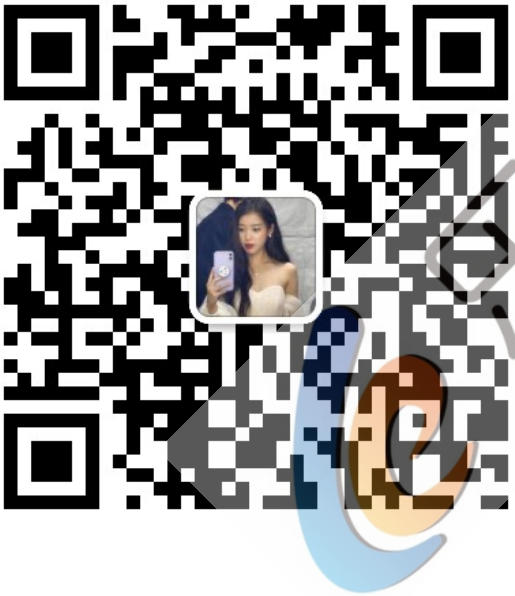
乐字节教育高级架构课程

正所谓“授人以鱼不如授人以渔”，你们想要的 Java 学习资料来啦！

不管你是学生，还是已经步入职场的同行，希望你们都要珍惜眼前的学习机会，奋斗没有终点，知识永不过时。

扫描下方二维码即可领取



乐字节官方交流群

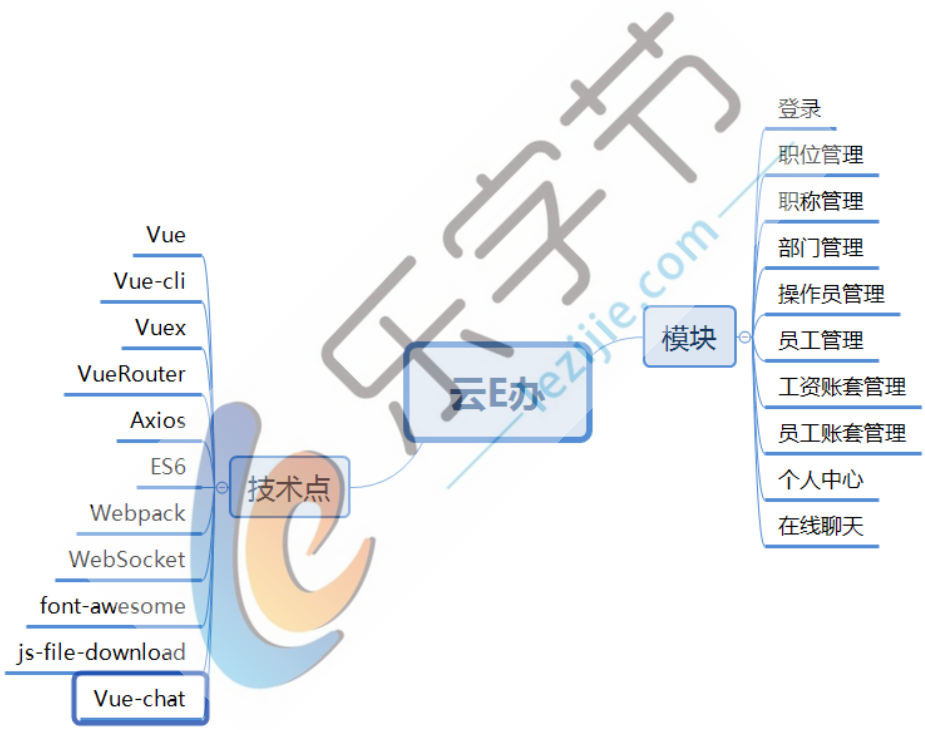
云E办(前端)

项目介绍

本项目目的是实现中小型企业的在线办公系统，云E办在线办公系统的是一个用来管理日常的办公事务的一个系统，他能够管的内容有：日常的各种流程审批，新闻，通知，公告，文件信息，财务，人事，费用，资产，行政，项目，移动办公等等。它的作用就是通过软件的方式，方便管理，更加简单，更加扁平。更加高效，更加规范，能够提高整体的管理运营水平。

本项目在技术方面采用最主流的前后端分离开发模式，使用业界最流行、社区非常活跃的开源框架Vue来构建前端端，旨在实现云E办在线办公系统。包括职位管理、职称管理、部门管理、员工管理、工资管理、在线聊天等模块。项目中还会使用业界主流的第三方组件扩展大家的知识面和技能池。

本项目主要模块及技术点如图



搭建项目

环境准备

环境准备

- Node.js（>= 6.x，首选 8.x）

安装 vue-cli

- 安装 Node.js

官网下载地址 http://nodejs.cn/download

当前版本: **12.14.1**

Windows 安装包 node-v12.14.1-x64.msi | macOS 安装包 node-v12.14.1.pkg | 源代码 node-v12.14.1.tar.gz

Windows 安装包 (.msi)	32 位	64 位
Windows 二进制文件 (.zip)	32 位	64 位
macOS 安装包 (.pkg)	64 位	
macOS 二进制文件 (.tar.gz)	64 位	
Linux 二进制文件 (x64)	64 位	
Linux 二进制文件 (ARM)	ARMv7	ARMv8
Docker 镜像	官方镜像	
全部安装包	阿里云镜像	

- 安装 Node.js 淘宝镜像加速器（`cnpm`）

```
npm install cnpm -g

# 或使用如下语句解决 npm 速度慢的问题
npm install --registry=https://registry.npm.taobao.org
```

```
Microsoft Windows [版本 10.0.18362.592]
(c) 2019 Microsoft Corporation。保留所有权利。

C:\WINDOWS\system32>npm install cnpm -g
C:\Users\Administrator\AppData\Roaming\npm\cnpm -> C:\Users\Administrator\AppData\Roaming\npm\node_modules\cnpm\bin\cnpm
+ cnpm@6.1.1
added 686 packages from 944 contributors in 86.005s

C:\WINDOWS\system32>
```

- 安装 vue-cli

```
npm install -g @vue/cli
```

- 测试是否安装成功

```
vue --version
```

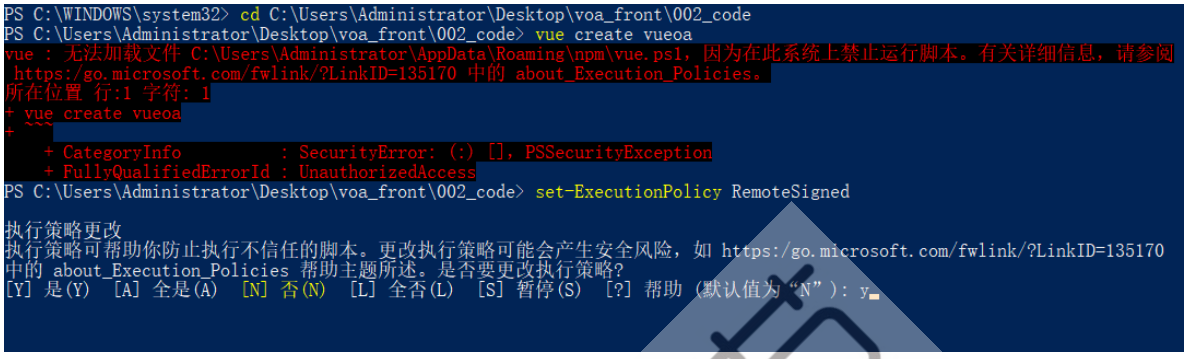
```
C:\Users\Administrator>vue --version
@vue/cli 4.2.3
```

打开Windows Power Shell(管理员模式运行)

```
vue create yeb
```

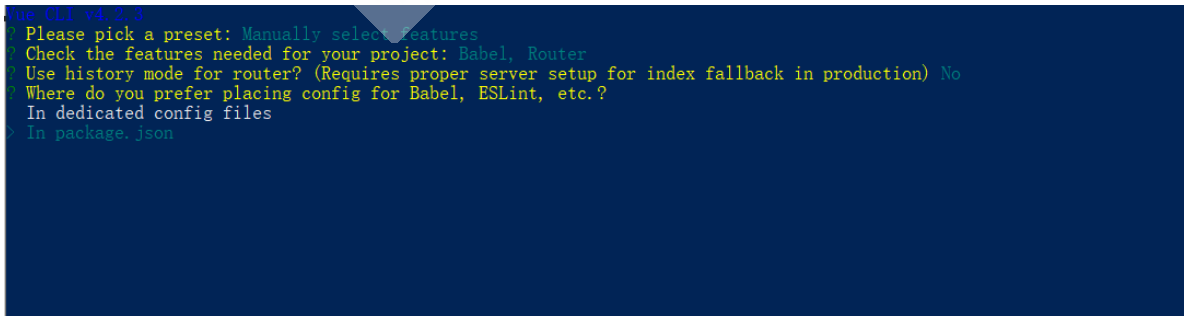
如果出现如下报错信息，表示系统禁止运行未知脚本

```
set-ExecutionPolicy RemoteSigned
# 选择Y
```



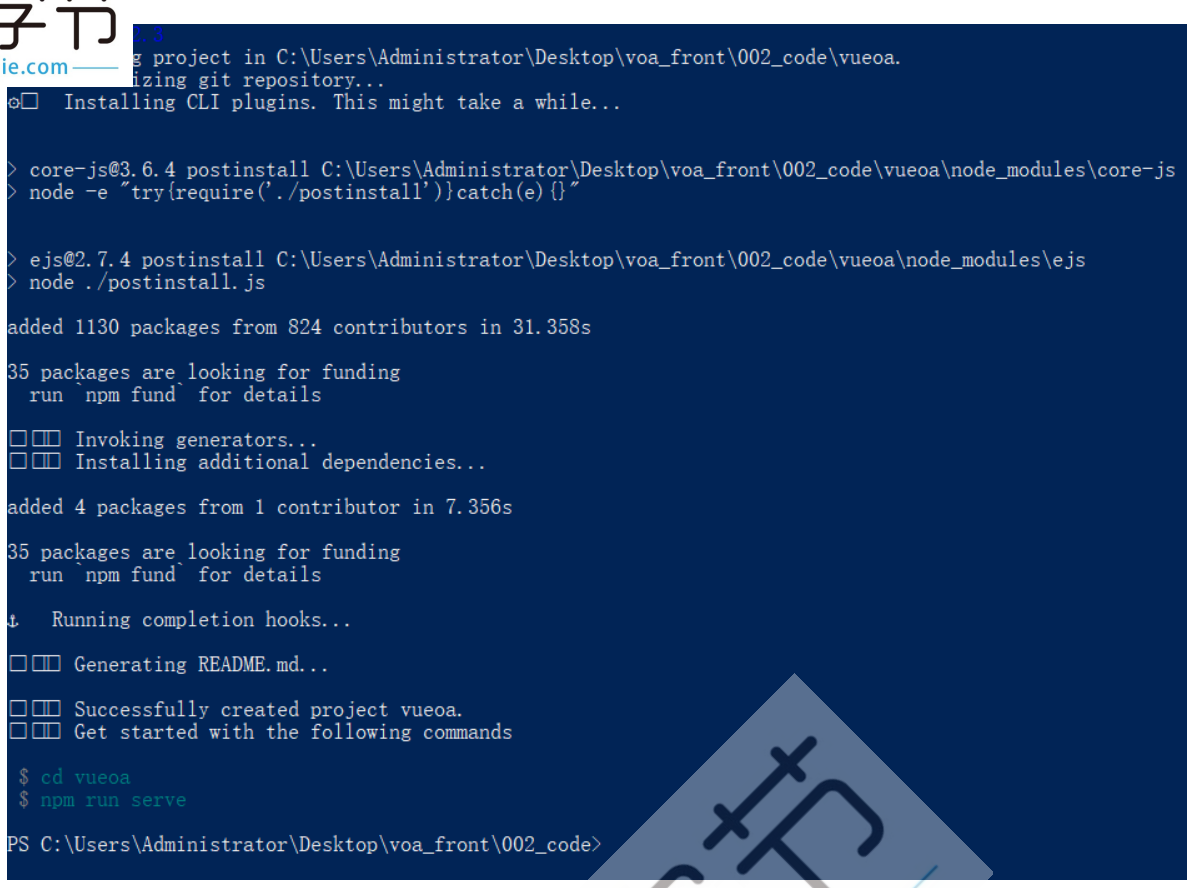
我们选择如下几个选项即可

- 空格表示选择
- 回车继续下一步操作



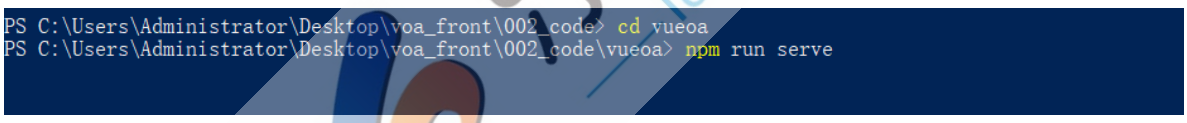
- 是否需要使用history模式，我们选择否
- 选择配置信息存储位置，我们选择package.json

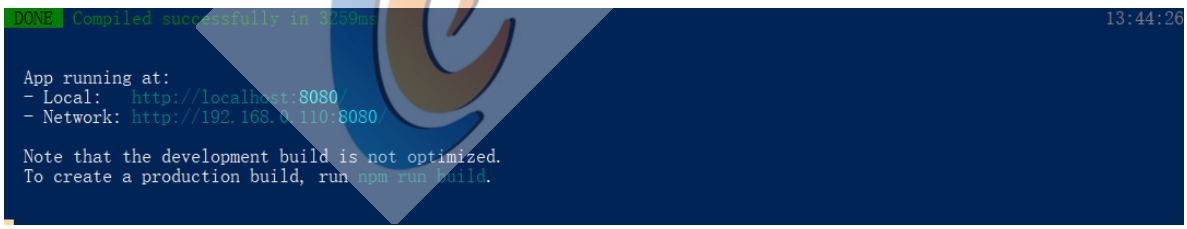
出现如下图所示表示项目创建完成



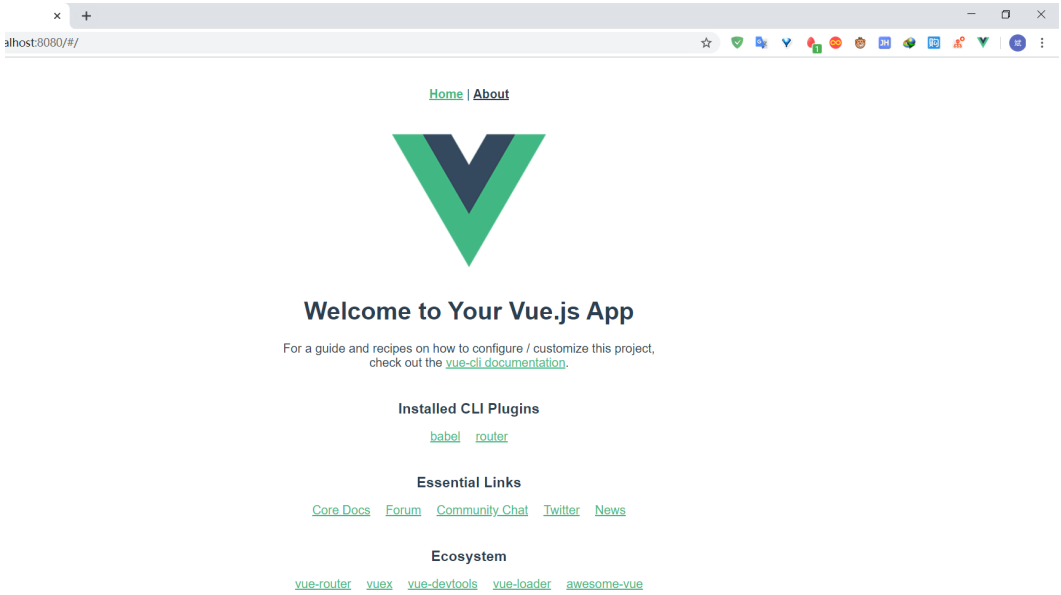
运行项目

```
cd yeb
npm run serve
```





安装并运行成功后在浏览器输入：http://localhost:8080



项目结构

vue-cli目录

- node_modules：用于存放项目的依赖文件
- Public:公共目录
- src：项目源码目录
- .gitignore：git 忽略的配置文件
- babel.config.js：Babel 配置文件，主要作用是将 ES6 转换为 ES5
- package.json：项目的配置文件
  - name：项目名称
  - version：项目版本
  - description：项目描述
  - author：项目作者
  - scripts：封装常用命令

