjectionKey } from 'vue'
import { createStore, useStore as baseUseStore, Store } from 'vuex'
import tabs,{TabsState} from '../store/modules/tabs'
import menu,{MenuState} from '../store/modules/menu'
import user,{UserState} from '../store/modules/user'
import { CommonStore } from './help'
//是一种规范
export type RootState = {
    tabs:TabsState,
    menu:MenuState,
    user:UserState
}
//导入所有的模块
export const modules = {
    tabs: tabs,
    menu: menu,
    user:user
}
export const key: InjectionKey<Store<RootState>> = Symbol()

export const store = createStore<RootState>({
    modules
}) as CommonStore

// 定义自己的 `useStore` 组合式函数
export function useStore(){
    return baseUseStore(key) as CommonStore
}
```

3、src下新建utils文件夹，然后新建auth.ts

```
enum Keys{
    Token = 'token',
    UserId = 'userId',
    ExpireTime = 'expireTime'
}
//存储token到session
export const setToken = (token:string)=>{
    sessionStorage.setItem(Keys.Token,token)
}
export const getToken = ()=>{
    return sessionStorage.getItem(Keys.Token)
}
//存储userId到sessionStorage
export const setUserId = (userId:number)=>{
    sessionStorage.setItem(Keys.UserId,JSON.stringify(userId))
}
export const getUserId = ()=>{
    return sessionStorage.getItem(Keys.Token)
}
//存储过期时间
export const setExpireTime = (time:number)=>{
    sessionStorage.setItem(Keys.ExpireTime,JSON.stringify(time))
}
export const getExpireTime = ()=>{
    return sessionStorage.getItem(Keys.ExpireTime)
}
```

4、composables/login/useLogin修改为如下

```
import { getCurrentInstance } from 'vue'
import { LoginParm } from "@/api/user/UserModel";
import { useStore } from '@/store';
import { useRouter } from 'vue-router';
export default function useLogin(loginModel: LoginParm) {
    const store = useStore();
    const router = useRouter();
    const { proxy } = getCurrentInstance() as any;
    //登录提交
    const login = () => {
        proxy.$refs.loginFormRef.validate(async (valid: boolean) => {
            if (valid) {
                store.dispatch('user/login',loginModel).then(res =>{
                    if(res.data.code == 200){
                        //跳转首页
                        router.push({path:'/'})
                    }
                })
                // await loginApi(loginModel).then(res => {
                //     if (res.code == 200) {
                //         console.log(res)
                //     }
                // })
            }
        })
    }
    return {
        login
    }
}
```

第26讲 用户权限信息接口对接

1.1、接口请求地址

```
{baseURL}/api/sysUser/getInfo
```

1.2、请求方式

```
GET
```

1.3、请求参数

```
请求头部携带token进行验证
```

1.4、响应参数

参数名称	类型	说明
id	string	用户id
avatar	string	头像地址
introduction	string	简介
name	string	姓名
roles	Array	权限集合

1.5、在api/user/UserModel.ts添加UserInfo

```
export interface UserInfo{
    avatar:string;
    id:string;
    introduction:string;
    name:string;
    roles:Array<string>
}
```

1.6、在api/user/user.ts添加如下方法

```
enum Api {
    login = '/api/user/login',
    getImg = '/api/sysUser/image',
    getInfo = '/api/sysUser/getInfo'
}
//获取用户信息
export const getInfo = async () => {
    return await http.get<UserInfo>(Api.getInfo)
}
```

1.7、store/modules/user.ts

1.7.1、UserState添加 permissions

```
export type UserState = {
    token:string,
    userId:string,
    permissions:string[]
}
//定义state
export const state : UserState = {
    token:'',
    userId:'',
    permissions:[]
}
```

1.7.2、actions添加getInfo方法

```
    //获取用户信息
    getInfo({commit}:ActionContext<UserState,RootState>){
        return new Promise((resolve,reject) =>{
            getInfoApi().then(res =>{
                //设置权限
                commit('setPermissions',res.data.roles)
                resolve(res.data)
            }).catch((error)=>{
                reject(error)
            })
        })
    },
```

1.7.3、getters添加getPermissions()方法

```
export const getters = {
    getToken(state:UserState){
        return state.token
    },
    getPermissions(state:UserState){
        return state.permissions
    }
}
```

第27讲 动态菜单数据接口对接

1.1、接口请求地址

```
{baseURL}/api/sysUser/getMenuList
```

1.2、请求方式

```
GET
```

1.3、请求参数

```
请求头部携带token进行验证
```

1.4、响应参数格式

```
{
    "msg": "菜单查询成功！",
    "code": 200,
    "data": [
        {
            "path": "/system",
            "component": "Layout",
            "alwaysShow": true,
            "name": "system",
            "meta": {
                "title": "系统管理",
                "icon": "el-icon-menu",
                "roles": [
                    "sys:manage"
                ]
            },
            "children": [
                {
                    "path": "/department",
                    "component": "/system/department/department",
                    "alwaysShow": false,
                    "name": "department",
                    "meta": {
                        "title": "机构管理",
                        "icon": "el-icon-document",
                        "roles": [
                            "sys:dept"
                        ]
                    }
                },
                {
                    "path": "/userList",
                    "component": "/system/User/UserList",
                    "alwaysShow": false,
                    "name": "userList",
                    "meta": {
                        "title": "用户管理",
                        "icon": "el-icon-s-custom",
                        "roles": [
                            "sys:user"
                        ]
                    }
                },
                {
                    "path": "/roleList",
                    "component": "/system/Role/RoleList",
                    "alwaysShow": false,
                    "name": "roleList",
                    "meta": {
                        "title": "角色管理",
                        "icon": "el-icon-s-tools",
                        "roles": [
                            "sys:role"
                        ]
                    }
                },
                {
                    "path": "/menuList",
                    "component": "/system/Menu/MenuList",
                    "alwaysShow": false,
                    "name": "menuList",
                    "meta": {
                        "title": "权限管理",
                        "icon": "el-icon-document",
                        "roles": [
                            "sys:menu"
                        ]
                    }
                }
            ]
        },
        {
            "path": "/goods",
            "component": "Layout",
            "alwaysShow": true,
            "name": "goods",
            "meta": {
                "title": "商品管理",
                "icon": "el-icon-document",
                "roles": [
                    "sys:goods"
                ]
            },
            "children": [
                {
                    "path": "/goodCategory",
                    "component": "/goods/goodsCategory/goodsCategoryList",
                    "alwaysShow": false,
                    "name": "goodCategory",
                    "meta": {
                        "title": "分类管理",
                        "icon": "el-icon-document",
                        "roles": [
                            "sys:goodsCategory"
                        ]
                    }
                }
            ]
        }
    ]
}
```

参数名称	类型	说明
path	string	路由地址
component	string	组件路径
name	string	路由名称
meta	string	菜单标题、图标、权限相关信息
children	Array	下级菜单

1.5、在api/menu下新建menu.ts

```
import http from "@/http/http"
enum Api {
    getMenuList = '/api/sysUser/getMenuList'
}
export const getMenuListApi = async ()=>{
    return await http.get(Api.getMenuList)
}
```

1.6、store/modules/menu修改

1.6.1、MenuState添加menuList

```
export type MenuState = {
    count: number,
    collapse: boolean,
    menuList: any
}
export const state: MenuState = {
    count: 0,
    collapse: false,
    menuList: [
        {
            path: '/dashboard',
            component: "Layout",
            meta: {
                title: "首页",
                icon: "HomeFilled",
                roles: ["sys:manage"]
            },
            children: []
        }
    ]
}
```

1.6.2、mutations添加setMenuList()

```
setMenuList: (state: MenuState, routes: Array<RouteRecordRaw>) => {
        state.menuList = state.menuList.concat(routes)
}
```

1.6.3、actions添加getMenuList()

```
import Layout from '@/layout/Index.vue'
const modules = import.meta.glob('../../views/**/*.vue')
```

```
getMenuList({ commit }: ActionContext<MenuState, RootState>, router: any) {
        return new Promise((resolve, reject) => {
            //存的是有权限的路由，是一个数组
            getMenuList().then(res => {
                // console.log(res)
                let accessedRoutes;
                if (res.code == 200) {
                    accessedRoutes = filterAsyncRoutes(res.data, router)
                }
                commit('setMenuList', accessedRoutes)
                resolve(accessedRoutes)
            }).catch(error => {
                reject(error)
            })
        })
    }
```

1.6.4、menu.ts添加如下方法

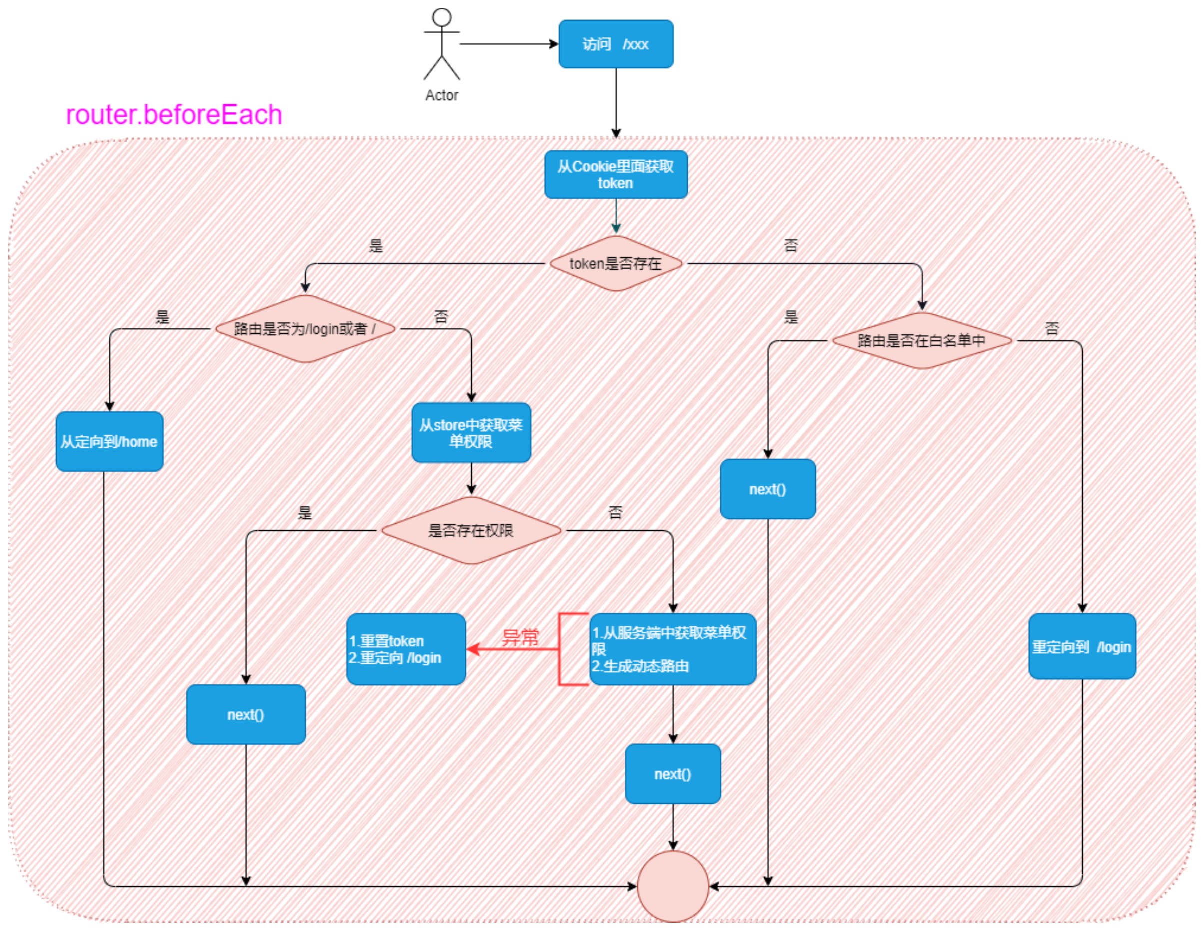
```
export function filterAsyncRoutes(routes: RouteRecordRaw[], router: any) {
    const res: Array<RouteRecordRaw> = []
    //循环每一个路由
    routes.forEach((route: any) => {
        const tmp = { ...route }
        const component = tmp.component;
        if (route.component) {
            if (component == 'Layout') {
                tmp.component = Layout;
            } else {
                tmp.component = modules[`../../views${component}.vue`]
            }
        }
        //判断是否有下级
        if (tmp.children) {
            tmp.children = filterAsyncRoutes(tmp.children, router)
        }
        router.addRoute(tmp)
        res.push(tmp)
    })
    return res
}
```

1.6.5、getters添加getMenuList()方法

```
getMenuList: (state: MenuState) => {
        return state.menuList;
},
```

第28讲 权限验证对接讲解

1、权限验证流程图



1、在router/index添加 constantRoutes

```
export const constantRoutes: Array<RouteRecordRaw> = [
    {
        path: '/',
        component: Layout,
        redirect: '/dashboard',
        children: [
            {
                path: '/dashboard',
                component: () => import('@/layout/dashboard/Index.vue'),
                name: 'dashboard',
                meta: {
                    title: '首页',
                    icon: '#icondashboard'
                }
            }
        ]
    },
    {
        path: '/login',
        name:'login',
        component: () => import('@/views/login/login.vue'),
    }
]
```

2、修改createRouter 为如下

```
//创建
const router = createRouter({
    history: createWebHistory(),
    routes:constantRoutes
})
```

3、在store/modules/menu.ts的getters中添加 getMenuList

```
getMenuList:(state:MenuState) =>{
        return state.menuList
}
```

4、修改MenuBar.vue中，获取菜单数据如下

```
//菜单数据
const menuList = computed(() =>{
    return store.getters['menu/getMenuList']
})
```

5、main.ts添加如下代码

```
const whiteList = ['/login'];
router.beforeEach(async (to, from, next) => {
    let token = getToken();
    if (token) { //token存在
        if (to.path === "/login" || to.path === "/") {
            next({ path: '/' })
        } else {
            console.log(store.state)
            let hasRoles = store.state.user.permissions &&
store.state.user.permissions.length > 0;
            if (hasRoles) {
                next();
            } else {
                try {
                    await store.dispatch('user/getInfo')
                    await store.dispatch('menu/getMenuList',router)
                    //确保动态添加的路由已经被完全加载上去
                    next({ ...to, replace: true })
                } catch (error) {
                    //重置token
                    removeToken();
                    //跳到登录
                    next({ path: '/login' })
                }

            }
        }
    } else { //token不存在 , 跳转的时候，需要注意 BredCum.vue里面判断first
        //判断是否存在白名单中
        if (whiteList.indexOf(to.path) !== -1) { //存在白名单中
            next();
        } else { //不存在白名单中,去登录
            next({ path: '/login' })
        }
    }
})
```

第29讲 部门管理列表页面制作

升级element plus

```
1、卸载  npm uninstall element-plus --save
2、安装  npm install element-plus --save
```

1、列表接口说明

1.1、接口请求地址

```
{baseURL}/api/department/list
```

1.2、请求方式

```
GET
```

1.3、请求参数

名称	类型	必填	备注
searchName	string	否	部门名称

1.4、响应参数格式

```
{
    "msg": "查询成功!",
    "code": 200,
    "data": [{
        "id": 1,
        "pid": 0,
        "likeId": "0",
        "parentName": "顶级部门",
        "manager": null,
        "name": "高原西北风科技有限公司",
        "deptCode": null,
        "deptAddress": null,
        "deptPhone": null,
        "orderNum": null,
        "open": null,
        "children": [{
            "id": 18,
            "pid": 1,
            "likeId": "0",
            "parentName": "高原西北风科技有限公司",
            "manager": "",
            "name": "人事管理部",
            "deptCode": "RSGL",
            "deptAddress": "",
            "deptPhone": "",
            "orderNum": null,
            "open": null,
            "children": [{
                "id": 40,
                "pid": 18,
                "likeId": "0",
                "parentName": "人事管理部",
                "manager": "",
                "name": "实习生管理部",
                "deptCode": "",
                "deptAddress": "",
                "deptPhone": "",
                "orderNum": null,
                "open": null,
                "children": []
            }]
        }, {
            "id": 19,
            "pid": 1,
            "likeId": "0",
            "parentName": "高原西北风科技有限公司",
            "manager": "",
            "name": "软件研发部",
            "deptCode": "RJYF",
            "deptAddress": "",
            "deptPhone": "",
            "orderNum": null,
            "open": null,
            "children": [{
                "id": 39,
                "pid": 19,
                "likeId": "0",
                "parentName": "软件研发部",
                "manager": "",
                "name": "java开发部",
                "deptCode": "",
                "deptAddress": "",
                "deptPhone": "",
                "orderNum": null,
                "open": null,
                "children": []
            }, {
                "id": 20,
                "pid": 19,
                "likeId": "0",
                "parentName": "软件研发部",
                "manager": "张三",
                "name": "前端开发",
                "deptCode": "QDKF",
                "deptAddress": "北京",
                "deptPhone": "18687116223",
                "orderNum": 1,
                "open": null,
                "children": []
            }]
        }, {
            "id": 12,
            "pid": 1,
            "likeId": "0",
            "parentName": "高原机械制造有限公司",
            "manager": "张丽荣",
            "name": "生产部门",
            "deptCode": "SCBM",
            "deptAddress": "",
            "deptPhone": "18787171906",
            "orderNum": 2,
            "open": null,
            "children": [{
                "id": 13,
                "pid": 12,
                "likeId": "0",
                "parentName": "生产部门",
                "manager": "李明",
                "name": "排产部门",
                "deptCode": "PCBM",
                "deptAddress": "",
                "deptPhone": "18687116223",
                "orderNum": 1,
                "open": null,
                "children": []
            }, {
                "id": 14,
                "pid": 12,
                "likeId": "0",
                "parentName": "生产部门",
                "manager": "崔浩",
                "name": "检验部门",
                "deptCode": "JYBM",
                "deptAddress": "",
                "deptPhone": "13314358245",
                "orderNum": 3,
                "open": null,
                "children": []
            }]
        }]
    }]
}
```

2、在src/api下新建dept目录，然后新建DeptModel.ts

```
export interface ListParm {
    searchName:string;
}
```

3、在src/api下新建dept目录，然后新建dept.ts

```
import http from '@/http/http'
import {ListParm} from '@/api/dept/DeptModel'
enum Api {
    getDeptList = '/api/department/list'
}
//部门列表
export const getDeptListApi = async(parms:ListParm) =>{
    return await http.get(Api.getDeptList,parms)
}
```

4、composables下新建department目录

然后新建useDept.ts、useBaseModel.ts、useDialog.ts、useDeptTable.ts 文件

4.1、useDeptTable.ts如下所示

