技术栈

使用vue-cli创建一个vue2的项目

Element-ui这个组件库

使用axios发起请求，同时使用axios的拦截器来展示loading的效果

使用proxy来实现跨域请求的代理

Vue-router路由

实现步骤

1. 初始化项目
2. 渲染用户表格的数据
3. 基于全局过滤器处理时间格式
4. 实现添加用户的操作
5. 实现删除用户的操作
6. 通过路由跳转到详情页

初始化项目

1. 使用powershell窗口，然后运行vue create 项目名称，下载好vue项目

项目创建完毕

1. 删除APP.vue里面的全部内容，自己初始化一个模板
2. 然后删除components组件文件夹里面的helloWord.vue文件
3. 然后终端运行npm run serve
4. 然后配置项目的的初始化，配置端口号，然后自动打开浏览器

初始化项目后，**项目根目录下**需要新建一个vue.config.js配置文件

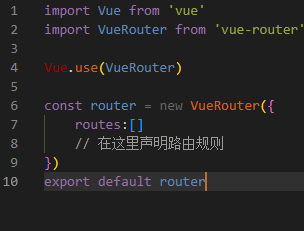


Ctrl+C两次，关闭服务器，然后从新运行npm run serve

这个时候端口号就会改变，然后自动打开浏览器

初始化路由

1. 运行npm install [vue-router@3.4.9](mailto:vue-router@3.4.9) -S 配置成当前项目使用
2. Src目录下新建router/index.js路由模块



1. 配置完毕后，在main.js中导入并且挂载路由模块

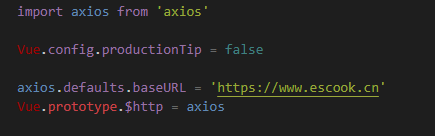
使用路由渲染UserList组件

1. 在components目录下新建UserList.vue组件
2. 然后再路由组件里面新增路由规则
3. 然后再App.vue里面使用我们的路由，使用路由占位view符来实现路由
4. **我们可以使用@来代表src路径**



安装和配置axios

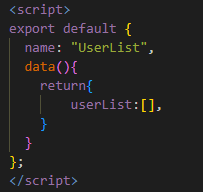
1. 首先安装axios
2. 然后再mian.js里面全局配置axios



这里根vue3不同的就是配置全局自定义属性的时候

请求用户列表数据并解决接口跨域问题

1. UserList.vue组件中声明一个data数据节点



1. 再created声明周期函数中预调用getUserList方法



1. mothds中声明getUserList方法



解决跨域

1. 首先把axios请求的根路径改成我们当前项目的路径



1. 然后再vue.config.js里面的devserve节点下面配置 proxy节点，配置成我们真正请求的地址就可以了



配置完毕记得重启项目

安装并配置element-ui

1. 运行npm i element-ui -S安装element-ui ，如果是vue3就用

element plush

1. 然后根据文档完整导入



import ElementUI from 'element-ui';

import 'element-ui/lib/theme-chalk/index.css';

Vue.use(ElementUI);

了解table组件的基本作用

1. 就是在element-ui的官方文档里面找到对应的表格组件
2. 我们查看源代码发现可以使用一个element-ui提供的<el-table>组件，然后里面有一个data自定义属性用v-bind绑定的，这个就是我们给表格提供的数据源
3. 然后我们就可以使用element-ui去编辑我们的userlist组件了
4. 然后使用< el-table-column>组件就可以指定每一列的数据了



里面prop属性指定要用的数据，，label指定我们的标题



1. 这里我们使用的数据，就是userList这个对象下的数据，就类似于我们使用作用域插槽slot的时候，把获取过来的自定义数据，给声明成为一个对象
2. 给这个表格添加一个效果，例如隔行变色和边框线



例如实现斑马线效果实现就是给el-table加上一个stripe属性

其余效果看文档即可

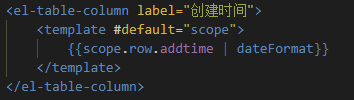
通过作用域插槽的形式自定义创建时间的渲染格式

1. 就是我们再定义一个组件，这个组件时一个用来创建时间渲染格式的组件，然后提供一个插槽，来供我们的userlist这个组件使用
2. 然后我们使用element-ui里面的自定义模板的方法拿到我们的数据

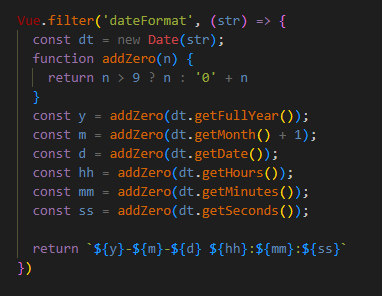


1. 这个意思就是我们可以通过对象的形式拿到，这个el-table范围内的全部数据
2. 就像我们原生的v-slot：slot名称：‘scope’那样赋值成一个scope对象
3. 然后这个里面的数据包括，row这一行的数据，column这一列的数据，vuex里store的数据
4. 然后我们拿到这一一行的数据