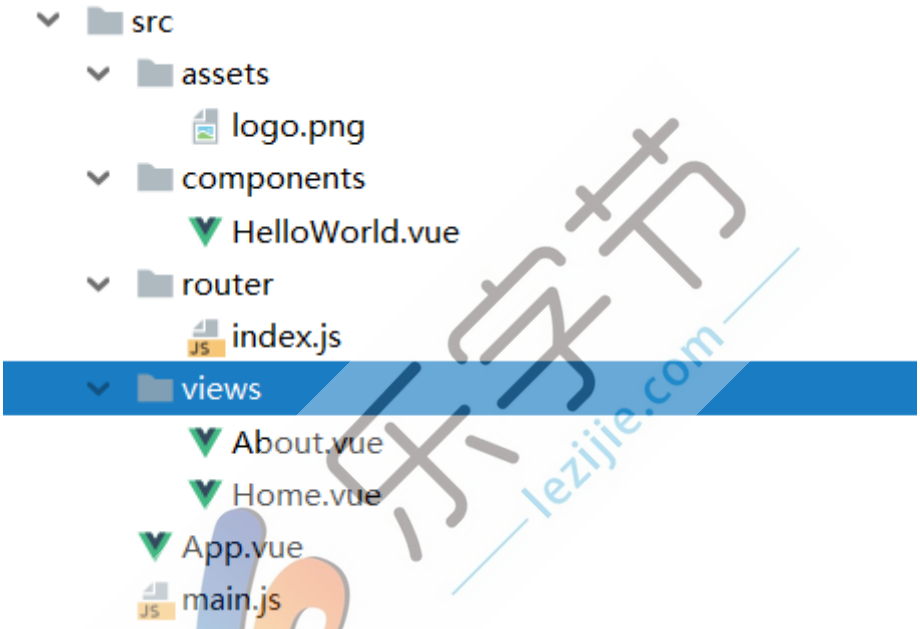
dependencies：生产环境依赖
devDependencies：开发环境依赖

Public目录

- favicon.ico:网页图标
- index.html:首页，仅作为模板页，实际开发时不使用

src目录

`src` 目录是项目的源码目录，所有代码都会写在这里



- `assets`:资源目录
- `components`：组件目录
- `router`：路由目录
- `views`：页面目录

main.js

项目的入口文件，我们知道所有的程序都会有一个入口

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'

Vue.config.productionTip = false

new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```

- `import Vue from 'vue'`：ES6 写法，会被转换成 `require("vue");` （require 是 NodeJS 提供的模块加载器）

rt App from './App.vue'：意思同上，但是指定了查找路径，./ 为当前目录

- Vue.config.productionTip = false：关闭浏览器控制台关于环境的相关提示
- new Vue({...})：实例化 Vue
  - router：使用router组件
  - render: h => h(App)：ES6写法，渲染App组件
  - .$mount('#app')：手动挂载，与之前的 el:'#app' 作用类似

App.vue

组件模板

```
<template>
  <div id="app">
    <div id="nav">
      <router-link to="/">Home</router-link> |
      <router-link to="/about">About</router-link>
    </div>
    <router-view/>
  </div>
</template>

<style>
#app {
  font-family: Avenir, Helvetica, Arial, sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  text-align: center;
  color: #2c3e50;
}

#nav {
  padding: 30px;
}

#nav a {
  font-weight: bold;
  color: #2c3e50;
}

#nav a.router-link-exact-active {
  color: #42b983;
}
</style>
```

- template：HTML 代码模板，会替换 <App /> 中的内容
- <router-link>：默认会被渲染成一个 <a> 标签，to 属性为指定链接
- <router-view/>： 用于渲染路由匹配到的组件

Home.vue

```
<template>
  <div class="home">
    <img alt="Vue logo" src="../assets/logo.png">
    <HelloWorld msg="Welcome to Your Vue.js App"/>
```

```
>
ate>

<script>
// @ is an alias to /src
import HelloWorld from '@/components/HelloWorld.vue'

export default {
  name: 'Home',
  components: {
    HelloWorld
  }
}
</script>
```

- `template`：HTML 代码模板，会替换 `<App />` 中的内容
- `export default{...}`：导出 NodeJS 对象，作用是可以通过`import`关键字导入
  - `name: 'Home'`：定义组件的名称
  - `components: { HelloWorld }`：定义子组件

HelloWorld.vue

```
<template>
  <div class="hello">
    <h1>{{ msg }}</h1>
    <p>
      For a guide and recipes on how to configure / customize this project,<br>
      check out the
      <a adminef="https://cli.vuejs.org" target="_blank" rel="noopener">vue-cli
documentation</a>.
    </p>
    <h3>Installed CLI Plugins</h3>
    <ul>
      <li><a adminef="https://github.com/vuejs/vue-
cli/tree/dev/packages/%40vue/cli-plugin-babel" target="_blank"
rel="noopener">babel</a></li>
      <li><a adminef="https://github.com/vuejs/vue-
cli/tree/dev/packages/%40vue/cli-plugin-router" target="_blank"
rel="noopener">router</a></li>
    </ul>
    <h3>Essential Links</h3>
    <ul>
      <li><a adminef="https://vuejs.org" target="_blank" rel="noopener">Core
Docs</a></li>
      <li><a adminef="https://forum.vuejs.org" target="_blank"
rel="noopener">Forum</a></li>
      <li><a adminef="https://chat.vuejs.org" target="_blank"
rel="noopener">Community Chat</a></li>
      <li><a adminef="https://twitter.com/vuejs" target="_blank"
rel="noopener">Twitter</a></li>
      <li><a adminef="https://news.vuejs.org" target="_blank"
rel="noopener">News</a></li>
    </ul>
    <h3>Ecosystem</h3>
    <ul>
```

```
li><a adminef="https://router.vuejs.org" target="_blank"
opener">vue-router</a></li>
      <li><a adminef="https://vuex.vuejs.org" target="_blank"
rel="noopener">vuex</a></li>
      <li><a adminef="https://github.com/vuejs/vue-devtools#vue-devtools"
target="_blank" rel="noopener">vue-devtools</a></li>
      <li><a adminef="https://vue-loader.vuejs.org" target="_blank"
rel="noopener">vue-loader</a></li>
      <li><a adminef="https://github.com/vuejs/awesome-vue" target="_blank"
rel="noopener">awesome-vue</a></li>
    </ul>
  </div>
</template>

<script>
export default {
  name: 'HelloWorld',
  props: {
    msg: String
  }
}
</script>

<!-- Add "scoped" attribute to limit CSS to this component only -->
<style scoped>
h3 {
  margin: 40px 0 0;
}
ul {
  list-style-type: none;
  padding: 0;
}
li {
  display: inline-block;
  margin: 0 10px;
}
a {
  color: #42b983;
}
</style>
```

- `template`：HTML 代码模板，会替换 `<App />` 中的内容
- `export default{...}`：导出 NodeJS 对象，作用是可以通过 `import` 关键字导入
  - `name: 'HelloWorld'`：定义组件的名称
  - `props: { msg: String }`：定义属性
- `<style scoped>`：CSS 样式仅在当前组件有效，声明了样式的作用域

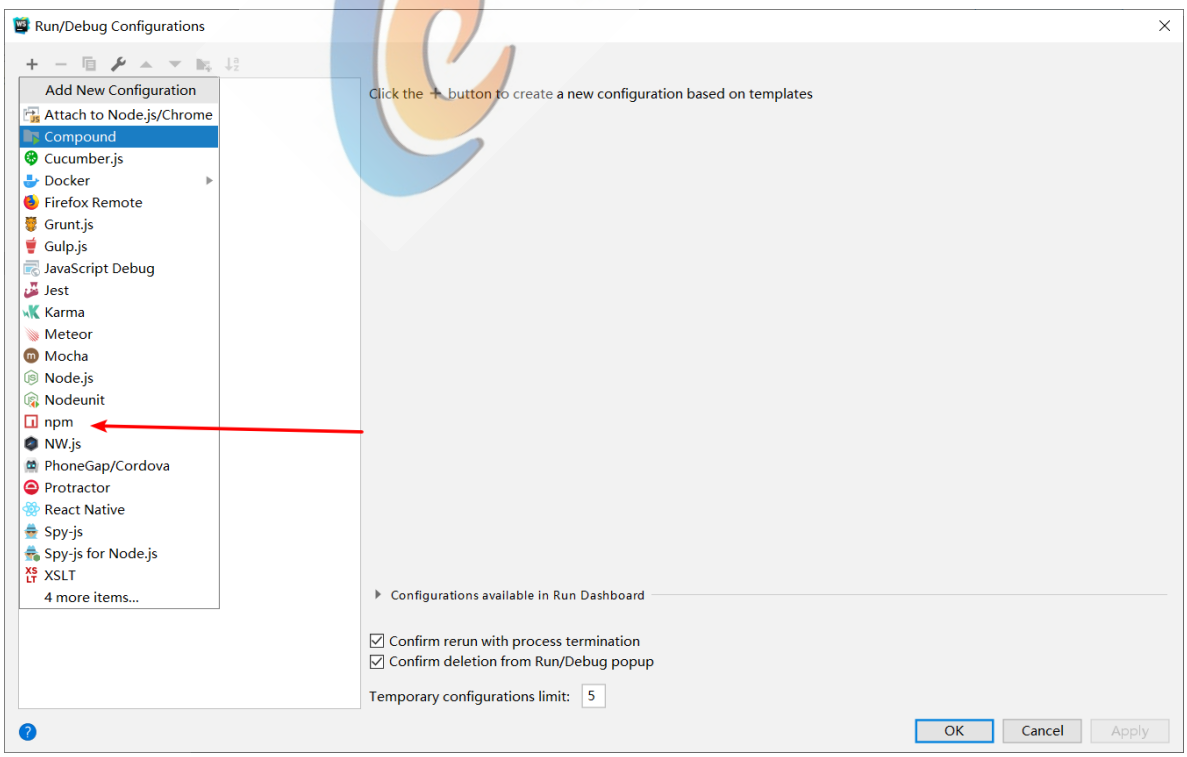
index.js（router目录）

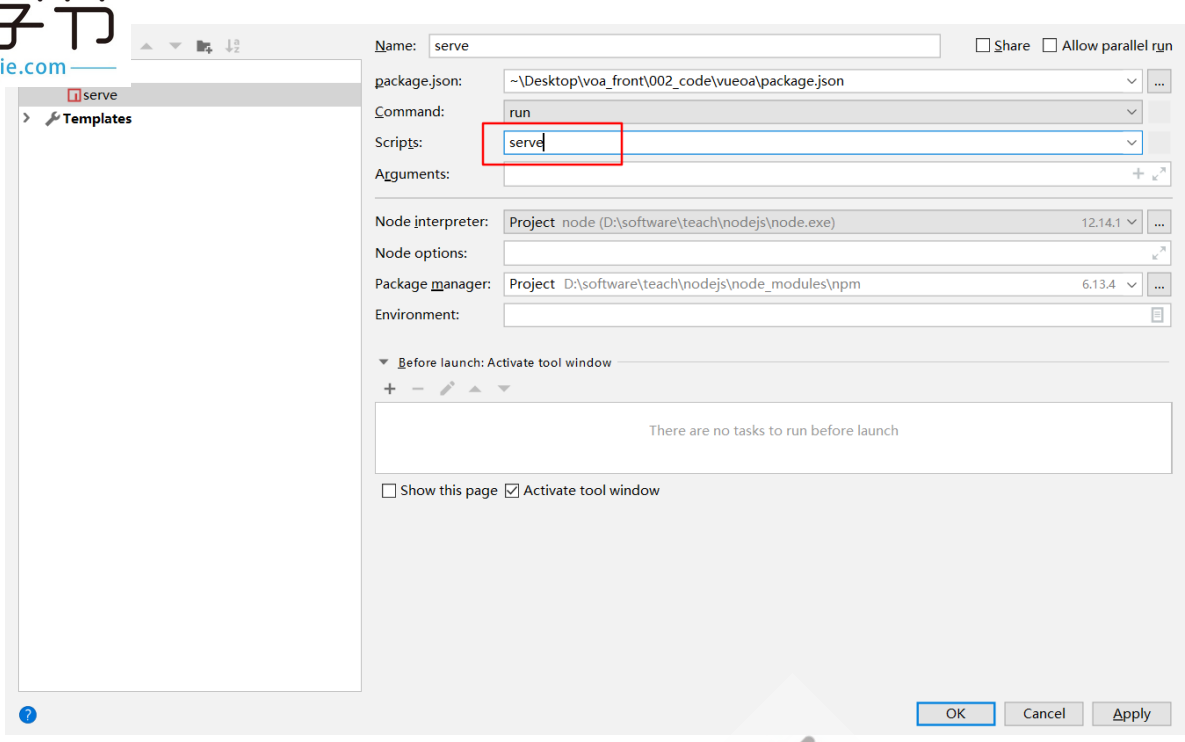
```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Home from '../views/Home.vue'

Vue.use(VueRouter)

const routes = [
```

```
    h: '/',
    name: 'Home',
    component: Home
  },
  {
    path: '/about',
    name: 'About',
    // route level code-splitting
    // this generates a separate chunk (about.[hash].js) for this route
    // which is lazy-loaded when the route is visited.
    component: () => import(/* webpackChunkName: "about" */
'../views/About.vue')
  }
]

const router = new VueRouter({
  routes
})

export default router
```

- `Vue.use(VueRouter)`：安装路由
- `const routes = []`：配置路由
- `path`：路由路径
- `name`：路由名称
- `component`：跳转的组件
- `component: () => import(/* webpackChunkName: "about" */ '../views/About.vue')`：动态导入，可以不用提前导入组件，在使用时再导入组件

配置项目启动





配置完成后启动项目并在浏览器输入：http://localhost:8080

登录页面

安装ElementUI

UI库我们使用了`ElementUI`,在正式开发之前我们需要先安装`ElementUI`

安装ElementUI

```
npm i element-ui -S
```

- `-S`：将模块安装到项目目录下，并在 package 文件的 dependencies 节点写入依赖

```
\Desktop\voa_front\002_code\vueoa>npm i element-ui -S
re-js@2.6.11: core-js@<3 is no longer maintained and not recommended for usage due to the number of issues. Please, upgrade your dependencies to the actual version of core-js@3.

> core-js@2.6.11 postinstall C:\Users\Administrator\Desktop\voa_front\002_code\vueoa\node_modules\babel-runtime\node_modules\core-js
> node -e "try{require('./postinstall')}catch(e){}"

Thank you for using core-js ( https://github.com/zloirock/core-js ) for polyfilling JavaScript standard library!

The project needs your help! Please consider supporting of core-js on Open Collective or Patreon:
> https://opencollective.com/core-js
> https://www.patreon.com/zloirock

Also, the author of core-js ( https://github.com/zloirock ) is looking for a good job -)

npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.12 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.12: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ element-ui@2.13.0
added 9 packages from 8 contributors in 13.179s

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

引入ElemetUI

修改入口文件`main.js`

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
// 引入ElementUI
import ElementUI from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'

Vue.config.productionTip = false
// 安装ElementUI
Vue.use(ElementUI)

new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```

创建登录页面视图

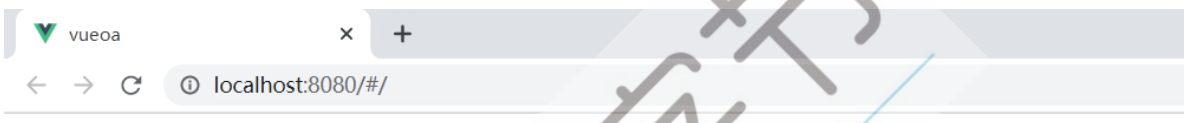
在`views`目录下创建一个名为`Login.vue`的视图组件

```
<template>
    <div>
        Login
    </div>
</template>

<script>
    export default {
        name: "Login"
    }
</script>

<style scoped>

</style>
```

修改`router`目录下的`index.js`

```
Vue from 'vue'
VueRouter from 'vue-router'
import Login from '../views/Login'

Vue.use(VueRouter)

  const routes = [
  {
    path: '/',
    name: 'Login',
    component: Login
  }
]

const router = new VueRouter({
  routes
})

export default router
```

启动测试查看首页是否为Login视图页面



编写登录视图页面

`el-*` 的元素为 ElementUI 组件;

```
<template>
    <div>
        <el-form :rules="rules" :model="loginForm" class="loginContainer">
            <h3 class="loginTitle">系统登录</h3>
            <el-form-item prop="username">
                <el-input type="text" v-model="loginForm.username" auto-
complete="false"
                          placeholder="请输入用户名"></el-input>
            </el-form-item>
            <el-form-item prop="password">
```