```
/**
 * 部门表格列表业务逻辑
 */
import { getDeptListApi } from '@/api/dept/dept'
import { ListParm } from '@/api/dept/DeptModel'
import {reactive,onMounted} from 'vue'
export default function useDeptTable(){
    //定义搜索数据
    const searchFrom = reactive<ListParm>({
        searchName:''
    })
    //定义表格数据
    const tableData = reactive({
        list:[]
    })
    //获取表格数据
    const getDeptList = async() =>{
        const res = await getDeptListApi(searchFrom);
        if(res && res.code == 200){
            console.log(res.data)
            tableData.list = res.data;
        }
    }
    //首次加载数据
    onMounted(() =>{
        getDeptList()
    })
    return{
        searchFrom,
        tableData,
        getDeptList
    }
}
```

4.2、useBaseModel.ts

```
import { reactive } from "vue";

/**
 * 部门模块基础数据
 */
export default function useBaseModel() {
    //表单验证规则
    const rules = reactive({})
    return {
        rules
    }
}
```

5、department.vue页面

```
<template>
    <el-main>
        <!-- 搜索栏 -->
        <el-form :model="searchFrom" :rules="rules" label-width="80px"
:inline="true" size="small">
            <el-form-item>
                <el-input v-model="searchFrom.searchName" placeholder="请输入部门名
称"></el-input>
            </el-form-item>
            <el-form-item>
                <el-button :icon="Search">查询</el-button>
                <!-- <el-button type="primary" :icon='Plus'>新增</el-button> -->
                <el-button size="mini" type='primary' :icon="Plus">新增</el-
button>
            </el-form-item>
        </el-form>
        <!-- 表格 -->
        <el-table
            :data="tableData.list"
            style="width: 100%"
            row-key="id"
            default-expand-all
            size="medium"
            border
            :tree-props="{ children: 'children', hasChildren: 'hasChildren' }"
        >
            <el-table-column prop="name" label="部门名称" />
            <el-table-column prop="deptCode" label="部门编码" />
            <el-table-column prop="deptPhone" label="部门电话" />
            <el-table-column width='200' align='center' label="操作">
                <template #default="scope">
                    <el-button size="mini" type='success' :icon="Edit">编辑</el-
button>

                    <el-button
                        size="mini"
                        type="danger"
                        :icon='Close'
                    >删除</el-button>
                </template>
            </el-table-column>
        </el-table>
    </el-main>
</template>
<script setup lang="ts">
import {Edit,Close,Plus,Search} from '@element-plus/icons'
import useBaseModel from '@/composables/department/useBaseModel';
import useDeptTable from '@/composables/department/useDeptTable';
//基础数据
const { rules } = useBaseModel();

//表格列表
const { searchFrom, tableData, getDeptList } = useDeptTable();
</script>
```

第30讲 通用弹框组件封装

1、前置知识

```
defineProps
defineEmits
```

2、弹框组件

```
<!-- 此处注意，使用 v-model="visible"绑定的形式
上线会报错

修改为如下方式
:model-value="visible"
 -->

<template>
  <el-dialog
    top="5vh"
    :title="title"
    :model-value="visible"
    :width="width + 'px'"
    :before-close="onClose"
    append-to-body
  >
    <div class="container" :style="{ height: height + 'px' }">
      <slot name="content"></slot>
    </div>
    <template #footer>
      <span class="dialog-footer">
        <el-button type="danger" @click="onClose">取 消</el-button>
        <el-button type="primary" @click="onConfirm">确 定</el-button>
      </span>
    </template>
  </el-dialog>
</template>

<script setup lang="ts">
import { watch } from 'vue'
```

```
const props = defineProps({
  title: {
    type: String,
    default: "标题",
  },
  visible: {
    type: Boolean,
    default: false,
  },
  width: {
    type: Number,
    default: 600,
  },
  height: {
    type: Number,
    default: 250,
  }
})
watch(() => props.visible, () => {
  console.log('组件监听到')
  console.log(props)
})
const emit = defineEmits(['onClose', 'onConfirm'])
//弹框取消事件
const onClose = () => {
  emit('onClose')
}
//弹框确定事件
const onConfirm = () => {
  emit('onConfirm')
}
</script>

<style lang="scss" scope>
.container {
  overflow-x: initial;
  overflow-y: auto;
}
.el-dialog {
  border-top-left-radius: 7px !important;
  border-top-right-radius: 7px !important;
  .el-dialog__header {
    border-top-left-radius: 7px !important;
    border-top-right-radius: 7px !important;
    background-color: #1890ff !important;
    .el-dialog__title {
      color: #fff;
      font-size: 16px;
      font-weight: 600;
    }
    .el-dialog__close {
      color: #fff;
    }
  }
  .el-dialog__body {
    padding: 10px;
  }
  .el-dialog__footer {
    border-top: 1px solid #e8eaec !important;
    padding: 10px;
  }
}
</style>
```

第31讲 弹框组件的使用

```
<template>
  <el-button type="primary" size="default" @click="addBtn">新增</el-button>
  <SysDialog
  :title="dialog.title"
  :visible="dialog.visible"
  :width="dialog.width"
  :height="dialog.height"
  @onClose='onClose'
  @onConfirm='onConfirm'
  >
      <template v-slot:content>
          弹框内容
      </template>
  </SysDialog>
</template>
<script setup lang='ts'>
import { reactive} from 'vue'
import SysDialog from './SysDialog.vue';
const dialog = reactive({
    title:'新增',
    height:280,
    width:630,
    visible:false
})
//按钮点击事件
const addBtn = () =>{
    dialog.visible = true;
}
//弹框关闭
const onClose = () =>{
    dialog.visible = false;
}
//弹框确定
const onConfirm = () =>{
    dialog.visible = false;
}
</script>
<style scoped lang='scss'>
</style>
```

第32讲 新增部门业务抽离与布局

1、抽离部门增删改查业务到useDept.ts

```
import { DeptModel,AddDeptModel } from '@/api/dept/DeptModel'
import { ref } from 'vue';
import { EditType } from '@/type/BaseEnum';
export default function useDept() {

    const addDeptRef = ref<{ show: (type: string) => void }>();
    //搜索
    const searchBtn = () => {

    }
    //新增
    const addBtn = () => {
        addDeptRef.value?.show(EditType.ADD);
    }
    //编辑
    const editBtn = (row: DeptModel) => {
        addDeptRef.value?.show(EditType.EDIT)
    }
    //删除
    const deleteBtn = (row: DeptModel) => {

    }
    //保存
    const save = (model:AddDeptModel) =>{
        console.log('保存')
        console.log(model)
    }
    return {
        searchBtn,
        addBtn,
        editBtn,
        deleteBtn,
        addDeptRef,
        save
    }
}
```

2、/api/dept/DeptModel下新建 DeptModel和AddDeptModel

```
//表格数据
export interface DeptModel {
    id: number;
    pid: number;
    likeId: number;
    parentName: string;
    manager: string;
    name: string;
    deptCode: string;
    deptAddress: string;
    deptPhone: string;
    orderNum: number;
    open: boolean;
    children: any
}
//表单提交的数据类型
export interface AddDeptModel {
    type: string;
    id: string | number;
    pid: string |number;
    parentName: string;
    manager: string;
    deptAddress: string;
    deptPhone: string;
    name: string;
    deptCode: string;
    orderNum: string;
}
```

3、src下新建hooks文件，然后新建useDialog.ts

```
import { reactive } from 'vue'
import { DialogModel } from "@/type/BaseType"
export default function useDialog() {
    //弹框属性
    const dialog = reactive<DialogModel>({
        title: '',
        visible: false,
        width: 630,
        height: 300
    })
    //弹框取消
    const onClose = () => {
        dialog.visible = false;
    }
    //弹框确定
    const onConfirm = () => {
        dialog.visible = false;
    }
    //弹框显示
    const onShow = () => {
        dialog.visible = true;
    }
    return {
        dialog,
        onClose,
        onConfirm,
        onShow
    }
}
```

4、src下新建type文件，新建BastType.ts 然后新建 DialogModel

```
//弹框属性类型
export type DialogModel = {
    title:string,
    visible:boolean,
    height:number,
    width:number
}
```

5、views/system/department下新建 AddAndEdit.vue组件

```
<template>
  <SysDialog
  :title="dialog.title"
  :visible="dialog.visible"
  :height="dialog.height"
  :width="dialog.width"
  @onClose='onClose'
  @onConfirm='confirm'
  >
      <template v-slot:content>
          弹框内容
      </template>
  </SysDialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import {EditType, Title} from '@/type/BaseEnum'
import useBaseModel from '@/composables/department/useBaseModel';

//基础数据
const {dialogModel} = useBaseModel();

//弹框
const {dialog,onClose,onShow} = useDialog();

//注册事件
const emit = defineEmits(['save'])
//弹框确定返回表单值给父组件
const confirm = () =>{
    //返回值
    emit('save',dialogModel)
    //关闭弹框
    onClose();
}

//父组件调用子组件展示弹框
const show = (type:string) =>{
    //显示弹框
    onShow();
    //设置弹框的标题
    type == EditType.ADD ? dialog.title = Title.ADD : dialog.title = Title.EDIT
    //设置type
    dialogModel.type = type;
}

defineExpose({
    show
})
</script>
<style scoped lang='scss'>
</style>
```

6、src/type下新建BaseEnum.ts

```
export enum EditType{
    ADD = '0',
    EDIT = '1'
}
export enum Title {
    ADD = '新增',
    EDIT = '编辑'
}
```

7、src/composables/department/useBaseModel 添加如下

```
import { reactive } from 'vue'
import { AddDeptModel } from '@/api/dept/DeptModel'
export default function useBaseModel() {
    //表单验证
    const rules = reactive({})

     //表单绑定的数据
     const dialogModel = reactive<AddDeptModel>({
        type: "",
        id: "",
        pid: "",
        parentName: "",
        manager: "",
        deptAddress: "",
        deptPhone: "",
        name: "",
        deptCode: "",
        orderNum: ""
    })
    return {
        rules,
        dialogModel
    }
}
```

第33讲 上级部门选择

1、上级树接口说明

1.1、接口请求地址

```
{baseURL}/api/department/parent
```

1.2、请求方式

```
GET
```

1.3、请求参数 无

1.4、响应参数格式

```
{
    "msg":"查询成功!",
    "code":200,
    "data":[
        {
            "id":0,
            "pid":-1,
            "likeId":"0,",
            "parentName":null,
            "manager":null,
            "name":"顶级部门",
            "deptCode":null,
            "deptAddress":null,
            "deptPhone":null,
            "orderNum":null,
            "open":null,
            "children":[
                {
                    "id":1,
                    "pid":0,
                    "likeId":"0",
                    "parentName":"顶级部门",
                    "manager":null,
                    "name":"高原西北风科技有限公司",
                    "deptCode":null,
                    "deptAddress":null,
                    "deptPhone":null,
                    "orderNum":null,
                    "open":null,
                    "children":[
                        {
                            "id":18,
                            "pid":1,
                            "likeId":"0",
                            "parentName":"高原西北风科技有限公司",
                            "manager":"",
                            "name":"人事管理部",
                            "deptCode":"RSGL",
                            "deptAddress":"",
                            "deptPhone":"",
                            "orderNum":null,
                            "open":null,
                            "children":[
                                {
                                    "id":40,
                                    "pid":18,
                                    "likeId":"0",
                                    "parentName":"人事管理部",
                                    "manager":"",
                                    "name":"实习生管理部",
                                    "deptCode":"",
                                    "deptAddress":"",
                                    "deptPhone":"",
                                    "orderNum":null,
                                "open":null,
                                "children":[

                                ]
                            }
                        ]
                    },
                    {
                        "id":19,
                        "pid":1,
                        "likeId":"0",
                        "parentName":"高原西北风科技有限公司",
                        "manager":"",
                        "name":"软件研发部",
                        "deptCode":"RJYF",
                        "deptAddress":"",
                        "deptPhone":"",
                        "orderNum":null,
                        "open":null,
                        "children":[
                            {
                                "id":39,
                                "pid":19,
                                "likeId":"0",
                                "parentName":"软件研发部",
                                "manager":"",
                                "name":"java开发部",
                                "deptCode":"",
                                "deptAddress":"",
                                "deptPhone":"",
                                "orderNum":null,
                                "open":null,
                                "children":[

                                ]
                            },
                            {
                                "id":20,
                                "pid":19,
                                "likeId":"0",
                                "parentName":"软件研发部",
                                "manager":"张三",
                                "name":"前端开发",
                                "deptCode":"QDKF",
                                "deptAddress":"北京",
                                "deptPhone":"18687116223",
                                "orderNum":1,
                                "open":null,
                                "children":[

                                ]
                            }
                        ]
                    },
                    {
                        "id":12,
                        "pid":1,
                        "likeId":"0",
                            "parentName":"高原机械制造有限公司",
                            "manager":"张丽荣",
                            "name":"生产部门",
                            "deptCode":"SCBM",
                            "deptAddress":"",
                            "deptPhone":"18787171906",
                            "orderNum":2,
                            "open":null,
                            "children":[
                                {
                                    "id":13,
                                    "pid":12,
                                    "likeId":"0",
                                    "parentName":"生产部门",
                                    "manager":"李明",
                                    "name":"排产部门",
                                    "deptCode":"PCBM",
                                    "deptAddress":"",
                                    "deptPhone":"18687116223",
                                    "orderNum":1,
                                    "open":null,
                                    "children":[

                                    ]
                                },
                                {
                                    "id":14,
                                    "pid":12,
                                    "likeId":"0",
                                    "parentName":"生产部门",
                                    "manager":"崔浩",
                                    "name":"检验部门",
                                    "deptCode":"JYBM",
                                    "deptAddress":"",
                                    "deptPhone":"13314358245",
                                    "orderNum":3,
                                    "open":null,
                                    "children":[

                                    ]
                                }
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

2、页面

```
<template>
    <sys-dialog
        :title="parentDialog.title"
        :width="parentDialog.width"
        :height="parentDialog.height"
        :visible="parentDialog.visible"
        @onClose="parentOnClose"
        @onConfirm="parentOnConfirm"
    >
        <template v-slot:content>
            <el-tree
                style="font-size: 14px"
                ref="parentTree"
                :data="treeData.data"
                node-key="id"
                :props="defaultProps"
                default-expand-all
                :highlight-current="true"
                :expand-on-click-node="false"
                @node-click="handleNodeClick"
            >
                <template #default="{ node, data }">
                    <div class="custom-tree-container">
                        <!-- 长度为0说明没有下级 -->
                        <span v-if="data.children.length == 0">
                            <DocumentRemove style="width: 1.3em; height: 1.3em;
margin-right: 5px;color: #8c8c8c;"></DocumentRemove>
                        </span>
                        <span v-else @click.stop="openBtn(data)">
                            <component style="width: 1.1em; height: 1.1em;
margin-right: 5px;color: #8c8c8c;" :is="data.open ? Plus : Minus" />
                        </span>
                        <span>{{ node.label }}</span>
                    </div>
                </template>
            </el-tree>
        </template>
    </sys-dialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue'
import { ref, reactive } from 'vue'
import useBaseModel from '@/composables/department/useBaseModel'
import { getDeptParentApi } from '@/api/dept/dept'
import { ElTree } from 'element-plus'
import { DocumentRemove, Plus, Minus } from '@element-plus/icons'
const { parentDialog } = useBaseModel();
//关闭弹框
const parentOnClose = () => {
    parentDialog.visible = false;
}
//确定弹框
const parentOnConfirm = () => {
    emits('select',selectNode)
    parentDialog.visible = false;
```

```
}
//显示弹框
const treeData = reactive({
    data: []
})
//树属性绑定
const defaultProps = reactive({
    children: "children",
    label: "name",
    disabled:'false'
})
const showParent = async () => {
    console.log('来了')
    //获取部门树数据
    const res = await getDeptParentApi()
    console.log(res)
    treeData.data = res.data;
    parentDialog.visible = true;
}

//返回选中的数据
const selectNode = reactive({
    parentId:'',
    parentName:''
})
//树节点点击事件
const handleNodeClick = (node) => {
    console.log(node)
    selectNode.parentId = node.id
    selectNode.parentName = node.name;
}
const parentTree = ref<InstanceType<typeof ElTree>>()
const openBtn = (data) => {
    data.open = !data.open;
    parentTree.value.store.nodesMap[data.id].expanded = !data.open;
}
//暴露给外部调用
defineExpose({
    showParent
})
const emits = defineEmits(['select'])

</script>

<style lang="scss">
.el-tree {
  // 将每一行的设置为相对定位 方便后面before after 使用绝对定位来固定位置
  .el-tree-node {
    position: relative;
    padding-left: 10px;
  }
  // 子集像右偏移 给数线留出距离
  .el-tree-node__children {
    padding-left: 20px;
  }
  //这是竖线
  .el-tree-node :last-child:before {
    height: 40px;
  }
  .el-tree > .el-tree-node:before {
    border-left: none;
  }
  .el-tree > .el-tree-node:after {
    border-top: none;
  }
  //这自定义的线 的公共部分
  .el-tree-node:before,
  .el-tree-node:after {
    content: "";
    left: -4px;
    position: absolute;
    right: auto;
    border-width: 1px;
  }
  .tree :first-child .el-tree-node:before {
    border-left: none;
  }
  // 竖线
  .el-tree-node:before {
    border-left: 1px dotted #d9d9d9;
    bottom: 0px;
    height: 100%;
    top: -25px;
    width: 1px;
  }
  //横线
  .el-tree-node:after {
    border-top: 1px dotted #d9d9d9;
    height: 20px;
    top: 14px;
    width: 24px;
  }
  .el-tree-node__expand-icon.is-leaf {
    width: 8px;
  }
  //去掉elementui自带的展开按钮  一个向下的按钮,打开时向右
  .el-tree-node__content > .el-tree-node__expand-icon {
    display: none;
  }
  //每一行的高度
  .el-tree-node__content {
    line-height: 30px;
    height: 30px;
    padding-left: 10px !important;
  }
}
//去掉最上级的before  after 即是去电最上层的连接线
.el-tree > div {
  &::before {
    display: none;
  }
  &::after {
    display: none;
  }
}
</style>
```

第34讲 工具类的封装与全局挂载

1、给request.ts添加token过期的处理

```
cleanSession();
//跳到登录
window.location.href = "/login";
```