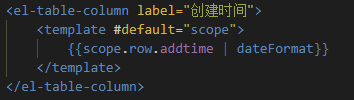




1. **这里我们不建议使用原来的slot-scope的方式拿到数据，我们用vue3的新方法，这个在新的vue2里面也支持**
2. 然后我们再通过scope.row.addtime的形式拿到我们的数据
3. 然后通过声明全局过滤器的形式去格式化我们的数据

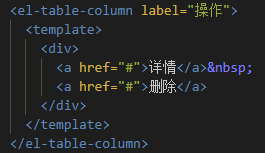


1. **注意模板字符串的引号是这个``**
2. 然后在userlist里面使用这个过滤器



通过插槽的形式操作列的模板机构

1. 就是通过插槽的形式自定义我们一列里面的每一项里面要渲染成什么样式，
2. 这里我们渲染成一个a链接的形式，这里其实是使用是，slot不是声明slot





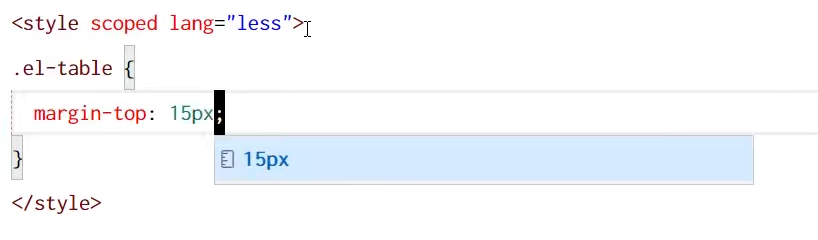
点击按钮添加用户的Dialog组件

1. 就是放一个添加按钮
2. 点击按钮之后显示一个弹出框提示我们信息
3. 我们使用element-ui里面的Dialog组件



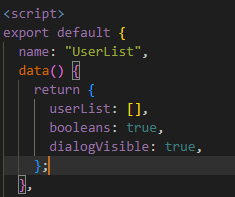
用法就是用我们的el-dialog去关联我们的按钮来实现功能

1. 然后就是我们element-ui里面的组件，这个组件名就是它的类名，所以我们要改变它的样式，就直接用组件名在下面style里面声明就行了





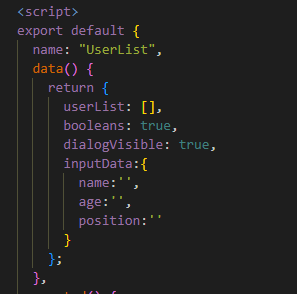
1. 这里还用了sync修饰符
2. 然后这个使用的dialogvisible要在下面data数据里面声明



渲染添加新用户的form表单

1. 就是我们的这个dialog里面可以包裹任何内容进行显示
2. 我们要在dialog里面渲染一个增加新用户的form表单
3. 就是使用<el-form>
4. 这个<el-form>有一个：model=‘对象’，就是使用属性绑定，把我们整个表单的信息都传给我们绑定的整个对象





实现form表单的数据验证



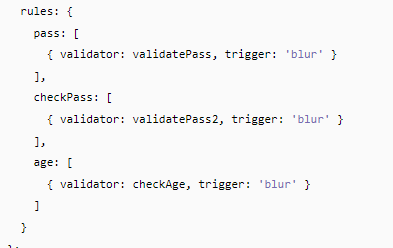
1. 就是加上一个校验条件，例如不能为空

自定义form表单的验证规则

1. 就是弄一个validator校验器函数，这个函数就定义在上面的rules规则里面就行



1. 声明这个校验器函数，第一个是rule，第二个是拿到的js表达式，第三个是我们的回调函数，就是调用callback就推出校验器函数
2. 然后把data里面声明好的校验器函数，就放在rules里面



这里注意，data里面数据是用return返回回去，但是方法不用return返回回去

在对话框取消的时候重置form表单

1. 首先监听对话框关闭的事件，就是监听dialog的close事件
2. 然后通过ref属性拿到我们的dialog

然后就是调用这个表单的方法



实现添加用户之前的表单验证

1. 就是我们点击确认按钮添加的数据之前，不能直接发起aixos请求，应该现判断用户输入的内容是否合法



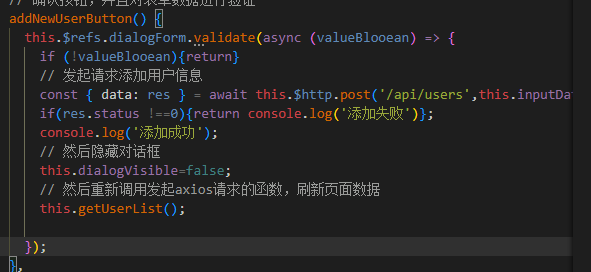
这回用的是validate方法

1. 这个是validate方法里面是一个函数，然后函数有两个值，一个是是否验证成功的一个bootean的布尔值，第二个是如果没有通过校验，会把没有通过的js表达式发过来