```
                <el-input type="password" v-model="loginForm.password" auto-
e="false"
                          placeholder="请输入密码"></el-input>
            </el-form-item>
            <el-form-item prop="code">
                <el-input size="normal" type="text" v-model="loginForm.code"
auto-complete="false"
                          placeholder="点击图片更换验证码" style="width:
250px;margin-right: 5px"></el-input>
                <img :src="captchaUrl">
            </el-form-item>
            <el-checkbox class="loginRemember" v-model="checked">记住我</el-
checkbox>
            <el-button type="primary" style="width: 100%">登录</el-button>
        </el-form>
    </div>
</template>

<script>
    export default {
        name: "Login",
        data() {
            return {
                captchaUrl: '',
                loginForm: {
                    username: 'admin',
                    password: '123',
                    code:''
                },
                checked: true,
                rules: {
                    username: [{required: true, message: '请输入用户名', trigger:
'blur'}],
                    password: [{required: true, message: '请输入密码', trigger:
'blur'}],
                    code: [{required: true, message: '请输入验证码', trigger:
'blur'}]
                }
            }
        }
    }
</script>

<style>
    .loginContainer {
        border-radius: 15px;
        background-clip: padding-box;
        margin: 180px auto;
        width: 350px;
        padding: 15px 35px 15px 35px;
        background: #fff;
        border: 1px solid #eaeaea;
        box-shadow: 0 0 25px #cac6c6;
    }

    .loginTitle {
        margin: 0px auto 40px auto;
        text-align: center;
```

```
        color: #505458;
    }

    .loginRemember {
        text-align: left;
        margin: 0px 0px 15px 0px;
    }

    .el-form-item__content{
        display: flex;
        align-items: center;
    }
</style>
```

显示效果如下

处理登录事件

login.vue

```
<template>
    <div>
        <el-form :rules="rules" ref="loginForm" :model="loginForm"
class="loginContainer">
            <h3 class="loginTitle">系统登录</h3>
            <el-form-item prop="username">
```

```
                <el-input type="text" v-model="loginForm.username" auto-
e="false"
                          placeholder="请输入用户名"></el-input>
            </el-form-item>
            <el-form-item prop="password">
                <el-input type="password" v-model="loginForm.password" auto-
complete="false"
                          placeholder="请输入密码"></el-input>
            </el-form-item>
            <el-form-item prop="code">
                <el-input size="normal" type="text" v-model="loginForm.code"
auto-complete="false" placeholder="点击图片更换验证码" style="width: 250px;margin-
right: 5px"></el-input>
                <img :src="captchaUrl">
            </el-form-item>
            <el-checkbox class="loginRemember" v-model="checked">记住我</el-
checkbox>
            <el-button type="primary" style="width: 100%" @click="submitLogin">
登录</el-button>
        </el-form>
    </div>
</template>

<script>
    export default {
        name: "Login",
        data() {
            return {
                captchaUrl: '',
                loginForm: {
                    username: 'admin',
                    password: '123',
                    code:''
                },
                checked: true,
                rules: {
                    username: [{required: true, message: '请输入用户名', trigger:
'blur'}],
                    password: [{required: true, message: '请输入密码', trigger:
'blur'}],
                    code: [{required: true, message: '请输入验证码', trigger:
'blur'}]
                }
            }
        },
        methods:{
            submitLogin(){
                this.$refs.loginForm.validate((valid)=>{
                    if (valid){
                        alert('aaa');
                    } else {
                        this.$message.error('请输入所有字段');
                        return false;
                    }
                })
            }
        }
    }
```

```
t>
<style>
    .loginContainer {
        border-radius: 15px;
        background-clip: padding-box;
        margin: 180px auto;
        width: 350px;
        padding: 15px 35px 15px 35px;
        background: #fff;
        border: 1px solid #eaeaea;
        box-shadow: 0 0 25px #cac6c6;
    }

    .loginTitle {
        margin: 0px auto 40px auto;
        text-align: center;
        color: #505458;
    }

    .loginRemember {
        text-align: left;
        margin: 0px 0px 15px 0px;
    }

    .el-form-item__content{
        display: flex;
        align-items: center;
    }
</style>
```

调用后端接口完成登录功能

安装Axios

```
npm install axios
```

```
C:\Users\Administrator\Desktop\voa_front\002_code\vueoa>npm install axios
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.12 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.12: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ axios@0.19.2
added 4 packages from 7 contributors in 9.058s

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

封装请求

每一次网络请求都有可能成功或者失败。成功有成功的处理方式，失败一般都是提示错误信息。我们希望将所有的失败统一进行处理提示错误信息而不是每失败一次处理一次，因此我们将网络请求进行封装达到统一处理失败信息的目的。

`utils`目录下新建`api.js`

```
import axios from 'axios'
import {Message} from 'element-ui';
```

```
router from '../router'
//响应拦截器
axios.interceptors.response.use(success => {
    //业务逻辑错误
    if (success.status && success.status == 200) {
        if (success.data.code == 500 || success.data.code == 401 ||
success.data.code == 403) {
            Message.error({message: success.data.message});
            return;
        }
        if (success.data.message) {
            Message.success({message: success.data.message});
        }
    }
    return success.data;
}, error => {
    if (error.response.code == 504 || error.response.code == 404) {
        Message.error({message: '服务器被吃了o(╯□╰)o'});
    } else if (error.response.code == 403) {
        Message.error({message: '权限不足，请联系管理员！'});
    } else if (error.response.code == 401) {
        Message.error({message: '尚未登录，请登录'});
        router.replace('/');
    } else {
        if (error.response.data.message) {
            Message.error({message: error.response.data.message});
        } else {
            Message.error({message: '未知错误！'});
        }
    }
    return;
});

let base = '';

//传送json格式的post请求
export const postRequest = (url, params) => {
    return axios({
        method: 'post',
        url: `${base}${url}`,
        data: params
    })
}
```

配置请求转发解决跨域

新建vue.config.js

```
let proxyObj = {}
proxyObj['/'] = {
    //websocket
    ws: false,
    //目标地址
    target: 'http://localhost:8081',
    //发送请求头中host会设置成target
    changeOrigin: true,
```

```
//重写请求地址
    pathRewrite: {
        '^/': '/'
    }
}


module.exports = {
    devServer: {
        host: 'localhost',
        port: 8080,
        proxy: proxyObj
    }
}
```

处理验证码

验证码后端直接返回图片，直接通过`img`标签的src属性即可获取。添加图片的点击事件，请求时`time`参数是为了确保验证码能够正确刷新

Login.vue

```
<template>
    <div>
        <el-form :rules="rules" ref="loginForm" :model="loginForm"
class="loginContainer">
            <h3 class="loginTitle">系统登录</h3>
            <el-form-item prop="username">
                <el-input type="text" v-model="loginForm.username" auto-
complete="false"
                          placeholder="请输入用户名"></el-input>
            </el-form-item>
            <el-form-item prop="password">
                <el-input type="password" v-model="loginForm.password" auto-
complete="false"
                          placeholder="请输入密码"></el-input>
            </el-form-item>
            <el-form-item prop="code">
                <el-input size="normal" type="text" v-model="loginForm.code"
auto-complete="false"
                          placeholder="点击图片更换验证码" style="width:
250px;margin-right: 5px"></el-input>
                <img :src="captchaUrl" @click="updateCaptcha">
            </el-form-item>
            <el-checkbox class="loginRemember" v-model="checked">记住我</el-
checkbox>
            <el-button type="primary" style="width: 100%" @click="submitLogin">
登录</el-button>
        </el-form>
    </div>
</template>

<script>
    export default {
        name: "Login",
        data() {
            return {
```

```
            captchaUrl: '/captcha?time=' + new Date(),
            loginForm: {
                username: 'admin',
                password: '123',
                code: ''
            },
            checked: true,
            rules: {
                username: [{required: true, message: '请输入用户名', trigger:
'blur'}],
                password: [{required: true, message: '请输入密码', trigger:
'blur'}],
                code: [{required: true, message: '请输入验证码', trigger:
'blur'}]
            }
        }
    },
    methods: {
        updateCaptcha() {
            this.captchaUrl = '/captcha?time=' + new Date();
        },
        submitLogin(){
            this.$refs.loginForm.validate((valid)=>{
                if (valid){
                    alert('aaa');
                } else {
                    this.$message.error('请输入所有字段');
                    return false;
                }
            })
        }
    }
}
</script>

<style>
    .loginContainer {
        border-radius: 15px;
        background-clip: padding-box;
        margin: 180px auto;
        width: 350px;
        padding: 15px 35px 15px 35px;
        background: #fff;
        border: 1px solid #eaeaea;
        box-shadow: 0 0 25px #cac6c6;
    }

    .loginTitle {
        margin: 0px auto 40px auto;
        text-align: center;
        color: #505458;
    }

    .loginRemember {
        text-align: left;
        margin: 0px 0px 15px 0px;
    }
```

```
-form-item__content{
        display: flex;
        align-items: center;
    }
</style>
```

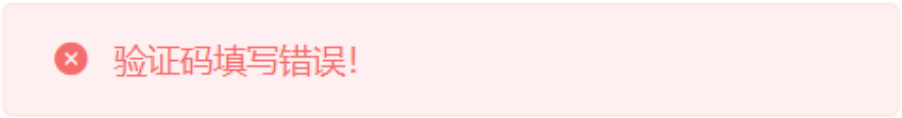
修改登录事件

Login.vue

```
<script>
    import {postRequest} from "../utils/api";

    export default {
        name: "Login",
        data() {
            return {
                captchaUrl: '/captcha?time=' + new Date(),
                loginForm: {
                    username: 'admin',
                    password: '123',
                    code: ''
                },
                checked: true,
                rules: {
                    username: [{required: true, message: '请输入用户名', trigger:
'blur'}],
                    password: [{required: true, message: '请输入密码', trigger:
'blur'}],
                    code: [{required: true, message: '请输入验证码', trigger:
'blur'}]
                }
            }
        },
        methods: {
            updateCaptcha() {
                this.captchaUrl = '/captcha?time=' + new Date();
            },
            submitLogin() {
                this.$refs.loginForm.validate((valid) => {
                    if (valid) {
                        postRequest('/login', this.loginForm).then(resp => {
                            if (resp) {
                                alert(JSON.stringify(resp));
                            }
                        })
                    } else {
                        this.$message.error('请输入所有字段');
                        return false;
                    }
                })
            }
        }
    }
</script>
```

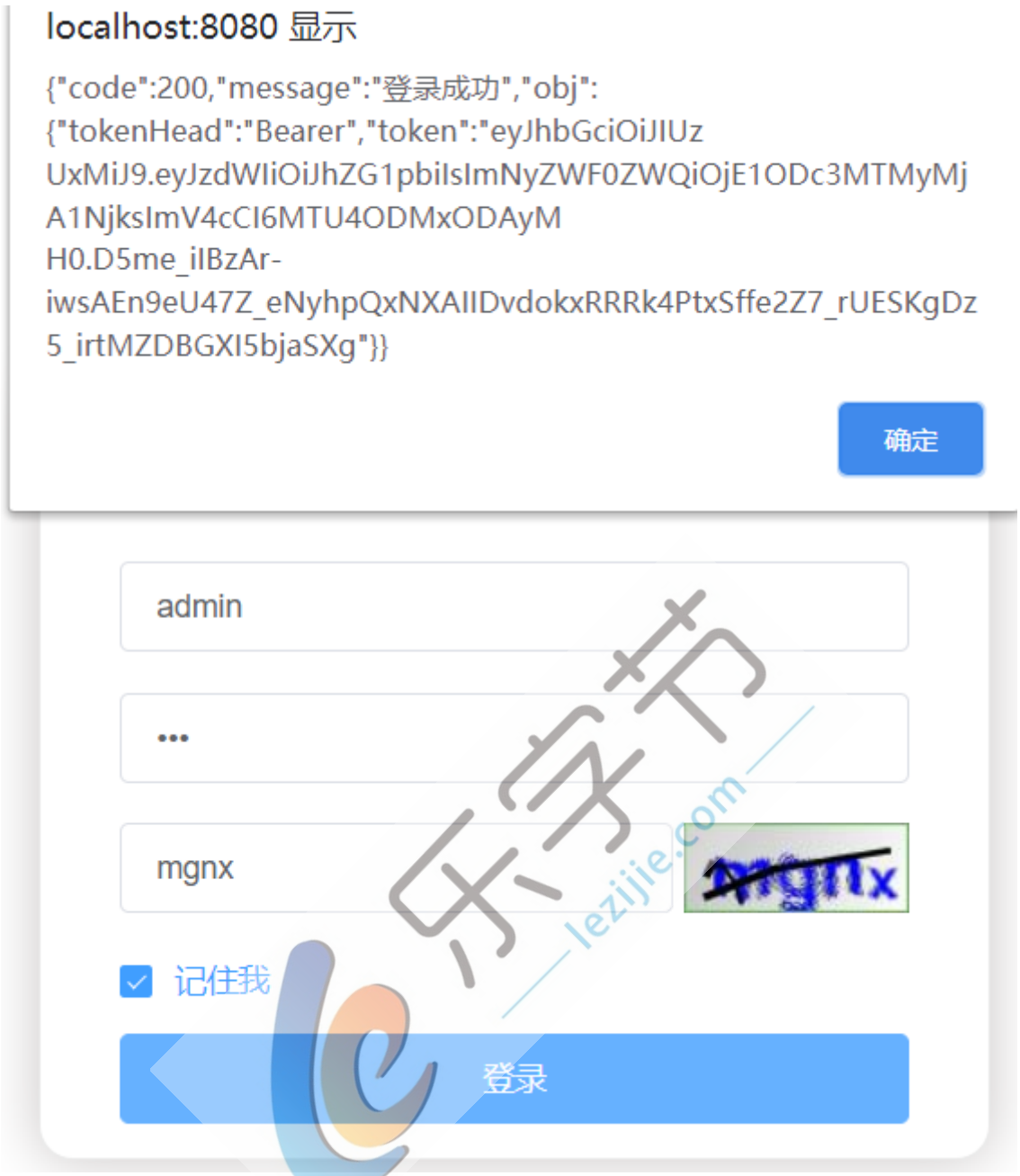
需要后台接口服务器先启动

验证码错误，提示错误信息（接口返回）



账户名或密码输入错误，提示错误信息（接口返回）

账户名和密码输入正确，弹出用户token（接口返回）



登录成功后的页面跳转

`views`目录新建一个`Home.vue`

```
<template>
    <div>
        Home
    </div>
</template>

<script>
    export default {
        name: "Home"
    }
</script>

<style scoped>
```

```
>
```

配置路由：`router`目录下`index.js`

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Login from '../views/Login'
import Home from '../views/Home'

Vue.use(VueRouter)

const routes = [
    {
        path: '/',
        name: 'Login',
        component: Login
    },
    {
        path: '/home',
        name: 'Home',
        component: Home
    }
]

const router = new VueRouter({
    routes
})

export default router
```

Login.vue

```
<template>
    <div>
        <el-form v-loading="loading"
                 element-loading-text="正在登录..."
                 element-loading-spinner="el-icon-loading"
                 element-loading-background="rgba(0, 0, 0, 0.8)"
                 :rules="rules"
                 ref="loginForm"
                 :model="loginForm"
                 class="loginContainer">
            <h3 class="loginTitle">系统登录</h3>
            <el-form-item prop="username">
                <el-input type="text" v-model="loginForm.username" auto-
complete="false"
                          placeholder="请输入用户名"></el-input>
            </el-form-item>
            <el-form-item prop="password">
                <el-input type="password" v-model="loginForm.password" auto-
complete="false"
                          placeholder="请输入密码"></el-input>
            </el-form-item>
            <el-form-item prop="code">
                <el-input size="normal" type="text" v-model="loginForm.code"
auto-complete="false"
```

```
                placeholder="点击图片更换验证码"
n.enter.native="submitLogin" style="width: 250px;margin-right: 5px"></el-
input>
                <img :src="captchaUrl" @click="updateCaptcha">
            </el-form-item>
            <el-checkbox class="loginRemember" v-model="checked">记住我</el-
checkbox>
            <el-button type="primary" style="width: 100%" @click="submitLogin">
登录</el-button>
        </el-form>
    </div>
</template>