2、信息确定弹框的封装 myconfirm.ts

```
//信息确认弹框封装
import { ElMessageBox } from "element-plus";
export default function myconfirm(text) {
    return new Promise((resolve, reject) => {
        ElMessageBox.confirm(text, '系统提示', {
            confirmButtonText: '确定',
            cancelButtonText: '取消',
            type: 'warning'
        }).then(() => {
            resolve(true);
        }).catch(() => {
            reject(false)
        })
    }).catch(() => {
        return false;
    })
}
```

3、对象复制的封装 objCoppy.ts

```
//对象的快速复制
export default function objCoppy(obj1,obj2){
    Object.keys(obj2).forEach(key =>{
        obj2[key] = obj1[key]
    })
}
```

4、表单清空封装 resetForm.ts

```
//表单清空
// formRef: 表单的ref属性  obj表单的数据域
export default function resetForm(formRef, obj) {
    //清空数据域
    Object.keys(obj).forEach(key => {
        obj[key] = '';
    })
    //清空表单
    if (formRef) {
        formRef.resetFields();
        formRef.clearValidate();
    }
}
```

5、main.ts引入挂载

```
import resetForm from './utils/resetForm'
import objCoppy from './utils/objCoppy'
import myconfirm from './utils/myconfirm'

//清空表单
app.config.globalProperties.$resetForm = resetForm;
//对象复制
app.config.globalProperties.$objCoppy = objCoppy;
//确定弹框
app.config.globalProperties.$myconfirm = myconfirm;
```

6、useInstance的封装

```
import { ComponentInternalInstance, getCurrentInstance } from 'vue'
export default function useInstance() {
    const { appContext,proxy } = getCurrentInstance() as
ComponentInternalInstance
    const global = appContext.config.globalProperties
    return {
        proxy,
        global
    }
}
```

7、使用

```
import useInstance from '@/hooks/useInstance';

const {global} = useInstance();

//信息确认框使用
 let confirm = await global.$myconfirm('确定删除该数据吗?')

//element plus全局信息提示信息使用
 global.$message.success('新增成功');
```

8、解决any的验证

implicitly has an 'any' type

在tsconfig.json添加如下配置

```
"noImplicitAny": false,
```

第35讲 新增、编辑部门接口对接

1、新增接口参数

1.1、接口请求地址

```
{baseURL}/api/department
```

1.2、请求方式

```
POST
```

1.3、请求参数

名称	类型	必填	备注
id	string	否	部门id
pid	string	是	上级部门id
parentName	string	是	上级部门名称
manager	string	否	部门经理
deptAddress	string	否	部门地址
deptPhone	string	否	部门电话
name	string	是	部门名称
deptCode	string	否	部门编码
orderNum	string	否	序号

1.4、响应参数格式

```
{
    "msg":"新增部门成功!",
    "code":200,
    "data":null
}
```

2、编辑接口参数

2.1、接口请求地址

```
{baseURL}/api/department
```

2.2、请求方式

```
PUT
```

2.3、请求参数

名称	类型	必填	备注
id	string	是	部门id
pid	string	是	上级部门id
parentName	string	是	上级部门名称
manager	string	否	部门经理
deptAddress	string	否	部门地址
deptPhone	string	否	部门电话
name	string	是	部门名称
deptCode	string	否	部门编码
orderNum	string	否	序号

2.4、响应参数格式

```
{
    "msg":"编辑部门成功!",
    "code":200,
    "data":null
}
```

第36讲 删除部门接口对接

1、删除接口参数说明

1.1、接口请求地址

```
{baseURL}/api/department
```

1.2、请求方式

```
DELETE
```

1.3、请求参数

名称	类型	必填	备注
id	string	是	部门id

1.4、响应参数格式

```
{
    "msg":"删除部门成功!",
    "code":200,
    "data":null
}
```

第37讲 用户管理左侧部门树

1、左侧部门树接口说明

1.1、接口请求地址

```
{baseURL}/api/department/list
```

1.2、请求方式

```
GET
```

1.3、请求参数 无

2、新建LeftTree.vue组件

```
<template>
    <el-tree
        ref="parentTree"
        :data="treeData.data"
        node-key="id"
        :props="defaultProps"
        default-expand-all
        :highlight-current="true"
        :expand-on-click-node="false"
        @node-click="handleNodeClick"
    >
        <template #default="{ node, data }">
            <div class="custom-tree-container">
                <!-- 长度为0说明没有下级 -->
                <span v-if="data.children.length == 0">
                    <DocumentRemove
                        style="width: 1.3em; height: 1.3em; margin-right:
5px;color: #8c8c8c;"
                    ></DocumentRemove>
                </span>
                <!-- 点击展开和关闭 -->
                <span v-else @click.stop="openBtn(data)">
                    <component
                        style="width: 1.1em; height: 1.1em; margin-right:
5px;color: #8c8c8c;"
                        :is="data.open ? Plus : Minus"
                    />
                </span>
                <span>{{ node.label }}</span>
            </div>
        </template>
    </el-tree>
</template>
<script setup lang='ts'>
import { DocumentRemove, Plus, Minus } from '@element-plus/icons'
import useLeftTree from '@/composables/user/useLeftTree';
//树相关数据
const { treeData, defaultProps, handleNodeClick, getTreeData, openBtn,
parentTree, selectNode } = useLeftTree();

</script>

<style lang="scss">
.el-tree {
    font-size: 14px;
  .el-tree-node {
    position: relative;
    padding-left: 10px;
  }
  // 子集像右偏移 给数线留出距离
  .el-tree-node__children {
    padding-left: 20px;
  }
  //这是竖线
  .el-tree-node :last-child:before {
    height: 40px;
  }
  .el-tree > .el-tree-node:before {
    border-left: none;
  }
  .el-tree > .el-tree-node:after {
    border-top: none;
  }
  //这自定义的线 的公共部分
  .el-tree-node:before,
  .el-tree-node:after {
    content: "";
    left: -4px;
    position: absolute;
    right: auto;
    border-width: 1px;
  }
  .tree :first-child .el-tree-node:before {
    border-left: none;
  }
  // 竖线
  .el-tree-node:before {
    border-left: 1px solid #d9d9d9;
    bottom: 0px;
    height: 100%;
    top: -25px;
    width: 1px;
  }
  //横线
  .el-tree-node:after {
    border-top: 1px solid #d9d9d9;
    height: 20px;
    top: 14px;
    width: 12px;
  }
  .el-tree-node__expand-icon.is-leaf {
    width: 8px;
  }
  //去掉elementui自带的展开按钮  一个向下的按钮,打开时向右
  .el-tree-node__content > .el-tree-node__expand-icon {
    display: none;
  }
  //每一行的高度
  .el-tree-node__content {
    line-height: 30px;
    height: 30px;
    padding-left: 10px !important;
  }
}
//去掉最上级的before  after 即是去电最上层的连接线
.el-tree > div {
  &::before {
    display: none;
  }
  &::after {
    display: none;
  }
}

</style>
```

3、在composables/user下新建useLeftTree.ts

```
import { DeptModel, SelectNode } from '@/api/dept/DeptModel'
import { reactive, ref ,onMounted,nextTick} from 'vue'
import { getLeftTreeApi } from '@/api/user/user'
import { ElTree } from 'element-plus';
export default function useLeftTree(){
     //树的ref属性
     const parentTree = ref<InstanceType<typeof ElTree>>();
     //上级树的数据
     const treeData = reactive({
         data: []
     })
     //选中的数据
     const selectNode = reactive<SelectNode>({
         id: '',
         name: ''
     })
     //树的属性
     const defaultProps = reactive({
         children: 'children', //设置树的children
         label: 'name', //设置树的名字属性字段
     })

     //树的点击事件
     const handleNodeClick = (data: DeptModel) => {
         console.log('点击了树了')
         selectNode.id = data.id;
         selectNode.name = data.name
         console.log(selectNode)
     }
     //获取树的数据
```

```
        const getTreeData = async () => {
            let res = await getLeftTreeApi();
            if (res && res.code == 200) {
                treeData.data = res.data;
                nextTick(() =>{
                   const firstNode = document.querySelector(".el-tree-node") as
any;
                   if(firstNode){
                       firstNode.click();
                   }
                })
            }
        }
        onMounted(() =>{
           getTreeData();
        })

        //加号和减号的点击事件
        const openBtn = (data: DeptModel) => {
            console.log(data)
            //设置展开或者关闭
            data.open = !data.open;
            if (parentTree.value) {
                parentTree.value.store.nodesMap[data.id].expanded = !data.open;
            }
        }
        return {
            treeData,
            defaultProps,
            handleNodeClick,
            getTreeData,
            openBtn,
            parentTree,
            selectNode
        }
}
```

4、UserList.vue组件

```
<template>
    <el-container :style="{ height: containerHeight + 'px' }">
        <el-aside width="200px" style="border-right: 1px solid #dfe6ec">
            <!-- Aside content -->
            <LeftTree></LeftTree>
        </el-aside>
        <el-main height>用户列表</el-main>
    </el-container>
</template>
<script setup lang="ts">
import LeftTree from './LeftTree.vue';
import {ref,onMounted} from 'vue'
const containerHeight = ref(0);
onMounted(() =>{
    containerHeight.value = window.innerHeight - 100
})
```

```
</script>
```

第38讲 用户管理列表数据接口对接

1、用户列表接口参数说明

1.1、接口请求地址

```
{baseURL}/api/user/list
```

1.2、请求方式

```
GET
```

1.3、请求参数

名称	类型	必填	备注
deptId	number	否	部门id
currentPage	number	是	当前页
pageSize	number	是	页容量
loginName	string	否	用户姓名

1.4、响应参数格式

```
{
    "msg": "查询成功!",
    "code": 200,
    "data": {
        "records": [
            {
                "id": 9,
                "username": "admin",
                "loginName": "超级管理员",
                "password":
"$2a$10$rDkPvvAFV8kqwvKJzwlRv.i.q.wz1w1pzOSFsHn/55jNeZFQv/eCm",
                "authorities": null,
                "nickName": "超级管理员",
                "mobile": "110",
                "email": "admin@163.com",
                "deptId": 1,
                "deptName": "高原机械制造有限公司",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2019-12-16 10:25:53",
                "isAdmin": "1",
                "permissionList": null,
                "sex": "0",
                "postId": null,
                "postName": null,
                "enabled": true,
                "accountNonExpired": true,
                "accountNonLocked": true,
                "credentialsNonExpired": true
            },
            {
                "id": 70,
                "username": "zsf",
                "loginName": "张三丰",
                "password": "1234",
                "authorities": null,
                "nickName": "菜鸟一只",
                "mobile": "18687116223",
                "email": "3501754007@qq.com",
                "deptId": 1,
                "deptName": "云南南天信息",
                "createTime": "2021-01-31 16:06:21",
                "updateTime": "2021-01-31 16:06:21",
                "isAdmin": "0",
                "permissionList": null,
                "sex": "1",
                "postId": null,
                "postName": null,
                "enabled": true,
                "accountNonExpired": true,
                "accountNonLocked": true,
                "credentialsNonExpired": true
            },
            {
                "id": 71,
                "username": "zwj",
                "loginName": "张无忌33",
                "password": "1234",
                "authorities": null,
                "nickName": "黑白双侠33",
                "mobile": "18787171933",
                "email": "3501754007@qq.com",
                "deptId": 1,
                "deptName": "云南南天信息",
                "createTime": "2021-01-31 16:10:14",
                "updateTime": "2021-01-31 16:10:14",
                "isAdmin": "0",
                "permissionList": null,
                "sex": "0",
                "postId": null,
                "postName": null,
                "enabled": true,
                "accountNonExpired": true,
                "accountNonLocked": true,
                "credentialsNonExpired": true
            },
            {
                "id": 76,
                "username": "lcy",
                "loginName": "lcy",
                "password":
"$2a$10$wW1o9nFm9nqdXoLZPoJOUOTRfGgESANiMf2kuD1n/yuLncbc7n1SO",
                "authorities": null,
                "nickName": "",
                "mobile": "18314358245",
                "email": "",
                "deptId": 1,
                "deptName": "高原西北风科技有限公司",
                "createTime": "2021-04-09 08:34:53",
                "updateTime": "2021-04-09 08:34:53",
                "isAdmin": "0",
                "permissionList": null,
                "sex": "0",
                "postId": null,
                "postName": null,
                "enabled": true,
                "accountNonExpired": true,
                "accountNonLocked": true,
                "credentialsNonExpired": true
            }
        ],
        "total": 4,
        "size": 10,
        "current": 1,
        "orders": [ ],
        "optimizeCountSql": true,
        "hitCount": false,
        "searchCount": true,
        "pages": 1
    }
}
```

2、src/api/user下新增如下方法

```
//获取用户列表
export const getUserListApi = async(parm:UserListParm) =>{
    return await http.get(Api.getUserList,parm)
}
```

src/api/user/UserModel添加如下代码

```
/**
 * 用户列表查询参数
 */
export interface UserListParm{
    deptId:string | number;
    loginName:string;
    currentPage:number;
    pageSize:number;
    total:number;
}
```

3、在src/composables/user下新建useUserTable.ts

```
import {reactive} from 'vue'
import { UserListParm } from '@/api/user/UserModel'
import { getUserListApi } from '@/api/user/user'
export default function useUserTable(){
    //列表参数
    const listParm = reactive<UserListParm>({
        deptId:'',
        currentPage:1,
        pageSize:10,
        loginName:'',
        total:0
    })
    //表格数据
    const tableData = reactive({
        list:[]
    })

    //获取表格数据
    const getUserList = async() =>{
        let res = await getUserListApi(listParm)
        console.log(res)
        if(res && res.code == 200){
            tableData.list = res.data.records
            listParm.total = res.data.total;
        }
    }
    //树点击数据
    const treeClick = (deptId:number) =>{
        console.log('父组件收到')
        //设置点击的部门id
        listParm.deptId = deptId;
        //获取部门列表
        getUserList();
    }
    //页容量改变触发
    const sizeChange = (size:number) =>{
        listParm.pageSize = size;
        getUserList();
    }
    //页数改变触发
    const currentChange = (page:number) =>{
        listParm.currentPage = page
        getUserList();
    }
    return{
        listParm,
        tableData,
        getUserList,
        treeClick,
        sizeChange,
        currentChange
    }
}
```

4、UserList.vue组件

```
<template>
    <el-container :style="{ height: containerHeight + 'px' }">
        <el-aside width="200px" style="border-right: 1px solid #dfe6ec">
            <!-- Aside content -->
            <LeftTree ref="userLeftTree" @treeClick="treeClick"></LeftTree>
        </el-aside>
        <el-main>
            <!-- 搜索栏 -->
            <el-form :model="listParm" label-width="80px" :inline="true"
size="mini">
                <el-form-item label>
                    <el-input v-model="listParm.loginName"></el-input>
                </el-form-item>
                <el-form-item>
                    <el-button size="mini" :icon="Search">搜索</el-button>
                    <el-button size="mini" :icon="Close">重置</el-button>
                    <el-button size="mini" type="primary" :icon="Plus">新增</el-
button>
                </el-form-item>
            </el-form>
            <!-- 用户表格 -->
            <el-table :height="tableHeight" :data="tableData.list" border
stripe>
                <el-table-column prop="loginName" label="用户名"></el-table-
column>
                <el-table-column prop="deptName" label="所属部门"></el-table-
column>
                <el-table-column prop="mobile" label="电话"></el-table-column>
                <el-table-column prop="email" label="邮箱"></el-table-column>
                <el-table-column align="center" width="290" label="操作">
                    <template #default="scope">
                        <el-button type="primary" size="mini" :icon="Edit"
@click=''>编辑</el-button>
                        <el-button type="danger" size="mini" :icon="Delete"
@click=''>删除</el-button>
                    </template>
                </el-table-column>
            </el-table>
            <!-- 分页 -->
            <el-pagination
                @size-change="sizeChange"
                @current-change="currentChange"
                :current-page.sync="listParm.currentPage"
                :page-sizes="[10, 20, 40, 80, 100]"
                :page-size="listParm.pageSize"
                :total="listParm.total"
                background
            ></el-pagination>
        </el-main>
    </el-container>
</template>
<script setup lang="ts">
import { Search, Close, Plus, Delete, Edit } from '@element-plus/icons';
import LeftTree from './LeftTree.vue';
import useUserTable from '@/composables/user/useUserTable';
import { ref, onMounted, nextTick } from 'vue'
//容器高度
const containerHeight = ref(0);
//表格高度
const tableHeight = ref(0);
//表格数据
const { listParm, tableData, getUserList, treeClick, sizeChange, currentChange }
= useUserTable();
onMounted(() => {
    nextTick(() => {
        containerHeight.value = window.innerHeight - 100
        tableHeight.value = window.innerHeight - 220
    })
})
</script>
```

5、解决分页英文的问题

在main.ts中引入如下代码

```
import locale from 'element-plus/lib/locale/lang/zh-cn'
app.use(ElementPlus,{locale})
```

第39讲 新增用户弹框展示与业务抽离

1、新建AddAndEdit.vue组件

```
<template>
  <SysDialog
  :title="dialog.title"
  :width="dialog.width"
  :height="dialog.height"
  :visible="dialog.visible"
  @onClose='onClose'
  @onConfirm='confirm'
  >
    <template v-slot:content>
      新增用户
    </template>
  </SysDialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import useUserAddAndEdit from '@/api/user/useUserAddAndEdit';
//弹框属性
const {dialog,onShow,onClose} = useDialog();


const {show,confirm} = useUserAddAndEdit(dialog,onShow,onClose);

//暴露方法
defineExpose({
  show
})

</script>
<style scoped lang='scss'>
</style>
```

2、在src/composables/user下新建useUserAddAndEdit.ts

```
import { EditType, Title } from "@/type/BaseEnum"
import { DialogModel } from "@/type/BastType"

export default function useUserAddAndEdit(dialog:DialogModel,onShow,onClose){
    const show = (type:string) =>{
        console.log('进入子组件了')
        console.log(type)
        //设置弹框属性
        type == EditType.ADD ? dialog.title = Title.ADD : dialog.title = 
Title.EDIT
        console.log(dialog)
        //显示弹框
        onShow();
    }
    //确定
    const confirm = () =>{
       onClose();
    }
    return {
        show,
        confirm
    }
}
```

3、在src/composables/user下新建useUser.ts

```
import { EditType } from '@/type/BaseEnum';
import { ref } from 'vue'
export default function useUser() {
    //新增弹框ref属性
    const userAddRef = ref<{ show: (type: string) => void }>();
    //新增
    const addBtn = () => {
        userAddRef.value?.show(EditType.ADD)
    }
    //编辑
    const editBtn = () => {
        userAddRef.value?.show(EditType.EDIT)
    }
    //删除
    const deleteBtn = () => {

    }
    //分配角色
    const assignBtn = () =>{

    }
    return {
        addBtn,
        editBtn,
        deleteBtn,
        userAddRef,
        assignBtn
    }
}
```