点击确认按钮，发起axios请求，然后添加用户的信息

1. 我们点击确认按钮，发起请求
2. 然后提示用户数据提交成功
3. 然后其实我们发送的数据就是我们的inputData，就是我们v-molel绑定然后存储过来的数据



其实就是我们点击这个确认按钮之后，

1. 首先对这个表单的数据进行验证判断
2. 判断成功了，valueBlooean式true就不会执行这个if
3. 然后继续执行后面的代码，发起axios的post请求，给服务器提交数据
4. 然后判断提交是否成功
5. 提交成功执行后面的代码
6. 前面的判断完毕后，执行隐藏对话框
7. 然后重新调用axios的get请求，刷新页面数据

用message组件实现信息提示的效果

1. 就是我们在console终端里面提示，但是我们的用户不知道看这个，所以我们要用这个消息提示组件

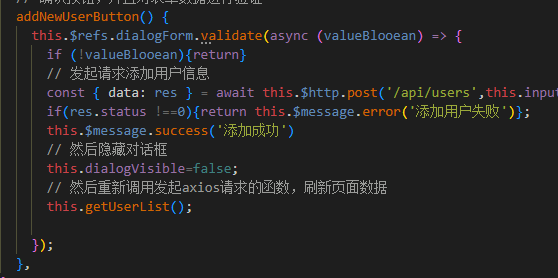
实现原理就是调用this.$message



1. 然后不同的主题就是切换不同的修饰，有这些方法



1. 然后把原来log提示的变成这个message提示



使用messageBox组件来询问用户是否真的删除这个数据

1. 就是我们点击删除按钮
2. 会弹出一个消息框，让你确定是不是真的删除
3. 点击确认后，再发起删除的ajax请求



这里用的式this.$confirm

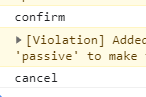
注意这个方法返回的是要给promise实例对象，我们就用await修饰

然后我们分别点击确认和取消都会返回值，这个时候如果点击取消会报错



所以我们用一个.catch（）捕获错误信息，把这个错误信息捕获，再做为参数返回给我们的confirm的promise实例对象

然后我们再点击按钮，正确会返回confirm，错误会返回cancel



然后对这个值进行if判断

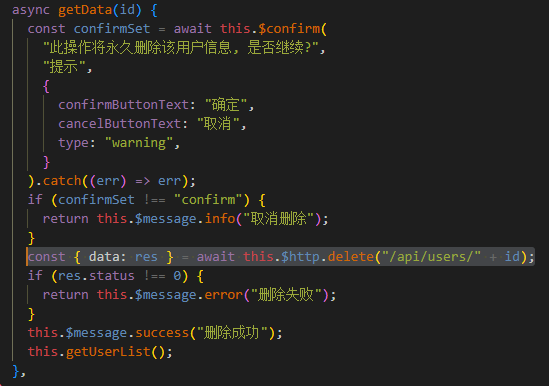


发起请求实现用户数据删除的功能

1. 我们点击删除的时候拿到这一行的id
2. 然后我们就给这个删除的template变成一个作用域插槽，用来接收element-ui组件传递来的数据，这个数据给它命名成一个名为scpoe的对象



然后我们把这个拿到的id值，做为参数传递给我们的getData方法，同时发起删除的请求



通过声明式导航跳转到用户详情页

1. 就是点击详情链接的时候，跳转到指定页面
2. 首先把这个链接改造成一个router-link组件
3. 点击的时候动态的绑定to属性值，从而实现导航
4. 然后我们创建一个用户详情页，然后通过路由实现跳转
5. 首先创建一个路由组件UserComponentSon
6. 然后把这个路由组件注册在路由模块里面
7. 然后声明一个动态的路由，名字为/users/:id，
8. 然后给router-link去动态绑定这个路由就行了

--------------------------------------------------------------

给渲染用户详情页的数据

1. 首先在路由规则上面开启接收props数据



然后我们就可以在对应的组件UserComponentSon里面，通过props节点来接收数据了

我们要接收的是hash值得动态得参数id



1. 然后再UserComponentSon里面发起ajax请求，注意请求要在created里面声明
2. 然后下面发起请求，再把请求来的数据存到userData里面
3. 然后为了展示好看，我们用element-ui里面的卡片组件crad

通过axios拦截器实现loading的效果

1. 首先声明axios的声明和相应的拦截器
2. 然后调用element-ui里面的loading组件实现loading效果