<script>
    import {postRequest} from "../utils/api";

    export default {
        name: "Login",
        data() {
            return {
                captchaUrl: '/captcha?time=' + new Date(),
                loginForm: {
                    username: 'admin',
                    password: '123',
                    code: ''
                },
                loading: false,
                checked: true,
                rules: {
                    username: [{required: true, message: '请输入用户名', trigger:
'blur'}],
                    password: [{required: true, message: '请输入密码', trigger:
'blur'}],
                    code: [{required: true, message: '请输入验证码', trigger:
'blur'}]
                }
            }
        },
        methods: {
            updateCaptcha() {
                this.captchaUrl = '/captcha?time=' + new Date();
            },
            submitLogin() {
                this.$refs.loginForm.validate((valid) => {
                    if (valid) {
                        this.loading = true;
                        postRequest('/login', this.loginForm).then(resp => {
                            this.loading = false;
                            if (resp) {
                                this.$router.replace('/home');
                            }
                        })
                    } else {
                        this.$message.error('请输入所有字段');
                        return false;
                    }
                })
            }
```

```
    }
</script>

<style>
    .loginContainer {
        border-radius: 15px;
        background-clip: padding-box;
        margin: 180px auto;
        width: 350px;
        padding: 15px 35px 15px 35px;
        background: #fff;
        border: 1px solid #eaeaea;
        box-shadow: 0 0 25px #cac6c6;
    }

    .loginTitle {
        margin: 0px auto 40px auto;
        text-align: center;
        color: #505458;
    }

    .loginRemember {
        text-align: left;
        margin: 0px 0px 15px 0px;
    }

    .el-form-item__content {
        display: flex;
        align-items: center;
    }
</style>
```

- `push`:添加页面，可以通过浏览器的后退按钮回到之前页面
- `replace`：替换页面，不可以通过浏览器的后退按钮回到之前页面
- `element-loading-text`：加载文案，显示在加载图标的下方
- `element-loading-spinner`：加载图标类名
- `element-loading-background`：加载背景色值

封装请求

完善api.js

后端的登录接口是返回一个token，我们除了登录接口外每一次请求都要在请求头里携带这个token方便后端校验，因此我们需要对请求进行拦截。

`utils`目录下的`api.js`

```
import axios from 'axios'
import {Message} from "element-ui";
import router from '../router'

// 请求拦截器
axios.interceptors.request.use(config=>{
    if (window.sessionStorage.getItem('tokenStr')) {
        //请求携带自定义token
        config.headers['Authorization'] =
window.sessionStorage.getItem('tokenStr');
    }
    return config
},error => {
    console.log(error);
})


//响应拦截器
axios.interceptors.response.use(success => {
    //业务逻辑错误
    if (success.status && success.status == 200 && success.data.code == 500) {
        Message.error({message: success.data.message})
        return;
    }
    if (success.data.message){
        Message.success({message:success.data.message})
```

```
    urn success.data;
}, error => {
    if (error.response.code == 504 || error.response.code == 404) {
        Message.error({message: '服务器被吃了(ﾉ□ﾉ)'})
    } else if (error.response.code == 403) {
        Message.error({message: '权限不足，请联系管理员'})
    } else if (error.response.code == 401) {
        Message.error({message: '尚未登录，请登录'});
        router.replace('/')
    } else {
        if (error.response.data.message) {
            Message.error({message: error.response.data.message})
        } else {
            Message.error({message: '未知错误！'})
        }
    }
    return;
})

let base = '';

//传递json的post请求
export const postRequest = (url, params) => {
    return axios({
        method: 'post',
        url: `${base}${url}`,
        data: params
    })
}

//传递json的put请求
export const putRequest = (url, params) => {
    return axios({
        method: 'put',
        url: `${base}${url}`,
        data: params
    })
}

//传递json的get请求
export const getRequest = (url, params) => {
    return axios({
        method: 'get',
        url: `${base}${url}`,
        data: params
    })
}

//传递json的delete请求
export const deleteRequest = (url, params) => {
    return axios({
        method: 'delete',
        url: `${base}${url}`,
        data: params
    })
}
```

式使用请求

main.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'

// 引入ElementUI
import ElementUI from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'

Vue.config.productionTip = false
// 安装ElementUI
Vue.use(ElementUI)

import {postRequest} from "./utils/api";
import {putRequest} from "./utils/api";
import {getRequest} from "./utils/api";
import {deleteRequest} from "./utils/api";

//插件
Vue.prototype.postRequest = postRequest;
Vue.prototype.putRequest = putRequest;
Vue.prototype.getRequest = getRequest;
Vue.prototype.deleteRequest = deleteRequest;

new Vue({
  router,
  render: h => h(App)
}).$mount('#app')
```

修改登录请求

Login.vue

```
submitLogin() {
    this.$refs.loginForm.validate((valid) => {
        if (valid) {
            this.postRequest('/login', this.loginForm).then(resp => {
                if (resp) {
                    //存储用户token
                    const tokenStr = resp.obj.tokenHead+resp.obj.token;
                    window.sessionStorage.setItem('tokenStr', tokenStr);
                    this.$router.replace('/home');
                }
            })
        } else {
            this.$message.error('请输入所有字段');
            return false;
        }
    })
}
```

首页页面

单

准备两个测试跳转组件页面

Test1.vue

```
<template>
    <div>
        Test1
    </div>
</template>

<script>
    export default {
        name: "Test1"
    }
</script>

<style scoped>

</style>
```

T

```
<template>
    <div>
        Test1
    </div>
</template>

<script>
    export default {
        name: "Test1"
    }
</script>

<style scoped>

</style>
```

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Login from '../views/Login'
import Home from '../views/Home'
import Test1 from '../views/Test1'
import Test2 from '../views/Test2'

Vue.use(VueRouter)

const routes = [
    {
        path: '/',
```

```
        name: 'Login',
        component: Login
    },
    {
        path: '/home',
        name: 'Home',
        component: Home
    },
    {
        path: '/test1',
        name: 'Test1',
        component: Test1
    },
    {
        path: '/test2',
        name: 'Test2',
        component: Test2
    }
]

const router = new VueRouter({
    routes
})

export default router
```

编写导航菜单

Home.vue

```
<template>
    <div>
        <el-container>
            <el-header class="homeHeader">
                <div class="title">云E办系统</div>
            </el-header>
            <el-container>
                <el-aside width="200px">
                    <el-menu @select="menuClick">
                        <el-submenu index="1">
                            <template slot="title">
                                <i class="el-icon-location"></i>
                                <span>导航一</span>
                            </template>
                            <el-menu-item index="/test1">选项1</el-menu-item>
                            <el-menu-item index="/test2">选项2</el-menu-item>
                        </el-submenu>
                    </el-menu>
                </el-aside>
                <el-main>
                    <router-view/>
                </el-main>
            </el-container>
        </el-container>
    </div>
</template>
```

```
>
ort default {
    name: 'Home',
    methods: {
        menuClick(index) {
            this.$router.push(index);
        }
    }
}
</script>

<style>

</style>
```

select： 菜单激活回调

index： 选中菜单项的 index

index Path： 选中菜单项以及所属父菜单项的index数组

测试

页面跳转Bug解决

点击选项后跳转的页面没有出现在右边而是直接跳转到新的页面

原因： <router-view/> 标签同时出现在 Home.vue 和 App.vue 中，所以默认vue的路由页面跳转是跳到 App.vue 中的 <router-view/> 标签

解决：修改路由配置,将 Test1 和 Test2 放在 Home 的子路由下面

router 目录下的 index.js

```
const routes = [
    {
        path: '/',
        name: 'Login',
        component: Login
    },
    {
        path: '/home',
        name: 'Home',
        component: Home,
        children:[
            {
                path: '/test1',
                name: 'Test1',
                component: Test1
            },
            {
                path: '/test2',
                name: 'Test2',
                component: Test2
            }
        ]
    }
]
```

系统

渲染路由数据

我们需要频繁添加菜单选项的时候会发现操作的步骤比较重复，因此我们可以将菜单和路由数据统一起来，将路由数据动态渲染到菜单上

router 目录下的 index.js

```
const routes = [
    {
        path: '/',
        name: 'Login',
        component: Login,
        hidden:true
    },
    {
        path: '/home',
        name: '导航一',
        component: Home,
        children: [
            {
                path: '/test1',
                name: '选项1',
                component: Test1
            },
            {
                path: '/test2',
                name: '选项2',
                component: Test2
            }
```

```
]
```

Home.vue

```
<el-menu router>
    <el-submenu index="1" v-for="(item,index) in this.$router.options.routes" v-
if="!item.hidden"
                :key="index">
        <template slot="title">
            <i class="el-icon-location"></i>
            <span>{{item.name}}</span>
        </template>
        <el-menu-item :index="children.path" v-for="(children,indexj) in
item.children"
                      :key="indexj">{{children.name}}
        </el-menu-item>
    </el-submenu>
</el-menu>
```

router： 是否使用 vue-router 的模式，启用该模式会在激活导航时以 index 作为 path 进行路由跳转。我们可以取消之前的 select 事件

获取菜单接口数据

安装Vuex

个专为 Vue.js 应用程序开发的 **状态管理模式**。它采用集中式存储管理应用的所有组件的状态，并以相应的规则保证状态以一种可预测的方式发生变化。

安装Vuex

```
npm install vuex --save
```

```
C:\Users\Administrator\Desktop\voa_front\002_code\vueoa>npm install vuex --save
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.12 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.12: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ vuex@3.1.3
added 1 package from 1 contributor in 8.113s

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

配置Vuex

在 src 目录下创建一个名为 store 的目录并新建一个名为 index.js 文件用来配置 Vuex

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

export default new Vuex.Store({
    state: {
        routes: []
    },

    mutations: {
        initRoutes(state, data) {
            state.routes = data;
        }
    },

    actions: {}
})
```

state： 全局state对象,用于保存所有组件的公共数据
getters： 监听state值的最新状态（计算属性）
mutations： 唯一可以改变state值的方法(同步执行)
actions： 异步执行mutations方法

修改 main.js 增加刚才配置的 store/index.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

// 引入ElementUI
import ElementUI from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'

Vue.config.productionTip = false
// 安装ElementUI
```

```
import {postRequest} from "./utils/api";
import {putRequest} from "./utils/api";
import {getRequest} from "./utils/api";
import {deleteRequest} from "./utils/api";

//插件
Vue.prototype.postRequest = postRequest;
Vue.prototype.putRequest = putRequest;
Vue.prototype.getRequest = getRequest;
Vue.prototype.deleteRequest = deleteRequest;

new Vue({
    router,
    store,
    render: h => h(App)
}).$mount('#app')
```

封装菜单请求工具类

后端接口返回的数据中 component 的值为String，我们需要将其转换为前端所需的对象并且我们需要将数据放入到路由的配置里。所以我们需要封装菜单请求工具类实现我们的需求。

utils 目录下新建 menus.js

```
import {getRequest} from "./api";

export const initMenu = (router, store) => {
    if (store.state.routes.length > 0) {
        return;
    }
    getRequest("/system/config/menu").then(data => {
        if (data) {
            //格式化router
            let fmtRoutes = formatRoutes(data);
            //添加到router
            router.addRoutes(fmtRoutes);
            //将数据存入vuex
            store.commit('initRoutes',fmtRoutes);
        }
    })
}

export const formatRoutes = (routes) => {
    let fmRoutes = [];
    routes.forEach(router => {
        let {
            path,
            component,
            name,
            meta,
            iconCls,
            children,
        } = router;
        if (children && children instanceof Array) {
            //递归
```

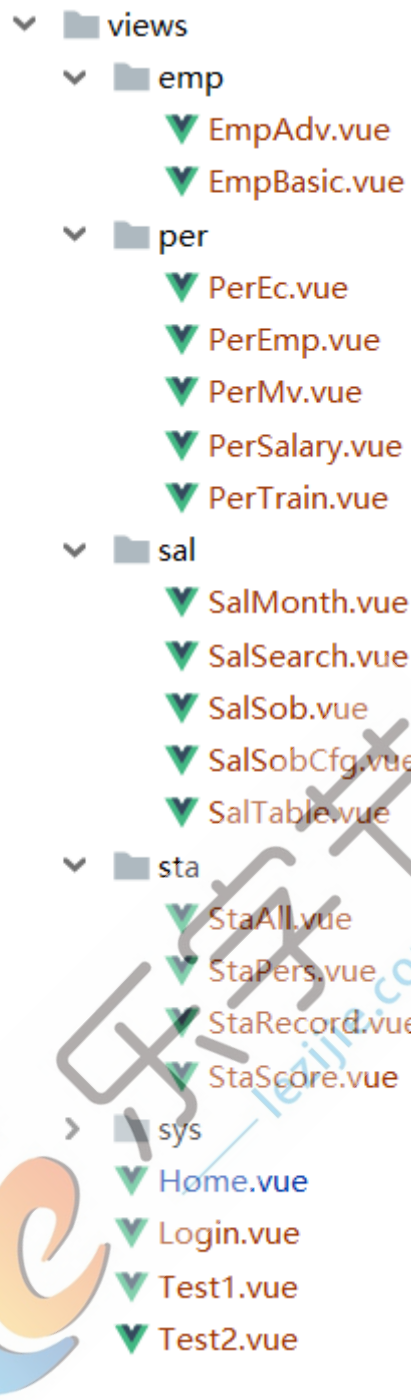
```
            children = formatRoutes(children);
        }
        let fmRouter = {
            path: path,
            name: name,
            meta: meta,
            iconCls: iconCls,
            children: children,
            component(resolve) {
                require(['../views/' + component + '.vue'], resolve);
            }
        }
        fmRoutes.push(fmRouter);
    })
    return fmRoutes;
}
```

添加页面

根据表格添加对应页面

7	/employee/basic/**	/emp/basic	EmpBasic	基本资料
8	/employee/advanced/**	/emp/adv	EmpAdv	高级资料
9	/personnel/emp/**	/per/emp	PerEmp	员工资料
10	/personnel/ec/**	/per/ec	PerEc	员工奖惩
11	/personnel/train/**	/per/train	PerTrain	员工培训
12	/personnel/salary/**	/per/salary	PerSalary	员工调薪
13	/personnel/remove/**	/per/mv	PerMv	员工调动
14	/salary/sob/**	/sal/sob	SalSob	工资账套管理
15	/salary/sobcfg/**	/sal/sobcfg	SalSobCfg	员工账套设置
16	/salary/table/**	/sal/table	SalTable	工资表管理
17	/salary/month/**	/sal/month	SalMonth	月末处理
18	/salary/search/**	/sal/search	SalSearch	工资表查询
19	/statistics/all/**	/sta/all	StaAll	综合信息统计
20	/statistics/score/**	/sta/score	StaScore	员工积分统计
21	/statistics/personnel/**	/sta/pers	StaPers	人事信息统计
22	/statistics/recored/**	/sta/record	StaRecord	人事记录统计
23	/system/basic/**	/sys/basic	SysBasic	基础信息设置
24	/system/cfg/**	/sys/cfg	SysCfg	系统管理
25	/system/log/**	/sys/log	SysLog	操作日志管理
26	/system/admin/**	/sys/admin	SysAdmin	操作员管理
27	/system/data/**	/sys/data	SysData	备份恢复数据库
28	/system/init/**	/sys/init	SysInit	初始化数据库

完成后目录如图



完善菜单请求工具类

我们现在的页面是在不同的目录下，而之前的工具类加载是直接在 views 目录下加载，这样无法找到对应页面，我们可以根据 component 的命名区分不同的子目录。

menus.js

```
import {getRequest} from "./api";

export const initMenu = (router, store) => {
    if (store.state.router.length > 0) {
        return;
    }
    getRequest("/system/config/menu").then(data => {
        if (data) {
            //格式化router
            let fmtRoutes = formatRoutes(data);
            //添加到router
            router.addRoutes(fmtRoutes);
```

```
            //将数据存入vuex
            store.commit('initRoutes', fmtRoutes);
        }
    })
}

export const formatRoutes = (routes) => {
    let fmRoutes = [];
    routes.forEach(router => {
        let {
            path,
            component,
            name,
            meta,
            iconCls,
            children,
        } = router;
        if (children && children instanceof Array) {
            //递归
            children = formatRoutes(children);
        }
        let fmRouter = {
            path: path,
            name: name,
            meta: meta,
            iconCls: iconCls,
            children: children,
            component(resolve) {
                if (component.startsWith("Emp")) {
                    require(['../views/emp/' + component + '.vue'], resolve);
                } else if (component.startsWith("Per")) {
                    require(['../views/per/' + component + '.vue'], resolve);
                } else if (component.startsWith("Sal")) {
                    require(['../views/sal/' + component + '.vue'], resolve);
                } else if (component.startsWith("Sta")) {
                    require(['../views/sta/' + component + '.vue'], resolve);
                } else if (component.startsWith("Sys")) {
                    require(['../views/sys/' + component + '.vue'], resolve);
                }
            }
        }
        fmRoutes.push(fmRouter);
    })
    return fmRoutes;
}
```