4、UserList.vue中使用AddAndEdit.vue组件

```
<template>
    <el-container :style="{ height: containerHeight + 'px' }">
        <el-aside width="200px" style="border-right: 1px solid #dfe6ec">
            <!-- Aside content -->
            <LeftTree ref="userLeftTree" @treeClick="treeClick"></LeftTree>
        </el-aside>
        <el-main>
            <!-- 搜索栏 -->
            <el-form :model="listParm" label-width="80px" :inline="true"
size="mini">
                <el-form-item label>
                    <el-input v-model="listParm.loginName"></el-input>
                </el-form-item>
                <el-form-item>
                    <el-button size="mini" :icon="Search">搜索</el-button>
                    <el-button size="mini" :icon="Close">重置</el-button>
                    <el-button size="mini" type="primary" @click="addBtn"
:icon="Plus">新增</el-button>
                </el-form-item>
            </el-form>
            <!-- 用户表格 -->
            <el-table :height="tableHeight" :data="tableData.list" border
stripe>
                <el-table-column prop="loginName" label="用户名"></el-table-
column>
                <el-table-column prop="deptName" label="所属部门"></el-table-
column>
                <el-table-column prop="mobile" label="电话"></el-table-column>
                <el-table-column prop="email" label="邮箱"></el-table-column>
                <el-table-column align="center" width="310" label="操作">
                    <template #default="scope">
                        <el-button type="primary" size="mini" :icon="Edit"
@click='editBtn'>编辑</el-button>
                        <el-button type="primary" size="mini" :icon="Edit"
@click='assignBtn'>分配角色</el-button>
                        <el-button type="danger" size="mini" :icon="Delete"
@click='deleteBtn'>删除</el-button>
                    </template>
                </el-table-column>
            </el-table>
            <!-- 分页 -->
            <el-pagination
                @size-change="sizeChange"
                @current-change="currentChange"
                :current-page.sync="listParm.currentPage"
                :page-sizes="[10, 20, 40, 80, 100]"
                :page-size="listParm.pageSize"
                :total="listParm.total"
                layout="total, sizes, prev, pager, next, jumper"
                background
            ></el-pagination>
        </el-main>
    </el-container>
    <!-- 新增、编辑弹框 -->
    <AddAndEdit ref="userAddRef"></AddAndEdit>
</template>
<script setup lang="ts">
import { Search, Close, Plus, Delete, Edit } from '@element-plus/icons';
import AddAndEdit from './AddAndEdit.vue';
import LeftTree from './LeftTree.vue';
import useUserTable from '@/composables/user/useUserTable';
import useUser from '@/composables/user/useUser';
import { ref, onMounted, nextTick } from 'vue'

//容器高度
const containerHeight = ref(0);
//表格高度
const tableHeight = ref(0);
//表格数据
const { listParm, tableData, getUserList, treeClick, sizeChange, currentChange }
= useUserTable();

//新增、编辑、删除
const { userAddRef,addBtn,editBtn,deleteBtn,assignBtn } = useUser();
onMounted(() => {
    nextTick(() => {
        containerHeight.value = window.innerHeight - 100
        tableHeight.value = window.innerHeight - 220
    })
})
</script>
```

第40讲 新增用户页面布局

1、AddAndEdit.vue组件

```
<template>
    <SysDialog
        :title="dialog.title"
        :width="dialog.width"
        :height="dialog.height"
        :visible="dialog.visible"
        @onClose="onClose"
        @onConfirm="confirm"
    >
        <template v-slot:content>
            <el-form
                :model="addModel"
                ref="addUserForm"
                :rules="rules"
                label-width="80px"
                size="mini"
            >
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop="deptName" label="所属部门">
                            <el-input v-model="addModel.deptName"></el-input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop="loginName" label="姓名">
                            <el-input v-model="addModel.loginName"></el-input>
                        </el-form-item>
                    </el-col>
                </el-row>
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop="mobile" label="电话">
                            <el-input v-model="addModel.mobile"></el-input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="昵称">
                            <el-input v-model="addModel.nickName"></el-input>
                        </el-form-item>
                    </el-col>
                </el-row>
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="邮箱">
                            <el-input v-model="addModel.email"></el-input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop="username" label="登录名">
                            <el-input v-model="addModel.username"></el-input>
                        </el-form-item>
                    </el-col>
                </el-row>
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop="password" label="密码">
                            <el-input v-model="addModel.password"></el-input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop="sex" label="性别">
                            <el-radio-group v-model="addModel.sex">
                                <el-radio :label="0">男</el-radio>
                                <el-radio :label="1">女</el-radio>
                            </el-radio-group>
                        </el-form-item>
                    </el-col>
                </el-row>
            </el-form>
        </template>
    </SysDialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import useUseAddAndEdit from '@/composables/user/useUseAddAndEdit';
import useBaseModel from '@/composables/user/useBaseModel';

//声明事件
```

```
const emit = defineEmits(['save'])

//基础属性
const { addModel, rules } = useBaseModel();

//弹框属性
const { dialog, onShow, onClose } = useDialog()

const { confirm, show, addUserForm } = useUseAddAndEdit(dialog, onShow, onClose,
addModel,emit)


//暴露方法给外部使用
defineExpose({
    show
})
</script>
<style scoped lang='scss'>
</style>
```

第41讲 上级部门选择

1、上级树接口说明

1.1、接口请求地址

```
{baseURL}/api/department/parent
```

1.2、请求方式

```
GET
```

1.3、请求参数 无

第42讲 新增、编辑用户数据接口对接

1、新增接口参数

1.1、接口请求地址

```
{baseURL}/api/user
```

1.2、请求方式

```
POST
```

1.3、请求参数

名称	类型	必填	备注
deptId	string	是	部门id
deptName	string	是	部门名称
email	string	是	邮箱
loginName	string	是	姓名
mobile	string	是	手机号
nickName	string	否	昵称
password	string	是	登录密码
username	string	是	登录账户
sex	string	是	性别

1.4、响应参数格式

```
{
    "msg":"新增用户成功!",
    "code":200,
    "data":null
}
```

2、编辑接口参数

2.1、接口请求地址

```
{baseURL}/api/user
```

2.2、请求方式

```
PUT
```

1.3、请求参数

名称	类型	必填	备注
id	string	是	用户id
deptId	string	是	部门id
deptName	string	是	部门名称
email	string	是	邮箱
loginName	string	是	姓名
mobile	string	是	手机号
nickName	string	否	昵称
password	string	是	登录密码
username	string	是	登录账户
sex	string	是	性别

1.4、响应参数格式

```
{
    "msg":"编辑成功!",
    "code":200,
    "data":null
}
```

第43讲 删除用户接口对接

1、删除接口参数说明

1.1、接口请求地址

```
{baseURL}/api/user
```

1.2、请求方式

```
DELETE
```

1.3、请求参数

名称	类型	必填	备注
id	string	是	用户id

1.4、响应参数格式

```
{
    "msg":"删除成功!",
    "code":200,
    "data":null
}
```

第44讲 角色管理列表数据获取

1、角色列表接口参数说明

1.1、接口请求地址

```
{baseURL}/api/role/list
```

1.2、请求方式

```
GET
```

1.3、请求参数

名称	类型	必填	备注
userId	number	否	用户id
currentPage	number	是	当前页
pageSize	number	是	页容量
name	string	否	角色名称

1.4、响应数据格式

```
{
    "msg": "查询成功!",
    "code": 200,
    "data": {
        "records": [
            {
                "id": 9,
                "createUser": 9,
                "name": "系统管理员",
                "remark": "系统管理员",
                "createTime": "2020-05-20 07:51:28",
                "updateTime": "2020-05-20 07:51:28"
            },
            {
                "id": 10,
                "createUser": 9,
                "name": "超级管理员",
                "remark": "超级管理员",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-08 11:11:11"
            },
            {
                "id": 13,
                "createUser": 9,
                "name": "销售管理员",
                "remark": "管理销售人员",
                "createTime": "2020-04-14 12:22:53",
                "updateTime": "2020-04-14 12:22:53"
            },
            {
                "id": 14,
                "createUser": 9,
                "name": "财务管理员",
                "remark": "管理公司财务",
                "createTime": "2020-04-14 12:23:10",
                "updateTime": "2020-04-14 12:23:10"
            },
            {
                "id": 15,
                "createUser": 9,
                "name": "人才管理员",
                "remark": "人才管理员",
                "createTime": "2020-04-18 09:58:05",
                "updateTime": "2020-04-18 09:58:05"
            },
            {
                "id": 59,
                "createUser": 76,
                "name": "lcy测试",
                "remark": "lcy测试",
                "createTime": "2021-04-09 08:45:55",
                "updateTime": "2021-04-09 08:45:55"
            },
            {
                "id": 60,
                "createUser": 9,
                "name": "lcy222",
                "remark": "lcy222",
                "createTime": "2021-04-10 22:58:38",
                "updateTime": "2021-04-10 22:58:38"
            }
        ],
        "total": 7,
        "size": 10,
        "current": 1,
        "orders": [ ],
        "optimizeCountSql": true,
        "hitCount": false,
        "searchCount": true,
        "pages": 1
    }
}
```

2、在src/api下新建role文件夹，然后新建role.ts 和 RoleModel.ts

RoleModel.ts

```
/**
 * 角色列表查询参数
 */
export interface RoleListParm{
    userId:string | number;
    currentPage:number;
    pageSize:number;
    name:string;
}
```

role.ts

```
import http from "@/http/http"
import { RoleListParm } from "./RoleModel"
enum Api{
    getList = '/api/role/list'
}
//角色列表
export const getRoleListApi = async(parm:RoleListParm) =>{
    return await http.get(Api.getList,parm)
}
```

3、composables/role/useRoleTable.ts

```
import { getRoleListApi } from '@/api/role/role'
import { RoleListParm } from '@/api/role/RoleModel'
import { reactive,onMounted } from 'vue'
import { getUserId } from '@/utils/auth'
export default function useRoleTable() {
    //表格数据
    const roleTable = reactive({
        list: []
    })

    //表格查询参数
    const listParm = reactive<RoleListParm>({
        userId: getUserId() || '',
        currentPage: 1,
        pageSize: 10,
        name: ''
    })
    //获取表格数据
    const getRoleList = async () => {
        let res = await getRoleListApi(listParm)
        if(res && res.code == 200){
            console.log('表格数据')
            console.log(res)
        }
    }
    onMounted(() =>{
        getRoleList()
    })

    return{
        listParm,
        roleTable,
        getRoleList
    }
}
```

4、RoleList.vue

```
<template>
<div>角色管理</div>
</template>
<script setup lang="ts">
import useRoleTable from '@/composables/role/useRoleTable';

//表格列表
const {listParm,getRoleList,roleTable} = useRoleTable()
</script>
```

第45讲 角色列表页面制作

1、src/composables/role/useRoleTable.ts添加 earchBtn, resetBtn,sizeChange,currentChange方法

```
import { getRoleListApi } from '@/api/role/role'
import { RoleListParm } from '@/api/role/RoleModel'
import { reactive,onMounted } from 'vue'
import { getUserId } from '@/utils/auth'
export default function useRoleTable() {
    //表格数据
    const roleTable = reactive({
        list: []
    })

    //表格查询参数
    const listParm = reactive<RoleListParm>({
        userId: getUserId() || '',
        currentPage: 1,
        pageSize: 10,
        name: '',
        total:0
    })
    //获取表格数据
    const getRoleList = async () => {
        let res = await getRoleListApi(listParm)
        if(res && res.code == 200){
            roleTable.list = res.data.records;
            listParm.total = res.data.total;
        }
    }
    //搜索按钮
    const searchBtn = () =>{
        getRoleList()
    }
    //重置按钮
    const resetBtn = () =>{
        listParm.name = '';
        getRoleList()
    }
    //页容量改变触发
    const sizeChange = (size:number) =>{
        listParm.pageSize = size;
        getRoleList();
    }
    //页数改变触发
    const currentChange = (page:number) =>{
        listParm.currentPage = page;
        getRoleList();
    }

    onMounted(() =>{
        getRoleList()
    })

    return{
        listParm,
        roleTable,
        getRoleList,
        searchBtn,
        resetBtn,
        sizeChange,
        currentChange
    }
}
```

2、RoleList.vue组件

```
<template>
    <el-main>
        <!-- 搜索栏 -->
        <el-form :model="listParm" label-width="80px" :inline="true"
size="mini">
            <el-form-item>
                <el-input placeholder="请输入角色名称" v-model="listParm.name">
</el-input>
            </el-form-item>
            <el-form-item>
                <el-button :icon="Search" @click="searchBtn">搜索</el-button>
                <el-button :icon="Close" style="color: #FF7670;"
@click="resetBtn">重置</el-button>
                <el-button type="primary" :icon="Plus">新增</el-button>
            </el-form-item>
        </el-form>
        <!-- 表格 -->
        <el-table :height="tableHeight" :data="roleTable.list" border stripe>
```

```
                <el-table-column prop="name" label="角色名称"></el-table-column>
                <el-table-column prop="remark" label="角色备注"></el-table-column>
                <el-table-column label="操作" align="center" width="300">
                    <template #default>
                        <el-button type="primary" size="mini" :icon="Edit">编辑</el-
button>
                        <el-button type="primary" size="mini" :icon="Setting">分配权限
</el-button>
                        <el-button type="danger" size="mini" :icon="Delete">删除</el-
button>
                    </template>
                </el-table-column>
            </el-table>
            <!-- 分页 -->
            <el-pagination
                @size-change="sizeChange"
                @current-change="currentChange"
                :current-page.sync="listParm.currentPage"
                :page-sizes="[10, 20, 40, 80, 100]"
                :page-size="listParm.pageSize"
                layout="total, sizes, prev, pager, next, jumper"
                :total="listParm.total"
                background
            ></el-pagination>
        </el-main>
</template>
<script setup lang="ts">
import { ref, nextTick, onMounted } from 'vue';
import { Search, Close, Plus, Delete, Edit, Setting } from '@element-
plus/icons';
import useRoleTable from '@/composables/role/useRoleTable';
//表格高度
const tableHeight = ref(0);

//表格列表
const { listParm, getRoleList, roleTable, searchBtn, resetBtn, sizeChange,
currentChange } = useRoleTable()

onMounted(() => {
    nextTick(() => {
        tableHeight.value = window.innerHeight - 220
    })
})
</script>
```

第46讲 新增角色弹框展示与业务抽离

1、src/composables/role下新建useRole

```
import { EditType } from '@/type/BaseEnum'
import {ref} from 'vue'
export default function useRole(){
    // 弹框组件ref属性
    const addRoleRef = ref<{show:(type:string) => void}>()
    //新增
    const addBtn = () =>{
        addRoleRef.value?.show(EditType.ADD)
    }
    //编辑
    const editBtn = () =>{
        addRoleRef.value?.show(EditType.EDIT)
    }
    //删除
    const deleteBtn = () =>{

    }
    //保存
    const save = () =>{

    }
    //分配权限
    const assignPermission = () =>{

    }
    return{
        addBtn,
        editBtn,
        deleteBtn,
        save,
        assignPermission,
        addRoleRef
    }
}
```

2、views/role下新建AddRole.vue组件

```
<template>
  <SysDialog
  :title="dialog.title"
  :width="dialog.width"
  :visible="dialog.visible"
  :height="dialog.height"
  @onClose='onClose'
  @onConfirm='confirm'
  >
      <template v-slot:content>
          新增角色
      </template>
  </SysDialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import useAddRole from '@/composables/role/useAddRole';
//弹框属性
const {dialog,onClose,onShow} = useDialog()

const {confirm,show} = useAddRole(dialog,onClose,onShow)

//暴露方法
defineExpose({
    show
})
</script>
<style scoped lang='scss'>
</style>
```

3、src/composables/role下新建useAddRole.ts

```
import { EditType, Title } from "@/type/BaseEnum";
import { DialogModel } from "@/type/BastType";

export default function useAddRole(dialog:DialogModel,onClose,onShow){
    //确定
    const confirm = () =>{
        onClose();
    }
    //显示弹框
    const show = (type:string) =>{
        //设置弹框标题
        type == EditType.ADD ? dialog.title = Title.ADD : Title.EDIT
        //显示弹框
        onShow();
    }
    return{
        confirm,
        show
    }
}
```

4、RoleList.vue组件使用新增弹框

```
<template>
    <el-main>
        <!-- 搜索栏 -->
        <el-form :model="listParm" label-width="80px" :inline="true"
size="mini">
            <el-form-item>
                <el-input placeholder="请输入角色名称" v-model="listParm.name">
</el-input>
            </el-form-item>
            <el-form-item>
                <el-button :icon="Search" @click="searchBtn">搜索</el-button>
                <el-button style="color: #FF7670;" :icon="Close"
@click="resetBtn">重置</el-button>
                <el-button type="primary" :icon="Plus" @click="addBtn">新增</el-
button>
            </el-form-item>
        </el-form>
        <!-- 表格 -->
        <el-table :height="tableHeight" :data="roleTable.list" border stripe>
            <el-table-column prop="name" label="角色名称"></el-table-column>
            <el-table-column prop="remark" label="角色备注"></el-table-column>
            <el-table-column label="操作" align="center" width="300">
                <template #default>
                    <el-button type="primary" size="mini" :icon="Edit"
@click="editBtn">编辑</el-button>
                    <el-button type="primary" size="mini" :icon="Setting"
@click="assignPermission">分配权限</el-button>
```

```
                    <el-button type="danger" size="mini" :icon="Delete"
@click="deleteBtn">删除</el-button>
                </template>
            </el-table-column>
        </el-table>
        <!-- 分页 -->
        <el-pagination
            @size-change="sizeChange"
            @current-change="currentChange"
            :current-page.sync="listParm.currentPage"
            :page-sizes="[10, 20, 40, 80, 100]"
            :page-size="listParm.pageSize"
            layout="total, sizes, prev, pager, next, jumper"
            :total="listParm.total"
            background
        ></el-pagination>
    </el-main>
    <!-- 新增、编辑 -->
    <AddRole ref='addRoleRef'></AddRole>
</template>
<script setup lang="ts">
import { ref, onMounted, nextTick } from 'vue';
import { Search, Edit, Plus, Setting, Close, Delete } from '@element-
plus/icons';
import useRoleTable from '@/composables/role/useRoleTable';
import useRole from '@/composables/role/useRole';
import AddRole from './AddRole.vue';
//表格的高度
const tableHeight = ref(0);
//表格列表
const { listParm, roleTable, getRoleList, sizeChange,
currentChange,searchBtn,resetBtn } = useRoleTable()

//新增、编辑
const {addBtn,editBtn,deleteBtn,save,assignPermission,addRoleRef} = useRole()

onMounted(() => {
    nextTick(() => {
        tableHeight.value = window.innerHeight - 220
    })
})
</script>
```

第47讲 新增角色页面制作与数据对接

1、新增接口参数

1.1、接口请求地址

```
{baseURL}/api/role
```

1.2、请求方式

```
POST
```

1.3、请求参数

名称	类型	必填	备注
name	string	是	角色名称
remark	string	否	角色描述
createUser	string	是	创建人id

1.4、响应参数格式

```
{
    "msg":"新增成功!",
    "code":200,
    "data":null
}
```

2、编辑接口参数

2.1、接口请求地址

```
{baseURL}/api/role
```

2.2、请求方式

```
PUT
```

1.3、请求参数

名称	类型	必填	备注
id	string	是	角色id
name	string	是	角色名称
remark	string	否	角色备注
createUser	string	是	创建人id

1.4、响应参数格式

```
{
    "msg":"编辑成功!",
    "code":200,
    "data":null
}
```

第48讲 删除角色接口对接

1、删除接口参数说明

1.1、接口请求地址

```
{baseURL}/api/role
```

1.2、请求方式

```
DELETE
```

1.3、请求参数

名称	类型	必填	备注
id	string	是	角色id

1.4、响应参数格式

```
{
    "msg":"删除成功!",
    "code":200,
    "data":null
}
```

第49讲 权限管理列表数据获取与业务抽离

1、权限列表接口参数说明

1.1、接口请求地址

```
{baseURL}/api/menu/list
```

1.2、请求方式

```
GET
```

1.3、请求参数

```
无
```

1.4、响应数据格式

```
{
    "msg": "查询成功",
    "code": 200,
    "data": [{
        "id": 17,
        "parentId": 0,
        "parentName": "顶级菜单",
        "label": "系统管理",
        "code": "sys:manage",
        "path": "/system",
        "name": "system",
        "url": "/system/system",
        "orderNum": 1,
        "type": "0",
        "icon": "el-icon-menu",
        "remark": null,
        "createTime": "2023-08-08 11:11:11",
        "updateTime": "2023-08-09 15:26:28",
        "children": [{
            "id": 33,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "机构管理",
            "code": "sys:dept",
            "path": "/department",
            "name": "department",
            "url": "/system/department/department",
            "orderNum": 2,
            "type": "1",
            "icon": "el-icon-document",
            "remark": "机构管理",
            "createTime": "2020-04-12 22:58:29",
            "updateTime": "2020-04-08 17:12:19",
            "children": [{
                "id": 46,
                "parentId": 33,
                "parentName": "机构管理",
                "label": "新增",
                "code": "sys:addDepartment",
                "path": "",
                "name": "",
                "url": null,
                "orderNum": 0,
                "type": "2",
                "icon": "el-icon-document",
                "remark": null,
                "createTime": "2020-04-12 19:58:48",
                "updateTime": "2020-04-12 19:58:48",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 76,
                "parentId": 33,
                "parentName": "机构管理",
                "label": "编辑",
                "code": "sys:editDept",
                "path": "",
                "name": "",
                "url": null,
                "orderNum": 1,
                "type": "2",
                "icon": "el-icon-document",
                "remark": null,
                "createTime": "2020-04-12 20:42:20",
                "updateTime": "2020-04-12 20:42:20",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 78,
                "parentId": 33,
                "parentName": "机构管理",
                "label": "删除",
                "code": "sys:deleteDept",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 3,
                "type": "2",
                "icon": "el-icon-document",
                "remark": null,
                "createTime": "2020-04-18 10:25:55",
                "updateTime": "2020-04-18 10:25:55",
                "children": [],
                "value": "2",
                "open": null
            }],
            "value": "1",
            "open": null
        }, {
            "id": 18,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "用户管理",
            "code": "sys:user",
            "path": "/userList",
            "name": "userList",
            "url": "/system/User/UserList",
            "orderNum": 3,
            "type": "1",
            "icon": "el-icon-s-custom",
            "remark": null,
            "createTime": "2023-08-08 11:11:11",
            "updateTime": "2023-08-09 15:26:28",
            "children": [{
                "id": 20,
                "parentId": 18,
                "parentName": null,
                "label": "新增",
                "code": "sys:user:add",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "新增用户",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 21,
                "parentId": 18,
                "parentName": null,
                "label": "修改",
                "code": "sys:user:edit",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "修改用户",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 22,
                "parentId": 18,
                "parentName": null,
                "label": "删除",
                "code": "sys:user:delete",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "删除用户",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 80,
                "parentId": 18,
                "parentName": "用户管理",
                "label": "分配角色",
                "code": "sys:user:assign",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 0,
                "type": "2",
                "icon": "el-icon-document",
                "remark": null,
                "createTime": "2020-04-18 10:50:14",
                "updateTime": "2020-04-18 10:50:14",
                "children": [],
                "value": "2",
                "open": null
            }],
            "value": "1",
            "open": null
        }, {
            "id": 23,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "角色管理",
            "code": "sys:role",
            "path": "/roleList",
            "name": "roleList",
            "url": "/system/Role/RoleList",
            "orderNum": 4,
            "type": "1",
            "icon": "el-icon-s-tools",
            "remark": null,
            "createTime": "2023-08-08 11:11:11",
            "updateTime": "2023-08-09 15:26:28",
            "children": [{
                "id": 25,
                "parentId": 23,
                "parentName": null,
                "label": "新增",
                "code": "sys:role:add",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "新增角色",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 26,
                "parentId": 23,
                "parentName": null,
                "label": "修改",
                "code": "sys:role:edit",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "修改角色",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 27,
                "parentId": 23,
                "parentName": null,
                "label": "删除",
                "code": "sys:role:delete",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "删除角色",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 79,
                "parentId": 23,
                "parentName": "角色管理",
                "label": "分配权限",
                "code": "sys:role:assign",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 0,
                "type": "2",
                "icon": "el-icon-document",
                "remark": null,
                "createTime": "2020-04-18 10:31:05",
                "updateTime": "2020-04-18 10:31:05",
                "children": [],
                "value": "2",
                "open": null
            }],
            "value": "1",
            "open": null
        }, {
            "id": 28,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "权限管理",
            "code": "sys:menu",
            "path": "/menuList",
            "name": "menuList",
            "url": "/system/Menu/MenuList",
            "orderNum": 5,
            "type": "1",
            "icon": "el-icon-document",
            "remark": null,
            "createTime": "2023-08-08 11:11:11",
            "updateTime": "2023-08-09 15:26:28",
            "children": [{
                "id": 30,
                "parentId": 28,
                "parentName": null,
                "label": "新增",
                "code": "sys:menu:add",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "新增权限",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 31,
                "parentId": 28,
                "parentName": null,
                "label": "修改",
                "code": "sys:menu:edit",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "修改权限",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 32,
                "parentId": 28,
                "parentName": null,
                "label": "删除",
                "code": "sys:menu:delete",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "删除权限",
                "createTime": "2023-08-08 11:11:11",
                "updateTime": "2023-08-09 15:26:28",
                "children": [],
                "value": "2",
                "open": null
            }],
            "value": "1",
            "open": null
        }],
        "value": "0",
        "open": null
    }, {
        "id": 34,
        "parentId": 0,
        "parentName": "顶级菜单",
        "label": "商品管理",
        "code": "sys:goods",
        "path": "/goods",
        "name": "goods",
        "url": "/goods/index",
        "orderNum": 2,
        "type": "0",
        "icon": "el-icon-document",
        "remark": null,
        "createTime": "2020-04-12 22:49:47",
        "updateTime": "2020-04-12 17:22:03",
        "children": [{
            "id": 36,
            "parentId": 34,
            "parentName": "商品管理",
            "label": "分类管理",
            "code": "sys:goodsCategory",
            "path": "/goodCategory",
            "name": "goodCategory",
            "url": "/goods/goodsCategory/goodsCategoryList",
            "orderNum": 1,
            "type": "1",
            "icon": "el-icon-document",
            "remark": null,
            "createTime": "2020-04-12 22:54:32",
            "updateTime": "2020-04-12 17:26:30",
            "children": [{
                "id": 38,
                "parentId": 36,
                "parentName": "分类管理",
                "label": "新增",
                "code": "sys:addGoodsCategory",
                "path": "",
                "name": "",
                "url": null,
                "orderNum": 0,
                "type": "2",
                "icon": "el-icon-document",
                "remark": null,
                "createTime": "2020-04-12 17:33:58",
                "updateTime": "2020-04-12 17:33:58",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 39,
                "parentId": 36,
                "parentName": "分类管理",
                "label": "编辑",
                "code": "sys:editGoodsCategory",
                "path": "",
                "name": "",
                "url": null,
                "orderNum": 1,
                "type": "2",
                "icon": "el-icon-document",
                "remark": null,
                "createTime": "2020-04-12 17:35:30",
                "updateTime": "2020-04-12 17:35:30",
                "children": [],
                "value": "2",
                "open": null
            }],
            "value": "1",
            "open": null
        }, {
            "id": 37,
            "parentId": 34,
            "parentName": null,
            "label": "品牌管理",
            "code": "sys:goodsBrand",
            "path": "/goodsBrand",
            "name": "goodsBrand",
            "url": "/goods/goodsBrand/goodsBrandList",
            "orderNum": 2,
            "type": "1",
            "icon": "el-icon-document",
            "remark": null,
            "createTime": "2020-04-12 17:32:04",
            "updateTime": "2020-04-12 17:32:04",
            "children": [{
                "id": 87,
                "parentId": 37,
                "parentName": "品牌管理",
                "label": "新增",
                "code": "sys:brand:add",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 0,
                "type": "2",
                "icon": "",
                "remark": null,
                "createTime": "2021-02-03 09:08:24",
                "updateTime": "2021-02-03 09:08:24",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 88,
                "parentId": 37,
                "parentName": "品牌管理",
                "label": "编辑",
                "code": "sys:brand:edit",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 1,
                "type": "2",
                "icon": "",
                "remark": null,
                "createTime": "2021-02-03 09:08:50",
                "updateTime": "2021-02-03 09:08:50",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 89,
                "parentId": 37,
                "parentName": "品牌管理",
                "label": "删除",
                "code": "sys:brand:delete",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 2,
                "type": "2",
                "icon": "",
                "remark": null,
                "createTime": "2021-02-03 09:09:18",
                "updateTime": "2021-02-03 09:09:18",
                "children": [],
                "value": "2",
                "open": null
            }],
            "value": "1",
            "open": null
        }],
        "value": "0",
        "open": null
    }, {
        "id": 42,
        "parentId": 0,
        "parentName": "顶级菜单",
        "label": "系统工具",
        "code": "sys:systenConfig",
        "path": "/systenConfig",
        "name": "systenConfig",
        "url": "/systenConfig",
        "orderNum": 3,
        "type": "0",
        "icon": "el-icon-document",
        "remark": null,
        "createTime": "2020-04-12 22:50:03",
        "updateTime": "2020-04-12 17:40:41",
        "children": [{
            "id": 43,
            "parentId": 42,
            "parentName": "系统工具",
            "label": "日志管理",
            "code": "sys:systemCode",
            "path": "/systemCode",
            "name": "systemCode",
            "url": "/system/config/code",
            "orderNum": 0,
            "type": "1",
            "icon": "el-icon-document",
            "remark": null,
            "createTime": "2020-04-16 12:44:42",
            "updateTime": "2020-04-12 17:44:06",
            "children": [{
                "id": 85,
                "parentId": 43,
                "parentName": "代码生成",
                "label": "新增",
                "code": "sys:code:add",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 0,
                "type": "2",
                "icon": "",
                "remark": null,
                "createTime": "2021-02-02 22:05:31",
                "updateTime": "2021-02-02 22:05:31",
                "children": [],
                "value": "2",
                "open": null
            }, {
                "id": 86,
                "parentId": 43,
                "parentName": "代码生成",
                "label": "编辑",
                "code": "sys:code:edit",
                "path": "",
                "name": "",
                "url": "",
                "orderNum": 1,
                "type": "2",
                "icon": "",
                "remark": null,
                "createTime": "2021-02-02 22:06:15",
                "updateTime": "2021-02-02 22:06:15",
                "children": [],
                "value": "2",
                "open": null
            }],
            "value": "1",
            "open": null
        }, {
            "id": 77,
            "parentId": 42,
            "parentName": "系统工具",
            "label": "接口文档",
            "code": "sys:document",
            "path": "/document",
            "name": "http://42.193.158.170:8089/swagger-ui/index.html",
            "url": "/system/config/systemDocument",
            "orderNum": 0,
            "type": "1",
            "icon": "el-icon-document",
            "remark": null,
            "createTime": "2020-04-13 11:31:45",
            "updateTime": "2020-04-13 11:31:45",
            "children": [],
            "value": "1",
            "open": null
        }, {
            "id": 93,
            "parentId": 42,
            "parentName": "系统工具",
            "label": "测试菜单",
            "code": "sys:document:tests",
            "path": "/test",
            "name": "test",
            "url": "/system/config/test",
            "orderNum": 2,
            "type": "1",
            "icon": "el-icon-s-order",
            "remark": null,
            "createTime": "2021-03-19 14:17:02",
            "updateTime": "2021-03-19 14:17:02",
            "children": [],
            "value": "1",
            "open": null
        }],
        "value": "0",
        "open": null
    }]
}
```

第50讲 权限列表页面制作

1、MenuList.vue

```
<template style='padding:0px 20px;'>
    <el-main>
        <!-- 新增按钮 -->
        <el-form :inline="true" size="mini">
            <el-form-item>
                <el-button type="primary" @click="addBtn" :icon="Plus">新增</el-
button>
            </el-form-item>
        </el-form>
        <!-- 表格 -->
        <el-table
            :height="tableHeight"
            :data="menuTable.list"
            style="width: 100%"
            row-key="id"
            border
            default-expand-all
            :tree-props="{ children: 'children', hasChildren: 'hasChildren' }"
        >
            <el-table-column prop="label" label="菜单名称" />
            <el-table-column prop="type" label="类型">
                <template #default="scope">
                    <el-tag v-if="scope.row.type == '0'">目录</el-tag>
                    <el-tag v-if="scope.row.type == '1'" type="success">菜单</el-
tag>

                    <el-tag v-if="scope.row.type == '2'" type="danger">按钮</el-
tag>
                </template>
            </el-table-column>
            <el-table-column prop="label" label="图标">
                <template #default="scope">
                    <Icon class="icons" :icon="scope.row.icon"></Icon>
                </template>
            </el-table-column>
            <el-table-column prop="name" label="路由名称" />
            <el-table-column prop="path" label="路由地址" />
            <el-table-column prop="url" label="组件路径" />
            <el-table-column label="操作" width="210" align="center">
                <template #default="scope">
```

```
                    <el-button type="primary" size="mini" @click="editBtn"
:icon="Edit">编辑</el-button>
                    <el-button type="danger" size="mini" @click="deleteBtn"
:icon="Close">删除</el-button>
                </template>
            </el-table-column>
        </el-table>
    </el-main>
</template>
<script setup lang="ts">
import { Plus, Edit, Close } from '@element-plus/icons';
import { ref, onMounted, nextTick } from 'vue';
import useMenuTable from '@/composables/menu/useMenuTable';
import useMenu from '@/composables/menu/useMenu';
//表格的高度
const tableHeight = ref(0)
//表格数据获取
const { menuTable, getMenuTable, defaultProps } = useMenuTable()

//表格的操作
const { addBtn, editBtn, deleteBtn, save } = useMenu()

onMounted(() => {
    nextTick(() => {
        tableHeight.value = window.innerHeight - 200
    })
})
</script>
<style scoped lang="scss">
.icons {
    width: 24px;
    height: 18px;
    margin-right: 5px;
}
</style>
```

第51讲 新增权限弹框展示和业务抽离

1、在views/system/Menu下新建AddMenu.vue组件

```
<template>
    <SysDialog
        :title="dialog.title"
        :width="dialog.width"
        :height="dialog.height"
        :visible="dialog.visible"
        @onConfirm="confrim"
        @onClose="onClose"
    >
        <template v-slot:content>新增权限</template>
    </SysDialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import useAddMenu from '@/composables/menu/useAddMenu';
//弹框属性
const { dialog, onClose, onShow } = useDialog()

const { confrim, show } = useAddMenu(dialog, onClose, onShow)

//暴露出去
defineExpose({
    show
})
</script>
<style scoped lang='scss'>
</style>
```

2、在src/composables/menu下新建useAddMenu.ts

```
import { EditType, Title } from "@/type/BaseEnum";
import { DialogModel } from "@/type/BastType"

export default function useAddMenu(dialog:DialogModel, onClose, onShow){
    //确定
    const confrim = () =>{

        //弹框关闭
        onClose();
    }
    //展示弹框
    const show = (type:string) =>{
        //设置弹框属性
        type == EditType.ADD ? dialog.title = Title.ADD : dialog.title =
Title.EDIT;
        onShow();
        //设置编辑的属性

    }
    return{
        confrim,
        show
    }
}
```

3、修改src/composables/menu下的useMenu.ts如下

```
import { EditType } from "@/type/BaseEnum";
import { ref } from "vue"
export default function useMenu(){
    //弹框的ref属性
    const addMenuRef = ref<{show:(type:string) =>void}>();
    //新增
    const addBtn = () =>{
        addMenuRef.value?.show(EditType.ADD)
    }
    //编辑
    const editBtn = () =>{
        addMenuRef.value?.show(EditType.EDIT)
    }
    //删除
    const deleteBtn = () =>{

    }
    //保存
    const save = () =>{

    }
    return {
        addBtn,
        editBtn,
        deleteBtn,
        save,
        addMenuRef
    }
}
```