导航守卫

菜单数据在用户点击刷新按钮时可能出现丢失的情况，解决办法

1. 每个页面添加初始化菜单的方法，这显然很麻烦
2. 路由导航守卫

`vue-router` 提供的导航守卫主要用来通过跳转或取消的方式守卫导航。有多种机会植入路由导航过程中：全局的, 单个路由独享的, 或者组件级的。

记住**参数或查询的改变并不会触发进入/离开的导航守卫**。我们可以通过观察 `$route` 对象来应对这些变化，或使用 `beforeRouteUpdate` 的组件内守卫。

我们可以使用 router.beforeEach 注册一个全局前置守卫：

```
const router = new VueRouter({ ... })

router.beforeEach((to, from, next) => {
  // ...
})
```

当一个导航触发时，全局前置守卫按照创建顺序调用。守卫是异步解析执行，此时导航在所有守卫 resolve 完之前一直处于 **等待中**。

每个守卫方法接收三个参数：

to: Route：即将要进入的目标 路由对象

from: Route：当前导航正要离开的路由

next: Function：一定要调用该方法来 **resolve** 这个钩子。执行效果依赖 next 方法的调用参数。

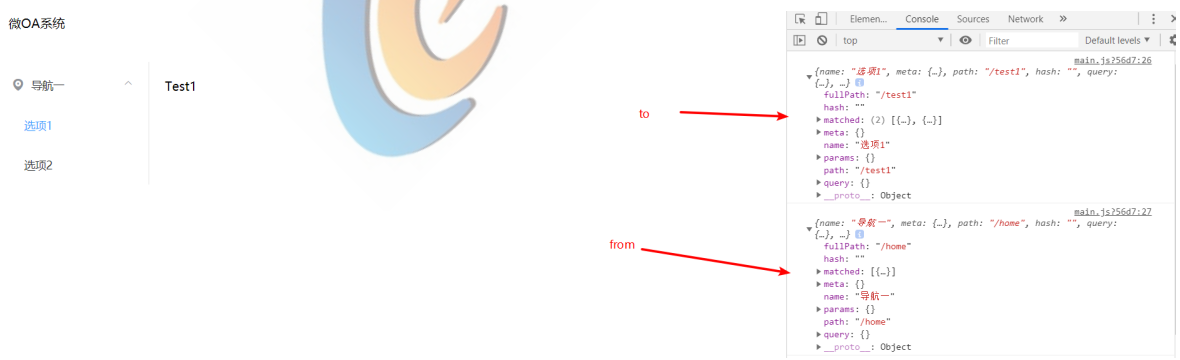
next()：进行管道中的下一个钩子。如果全部钩子执行完了，则导航的状态就是 **confirmed** (确认的)。

next(false)：中断当前的导航。如果浏览器的 URL 改变了 (可能是用户手动或者浏览器后退按钮)，那么 URL 地址会重置到 from 路由对应的地址。

next('/') **或者** next({ path: '/' })：跳转到一个不同的地址。当前的导航被中断，然后进行一个新的导航。你可以向 next 传递任意位置对象，且允许设置诸如 replace: true、 name: 'home' 之类的选项以及任何用在 router-link 的 to prop 或 router.push 中的选项。

next(error)：(2.4.0+) 如果传入 next 的参数是一个 Error 实例，则导航会被终止且该错误会被传递给 router.onError() 注册过的回调。

确保要调用 next 方法，否则钩子就不会被 resolved。



main.js

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

// 引入ElementUI
import ElementUI from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'

Vue.config.productionTip = false
```

lementUI
(ElementUI)

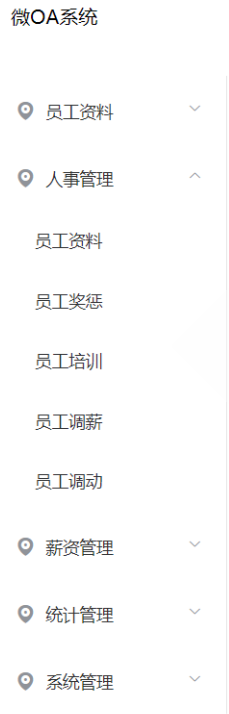
```
import {postRequest} from "./utils/api";
import {putRequest} from "./utils/api";
import {getRequest} from "./utils/api";
import {deleteRequest} from "./utils/api";
import {initMenu} from "./utils/menu";

//插件
Vue.prototype.postRequest = postRequest;
Vue.prototype.putRequest = putRequest;
Vue.prototype.getRequest = getRequest;
Vue.prototype.deleteRequest = deleteRequest;

router.beforeEach((to, from, next)=>{
   if (to.path=='/'){
       next()
   } else {
       initMenu(router,store);
       next();
   }
})


new Vue({
    router,
    store,
    render: h => h(App)
}).$mount('#app')
```

Home.vue

```
<template>
    <div>
        <el-container>
            <el-header class="homeHeader">
                <div class="title">云E办系统</div>
            </el-header>
            <el-container>
                <el-aside width="200px">
                    <el-menu router unique-opened>
                        <el-submenu :index="index+''" v-for="(item,index) in
routes"
                                    v-if="!item.hidden"
                                    :key="index">
                            <template slot="title">
                                <i class="el-icon-location"></i>
                                <span>{{item.name}}</span>
                            </template>
                            <el-menu-item :index="children.path" v-for="
(children,indexj) in item.children"
                                          :key="indexj">{{children.name}}
                            </el-menu-item>
                        </el-submenu>
                    </el-menu>
                </el-aside>
```

```
                <el-main>
                    <router-view/>
                </el-main>
            </el-container>
        </el-container>
    </div>
</template>

<script>
    export default {
        name: 'Home',
        computed: {
            routes() {
                return this.$store.state.routes;
            }
        }
    }
</script>

<style>

</style>
```

测试



菜单图标

我们使用了 Font Awesome 的图标做为菜单图标，使用前先安装 Font Awesome

```
npm install font-awesome
```

导入 Font Awesome (main.js)

```
import 'font-awesome/css/font-awesome.min.css'
```

```
<i style="color: #1accff;margin-right: 5px" :class="item.iconCls"></i>
```

测试

微OA系统

页面跳转Bug解决

我们发现点击菜单无法跳转到对应的页面，主要原因是我们之前封装的工具类里面没有对 Home 做相应的判断导致

menus.js

```
export const formatRoutes = (routes) => {
    let fmRoutes = [];
    routes.forEach(router => {
        let {
            path,
            component,
            name,
            meta,
            iconCls,
            children,
        } = router;
        if (children && children instanceof Array) {
            //递归
            children = formatRoutes(children);
        }
        let fmRouter = {
            path: path,
            name: name,
            meta: meta,
            iconCls: iconCls,
            children: children,
            component(resolve) {
                if (component.startsWith("Home")) {
```

```
                    require(['../views/' + component + '.vue'], resolve);
                } else if (component.startsWith("Emp")) {
                    require(['../views/emp/' + component + '.vue'], resolve);
                } else if (component.startsWith("Per")) {
                    require(['../views/per/' + component + '.vue'], resolve);
                } else if (component.startsWith("Sal")) {
                    require(['../views/sal/' + component + '.vue'], resolve);
                } else if (component.startsWith("Sta")) {
                    require(['../views/sta/' + component + '.vue'], resolve);
                } else if (component.startsWith("Sys")) {
                    require(['../views/sys/' + component + '.vue'], resolve);
                }
            }
        }
        fmRoutes.push(fmRouter);
    })
    return fmRoutes;
}
```

测试

标题

获取用户信息

之前登录时只是获取用户token，并没有获取用户信息进行展示。我们可以在全局前置守卫时获取用户信息并存储在 SessionStorage 中方便使用

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'

// 引入ElementUI
import ElementUI from 'element-ui'
import 'element-ui/lib/theme-chalk/index.css'

import 'font-awesome/css/font-awesome.min.css'

Vue.config.productionTip = false
// 安装ElementUI
Vue.use(ElementUI)

import {postRequest} from "./utils/api";
import {putRequest} from "./utils/api";
import {getRequest} from "./utils/api";
import {deleteRequest} from "./utils/api";
import {initMenu} from "./utils/menu";

//插件
Vue.prototype.postRequest = postRequest;
Vue.prototype.putRequest = putRequest;
Vue.prototype.getRequest = getRequest;
Vue.prototype.deleteRequest = deleteRequest;

router.beforeEach((to, from, next) => {
    if (window.sessionStorage.getItem("tokenStr")) {
        initMenu(router, store);
        if (!window.sessionStorage.getItem("user")) {
            return getRequest('/admin/info').then(resp => {
                if (resp) {
                    //存入用户信息
                    window.sessionStorage.setItem("user", JSON.stringify(resp));
                    next();
                }
            });
        }
        next();
    } else {
        if (to.path == '/') {
            next();
        }
    }
})

new Vue({
    router,
    store,
    render: h => h(App)
}).$mount('#app')
```

首页

```
<template>
    <div>
        <el-container>
            <el-header class="homeHeader">
                <div class="title">云E办系统</div>
                <el-dropdown class="userInfo" @command="commandHandler">
                  <span class="el-dropdown-link">
                    {{user.name}}<i><img :src="user.userFace"></i>
                  </span>
                    <el-dropdown-menu slot="dropdown">
                        <el-dropdown-item command="userinfo">个人中心</el-
dropdown-item>
                        <el-dropdown-item command="setting">设置</el-dropdown-
item>
                        <el-dropdown-item command="logout">注销登录</el-dropdown-
item>
                    </el-dropdown-menu>
                </el-dropdown>
            </el-header>
            <el-container>
                <el-aside width="200px">
                    <el-menu router unique-opened>
                        <el-submenu :index="index+''" v-for="(item,index) in
routes"
                                    v-if="!item.hidden"
                                    :key="index">
                            <template slot="title">
                                <i style="color: #1accff;margin-right: 5px"
:class="item.iconCls"></i>
                                <span>{{item.name}}</span>
                            </template>
                            <el-menu-item :index="children.path" v-for="
(children,indexj) in item.children"
                                          :key="indexj">{{children.name}}
                            </el-menu-item>
                        </el-submenu>
                    </el-menu>
                </el-aside>
                <el-main>
                    <router-view/>
                </el-main>
            </el-container>
        </el-container>
    </div>
</template>

<script>
    export default {
        name: 'Home',
        data() {
            return {
                user: JSON.parse(window.sessionStorage.getItem("user"))
            }
        },
        computed: {
```

```
            routes() {
                return this.$store.state.routes;
            }
        },
        methods: {
            commandHandler(cmd) {
                if (cmd == 'logout') {
                    this.$confirm('此操作将注销登录，是否继续？', '提示', {
                        confirmButtonText: '确定',
                        cancelButtonText: '取消',
                        type: 'warning'
                    }).then(() => {
                        //注销
                        this.postRequest("/logout");
                        //清除用户信息
                        window.sessionStorage.removeItem("user");
                        window.sessionStorage.removeItem("tokenStr");
                         //清空菜单
                        this.$store.commit('initRoutes',[]);
                        //跳转登录页
                        this.$router.replace("/")
                    }).catch(() => {
                        this.$message({
                            type: 'info',
                            message: '已取消操作'
                        })
                    })
                }
            }
        }
    }
</script>

<style>
    .homeHeader {
        background-color: #409eff;
        display: flex;
        align-items: center;
        justify-content: space-between;
        padding: 0px 15px;
        box-sizing: border-box;
    }

    .homeHeader .title {
        font-size: 30px;
        font-family: 华文行楷;
        color: #ffffff;
    }

    .homeHeader .userInfo {
        cursor: pointer;
    }

    .el-dropdown-link img {
        width: 48px;
        height: 48px;
        border-radius: 24px;
        margin-left: 8px;
```

```
    .el-dropdown-link {
        display: flex;
        align-items: center;
    }
</style>
```

command： 点击菜单项触发的事件回调

测试

首页完善

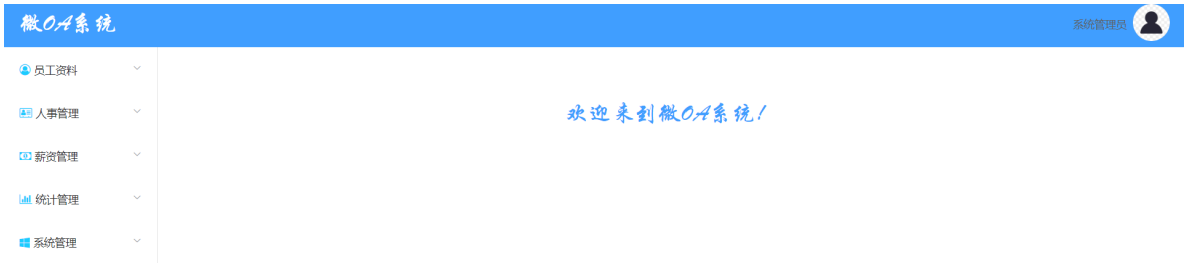
面包屑效果

Home.vue

```
<el-main>
    <el-breadcrumb separator-class="el-icon-arrow-right" v-
if="this.$router.currentRoute.path!='/home'">
        <el-breadcrumb-item :to="{ path: '/home' }">首页</el-breadcrumb-item>
        <el-breadcrumb-item>{{this.$router.currentRoute.name}}</el-breadcrumb-
item>
    </el-breadcrumb>
    <div class="homeWelcome" v-if="this.$router.currentRoute.path=='/home'">
        欢迎来到云E办系统!
    </div>
    <router-view/>
</el-main>


<style>
    .homeWelcome{
        text-align: center;
        font-size: 30px;
        font-family: 华文行楷;
        color: #409eff;
        padding-top: 50px;
    }
</style>
```

路由跳转对象，同 vue-router 的 to

效果



未登录的Bug

未登录时直接访问菜单会出现空白页面。

我们可以让用户未登录时输入菜单地址跳转至登录页，登录成功后直接跳转至用户之前输入的菜单地址

main.js

```
router.beforeEach((to, from, next) => {
    if (window.sessionStorage.getItem("tokenStr")) {
        initMenu(router, store);
        if (!window.sessionStorage.getItem("user")) {
            return getRequest('/admin/info').then(resp => {
                if (resp) {
                    //存入用户信息
                    window.sessionStorage.setItem("user", JSON.stringify(resp));
                    next();
                }
            });
        }
        next();
    } else {
        if (to.path == '/') {
            next();
        } else {
            next('/?redirect=' + to.path);
        }
    }
})
```

Login.vue

```
submitLogin() {
    this.$refs.loginForm.validate((valid) => {
        if (valid) {
            this.loading = true;
            this.postRequest('/login', this.loginForm).then(resp => {
                this.loading = false;
                if (resp) {
                    //存储用户token
                    const tokenStr = resp.obj.tokenHead+resp.obj.token;
                    window.sessionStorage.setItem('tokenStr', tokenStr);
                    //清空菜单
                    this.$store.commit('initRoutes',[]);
                    //页面跳转
                    let path = this.$route.query.redirect;
```

```
                this.$router.replace((path == '/' || path == undefined) ?
: path);
                }
            })
        } else {
            this.$message.error('请输入所有字段');
            return false;
        }
    })
}
```

基础信息设置

定义组件

Home.vue

```
<router-view class="homeRouterView"/>

<style>
    .homeRouterView{
        margin-top: 10px;
    }
</style>
```

SysBasic.vue

```
<template>
    <div>
        <el-tabs v-model="activeName" type="card">
            <el-tab-pane label="部门管理" name="depmana"><DepMana></DepMana></el-
tab-pane>
```

```
            <el-tab-pane label="职位管理" name="posmana"><PosMana></PosMana></el-
e>
            <el-tab-pane label="职称管理" name="JoblevelMana"><JoblevelMana>
</JoblevelMana></el-tab-pane>
            <el-tab-pane label="奖惩规则" name="ecmana"><EcMana></EcMana></el-
tab-pane>
            <el-tab-pane label="权限组" name="permissmana"><PermissMana>
</PermissMana></el-tab-pane>
        </el-tabs>
    </div>
</template>

<script>
    import DepMana from '../../components/sys/basic/DepMana'
    import EcMana from '../../components/sys/basic/EcMana'
    import JoblevelMana from '../../components/sys/basic/JoblevelMana'
    import PermissMana from '../../components/sys/basic/PermissMana'
    import PosMana from '../../components/sys/basic/PosMana'

    export default {
        name: "SysBasic",
        data(){
            return{
                activeName:'PosMana'
            }
        },
        components:{
            DepMana,
            EcMana,
            JoblevelMana,
            PermissMana,
            PosMana
        }
    }
</script>

<style>

</style>
```