4、MenuList.vue组件使用新增弹框

```
<template>
    <el-main>
        <!-- 按钮 -->
        <el-form  :inline="true" size="mini">
            <el-form-item>
                <el-button type="primary" @click="addBtn" :icon='Plus'>新增</el-
button>
            </el-form-item>
        </el-form>

        <!-- 表格 -->
        <el-table
            :height='tableHeight'
            :data="menuTable.list"
            style="width: 100%"
            row-key="id"
            border
            default-expand-all
            :tree-props="{ children: 'children', hasChildren: 'hasChildren' }"
        >
            <el-table-column prop="label" label="菜单名称" />
            <el-table-column prop="name" label="类型">
                <template #default="scope">
                    <el-tag v-if="scope.row.type == '0'">目录</el-tag>
                    <el-tag type="success" v-if="scope.row.type == '1'">菜单</el-
tag>
                    <el-tag type="danger" v-if="scope.row.type == '2'">按钮</el-
tag>
                </template>
            </el-table-column>
            <el-table-column prop="name" label="图标">
                <template #default="scope">
                    <Icon class="icons" :icon="scope.row.icon"></Icon>
                </template>
            </el-table-column>
            <el-table-column prop="name" label="路由名称" />
            <el-table-column prop="path" label="路由地址" />
            <el-table-column prop="url" label="组件路径" />
            <el-table-column label="操作" align="center" width="200">
```

```
                <el-button type="primary" size="mini" @click="editBtn"
:icon="Edit">新增</el-button>
                <el-button type="danger" size="mini" @click="deleteBtn"
:icon="Delete">删除</el-button>
            </el-table-column>
        </el-table>
    </el-main>
    <!-- 新增、编辑弹框 -->
    <AddMenu ref='addMenuRef'></AddMenu>
</template>
<script setup lang="ts">
import AddMenu from './AddMenu.vue';
import { ref,onMounted,nextTick } from 'vue';
import { Edit, Delete,Plus } from '@element-plus/icons';
import useMenuTable from '@/composables/menu/useMenuTable';
import useMenu from '@/composables/menu/useMenu';
//表格高度
const tableHeight = ref(0);
//表格数据获取
const { menuTable, getMenuTable } = useMenuTable()

//表格的操作
const { addBtn, editBtn, deleteBtn, save,addMenuRef } = useMenu()

onMounted(() =>{
    nextTick(() =>{
        tableHeight.value = window.innerHeight - 200
    })
})
</script>
<style lang="scss" scoped>
.icons {
    width: 24px;
    height: 18px;
    margin-right: 5px;
}
</style>
```

第52 新增权限页面布局

1、新增表单数据类型定义

```
export interface AddMenuModel {
    id: number;
    editType: string;
    type: string;
    parentId: string | number;
    parentName: string;
    label: string;
    icon: string;
    name: string;
    path: string;
    url: string;
    code: string;
    orderNum: string | number;
}
```

2、AddMenu.vue组件

```
<template>
    <SysDialog
        :title="dialog.title"
        :width="dialog.width"
        :height="dialog.height"
        :visible="dialog.visible"
        @onConfirm="confrim"
        @onClose="onClose"
    >
        <template v-slot:content>
            <el-form
                :model="addMenuModel"
                ref="addMenuForm"
                :rules="rules"
                label-width="80px"
                size="mini"
            >
                <el-row>
                    <el-col :span="24" :offset="0">
                        <el-form-item prop='type' label="菜单类型">
                            <el-radio-group v-model="addMenuModel.type">
                                <el-radio :label="'0'">目录</el-radio>
                                <el-radio :label="'1'">菜单</el-radio>
                                <el-radio :label="'2'">按钮</el-radio>
                            </el-radio-group>
                        </el-form-item>
                    </el-col>
                </el-row>
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop='parentName' label="上级菜单">
                            <el-input type="hidden" v-
model="addMenuModel.parentId"></el-input>
                            <el-input v-model="addMenuModel.parentName"></el-
input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item prop='label' label="菜单名称">
                            <el-input v-model="addMenuModel.label"></el-input>
                        </el-form-item>
                    </el-col>
                </el-row>
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="菜单图标">
                            <el-input  v-model="addMenuModel.icon"></el-input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="路由名称">
                            <el-input v-model="addMenuModel.name"></el-input>
                        </el-form-item>
                    </el-col>
                </el-row>
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="路由地址">
                            <el-input  v-model="addMenuModel.path"></el-input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="组件路径">
                            <el-input v-model="addMenuModel.url"></el-input>
                        </el-form-item>
                    </el-col>
                </el-row>
                <el-row>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="权限字段">
                            <el-input v-model="addMenuModel.code"></el-input>
                        </el-form-item>
                    </el-col>
                    <el-col :span="12" :offset="0">
                        <el-form-item label="菜单序号">
                            <el-input v-model="addMenuModel.orderNum"></el-
input>
                        </el-form-item>
                    </el-col>
                </el-row>
            </el-form>
        </template>
    </SysDialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import useAddMenu from '@/composables/menu/useAddMenu';
//注册事件
const emit = defineEmits(['save'])

//弹框属性
const { dialog, onClose, onShow } = useDialog()

const { confrim, show, addMenuModel, rules,addMenuForm } = useAddMenu(dialog,
onClose, onShow,emit)


//暴露出去
defineExpose({
    show
})
</script>
<style scoped lang='scss'>
</style>
```

第53讲 上级菜单选择组件

1、上级菜单接口参数说明

1.1、接口请求地址

```
{baseURL}/api/menu/parent
```

1.2、请求方式

```
GET
```

1.3、请求参数

```
无
```

1.4、响应数据格式

```
{
    "msg": "查询成功",
    "code": 200,
    "data": [
        {
            "id": 0,
            "parentId": -1,
            "parentName": null,
            "label": "顶级菜单",
            "code": null,
            "path": null,
            "name": null,
            "url": null,
            "orderNum": null,
            "type": null,
            "icon": null,
            "remark": null,
            "createTime": null,
            "updateTime": null,
            "children": [
                {
                    "id": 17,
                    "parentId": 0,
                    "parentName": "顶级菜单",
                    "label": "系统管理",
                    "code": "sys:manage",
                    "path": "/system",
                    "name": "system",
                    "url": "/system/system",
                    "orderNum": 1,
                    "type": "0",
                    "icon": "el-icon-menu",
                    "remark": null,
                    "createTime": "2023-08-08 11:11:11",
                    "updateTime": "2023-08-09 15:26:28",
                    "children": [
                        {
                            "id": 33,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "机构管理",
            "code": "sys:dept",
            "path": "/department",
            "name": "department",
            "url": "/system/department/department",
            "orderNum": 2,
            "type": "1",
            "icon": "el-icon-document",
            "remark": "机构管理",
            "createTime": "2020-04-12 22:58:29",
            "updateTime": "2020-04-08 17:12:19",
            "children": [ ],
            "value": null,
            "open": null
        },
        {
            "id": 18,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "用户管理",
            "code": "sys:user",
            "path": "/userList",
            "name": "userList",
            "url": "/system/User/UserList",
            "orderNum": 3,
            "type": "1",
            "icon": "el-icon-s-custom",
            "remark": null,
            "createTime": "2023-08-08 11:11:11",
            "updateTime": "2023-08-09 15:26:28",
            "children": [ ],
            "value": null,
            "open": null
        },
        {
            "id": 23,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "角色管理",
            "code": "sys:role",
            "path": "/roleList",
            "name": "roleList",
            "url": "/system/Role/RoleList",
            "orderNum": 4,
            "type": "1",
            "icon": "el-icon-s-tools",
            "remark": null,
            "createTime": "2023-08-08 11:11:11",
            "updateTime": "2023-08-09 15:26:28",
            "children": [ ],
            "value": null,
            "open": null
        },
        {
            "id": 28,
            "parentId": 17,
                            "parentName": "系统管理",
                            "label": "权限管理",
                            "code": "sys:menu",
                            "path": "/menuList",
                            "name": "menuList",
                            "url": "/system/Menu/MenuList",
                            "orderNum": 5,
                            "type": "1",
                            "icon": "el-icon-document",
                            "remark": null,
                            "createTime": "2023-08-08 11:11:11",
                            "updateTime": "2023-08-09 15:26:28",
                            "children": [ ],
                            "value": null,
                            "open": null
                        }
                    ],
                    "value": null,
                    "open": null
                },
                {
                    "id": 34,
                    "parentId": 0,
                    "parentName": "顶级菜单",
                    "label": "商品管理",
                    "code": "sys:goods",
                    "path": "/goods",
                    "name": "goods",
                    "url": "/goods/index",
                    "orderNum": 2,
                    "type": "0",
                    "icon": "el-icon-document",
                    "remark": null,
                    "createTime": "2020-04-12 22:49:47",
                    "updateTime": "2020-04-12 17:22:03",
                    "children": [
                        {
                            "id": 36,
                            "parentId": 34,
                            "parentName": "商品管理",
                            "label": "分类管理",
                            "code": "sys:goodsCategory",
                            "path": "/goodCategory",
                            "name": "goodCategory",
                            "url": "/goods/goodsCategory/goodsCategoryList",
                            "orderNum": 1,
                            "type": "1",
                            "icon": "el-icon-document",
                            "remark": null,
                            "createTime": "2020-04-12 22:54:32",
                            "updateTime": "2020-04-12 17:26:30",
                            "children": [ ],
                            "value": null,
                            "open": null
                        },
                        {
                            "id": 37,
                            "parentId": 34,
                    "parentName": null,
                    "label": "品牌管理",
                    "code": "sys:goodsBrand",
                    "path": "/goodsBrand",
                    "name": "goodsBrand",
                    "url": "/goods/goodsBrand/goodsBrandList",
                    "orderNum": 2,
                    "type": "1",
                    "icon": "el-icon-document",
                    "remark": null,
                    "createTime": "2020-04-12 17:32:04",
                    "updateTime": "2020-04-12 17:32:04",
                    "children": [ ],
                    "value": null,
                    "open": null
                }
            ],
            "value": null,
            "open": null
        },
        {
            "id": 42,
            "parentId": 0,
            "parentName": "顶级菜单",
            "label": "系统工具",
            "code": "sys:systenConfig",
            "path": "/systenConfig",
            "name": "systenConfig",
            "url": "/systenConfig",
            "orderNum": 3,
            "type": "0",
            "icon": "el-icon-document",
            "remark": null,
            "createTime": "2020-04-12 22:50:03",
            "updateTime": "2020-04-12 17:40:41",
            "children": [
                {
                    "id": 43,
                    "parentId": 42,
                    "parentName": "系统工具",
                    "label": "日志管理",
                    "code": "sys:systemCode",
                    "path": "/systemCode",
                    "name": "systemCode",
                    "url": "/system/config/code",
                    "orderNum": 0,
                    "type": "1",
                    "icon": "el-icon-document",
                    "remark": null,
                    "createTime": "2020-04-16 12:44:42",
                    "updateTime": "2020-04-12 17:44:06",
                    "children": [ ],
                    "value": null,
                    "open": null
                },
                {
                    "id": 77,
                    "parentId": 42,
                            "parentName": "系统工具",
                            "label": "接口文档",
                            "code": "sys:document",
                            "path": "/document",
                            "name": "http://42.193.158.170:8089/swagger-
ui/index.html",
                            "url": "/system/config/systemDocument",
                            "orderNum": 0,
                            "type": "1",
                            "icon": "el-icon-document",
                            "remark": null,
                            "createTime": "2020-04-13 11:31:45",
                            "updateTime": "2020-04-13 11:31:45",
                            "children": [ ],
                            "value": null,
                            "open": null
                        },
                        {
                            "id": 93,
                            "parentId": 42,
                            "parentName": "系统工具",
                            "label": "测试菜单",
                            "code": "sys:document:tests",
                            "path": "/test",
                            "name": "test",
                            "url": "/system/config/test",
                            "orderNum": 2,
                            "type": "1",
                            "icon": "el-icon-s-order",
                            "remark": null,
                            "createTime": "2021-03-19 14:17:02",
                            "updateTime": "2021-03-19 14:17:02",
                            "children": [ ],
                            "value": null,
                            "open": null
                        }
                    ],
                    "value": null,
                    "open": null
                }
            ],
            "value": null,
            "open": null
        }
    ]
}
```

第54讲 新增、编辑权限接口对接

1、新增接口参数

1.1、接口请求地址

```
{baseURL}/api/menu
```

1.2、请求方式

```
POST
```

1.3、请求参数

名称	类型	必填	备注
parentId	string	是	上级菜单id
parentName	string	是	上级菜单名称
label	string	是	菜单名称
code	string	否	权限标识
path	string	否	路由地址
name	string	否	路由名称
url	string	否	组件路径
orderNum	number	否	序号
type	string	是	菜单类型： 0 目录 1 菜单 2 按钮
icon	string	否	图标

1.4、响应参数格式

```
{
    "msg":"新增成功!",
    "code":200,
    "data":null
}
```

2、编辑接口参数

2.1、接口请求地址

```
{baseURL}/api/menu
```

2.2、请求方式

```
PUT
```

2.3、请求参数

名称	类型	必填	备注
id	string	是	菜单id
parentId	string	是	上级菜单id
parentName	string	是	上级菜单名称
label	string	是	菜单名称
code	string	否	权限标识
path	string	否	路由地址
name	string	否	路由名称
url	string	否	组件路径
orderNum	number	否	序号
type	string	是	菜单类型： 0 目录 1 菜单 2 按钮
icon	string	否	图标

2.4、响应参数格式

```
{
    "msg":"编辑成功!",
    "code":200,
    "data":null
}
```

3、新增表单验证规则

```
const rules = reactive({
    type: [
          {
            required: true,
            trigger: "change",
            message: "请选择菜单类型",
          },
        ],
        parentName: [
          {
            required: true,
            trigger: "change",
            message: "请选择上级菜单",
          },
        ],
        label: [
          {
            required: true,
            trigger: "change",
            message: "请填写菜单名称",
          },
        ],
        name: [
          {
            required: true,
            trigger: "change",
            message: "请填写路由名称",
          },
        ],
        path: [
          {
            required: true,
            trigger: "change",
            message: "请填写路由路径",
          },
        ],
        url: [
          {
            required: true,
            trigger: "change",
            message: "请填写组件路径",
          },
        ],
        code: [
          {
            required: true,
            trigger: "change",
            message: "请填写权限字段",
          },
        ]
})
```

第55讲 删除权限接口对接

1、删除接口参数说明

1.1、接口请求地址

```
{baseURL}/api/menu
```

1.2、请求方式

```
DELETE
```

1.3、请求参数

名称	类型	必填	备注
id	string	是	权限id

1.4、响应参数格式

```
{
    "msg":"删除成功!",
    "code":200,
    "data":null
}
```

第56讲 用户分配角色

1、在views/system/user下新建AssignRole.vue组件

```
<template>
    <SysDialog
        :title="dialog.title"
        :width="dialog.width"
        :visible="dialog.visible"
        :height="dialog.height"
        @onClose="onClose"
        @onConfirm="confirm"
    >
        <template v-slot:content>
            <el-table :height='tableHeight' size="mini"
:data="assignRoleList.list" border stripe>
                <el-table-column width="50" align="center" label="选中">
                    <template #default="scope">
                        <el-radio
                            v-model="selectRoleId"
                            :label="scope.row.id"
                            @change="getSlectRole(scope.row)"
                        >{{ "" }}</el-radio>
                    </template>
                </el-table-column>
                <el-table-column prop="name" label="角色名称"></el-table-column>
            </el-table>
            <!-- 分页 -->
            <el-pagination
                @size-change="sizeChange"
                @current-change="currentChange"
                :current-page.sync="parms.currentPage"
                :page-sizes="[10, 20, 40, 80, 100]"
                :page-size="parms.pageSize"
                layout="total, sizes, prev, pager, next, jumper"
                :total="parms.total"
                background
            ></el-pagination>
        </template>
    </SysDialog>
</template>
<script setup lang='ts'>
import { ref,onMounted,nextTick } from 'vue';
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import useAssignRole from '@/composables/user/useAssignRole';
//表格高度
```

```
const tableHeight = ref(0)
//弹框属性
const { dialog, onShow, onClose } = useDialog()

const { show, confirm, parms, assignRoleList, selectRoleId, getSlectRole,
sizeChange, currentChange } = useAssignRole(dialog, onShow, onClose)

//暴露方法
defineExpose({
    show
})

onMounted(() =>{
    nextTick(() =>{
        tableHeight.value = window.innerHeight - 690
    })
})
</script>
<style scoped lang='scss'>
</style>
```

2、在composables/user下新建useAssignRole.ts

```
import { DialogModel } from '@/type/BastType'
import { reactive, onMounted, ref } from 'vue';
import { assignRoleListApi, getRoleIdApi, assingRoleSaveApi } from
'@/api/user/user';
import { getUserId } from '@/utils/auth';
import { AssignRoleListParm } from '@/api/user/UserModel';
import useInstance from '@/hooks/useInstance';
export default function useAssignRole(dialog: DialogModel, onShow, onClose) {
    const { global } = useInstance()
    //选中用户的id
    const selectUserId = ref<string | number>('')
    //选中的角色id
    const selectRoleId = ref('');
    //表格数据
    const assignRoleList = reactive({
        list: []
    })
    //表格查询参数
    const parms = reactive<AssignRoleListParm>({
        userId: getUserId() || '',
        currentPage: 1,
        pageSize: 4,
        total: 0
    })
    //显示弹框
    const show = async (name: string, userId: number | number) => {
        selectRoleId.value = ''
        selectUserId.value = userId
        const res = await getRoleIdApi(userId)
        if (res.data) {
            selectRoleId.value = res.data.roleId
        }
        dialog.title = '为【' + name + '】分配角色';
        onShow();
    }

    //确定
    const confirm = async () => {
        if (!selectRoleId.value) {
            global.$message({ message: '请选择角色', type: 'warning' })
            return;
        }
        const res = await assingRoleSaveApi({ roleId: selectRoleId.value,
userId: selectUserId.value })
        if (res && res.code == 200) {
            global.$message({ message: res.msg, type: 'success' })
            onClose();
        }
    }
    //获取角色数据
    const getAssignList = async () => {
        let res = await assignRoleListApi(parms)
        if (res && res.code == 200) {
            assignRoleList.list = res.data.records;
            parms.total = res.data.total;
        }
    }
    //表格单选点击事件
    const getSlectRole = (row: any) => {
        selectRoleId.value = row.id
    }
    //页容量改变时触发
    const sizeChange = (size: number) => {
        parms.pageSize = size;
        getAssignList();
    }
    //页数改变时触发
    const currentChange = (page: number) => {
        parms.currentPage = page;
        getAssignList();
    }
    onMounted(() => {
        getAssignList()
    })
    return {
        show,
        confirm,
        assignRoleList,
        parms,
        selectRoleId,
        getSlectRole,
        sizeChange,
        currentChange
    }
}
```

3、在src/api/user/user.ts新建如下方法

```
enum Api{
    assignRoleList ='/api/user/getRolistForAssign',
    getRoleId = '/api/user/getRoleIdByUserId',
    assignSave = '/api/user/assingRole'
}

//分配角色列表
export const assignRoleListApi = async(parm:AssignRoleListParm) =>{
    return await http.get(Api.assignRoleList,parm)
}
//被分配角色的用户id
export const getRoleIdApi = async(userId:string | number) =>{
    return await http.getRestApi(Api.getRoleId,{userId:userId})
}
//分配角色保存
export const assingRoleSaveApi = async(parm) =>{
    return await http.post(Api.assignSave,parm)
}
```

4、UserList.vue引入AssignRole.vue组件

5、AssignRoleListParm

```
/**
 * 分配角色列表参数
 */
export interface AssignRoleListParm {
    currentPage: number,
    pageSize: number,
    userId: string | number,
    total: number,
}
```

6、接口参数

6.1、弹框角色列表

6.1.1、接口地址

```
{baseURL}/api/user/getRolistForAssign
```

6.1.2、请求方式

```
GET
```

6.1.3、请求参数

名称	类型	必填	备注
currentPage	number	是	当前页
pageSize	number	是	页容量
userId	number	是	当前登录系统的用户id
total	number	是	总条数

6.1.4、返回示例

```
{
    "msg": "成功",
    "code": 200,
    "data": {
        "records": [
            {
                "id": 9,
                "createUser": 9,
                "name": "系统管理员",
                "remark": "系统管理员",
                "createTime": "2020-05-19T23:51:28.000+00:00",
                "updateTime": "2020-05-19T23:51:28.000+00:00"
            },
            {
                "id": 10,
                "createUser": 9,
                "name": "超级管理员",
                "remark": "超级管理员",
                "createTime": "2023-08-08T03:11:11.000+00:00",
                "updateTime": "2023-08-08T03:11:11.000+00:00"
            },
            {
                "id": 13,
                "createUser": 9,
                "name": "销售管理员",
                "remark": "管理销售人员",
                "createTime": "2020-04-14T04:22:53.000+00:00",
                "updateTime": "2020-04-14T04:22:53.000+00:00"
            },
            {
                "id": 14,
                "createUser": 9,
                "name": "财务管理员",
                "remark": "管理公司财务",
                "createTime": "2020-04-14T04:23:10.000+00:00",
                "updateTime": "2020-04-14T04:23:10.000+00:00"
            }
        ],
        "total": 5,
        "size": 4,
        "current": 1,
        "orders": [ ],
        "optimizeCountSql": true,
        "hitCount": false,
        "countId": null,
        "maxLimit": null,
        "searchCount": true,
        "pages": 2
    }
}
```

6.2、被分配用户角色id

6.2.1、接口地址

```
{baseURL}/api/user/getRoleIdByUserId
```

6.2.2、请求方式（**注意：使用restfulapi格式的get请求**）

```
GET
```

6.2.3、请求参数

名称	类型	必填	备注
userId	number	是	被分配角色的用户的id

6.2.4、返回示例

```
{
    "msg": "成功",
    "code": 200,
    "data": {
        "id": 1,
        "userId": 9,
        "roleId": 9
    }
}
```

6.3、分配角色保存

6.3.1、接口地址

```
{baseURL}/api/user/assingRole
```

6.3.2、请求方式

```
post
```

6.3.3、请求参数

名称	类型	必填	备注
roleId	number	是	选择的角色的id
userId	number	是	被分配角色的用户的id

6.3.4、返回示例

```
{
    "msg": "成功",
    "code": 200,
    "data": null
}
```

第57讲 分配权限树显示制作

1、权限树接口参数

1.1、接口请求地址

```
{baseURL}/api/role/getAssignPermissionTree
```

1.2、请求方式

```
GET
```

1.3、请求参数

名称	类型	必填	备注
userId	string	是	当前登录系统的用户id
roleId	string	是	被分配角色的id

1.4、响应参数格式

名称	类型	备注
listmenu	array	权限树数据
checkList	array	角色原来拥有的权限id集合

```
{
    "msg": "成功",
    "code": 200,
    "data": {
        "listmenu": [
              {
                  "id": 17,
                  "parentId": 0,
                  "parentName": "顶级菜单",
                  "label": "系统管理",
                  "code": "sys:manage",
                  "path": "/system",
                  "name": "system",
                  "url": "/system/system",
                  "orderNum": 1,
                  "type": "0",
                  "icon": "Setting",
                  "remark": null,
                  "createTime": "2023-08-08T03:11:11.000+00:00",
                  "updateTime": "2023-08-09T07:26:28.000+00:00",
                  "children": [
                      {
                          "id": 33,
                          "parentId": 17,
                          "parentName": "系统管理",
                          "label": "机构管理",
                          "code": "sys:dept",
                          "path": "/department",
                          "name": "department",
                          "url": "/system/department/department",
                          "orderNum": 2,
                          "type": "1",
                          "icon": "List",
                          "remark": "机构管理",
                          "createTime": "2020-04-12T14:58:29.000+00:00",
                          "updateTime": "2020-04-08T09:12:19.000+00:00",
                          "children": [
                              {
                                  "id": 46,
                                  "parentId": 33,
                                  "parentName": "机构管理",
                                  "label": "新增",
                                  "code": "sys:addDepartment",
                                  "path": "",
                                  "name": "",
                                  "url": null,
                                  "orderNum": 0,
                                  "type": "2",
                                  "icon": "el-icon-document",
                                  "remark": null,
                                  "createTime": "2020-04-12T11:58:48.000+00:00",
                                  "updateTime": "2020-04-12T11:58:48.000+00:00",
                                  "children": [ ],
                                  "value": null,
                                  "open": null
                              },
                              {
                                  "id": 76,
                                  "parentId": 33,
                                  "parentName": "机构管理",
                                  "label": "编辑",
                                  "code": "sys:editDept",
                                  "path": "",
                    "name": "",
                    "url": null,
                    "orderNum": 1,
                    "type": "2",
                    "icon": "el-icon-document",
                    "remark": null,
                    "createTime": "2020-04-12T12:42:20.000+00:00",
                    "updateTime": "2020-04-12T12:42:20.000+00:00",
                    "children": [ ],
                    "value": null,
                    "open": null
                },
                {
                    "id": 78,
                    "parentId": 33,
                    "parentName": "机构管理",
                    "label": "删除",
                    "code": "sys:deleteDept",
                    "path": "",
                    "name": "",
                    "url": "",
                    "orderNum": 3,
                    "type": "2",
                    "icon": "el-icon-document",
                    "remark": null,
                    "createTime": "2020-04-18T02:25:55.000+00:00",
                    "updateTime": "2020-04-18T02:25:55.000+00:00",
                    "children": [ ],
                    "value": null,
                    "open": null
                }
            ],
            "value": null,
            "open": null
        },
        {
            "id": 18,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "用户管理",
            "code": "sys:user",
            "path": "/userList",
            "name": "userList",
            "url": "/system/User/UserList",
            "orderNum": 3,
            "type": "1",
            "icon": "UserFilled",
            "remark": null,
            "createTime": "2023-08-08T03:11:11.000+00:00",
            "updateTime": "2023-08-09T07:26:28.000+00:00",
            "children": [
                {
                    "id": 20,
                    "parentId": 18,
                    "parentName": null,
                    "label": "新增",
                    "code": "sys:user:add",
                    "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "User",
                "remark": "新增用户",
                "createTime": "2023-08-08T03:11:11.000+00:00",
                "updateTime": "2023-08-09T07:26:28.000+00:00",
                "children": [ ],
                "value": null,
                "open": null
            },
            {
                "id": 21,
                "parentId": 18,
                "parentName": null,
                "label": "修改",
                "code": "sys:user:edit",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "修改用户",
                "createTime": "2023-08-08T03:11:11.000+00:00",
                "updateTime": "2023-08-09T07:26:28.000+00:00",
                "children": [ ],
                "value": null,
                "open": null
            },
            {
                "id": 22,
                "parentId": 18,
                "parentName": null,
                "label": "删除",
                "code": "sys:user:delete",
                "path": null,
                "name": null,
                "url": "",
                "orderNum": null,
                "type": "2",
                "icon": "el-icon-document",
                "remark": "删除用户",
                "createTime": "2023-08-08T03:11:11.000+00:00",
                "updateTime": "2023-08-09T07:26:28.000+00:00",
                "children": [ ],
                "value": null,
                "open": null
            },
            {
                "id": 80,
                "parentId": 18,
                "parentName": "用户管理",
                "label": "分配角色",
                "code": "sys:user:assign",
                "path": "",
                "name": "",
                    "url": "",
                    "orderNum": 0,
                    "type": "2",
                    "icon": "el-icon-document",
                    "remark": null,
                    "createTime": "2020-04-18T02:50:14.000+00:00",
                    "updateTime": "2020-04-18T02:50:14.000+00:00",
                    "children": [ ],
                    "value": null,
                    "open": null
                }
            ],
            "value": null,
            "open": null
        },
        {
            "id": 23,
            "parentId": 17,
            "parentName": "系统管理",
            "label": "角色管理",
            "code": "sys:role",
            "path": "/roleList",
            "name": "roleList",
            "url": "/system/Role/RoleList",
            "orderNum": 4,
            "type": "1",
            "icon": "Wallet",
            "remark": null,
            "createTime": "2023-08-08T03:11:11.000+00:00",
            "updateTime": "2023-08-09T07:26:28.000+00:00",
            "children": [
                {
                    "id": 25,
                    "parentId": 23,
                    "parentName": null,
                    "label": "新增",
                    "code": "sys:role:add",
                    "path": null,
                    "name": null,
                    "url": "",
                    "orderNum": null,
                    "type": "2",
                    "icon": "el-icon-document",
                    "remark": "新增角色",
                    "createTime": "2023-08-08T03:11:11.000+00:00",
                    "updateTime": "2023-08-09T07:26:28.000+00:00",
                    "children": [ ],
                    "value": null,
                    "open": null
                },
                {
                    "id": 26,
                    "parentId": 23,
                    "parentName": null,
                    "label": "修改",
                    "code": "sys:role:edit",
                    "path": null,
                    "name": null,
```

```json
                                "url": "",
                                "orderNum": null,
                                "type": "2",
                                "icon": "el-icon-document",
                                "remark": "修改角色",
                                "createTime": "2023-08-08T03:11:11.000+00:00",
                                "updateTime": "2023-08-09T07:26:28.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            },
                            {
                                "id": 27,
                                "parentId": 23,
                                "parentName": null,
                                "label": "删除",
                                "code": "sys:role:delete",
                                "path": null,
                                "name": null,
                                "url": "",
                                "orderNum": null,
                                "type": "2",
                                "icon": "el-icon-document",
                                "remark": "删除角色",
                                "createTime": "2023-08-08T03:11:11.000+00:00",
                                "updateTime": "2023-08-09T07:26:28.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            },
                            {
                                "id": 79,
                                "parentId": 23,
                                "parentName": "角色管理",
                                "label": "分配权限",
                                "code": "sys:role:assign",
                                "path": "",
                                "name": "",
                                "url": "",
                                "orderNum": 0,
                                "type": "2",
                                "icon": "el-icon-document",
                                "remark": null,
                                "createTime": "2020-04-18T02:31:05.000+00:00",
                                "updateTime": "2020-04-18T02:31:05.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            }
                        ],
                        "value": null,
                        "open": null
                    },
                    {
                        "id": 28,
                        "parentId": 17,
                        "parentName": "系统管理",
                        "label": "权限管理",
                    "code": "sys:menu",
                    "path": "/menuList",
                    "name": "menuList",
                    "url": "/system/Menu/MenuList",
                    "orderNum": 5,
                    "type": "1",
                    "icon": "Menu",
                    "remark": null,
                    "createTime": "2023-08-08T03:11:11.000+00:00",
                    "updateTime": "2023-08-09T07:26:28.000+00:00",
                    "children": [
                        {
                            "id": 30,
                            "parentId": 28,
                            "parentName": null,
                            "label": "新增",
                            "code": "sys:menu:add",
                            "path": null,
                            "name": null,
                            "url": "",
                            "orderNum": null,
                            "type": "2",
                            "icon": "el-icon-document",
                            "remark": "新增权限",
                            "createTime": "2023-08-08T03:11:11.000+00:00",
                            "updateTime": "2023-08-09T07:26:28.000+00:00",
                            "children": [ ],
                            "value": null,
                            "open": null
                        },
                        {
                            "id": 31,
                            "parentId": 28,
                            "parentName": null,
                            "label": "修改",
                            "code": "sys:menu:edit",
                            "path": null,
                            "name": null,
                            "url": "",
                            "orderNum": null,
                            "type": "2",
                            "icon": "el-icon-document",
                            "remark": "修改权限",
                            "createTime": "2023-08-08T03:11:11.000+00:00",
                            "updateTime": "2023-08-09T07:26:28.000+00:00",
                            "children": [ ],
                            "value": null,
                            "open": null
                        },
                        {
                            "id": 32,
                            "parentId": 28,
                            "parentName": null,
                            "label": "删除",
                            "code": "sys:menu:delete",
                            "path": null,
                            "name": null,
                            "url": "",
                                "orderNum": null,
                                "type": "2",
                                "icon": "el-icon-document",
                                "remark": "删除权限",
                                "createTime": "2023-08-08T03:11:11.000+00:00",
                                "updateTime": "2023-08-09T07:26:28.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            }
                        ],
                        "value": null,
                        "open": null
                    }
                ],
                "value": null,
                "open": null
            },
            {
                "id": 34,
                "parentId": 0,
                "parentName": "顶级菜单",
                "label": "商品管理",
                "code": "sys:goods",
                "path": "/goods",
                "name": "goods",
                "url": "/goods/index",
                "orderNum": 2,
                "type": "0",
                "icon": "Tickets",
                "remark": null,
                "createTime": "2020-04-12T14:49:47.000+00:00",
                "updateTime": "2020-04-12T09:22:03.000+00:00",
                "children": [
                    {
                        "id": 36,
                        "parentId": 34,
                        "parentName": "商品管理",
                        "label": "分类管理",
                        "code": "sys:goodsCategory",
                        "path": "/goodCategory",
                        "name": "goodCategory",
                        "url": "/goods/goodsCategory/goodsCategoryList",
                        "orderNum": 1,
                        "type": "1",
                        "icon": "ShoppingBag",
                        "remark": null,
                        "createTime": "2020-04-12T14:54:32.000+00:00",
                        "updateTime": "2020-04-12T09:26:30.000+00:00",
                        "children": [
                            {
                                "id": 38,
                                "parentId": 36,
                                "parentName": "分类管理",
                                "label": "新增",
                                "code": "sys:addGoodsCategory",
                                "path": "",
                                "name": "",
                        "url": null,
                        "orderNum": 0,
                        "type": "2",
                        "icon": "el-icon-document",
                        "remark": null,
                        "createTime": "2020-04-12T09:33:58.000+00:00",
                        "updateTime": "2020-04-12T09:33:58.000+00:00",
                        "children": [ ],
                        "value": null,
                        "open": null
                    },
                    {
                        "id": 39,
                        "parentId": 36,
                        "parentName": "分类管理",
                        "label": "编辑",
                        "code": "sys:editGoodsCategory",
                        "path": "",
                        "name": "",
                        "url": null,
                        "orderNum": 1,
                        "type": "2",
                        "icon": "el-icon-document",
                        "remark": null,
                        "createTime": "2020-04-12T09:35:30.000+00:00",
                        "updateTime": "2020-04-12T09:35:30.000+00:00",
                        "children": [ ],
                        "value": null,
                        "open": null
                    }
                ],
                "value": null,
                "open": null
            },
            {
                "id": 37,
                "parentId": 34,
                "parentName": null,
                "label": "品牌管理",
                "code": "sys:goodsBrand",
                "path": "/goodsBrand",
                "name": "goodsBrand",
                "url": "/goods/goodsBrand/goodsBrandList",
                "orderNum": 2,
                "type": "1",
                "icon": "Management",
                "remark": null,
                "createTime": "2020-04-12T09:32:04.000+00:00",
                "updateTime": "2020-04-12T09:32:04.000+00:00",
                "children": [
                    {
                        "id": 87,
                        "parentId": 37,
                        "parentName": "品牌管理",
                        "label": "新增",
                        "code": "sys:brand:add",
                        "path": "",
                        "name": "",
                                "url": "",
                                "orderNum": 0,
                                "type": "2",
                                "icon": "",
                                "remark": null,
                                "createTime": "2021-02-03T01:08:24.000+00:00",
                                "updateTime": "2021-02-03T01:08:24.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            },
                            {
                                "id": 88,
                                "parentId": 37,
                                "parentName": "品牌管理",
                                "label": "编辑",
                                "code": "sys:brand:edit",
                                "path": "",
                                "name": "",
                                "url": "",
                                "orderNum": 1,
                                "type": "2",
                                "icon": "",
                                "remark": null,
                                "createTime": "2021-02-03T01:08:50.000+00:00",
                                "updateTime": "2021-02-03T01:08:50.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            },
                            {
                                "id": 89,
                                "parentId": 37,
                                "parentName": "品牌管理",
                                "label": "删除",
                                "code": "sys:brand:delete",
                                "path": "",
                                "name": "",
                                "url": "",
                                "orderNum": 2,
                                "type": "2",
                                "icon": "",
                                "remark": null,
                                "createTime": "2021-02-03T01:09:18.000+00:00",
                                "updateTime": "2021-02-03T01:09:18.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            }
                        ],
                        "value": null,
                        "open": null
                    }
                ],
                "value": null,
                "open": null
            },
            {
                "id": 42,
                "parentId": 0,
                "parentName": "顶级菜单",
                "label": "系统工具",
                "code": "sys:systenConfig",
                "path": "/systenConfig",
                "name": "systenConfig",
                "url": "/systenConfig",
                "orderNum": 3,
                "type": "0",
                "icon": "ChatLineSquare",
                "remark": null,
                "createTime": "2020-04-12T14:50:03.000+00:00",
                "updateTime": "2020-04-12T09:40:41.000+00:00",
                "children": [
                    {
                        "id": 43,
                        "parentId": 42,
                        "parentName": "系统工具",
                        "label": "日志管理",
                        "code": "sys:systemCode",
                        "path": "/systemCode",
                        "name": "systemCode",
                        "url": "/system/config/code",
                        "orderNum": 0,
                        "type": "1",
                        "icon": "Operation",
                        "remark": null,
                        "createTime": "2020-04-16T04:44:42.000+00:00",
                        "updateTime": "2020-04-12T09:44:06.000+00:00",
                        "children": [
                            {
                                "id": 85,
                                "parentId": 43,
                                "parentName": "代码生成",
                                "label": "新增",
                                "code": "sys:code:add",
                                "path": "",
                                "name": "",
                                "url": "",
                                "orderNum": 0,
                                "type": "2",
                                "icon": "",
                                "remark": null,
                                "createTime": "2021-02-02T14:05:31.000+00:00",
                                "updateTime": "2021-02-02T14:05:31.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            },
                            {
                                "id": 86,
                                "parentId": 43,
                                "parentName": "代码生成",
                                "label": "编辑",
                                "code": "sys:code:edit",
                                "path": "",
                                "name": "",
                                "url": "",
                                "orderNum": 1,
                                "type": "2",
                                "icon": "",
                                "remark": null,
                                "createTime": "2021-02-02T14:06:15.000+00:00",
                                "updateTime": "2021-02-02T14:06:15.000+00:00",
                                "children": [ ],
                                "value": null,
                                "open": null
                            }
                        ],
                        "value": null,
                        "open": null
                    },
                    {
                        "id": 77,
                        "parentId": 42,
                        "parentName": "系统工具",
                        "label": "接口文档",
                        "code": "sys:document",
                        "path": "/document",
                        "name": "http://42.193.158.170:8089/swagger-
ui/index.html",
                        "url": "/system/config/systemDocument",
                        "orderNum": 0,
                        "type": "1",
                        "icon": "DocumentCopy",
                        "remark": null,
                        "createTime": "2020-04-13T03:31:45.000+00:00",
                        "updateTime": "2020-04-13T03:31:45.000+00:00",
                        "children": [ ],
                        "value": null,
                        "open": null
                    }
                ],
                "value": null,
                "open": null
            }
        ],
        "checkList": [
            20,
            21,
            22,
            25,
            26,
            27,
            30,
            31,
            32,
            38,
            43,
            46,
            77,
            79,
            80,
            85,
            17,
            36,
            39,
            76,
            86,
            33,
            34,
            18,
            42,
            78,
            23,
            28
        ]
    }
}
```