职位管理

页面设计

PosMana.vue

```
<template>
    <div>
        <div>
            <el-input
                    size="small"
                    class="addPosInput"
                    placeholder="添加职位..."
                    prefix-icon="el-icon-plus"
                    v-model="pos.name">
            </el-input>
            <el-button icon="el-icon-plus" size="small" type="primary">添加</el-
button>
        </div>
        <div class="posManaMain">
            <el-table
                    :data="positions"
                    border
                    stripe
                    size="small"
                    style="width: 70%">
                <el-table-column
                        prop="id"
                        label="编号"
                        width="55">
                </el-table-column>
                <el-table-column
```

```
                    prop="name"
                    label="职位名称"
                    width="120">
                </el-table-column>
                <el-table-column
                    prop="createDate"
                    label="创建时间">
                </el-table-column>
            </el-table>
        </div>
    </div>
</template>

<script>
    export default {
        name: "PosMana",
        data(){
            return{
                pos:{
                    name:''
                },
                positions: []
            }
        }
    }
</script>

<style>
    .addPosInput{
        width: 300px;
        margin-right: 8px
    }
    .posManaMain{
       margin-top: 10px;
    }

</style>
```

stripe： 是否为斑马纹 table
border： 是否带有纵向边框

效果

调用接口

```
<template>
    <div>
        <div>
            <el-input
                    size="small"
                    class="addPosInput"
                    placeholder="添加职位..."
                    prefix-icon="el-icon-plus"
                    @keydown.enter.native="addPosition"
                    v-model="pos.name">
            </el-input>
            <el-button icon="el-icon-plus" size="small" type="primary"
@click="addPosition">添加</el-button>
        </div>
        <div class="posManaMain">
            <el-table
                    :data="positions"
                    border
                    stripe
                    size="small"
                    style="width: 70%">
                <el-table-column
                        type="selection"
                        width="55">
                </el-table-column>
                <el-table-column
                        prop="id"
                        label="编号"
                        width="55">
                </el-table-column>
                <el-table-column
                        prop="name"
                        label="职位名称"
                        width="120">
                </el-table-column>
                <el-table-column
                        prop="createDate"
                        label="创建时间">
                </el-table-column>
                <el-table-column label="操作">
                    <template slot-scope="scope">
                        <el-button
                                size="mini"
                                @click="handleEdit(scope.$index, scope.row)">编辑
                        </el-button>
                        <el-button
                                size="mini"
                                type="danger"
                                @click="handleDelete(scope.$index, scope.row)">
删除
                        </el-button>
                    </template>
                </el-table-column>
            </el-table>
        </div>
```

iv>
ate>

```
<script>
    export default {
        name: "PosMana",
        data() {
            return {
                pos: {
                    name: ''
                },
                positions: []
            }
        },
        mounted() {
            this.initPositions();
        },
        methods: {
            addPosition() {
                if (this.pos.name) {
                    this.postRequest('/system/basic/pos/', this.pos).then(resp
=> {
                        if (resp) {
                            this.initPositions();
                            this.pos.name = '';
                        }
                    })
                } else {
                    this.$message.error('职位名称不可以为空！');
                }
            },
            handleEdit(index, data) {

            },
            handleDelete(index, data) {
                this.$confirm('此操作将永久删除该【' + data.name + '】职位，是否继
续?', '提示', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(() => {
                    this.deleteRequest('/system/basic/pos/'+data.id).then(resp=>
{
                        if (resp){
                            this.initPositions();
                        }
                    });
                }).catch(() => {
                    this.$message({
                        type: 'info',
                        message: '已取消删除'
                    });
                });
            },
            initPositions() {
                this.getRequest('/system/basic/pos/').then(resp => {
                    if (resp) {
                        this.positions = resp;
```

```
                }
            })
        }
    }
}
</script>

<style>
    .addPosInput {
        width: 300px;
        margin-right: 8px
    }
    .posManaMain {
        margin-top: 10px;
    }

</style>
```

Scoped slot ： 可以获取到 row, column, $index 和 store（table 内部的状态管理）的数据

@keydown.enter.native： 键盘回车按下触发的事件

效果

编辑职位

PosMana.vue

```
<template>
    <div>
        <div>
            <el-input
                    size="small"
```

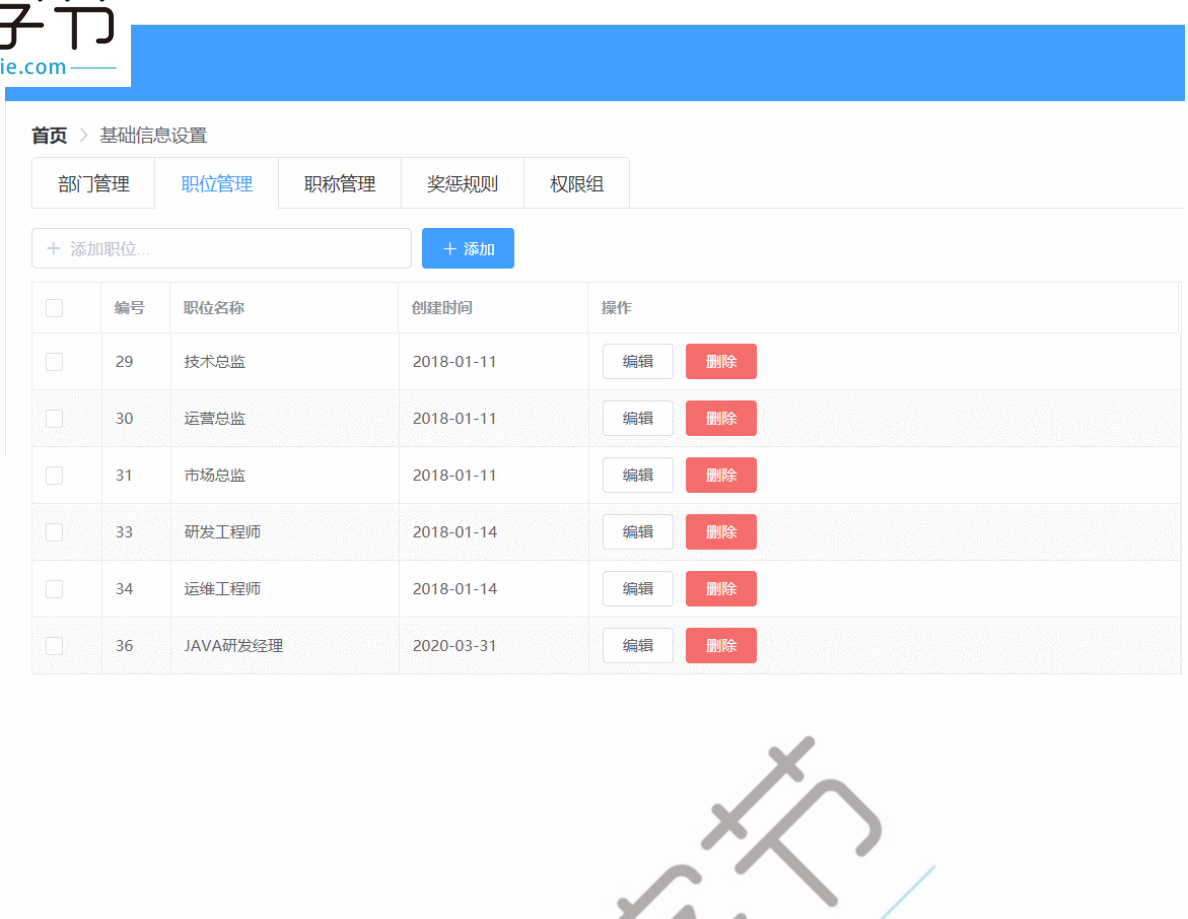
```
                    class="addPosInput"
                    placeholder="添加职位..."
                    prefix-icon="el-icon-plus"
                    @keydown.enter.native="addPosition"
                    v-model="pos.name">
                </el-input>
                <el-button icon="el-icon-plus" size="small" type="primary"
@click="addPosition">添加</el-button>
            </div>
            <div class="posManaMain">
                <el-table
                        :data="positions"
                        border
                        stripe
                        size="small"
                        style="width: 70%">
                    <el-table-column
                            type="selection"
                            width="55">
                    </el-table-column>
                    <el-table-column
                            prop="id"
                            label="编号"
                            width="55">
                    </el-table-column>
                    <el-table-column
                            prop="name"
                            label="职位名称"
                            width="180">
                    </el-table-column>
                    <el-table-column
                            prop="createDate"
                            width="150"
                            label="创建时间">
                    </el-table-column>
                    <el-table-column label="操作">
                        <template slot-scope="scope">
                            <el-button
                                    size="mini"
                                    @click="showEditView(scope.$index, scope.row)">
编辑
                            </el-button>
                            <el-button
                                    size="mini"
                                    type="danger"
                                    @click="handleDelete(scope.$index, scope.row)">
删除
                            </el-button>
                        </template>
                    </el-table-column>
                </el-table>
            </div>
            <el-dialog
                    title="编辑职位"
                    :visible.sync="dialogVisible"
                    width="30%">
                <div>
                    <el-tag>职位名称</el-tag>
```

```
                <el-input v-model="updatePos.name" size="small"
updatePosInput"></el-input>
            </div>
            <span slot="footer" class="dialog-footer">
                <el-button size="small" @click="dialogVisible = false">取 消</el-
button>
                <el-button size="small" type="primary" @click="doUpdate">确 定
</el-button>
            </span>
        </el-dialog>
    </div>
</template>

<script>
    export default {
        name: "PosMana",
        data() {
            return {
                pos: {
                    name: ''
                },
                dialogVisible: false,
                updatePos: {
                    name: ''
                },
                positions: []
            }
        },
        mounted() {
            this.initPositions();
        },
        methods: {
            addPosition() {
                if (this.pos.name) {
                    this.postRequest('/system/basic/pos/', this.pos).then(resp
=> {
                        if (resp) {
                            this.initPositions();
                            this.pos.name = '';
                        }
                    })
                } else {
                    this.$message.error('职位名称不可以为空！');
                }
            },
            showEditView(index, data) {
                Object.assign(this.updatePos,data);
                this.updatePos.createDate = '';
                this.dialogVisible = true;
            },
            doUpdate() {
                this.putRequest('/system/basic/pos/', this.updatePos).then(resp
=> {
                    if (resp) {
                        this.initPositions();
                        this.updatePos.name = '';
                        this.dialogVisible = false;
                    }
```

```
                })
            },
            handleDelete(index, data) {
                this.$confirm('此操作将永久删除该【' + data.name + '】职位，是否继
续?', '提示', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(() => {
                    this.deleteRequest('/system/basic/pos/' + data.id).then(resp
=> {
                        if (resp) {
                            this.initPositions();
                        }
                    });
                }).catch(() => {
                    this.$message({
                        type: 'info',
                        message: '已取消删除'
                    });
                });
            },
            initPositions() {
                this.getRequest('/system/basic/pos/').then(resp => {
                    if (resp) {
                        this.positions = resp;
                    }
                })
            }
        }
    }
</script>

<style>
    .addPosInput {
        width: 300px;
        margin-right: 8px
    }

    .posManaMain {
        margin-top: 10px;
    }

    .updatePosInput {
        width: 200px;
        margin-left: 8px;
    }

</style>
```

visible：是否显示 Dialog，支持 .sync 修饰符

效果



批量删除

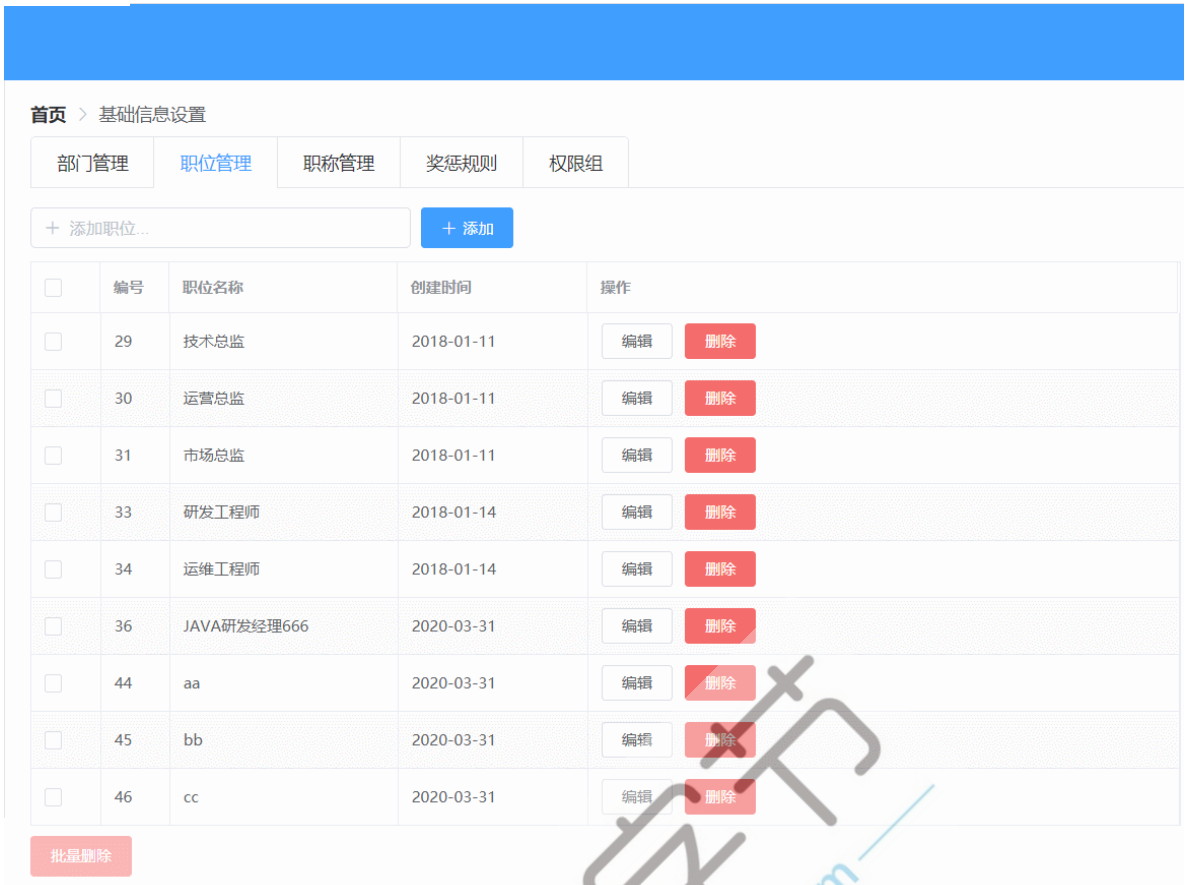
PosMana.vue

```
<template>
    <div>
        <div>
            <el-input
                    size="small"
                    class="addPosInput"
                    placeholder="添加职位..."
                    prefix-icon="el-icon-plus"
                    @keydown.enter.native="addPosition"
                    v-model="pos.name">
            </el-input>
            <el-button icon="el-icon-plus" size="small" type="primary"
@click="addPosition">添加</el-button>
        </div>
        <div class="posManaMain">
            <el-table
                    :data="positions"
                    border
                    stripe
                    size="small"
                    @selection-change="handleSelectionChange"
                    style="width: 70%">
                <el-table-column
                        type="selection"
                        width="55">
                </el-table-column>
                <el-table-column
                        prop="id"
```

```
                        label="编号"
                        width="55">
                </el-table-column>
                <el-table-column
                        prop="name"
                        label="职位名称"
                        width="120">
                </el-table-column>
                <el-table-column
                        prop="createDate"
                        label="创建时间">
                </el-table-column>
                <el-table-column label="操作">
                    <template slot-scope="scope">
                        <el-button
                                size="mini"
                                @click="showEditView(scope.$index, scope.row)">
编辑
                        </el-button>
                        <el-button
                                size="mini"
                                type="danger"
                                @click="handleDelete(scope.$index, scope.row)">
删除
                        </el-button>
                    </template>
                </el-table-column>
            </el-table>
            <el-button type="danger" size="small" style="margin-top: 8px"
:disabled="multipleSelection.length==0"
                       @click="deleteMany">批量删除
            </el-button>
        </div>
        <el-dialog
                title="编辑职位"
                :visible.sync="dialogVisible"
                width="30%">
            <div>
                <el-tag>职位名称</el-tag>
                <el-input v-model="updatePos.name" size="small"
class="updatePosInput"></el-input>
            </div>
            <span slot="footer" class="dialog-footer">
                <el-button size="small" @click="dialogVisible = false">取 消</el-
button>
                <el-button size="small" type="primary" @click="doUpdate">确 定
</el-button>
            </span>
        </el-dialog>
    </div>
</template>

<script>
    export default {
        name: "PosMana",
        data() {
            return {
                pos: {
```

```
                name: ''
            },
            dialogVisible: false,
            updatePos: {
                name: ''
            },
            multipleSelection: [],
            positions: []
        }
    },
    mounted() {
        this.initPositions();
    },
    methods: {
        deleteMany() {
            this.$confirm('此操作将永久删除【' + this.multipleSelection.length
+ '】条记录，是否继续?', '提示', {
                confirmButtonText: '确定',
                cancelButtonText: '取消',
                type: 'warning'
            }).then(() => {
                let ids = '?';
                this.multipleSelection.forEach(item => {
                    ids += 'ids=' + item.id + '&';
                })
                this.deleteRequest('/system/basic/pos/' + ids).then(resp =>
{
                    if (resp) {
                        this.initPositions();
                    }
                });
            }).catch(() => {
                this.$message({
                    type: 'info',
                    message: '已取消删除'
                });
            });
        },
        handleSelectionChange(val) {
            this.multipleSelection = val;
        },
        addPosition() {
            if (this.pos.name) {
                this.postRequest('/system/basic/pos/', this.pos).then(resp
=> {
                    if (resp) {
                        this.initPositions();
                        this.pos.name = '';
                    }
                })
            } else {
                this.$message.error('职位名称不可以为空！');
            }
        },
        showEditView(index, data) {
            Object.assign(this.updatePos, data);
            this.updatePos.createDate = '';
            this.dialogVisible = true;
```

```
            },
            doUpdate() {
                this.putRequest('/system/basic/pos/', this.updatePos).then(resp
=> {
                    if (resp) {
                        this.initPositions();
                        this.updatePos.name = '';
                        this.dialogVisible = false;
                    }
                })
            },
            handleDelete(index, data) {
                this.$confirm('此操作将永久删除该【' + data.name + '】职位, 是否继
续?', '提示', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(() => {
                    this.deleteRequest('/system/basic/pos/' + data.id).then(resp
=> {
                        if (resp) {
                            this.initPositions();
                        }
                    });
                }).catch(() => {
                    this.$message({
                        type: 'info',
                        message: '已取消删除'
                    });
                });
            },
            initPositions() {
                this.getRequest('/system/basic/pos/').then(resp => {
                    if (resp) {
                        this.positions = resp;
                    }
                })
            }
        }
    }
</script>

<style>
    .addPosInput {
        width: 300px;
        margin-right: 8px
    }

    .posManaMain {
        margin-top: 10px;
    }

    .updatePosInput {
        width: 200px;
        margin-left: 8px;
    }
</style>
```