2、新建AssignMenu.vue组件

```
<template>
    <SysDialog
        :title="dialog.title"
        :width="dialog.width"
        :visible="dialog.visible"
        :height="dialog.height"
        @onClose="onClose"
        @onConfirm="confirm"
    >
        <template v-slot:content>
            <el-tree
                ref="assigntree"
                :data="assignTreeData.list"
                node-key="id"
                :props="defaultProps"
                default-expand-all
                empty-text="暂无数据"
                :show-checkbox="true"
                :highlight-current="true"
            ></el-tree>
        </template>
    </SysDialog>
</template>
<script setup lang='ts'>
import SysDialog from '@/components/SysDialog.vue';
import useDialog from '@/hooks/useDialog';
import useAssignMenu from '@/composables/role/useAssignMenu';
//弹框属性
const { dialog, onShow, onClose } = useDialog();

const { confirm, show, assignTreeData, defaultProps } = useAssignMenu(dialog,
onShow, onClose)

//暴露方法
defineExpose({
    show
})
</script>
```

```
<style scoped lang='scss'>
</style>
```

3、src/composables/role下新建useAssignMenu.ts

```
import { DialogModel } from "@/type/BastType"
import { assignTreeApi } from "@/api/role/role";
import { AssignTreeeParm } from "@/api/role/RoleModel";
import { getUserId } from "@/utils/auth";
import { reactive } from "vue";
export default function useAssignMenu(dialog: DialogModel, onShow, onClose) {
    const defaultProps = reactive({
        children: 'children',
        label: 'label'
    })
    //树数据
    const assignTreeData = reactive({
        list:[],
        assignTreeChecked:[]
    })
    //确定
    const confirm = () => {
        onClose();
    }
    //显示弹框
    const show = (roleId: string, name: string) => {
        //获取弹框树数据
        let parm = { roleId: roleId, userId: getUserId() || ''}
        getAssignTree(parm);
        //设置弹属性
        dialog.width = 300
        dialog.height = 420
        dialog.title = '为【' + name + '】分配权限'
        onShow();
    }
    //获取权限树数据
    const getAssignTree = async (parm: AssignTreeeParm) => {
        let res = await assignTreeApi(parm);
        console.log('树数据加载完成')
        console.log(res)
        assignTreeData.list = res.data.listmenu
        assignTreeData.assignTreeChecked = res.data.checkList
    }
    return {
        confirm,
        show,
        assignTreeData,
        defaultProps
    }
}
```

4、role的api添加如下方法

```
//获取角色的权限树数据
export const assignTreeApi = async(parm:AssignTreeeParm) =>{
    return await http.get(Api.assignTree,parm)
}
```

第58讲 权限树回显与分配保存

1、权限分配保存接口参数

1.1、接口请求地址

```
{baseURL}/api/role/roleAssignSave
```

1.2、请求方式

```
POST
```

1.3、请求参数

名称	类型	必填	备注
roleId	string	是	被分配角色的id
list	array	是	角色的权限列表，是一个数组

1.4、响应参数格式

```
{
    "msg":"分配成功!",
    "code":200,
    "data":null
}
```

第59讲 退出登录、还原数据对接

1、src/layout/header下新建UserInfo.vue组件

```
<template>
    <div class="header-right">
        <el-dropdown>
            <span class="el-dropdown-link">
                <img class="userimg" src="@/assets/avatar.jpg" />
            </span>
            <template #dropdown>
                <el-dropdown-menu>
                    <el-dropdown-item @click="resetBtn">还原数据</el-dropdown-
item>
```

```
                    <el-dropdown-item @click="loginOutBtn">退出</el-dropdown-
item>
                </el-dropdown-menu>
            </template>
        </el-dropdown>
    </div>
</template>
<script setup lang='ts'>
import useInstance from '@/hooks/useInstance';
const { global } = useInstance();
import { loginOutApi, restoreApi } from '@/api/user/user'
import { getToken, cleanSession } from '@/utils/auth';
//退出登录
const loginOutBtn = async () => {
    let confirm = await global.$myconfirm('确定退出登录吗？')
    if (confirm) {
        let parm = {
            token: getToken()
        }
        let res = await loginOutApi(parm)
        if (res && res.code == 200) {
            //清空session
            cleanSession();
            //跳到登录
            window.location.href = "/login";
        }
    }
}
//还原数据
const resetBtn = async () => {
    let confirm = await global.$myconfirm('确定退出登录吗？')
    if (confirm) {
        let res = await restoreApi();
        if (res.code == 200) {
            global.$message.success(res.msg);
        }
    }
}
</script>
<style scoped lang='scss'>
.header-right {
    display: flex;
}

.userimg {
    height: 42px;
    width: 42px;
    border-radius: 50%;
}
</style>
```

2、Header.vue引入UserInfo.vue组件

```
<template>
    <div style="display: flex;align-items: center;">
        <Collapse></Collapse>
        <BredCum></BredCum>
    </div>
    <UserInfo></UserInfo>
</template>
<script setup lang="ts">
import Collapse from './Collapse.vue';
import BredCum from './BredCum.vue';
import UserInfo from './UserInfo.vue';
</script>
```

3、api/user.ts添加如下方法

```
enum Api{
    loginOut = '/api/sysUser/loginOut',
    restore = '/api/backup/restore'
}



//退出登录
export async function loginOutApi(parm) {
    return await http.post(Api.loginOut, parm)
}
//数据库还原
export async function restoreApi() {
    return await http.post(Api.restore, null)
}
```

第60讲 按钮权限讲解-自定义指令

1、解决sessionStorage中的tabsView不能清空的问题

1.1、修改src/layout/tabs.vue中的beforeRefresh()方法

```
//解决刷新数据丢失的问题,不是登录请求的时候，才缓存tabsView
const beforeRefresh = () => {
    if (route.path != '/login') {
        window.addEventListener('beforeunload', () => {
            sessionStorage.setItem('tabsView', JSON.stringify(tabsList.value))
        })
        let tabSesson = sessionStorage.getItem("tabsView");
        if (tabSesson) {
            let oldTabs = JSON.parse(tabSesson);
            if (oldTabs.length > 0) {
                store.state.tabs.tabsList = oldTabs;
            }
        }
    }
}
```

1.2、修改src/layout/header/UserInfo.vue 和 request.ts

把清空session放到跳转的后面

```
const loginOutBtn = async () => {
    let confirm = await global.$myconfirm('确定退出登录吗？')
    if (confirm) {
        let parm = {
            token: getToken()
        }
        let res = await loginOutApi(parm)
        if (res && res.code == 200) {
            //跳到登录
            window.location.href = "/login";
             //清空session
            cleanSession();
        }
    }
}
```

3、在store/modules/user模块的getters，添加getPermissions()

```
getPermissions(state:UserState){
        return state.roles
}
```

3、src下新建directives文件夹，然后新建permission.ts

```
import { Directive } from 'vue'
import { store } from '@/store/index'
export const permission: Directive = {
    mounted(el, binding) {
        const { value } = binding
        const permissions = store.getters['user/getPermissions']
        console.log(permissions)
        if (value && value instanceof Array && value.length > 0) {
            const permissionRoles = value
            const hasPermission = permissions.some((role: any) => {
                return permissionRoles.includes(role)
            })
            if (!hasPermission) {
                el.style.display = 'none'
            }
        } else {
            throw new Error('need roles! Like v-permission="
[\'add\',\'edit\']"')
        }
    }
}
```

4、main.ts注册指令

```
//按钮权限
import {permission} from '@/directives/permission'
app.directive('permission', permission)
```

5、页面使用permission指令

```
<el-button size="mini"  v-permission="['sys:addDepartment']"  type="primary"
:icon="Plus"  @click="addBtn">新增</el-button>
```

第61讲 项目整合echarts

1、安装

```
npm install echarts --save
```

2、main.ts引入全局挂载

```
//echarts
import * as echarts from 'echarts'

app.config.globalProperties.$echarts = echarts;
```

3、页面使用

```
<template>
  <el-main :style="{ height: mianHeight + 'px' }">
    <div style="display: flex;">
      <el-card style="flex: 1;">
        <template #header>
          <div class="card-header">
            <span>订单统计</span>
          </div>
        </template>
        <div ref="myChart" :style="{ width: '400px', height: '300px' }"></div>
      </el-card>
      <el-card style="margin-left: 20px;flex: 1;">
        <template #header>
          <div class="card-header">
            <span>订单统计</span>
          </div>
        </template>
        <div ref="myChart1" :style="{ width: '400px', height: '300px' }"></div>
      </el-card>
      <el-card style="margin-left: 20px;flex: 1;">
        <template #header>
          <div class="card-header">
            <span>订单统计</span>
          </div>
        </template>
        <div ref="myChart2" :style="{ width: '400px', height: '300px' }"></div>
      </el-card>
    </div>
  </el-main>
</template>
<script setup lang='ts'>
import { ref, nextTick, onMounted, reactive, onBeforeUnmount } from 'vue'
import useInstance from '@/hooks/useInstance';
const mianHeight = ref(0)
const { global } = useInstance()
const myChart = ref<HTMLElement>();
const myChart1 = ref<HTMLElement>();
const myChart2 = ref<HTMLElement>();

//柱状图
const charts1 = () => {
  const echartInstance = global.$echarts.init(myChart.value);
  let option = reactive({
    xAxis: {
      type: 'category',
      data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
    },
    yAxis: {
      type: 'value'
    },
    series: [
      {
        data: [120, 200, 150, 80, 70, 110, 130],
        type: 'bar'
      }
    ]
  });
  echartInstance.setOption(option)
}
//饼图
const charts2 = () => {
  const myChart = global.$echarts.init(myChart1.value);
  let option = reactive({
    title: {
      subtext: 'Fake Data',
      left: 'center'
    },
    tooltip: {
      trigger: 'item'
    },
    legend: {
      orient: 'vertical',
      left: 'left'
    },
    series: [
      {
        name: 'Access From',
        type: 'pie',
        radius: '50%',
        data: [
          { value: 1048, name: 'Search Engine' },
          { value: 735, name: 'Direct' },
          { value: 580, name: 'Email' },
          { value: 484, name: 'Union Ads' },
          { value: 300, name: 'Video Ads' }
        ],
        emphasis: {
          itemStyle: {
            shadowBlur: 10,
            shadowOffsetX: 0,
            shadowColor: 'rgba(0, 0, 0, 0.5)'
          }
        }
      }
    ]
  });
  myChart.setOption(option)
}
//环图
const charts3 = () => {
  const myChart = global.$echarts.init(myChart2.value);
  let option = reactive({
    tooltip: {
      trigger: 'item'
    },
    legend: {
      top: '5%',
      left: 'center'
    },
    series: [
      {
        name: 'Access From',
        type: 'pie',
        radius: ['40%', '70%'],
        avoidLabelOverlap: false,
        label: {
          show: false,
          position: 'center'
        },
        emphasis: {
          label: {
            show: true,
            fontSize: '40',
            fontWeight: 'bold'
          }
        },
        labelLine: {
          show: false
        },
        data: [
          { value: 1048, name: 'Search Engine' },
          { value: 735, name: 'Direct' },
          { value: 580, name: 'Email' },
          { value: 484, name: 'Union Ads' },
          { value: 300, name: 'Video Ads' }
        ]
      }
    ]
  });
  myChart.setOption(option)
```

```
}

onMounted(() => {
  charts1();
  charts2();
  charts3();
  nextTick(() => {
    mianHeight.value = window.innerHeight - 100
  })
})
</script>
<style scoped lang='scss'>
</style>
```

第62讲 封装hooks方式使用Echarts

1、在src/hooks下新建useEcharts.ts

```
import * as echarts from 'echarts'
import { EChartsOption } from 'echarts'
export default function useEcharts(el: HTMLElement) {
    //创建echarts实例
    const echartInstance = echarts.init(el)
    //设置options
    const setOptions = (options: any) => {
        echartInstance.setOption(options)
    }
    //监听窗口变化做自适应
    const resize = () => {
        echartInstance.resize()
    }
    return {
        setOptions,
        resize
    }
}
```

2、components下新建echarts文件夹，然后新建CommonEcharts.vue

```
<template>
    <div>
        <div ref="commonEchatRef" :style="{ height: height, width: width }">
</div>
    </div>
</template>
<script setup lang='ts'>
import { ref, onMounted, watchEffect } from 'vue'
import useEcharts from '@/hooks/useEcharts';

//echarts的 ref属性
const commonEchatRef = ref<HTMLElement>()

//接收父组件传来的值
const props = withDefaults(defineProps<{
    width?: string,
    height: string,
    options: any
}>(), {
    width: '100%',
    height: '360px'
})
onMounted(() => {
    const { setOptions, resize } = useEcharts(commonEchatRef.value!)
    watchEffect(() => {
        setOptions(props.options)
    })
    window.addEventListener('resize', () => {
        resize();
    })
})

</script>
<style scoped lang='scss'>
</style>
```

3、使用通用Echarts组件

```
<template>
  <el-main :style="{ height: mianHeight + 'px' }">
    <div style="display: flex;">
      <el-card style="flex: 1;">
        <template #header>
          <div class="card-header">
            <span>订单统计</span>
          </div>
        </template>
        <CommonEcharts :height="height" :options="options1"></CommonEcharts>
        <!-- <div id="myChart" ref="myChart" :style="{ width: '400px', height:
'300px' }"></div> -->
      </el-card>
      <el-card style="margin-left: 20px;flex: 1;">
        <template #header>
          <div class="card-header">
            <span>订单统计</span>
          </div>
        </template>
        <CommonEcharts :height="height" :options="options2"></CommonEcharts>
        <!-- <div id="myChart1" ref="myChart1" :style="{ width: '400px', height:
'300px' }"></div> -->
      </el-card>
      <el-card style="margin-left: 20px;flex: 1;">
        <template #header>
          <div class="card-header">
            <span>订单统计</span>
          </div>
        </template>
        <CommonEcharts :height="height" :options="options3"></CommonEcharts>
        <!-- <div id="myChart2" ref="myChart2" :style="{ width: '400px', height:
'300px' }"></div> -->
      </el-card>
    </div>
    <el-card style="flex: 1;">
      <template #header>
        <div class="card-header">
          <span>订单统计</span>
        </div>
      </template>
      <CommonEcharts height="290px" :options="options3"></CommonEcharts>
      <!-- <div id="myChart2" ref="myChart2" :style="{ width: '400px', height:
'300px' }"></div> -->
    </el-card>
  </el-main>
</template>
<script setup lang='ts'>
import { ref, nextTick, onMounted, reactive } from 'vue'
import CommonEcharts from '@/components/echarts/CommonEcharts.vue';
import useInstance from '@/hooks/useInstance';
const mianHeight = ref(0)
const { global } = useInstance()

//柱状图
const width = ref('400px')
const height = ref('300px')
let options1 = reactive({
  xAxis: {
    type: 'category' as 'category',
    data: ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
  },
  yAxis: {
    type: 'value' as 'value'
  },
  series: [
    {
      data: [120, 200, 150, 80, 70, 110, 130],
      type: 'bar' as 'bar'
    }
  ]
});

//饼图
let options2 = reactive({
  title: {
    subtext: 'Fake Data',
    left: 'center'
  },
  tooltip: {
    trigger: 'item'
  },
  legend: {
    orient: 'vertical',
    left: 'left'
  },
  series: [
    {
      name: 'Access From',
      type: 'pie' as 'pie',
      radius: '50%',
      data: [
          { value: 1048, name: 'Search Engine' },
          { value: 735, name: 'Direct' },
          { value: 580, name: 'Email' },
          { value: 484, name: 'Union Ads' },
          { value: 300, name: 'Video Ads' }
        ],
        emphasis: {
          itemStyle: {
            shadowBlur: 10,
            shadowOffsetX: 0,
            shadowColor: 'rgba(0, 0, 0, 0.5)'
          }
        }
      }
    ]
  });

  //环图
  let options3 = reactive({
    tooltip: {
      trigger: 'item'
    },
    legend: {
      top: '5%',
      left: 'center'
    },
    series: [
      {
        name: 'Access From',
        type: 'pie',
        radius: ['40%', '70%'],
        avoidLabelOverlap: false,
        label: {
          show: false,
          position: 'center'
        },
        emphasis: {
          label: {
            show: true,
            fontSize: '40',
            fontWeight: 'bold'
          }
        },
        labelLine: {
          show: false
        },
        data: [
          { value: 1048, name: 'Search Engine' },
          { value: 735, name: 'Direct' },
          { value: 580, name: 'Email' },
          { value: 484, name: 'Union Ads' },
          { value: 300, name: 'Video Ads' }
        ]
      }
    ]
  });

  onMounted(() => {
```

```
  nextTick(() => {
    mianHeight.value = window.innerHeight - 100
  })
})
</script>
<style scoped lang='scss'>
</style>
```