职称管理

页面设计

JoblevelMana.vue

```
<template>
    <div>
        <div>
            <el-input size="small" v-model="jl.name" style="width: 300px"
prefix-icon="el-icon-plus" placeholder="添加职称..."></el-input>
            <el-select v-model="jl.titleLevel" placeholder="职称等级"
size="small" style="margin-left: 6px;margin-right: 6px">
                <el-option
                        v-for="item in titleLevels"
                        :key="item"
                        :label="item"
                        :value="item">
                </el-option>
            </el-select>
            <el-button type="primary" icon="el-icon-plus" size="small">添加</el-
button>
        </div>
        <div style="margin-top: 10px">
            <el-table
                    :data="jls"
                    border
                    stripe
                    size="small"
                    style="width: 80%">
```

```
                <el-table-column
                        prop="id"
                        label="编号"
                        width="55">
                </el-table-column>
                <el-table-column
                        prop="name"
                        label="职称名称"
                        width="150">
                </el-table-column>
                <el-table-column
                        prop="titleLevel"
                        label="职称级别"
                        width="150">
                </el-table-column>
                <el-table-column
                        prop="createDate"
                        label="创建时间"
                        width="150">
                </el-table-column>
                <el-table-column
                        label="操作">
                    <template slot-scope="scope">
                        <el-button size="small">编辑</el-button>
                        <el-button size="small" type="danger">删除</el-button>
                    </template>
                </el-table-column>
            </el-table>
        </div>
    </div>
</template>

<script>
    export default {
        name: "JoblevelMana",
        data(){
            return{
                jl:{
                    name:'',
                    titleLevel:''
                },
                jls:[],
                titleLevels:[
                    '正高级',
                    '副高级',
                    '中级',
                    '初级',
                    '员级',
                ]
            }
        }
    }
</script>

<style >

</style>
```

调用接口

JoblevelMana.vue

```
<template>
    <div>
        <div>
            <el-input size="small" v-model="jl.name" style="width: 300px"
prefix-icon="el-icon-plus"
                      placeholder="添加职称..."></el-input>
            <el-select v-model="jl.titleLevel" placeholder="职称等级"
size="small"
                       style="margin-left: 6px;margin-right: 6px">
                <el-option
                        v-for="item in titleLevels"
                        :key="item"
                        :label="item"
                        :value="item">
                </el-option>
            </el-select>
            <el-button type="primary" icon="el-icon-plus" size="small"
@click="addJobLevel">添加</el-button>
        </div>
        <div style="margin-top: 10px">
            <el-table
                    :data="jls"
                    border
                    stripe
                    size="small"
                    style="width: 80%">
                <el-table-column
                        prop="id"
                        label="编号"
                        width="55">
                </el-table-column>
                <el-table-column
                        prop="name"
                        label="职称名称"
                        width="150">
                </el-table-column>
                <el-table-column
                        prop="titleLevel"
                        label="职称级别"
                        width="150">
                </el-table-column>
```

```
                <el-table-column
                        prop="createDate"
                        label="创建时间"
                        width="150">
                </el-table-column>
                <el-table-column
                        prop="enabled"
                        label="是否启用"
                        width="150">
                    <template slot-scope="scope">
                        <el-tag v-if="scope.row.enabled" type="success">已启用
</el-tag>
                        <el-tag v-else type="danger">未启用</el-tag>
                    </template>
                </el-table-column>
                <el-table-column
                        label="操作">
                    <template slot-scope="scope">
                        <el-button size="small"
@click="showEditView(scope.row)">编辑</el-button>
                        <el-button size="small" type="danger"
@click="deleteHandler(scope.row)">删除</el-button>
                    </template>
                </el-table-column>
            </el-table>
        </div>
        <el-dialog
                title="编辑职称"
                :visible.sync="dialogVisible"
                width="30%">
            <div>
                <table>
                    <tr>
                        <td>
                            <el-tag>职称名</el-tag>
                        </td>
                        <td>
                            <el-input v-model="updateJl.name" size="small"></el-
input>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <el-tag>职称级别</el-tag>
                        </td>
                        <td>
                            <el-select v-model="updateJl.titleLevel"
placeholder="职称等级" size="small"
                                       style="margin-left: 6px;margin-right:
6px">
                                <el-option
                                        v-for="item in titleLevels"
                                        :key="item"
                                        :label="item"
                                        :value="item">
                                </el-option>
                            </el-select>
                        </td>
```

```
                    </tr>
                    <tr>
                        <td>
                            <el-tag>是否启用</el-tag>
                        </td>
                        <td>
                            <el-switch v-model="updateJl.enabled" active-
color="#13ce66" inactive-color="#ff4949"
                                       active-text="启用" inactive-text="禁用">
</el-switch>
                        </td>
                    </tr>
                </table>

            </div>
            <span slot="footer" class="dialog-footer">
                <el-button size="small" @click="dialogVisible = false">取 消</el-
button>
                <el-button size="small" type="primary" @click="doUpdate">确 定
</el-button>
            </span>
        </el-dialog>
    </div>
</template>

<script>
    export default {
        name: "JobLevelMana",
        data() {
            return {
                jl: {
                    name: '',
                    titleLevel: ''
                },
                updateJl: {
                    name: '',
                    titleLevel: '',
                    enabled: false
                },
                dialogVisible: false,
                jls: [],
                titleLevels: [
                    '正高级',
                    '副高级',
                    '中级',
                    '初级',
                    '员级',
                ]
            }
        },
        mounted() {
            this.initJls();
        },
        methods: {
            doUpdate() {
                this.putRequest('/system/basic/joblevel/',
this.updateJl).then(resp => {
                    if (resp) {
```

```
                        this.initJls();
                        this.dialogVisible = false;
                    }
                })
            },
            showEditView(data) {
                Object.assign(this.updateJl, data);
                this.dialogVisible = true;
            },
            deleteHandler(data) {
                this.$confirm('此操作将永久删除【' + data.name + '】职称, 是否继续?',
'提示', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(() => {
                    this.deleteRequest('/system/basic/joblevel/' +
data.id).then(resp => {
                        if (resp) {
                            this.initJls();
                        }
                    });
                }).catch(() => {
                    this.$message({
                        type: 'info',
                        message: '已取消删除'
                    });
                });
            },
            addJobLevel() {
                if (this.jl.name && this.jl.titleLevel) {
                    this.postRequest('/system/basic/joblevel/',
this.jl).then(resp => {
                        if (resp) {
                            this.initJls();
                        }
                    })
                } else {
                    this.$message.error("字段不能为空！");
                }
            },
            initJls() {
                this.getRequest('/system/basic/joblevel/').then(resp => {
                    if (resp) {
                        this.jls = resp;
                        this.jl = {
                            name: '',
                            titleLevel: ''
                        }
                    }
                })
            }
        }
    }
</script>

<style scoped>
```

active-text： switch 打开时的文字描述
inactive-text： switch 关闭时的文字描述
active-color： switch 打开时的背景色
inactive-color： switch 关闭时的背景色

效果

批量删除

JoblevelMana.vue

```
<template>
    <div>
        <div>
            <el-input size="small" v-model="jl.name" style="width: 300px"
prefix-icon="el-icon-plus"
                      placeholder="添加职称..."></el-input>
            <el-select v-model="jl.titleLevel" placeholder="职称等级"
size="small"
                       style="margin-left: 6px;margin-right: 6px">
                <el-option
                        v-for="item in titleLevels"
                        :key="item"
                        :label="item"
                        :value="item">
                </el-option>
            </el-select>
            <el-button type="primary" icon="el-icon-plus" size="small"
@click="addJobLevel">添加</el-button>
        </div>
```

```
<div style="margin-top: 10px">
    <el-table
            :data="jls"
            border
            stripe
            size="small"
            @selection-change="handleSelectionChange"
            style="width: 80%">
        <el-table-column
                type="selection"
                width="55">
        </el-table-column>
        <el-table-column
                prop="id"
                label="编号"
                width="55">
        </el-table-column>
        <el-table-column
                prop="name"
                label="职称名称"
                width="150">
        </el-table-column>
        <el-table-column
                prop="titleLevel"
                label="职称级别"
                width="150">
        </el-table-column>
        <el-table-column
                prop="createDate"
                label="创建时间"
                width="150">
        </el-table-column>
        <el-table-column
                prop="enabled"
                label="是否启用"
                width="150">
            <template slot-scope="scope">
                <el-tag v-if="scope.row.enabled" type="success">已启用
</el-tag>
                <el-tag v-else type="danger">未启用</el-tag>
            </template>
        </el-table-column>
        <el-table-column
                label="操作">
            <template slot-scope="scope">
                <el-button size="small"
@click="showEditView(scope.row)">编辑</el-button>
                <el-button size="small" type="danger"
@click="deleteHandler(scope.row)">删除</el-button>
            </template>
        </el-table-column>
    </el-table>
    <el-button type="danger" size="small" style="margin-top: 10px"
:disabled="multipleSelection.length==0"
               @click="deleteMany">批量删除
    </el-button>
</div>
<el-dialog
```

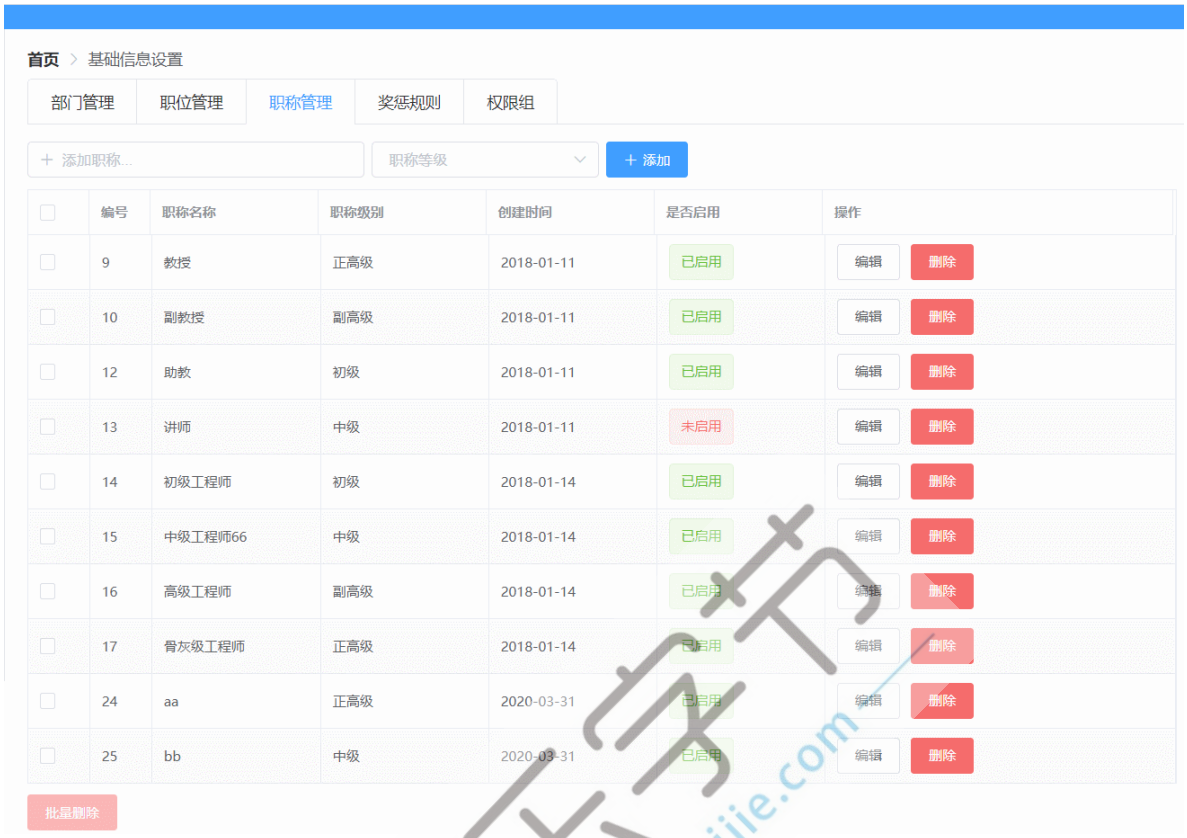
```
                title="编辑职称"
                :visible.sync="dialogVisible"
                width="30%">
            <div>
                <table>
                    <tr>
                        <td>
                            <el-tag>职称名</el-tag>
                        </td>
                        <td>
                            <el-input v-model="updateJl.name" size="small"></el-
input>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <el-tag>职称级别</el-tag>
                        </td>
                        <td>
                            <el-select v-model="updateJl.titleLevel"
placeholder="职称等级" size="small"
                                       style="margin-left: 6px;margin-right:
6px">
                                <el-option
                                        v-for="item in titleLevels"
                                        :key="item"
                                        :label="item"
                                        :value="item">
                                </el-option>
                            </el-select>
                        </td>
                    </tr>
                    <tr>
                        <td>
                            <el-tag>是否启用</el-tag>
                        </td>
                        <td>
                            <el-switch v-model="updateJl.enabled" active-
color="#13ce66" inactive-color="#ff4949"
                                       active-text="启用" inactive-text="禁用">
</el-switch>
                        </td>
                    </tr>
                </table>

            </div>
            <span slot="footer" class="dialog-footer">
                <el-button size="small" @click="dialogVisible = false">取 消</el-
button>
                <el-button size="small" type="primary" @click="doUpdate">确 定
</el-button>
            </span>
        </el-dialog>
    </div>
</template>

<script>
    export default {
```

```
    name: "JoblevelMana",
    data() {
        return {
            jl: {
                name: '',
                titleLevel: ''
            },
            updateJl: {
                name: '',
                titleLevel: '',
                enabled: false
            },
            dialogVisible: false,
            jls: [],
            multipleSelection: [],
            titleLevels: [
                '正高级',
                '副高级',
                '中级',
                '初级',
                '员级',
            ]
        }
    },
    mounted() {
        this.initJls();
    },
    methods: {
        deleteMany() {
            this.$confirm('此操作将永久删除【' + this.multipleSelection.length
+ '】条记录，是否继续?', '提示', {
                confirmButtonText: '确定',
                cancelButtonText: '取消',
                type: 'warning'
            }).then(() => {
                let ids = '?';
                this.multipleSelection.forEach(item => {
                    ids += 'ids=' + item.id + '&';
                })
                this.deleteRequest('/system/basic/joblevel/' +
ids).then(resp => {
                    if (resp) {
                        this.initJls();
                    }
                });
            }).catch(() => {
                this.$message({
                    type: 'info',
                    message: '已取消删除'
                });
            });
        },
        handleSelectionChange(val) {
            this.multipleSelection = val;
        },
        doUpdate() {
            this.putRequest('/system/basic/joblevel/',
this.updateJl).then(resp => {
```

```
                    if (resp) {
                        this.initJls();
                        this.dialogVisible = false;
                    }
                })
            },
            showEditView(data) {
                Object.assign(this.updateJl, data);
                this.updateJl.createDate = '';
                this.dialogVisible = true;
            },
            deleteHandler(data) {
                this.$confirm('此操作将永久删除【' + data.name + '】职称, 是否继续?',
'提示', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(() => {
                    this.deleteRequest('/system/basic/joblevel/' +
data.id).then(resp => {
                        if (resp) {
                            this.initJls();
                        }
                    });
                }).catch(() => {
                    this.$message({
                        type: 'info',
                        message: '已取消删除'
                    });
                });
            },
            addJobLevel() {
                if (this.jl.name && this.jl.titleLevel) {
                    this.postRequest('/system/basic/joblevel/',
this.jl).then(resp => {
                        if (resp) {
                            this.initJls();
                        }
                    })
                } else {
                    this.$message.error("字段不能为空！");
                }
            },
            initJls() {
                this.getRequest('/system/basic/joblevel/').then(resp => {
                    if (resp) {
                        this.jls = resp;
                        this.jl = {
                            name: '',
                            titleLevel: ''
                        }
                    }
                })
            }
        }
    }
</script>
```

```
</style>
```

效果



权限组

页面设计

PermissMana.vue

```
<template>
    <div>
        <div class="permissManaTool">
            <el-input size="small" placeholder="请输入角色英文名" v-
model="role.name">
                <template slot="prepend">ROLE_</template>
            </el-input>
            <el-input size="small" v-model="role.nameZh" placeholder="请输入角色中
文名"></el-input>
            <el-button type="primary" size="small" icon="el-icon-plus">添加角色
</el-button>
        </div>
        <div class="permissManaMain">
            <el-collapse v-model="activeName" accordion>
                <el-collapse-item title="一致性 Consistency" name="1">
                    <div>与现实生活一致：与现实生活的流程、逻辑保持一致，遵循用户习惯的语言
和概念；</div>
                    <div>在界面中一致：所有的元素和结构需保持一致，比如：设计样式、图标和文
本、元素的位置等。</div>
                </el-collapse-item>
                <el-collapse-item title="反馈 Feedback" name="2">
```

```
                    <div>控制反馈：通过界面样式和交互动效让用户可以清晰的感知自己的操作；
                    <div>页面反馈：操作后，通过页面元素的变化清晰地展现当前状态。</div>
                </el-collapse-item>
                <el-collapse-item title="效率 Efficiency" name="3">
                    <div>简化流程：设计简洁直观的操作流程；</div>
                    <div>清晰明确：语言表达清晰且表意明确，让用户快速理解进而作出决策；
</div>
                    <div>帮助用户识别：界面简单直白，让用户快速识别而非回忆，减少用户记忆负
担。</div>
                </el-collapse-item>
                <el-collapse-item title="可控 Controllability" name="4">
                    <div>用户决策：根据场景可给予用户操作建议或安全提示，但不能代替用户进行
决策；</div>
                    <div>结果可控：用户可以自由的进行操作，包括撤销、回退和终止当前操作等。
</div>
                </el-collapse-item>
            </el-collapse>
        </div>
    </div>
</template>

<script>
    export default {
        name: "PermissMana",
        data(){
            return{
                activeName:'2',
                role:{
                    name:'',
                    nameZh:''
                }
            }
        }
    }
</script>

<style>
    .permissManaTool{
        display: flex;
        justify-content: flex-start;
    }
    .permissManaTool .el-input{
        width: 300px;
        margin-right: 6px;
    }
    .permissManaMain{
        margin-top: 10px;
        width: 700px
    }
</style>
```

accordion： 是否手风琴模式

效果

调用接口显示所有角色

PremissMana.vue

```
<template>
    <div>
        <div class="permissManaTool">
            <el-input size="small" placeholder="请输入角色英文名" v-
model="role.name">
                <template slot="prepend">ROLE_</template>
            </el-input>
            <el-input size="small" v-model="role.nameZh" placeholder="请输入角色中
文名"></el-input>
            <el-button type="primary" size="small" icon="el-icon-plus">添加角色
</el-button>
        </div>
        <div class="permissManaMain">
            <el-collapse accordion>
                <el-collapse-item :title="r.nameZh" :name="r.id" v-for="
(r,index) in roles" :key="index">
                    <el-card class="box-card">
                        <div slot="header" class="clearfix">
                            <span>可访问资源</span>
                            <el-button style="float: right; padding: 3px
0;color: #ff0000" icon="el-icon-delete" type="text"></el-button>
                        </div>
                        <div>

                        </div>
                    </el-card>
                </el-collapse-item>
            </el-collapse>
        </div>
    </div>
</template>
```

```
>
ort default {
    name: "PermissMana",
    data() {
        return {
            activeName: '2',
            role: {
                name: '',
                nameZh: ''
            },
            roles: []
        }
    },
    mounted() {
        this.initRoles();
    },
    methods: {
        initRoles() {
            this.getRequest('/system/basic/permiss/').then(resp => {
                if (resp) {
                    this.roles = resp;
                }
            })
        }
    }
}
</script>

<style>
    .permissManaTool {
        display: flex;
        justify-content: flex-start;
    }

    .permissManaTool .el-input {
        width: 300px;
        margin-right: 6px;
    }

    .permissManaMain {
        margin-top: 10px;
        width: 700px
    }
</style>
```

效果

菜单树形展示

PermissMana.vue

```
<template>
    <div>
        <div class="permissManaTool">
            <el-input size="small" placeholder="请输入角色英文名" v-
model="role.name">
                <template slot="prepend">ROLE_</template>
            </el-input>
            <el-input size="small" v-model="role.nameZh" placeholder="请输入角色中
文名"></el-input>
            <el-button type="primary" size="small" icon="el-icon-plus">添加角色
</el-button>
        </div>
        <div class="permissManaMain">
            <el-collapse accordion @change="change">
                <el-collapse-item :title="r.nameZh" :name="r.id" v-for="
(r,index) in roles" :key="index">
                    <el-card class="box-card">
                        <div slot="header" class="clearfix">
                            <span>可访问资源</span>
                            <el-button style="float: right; padding: 3px
0;color: #ff0000" icon="el-icon-delete"
                                       type="text"></el-button>
                        </div>
                        <div>
                            <el-tree
                                    show-checkbox
                                    node-key="id"
```

```
                                :key="index"
                                :default-checked-keys="selectedMenus"
                                :data="allMenus" :props="defaultProps"></el-
tree>
                        </div>
                    </el-card>
                </el-collapse-item>
            </el-collapse>
        </div>
    </div>
</template>

<script>
    export default {
        name: "PermissMana",
        data() {
            return {
                role: {
                    name: '',
                    nameZh: ''
                },
                roles: [],
                allMenus: [],
                selectedMenus: [],
                defaultProps: {
                    children: 'children',
                    label: 'name'
                }
            }
        },
        mounted() {
            this.initRoles();
        },
        methods: {
            change(rid) {
                if (rid) {
                    this.initAllMenus();
                    this.initSelectedMenus(rid);
                }
            },
            initSelectedMenus(rid) {
                this.getRequest('/system/basic/permiss/mid/' + rid).then(resp =>
{
                    this.selectedMenus = resp;
                })
            },
            initAllMenus() {
                this.getRequest('/system/basic/permiss/menus').then(resp => {
                    this.allMenus = resp;
                })
            },
            initRoles() {
                this.getRequest('/system/basic/permiss/').then(resp => {
                    if (resp) {
                        this.roles = resp;
                    }
                })
            },
```

```
}
</script>

<style>
    .permissManaTool {
        display: flex;
        justify-content: flex-start;
    }

    .permissManaTool .el-input {
        width: 300px;
        margin-right: 6px;
    }

    .permissManaMain {
        margin-top: 10px;
        width: 700px
    }

</style>
```

change： 当前激活面板改变时触发(如果是手风琴模式，参数 activeNames 类型为 string，否则为 array)

prop： 属性

label： 指定节点标签为节点对象的某个属性值

children： 指定子树为节点对象的某个属性值

效果

角色菜单修改

```
<template>
    <div>
        <div class="permissManaTool">
            <el-input size="small" placeholder="请输入角色英文名" v-
model="role.name">
                <template slot="prepend">ROLE_</template>
            </el-input>
            <el-input size="small" v-model="role.nameZh" placeholder="请输入角色中
文名"></el-input>
            <el-button type="primary" size="small" icon="el-icon-plus">添加角色
</el-button>
        </div>
        <div class="permissManaMain">
            <el-collapse v-model="activeName" accordion @change="change">
                <el-collapse-item :title="r.nameZh" :name="r.id" v-for="
(r,index) in roles" :key="index">
                    <el-card class="box-card">
                        <div slot="header" class="clearfix">
                            <span>可访问资源</span>
                            <el-button style="float: right; padding: 3px
0;color: #ff0000" icon="el-icon-delete"
                                       type="text"></el-button>
                        </div>
                        <div>
                            <el-tree
                                    show-checkbox
                                    node-key="id"
                                    ref="tree"
                                    :default-checked-keys="selectedMenus"
                                    :data="allMenus" :props="defaultProps"></el-
tree>
                            <div style="display: flex;justify-content: flex-
end">
                                <el-button @click="cancelUpdate">取消修改</el-
button>
                                <el-button type="primary"
@click="doUpdate(r.id,index)">确认修改</el-button>
                            </div>
                        </div>
                    </el-card>
                </el-collapse-item>
            </el-collapse>
        </div>
    </div>
</template>

<script>
    export default {
        name: "PermissMana",
        data() {
            return {
                role: {
                    name: '',
                    nameZh: ''
                },
```

```
            roles: [],
            allMenus: [],
            selectedMenus: [],
            activeName: -1,
            defaultProps: {
                children: 'children',
                label: 'name'
            }
        }
    },
    mounted() {
        this.initRoles();
    },
    methods: {
        cancelUpdate() {
            this.activeName = -1;
        },
        doUpdate(rid, index) {
            let tree = this.$refs.tree[index];
            let selectedKeys = tree.getCheckedKeys(true);
            let url = '/system/basic/permiss/?rid=' + rid;
            selectedKeys.forEach(key => {
                url += '&mids=' + key;
            })
            this.putRequest(url).then(resp => {
                if (resp) {
                    this.initRoles();
                    this.activeName = -1;
                }
            })
        },
        change(rid) {
            if (rid) {
                this.initAllMenus();
                this.initSelectedMenus(rid);
            }
        },
        initSelectedMenus(rid) {
            this.getRequest('/system/basic/permiss/mid/' + rid).then(resp =>
{
                this.selectedMenus = resp;
            })
        },
        initAllMenus() {
            this.getRequest('/system/basic/permiss/menus').then(resp => {
                this.allMenus = resp;
            })
        },
        initRoles() {
            this.getRequest('/system/basic/permiss/').then(resp => {
                if (resp) {
                    this.roles = resp;
                }
            })
        }
    }
}
</script>
```

```
        .permissManaTool {
            display: flex;
            justify-content: flex-start;
        }

        .permissManaTool .el-input {
            width: 300px;
            margin-right: 6px;
        }

        .permissManaMain {
            margin-top: 10px;
            width: 700px
        }

    </style>
```

效果

角色操作

PermissMana.vue

```
<template>
    <div>
        <div class="permissManaTool">
```

```
            <el-input size="small" placeholder="请输入角色英文名" v-
        role.name">
                <template slot="prepend">ROLE_</template>
            </el-input>
            <el-input size="small" v-model="role.nameZh" placeholder="请输入角色中
文名"
                      @keydown.enter.native="doAddRole"></el-input>
            <el-button type="primary" size="small" icon="el-icon-plus"
@click="doAddRole">添加角色</el-button>
        </div>
        <div class="permissManaMain">
            <el-collapse v-model="activeName" accordion @change="change">
                <el-collapse-item :title="r.nameZh" :name="r.id" v-for="
(r,index) in roles" :key="index">
                    <el-card class="box-card">
                        <div slot="header" class="clearfix">
                            <span>可访问资源</span>
                            <el-button style="float: right; padding: 3px
0;color: #ff0000" icon="el-icon-delete" type="text" @click="doDeleteRole(r)">
</el-button>
                        </div>
                        <div>
                            <el-tree
                                    show-checkbox
                                    node-key="id"
                                    ref="tree"
                                    :key="index"
                                    :default-checked-keys="selectedMenus"
                                    :data="allMenus" :props="defaultProps">
                            </el-tree>
                            <div style="display: flex;justify-content: flex-
end">
                                <el-button @click="cancelUpdate">取消修改</el-
button>
                                <el-button type="primary"
@click="doUpdate(r.id,index)">确认修改</el-button>
                            </div>
                        </div>
                    </el-card>
                </el-collapse-item>
            </el-collapse>
        </div>
    </div>
</template>

<script>
    export default {
        name: "PermissMana",
        data() {
            return {
                role: {
                    name: '',
                    nameZh: ''
                },
                roles: [],
                allMenus: [],
                selectedMenus: [],
                activeName: -1,
```

```
                defaultProps: {
                    children: 'children',
                    label: 'name'
                }
            }
        },
        mounted() {
            this.initRoles();
        },
        methods: {
            doDeleteRole(role){
                this.$confirm('此操作将永久删除该【' + role.nameZh + '】角色，是否继续?', '提示', {
                    confirmButtonText: '确定',
                    cancelButtonText: '取消',
                    type: 'warning'
                }).then(() => {
                    this.deleteRequest('/system/basic/permiss/role/' + role.id).then(resp => {
                        if (resp) {
                            this.initRoles();
                        }
                    });
                }).catch(() => {
                    this.$message({
                        type: 'info',
                        message: '已取消删除'
                    });
                });
            },
            doAddRole(){
                if (this.role.name&&this.role.nameZh){

 this.postRequest('/system/basic/permiss/role',this.role).then(resp=>{
                        if (resp){
                            this.name='';
                            this.nameZh='';
                            this.initRoles();
                        }
                    })
                } else {
                    this.$message.error('字段不能为空!');
                }
            },
            cancelUpdate() {
                this.activeName = -1;
            },
            doUpdate(rid, index) {
                let tree = this.$refs.tree[index];
                let selectedKeys = tree.getCheckedKeys(true);
                let url = '/system/basic/permiss/?rid=' + rid;
                selectedKeys.forEach(key => {
                    url += '&mids=' + key;
                })
                this.putRequest(url).then(resp => {
                    if (resp) {
                        this.activeName = -1;
                    }
```

```
                })
            },
            change(rid) {
                if (rid) {
                    this.initAllMenus();
                    this.initSelectedMenus(rid);
                }
            },
            initSelectedMenus(rid) {
                this.getRequest('/system/basic/permiss/mid/' + rid).then(resp =>
{
                    this.selectedMenus = resp;
                })
            },
            initAllMenus() {
                this.getRequest('/system/basic/permiss/menus').then(resp => {
                    this.allMenus = resp;
                })
            },
            initRoles() {
                this.getRequest('/system/basic/permiss/').then(resp => {
                    if (resp) {
                        this.roles = resp;
                    }
                })
            }
        }
    }
</script>

<style>
    .permissManaTool {
        display: flex;
        justify-content: flex-start;
    }

    .permissManaTool .el-input {
        width: 300px;
        margin-right: 6px;
    }

    .permissManaMain {
        margin-top: 10px;
        width: 700px
    }
</style>
```

效果