# 一、使用element ui模板来开发

## 1.安装element ui模板

1）输入命令 npm i element-ui -S 安装

2）在项目中引入element-ui

|  |
| --- |
| import Vue from 'vue'  import ElementUI from 'element-ui';  import 'element-ui/lib/theme-chalk/index.css';  import App from './App.vue'  Vue.config.productionTip = false  Vue.use(ElementUI); |

## 2.尝试使用element ui中的组件

将组件的相应的部分，挪到我们的组件中进行调整。

|  |
| --- |
| <template>  <div id="app">  <template>  <div id="app">  <el-form ref="form" class="form-box" :model="form" label-width="80px">  <h3 align="center" class="login\_title">欢迎登录</h3>  <el-form-item label="用户名">  <el-input v-model="form.name"></el-input>  </el-form-item>  <el-form-item label="密码">  <el-input v-model="form.password"></el-input>  </el-form-item>  <el-form-item>  <el-button type="primary" @click="onSubmit">登录</el-button>  </el-form-item>  </el-form>  </div>  </template>  <script>  export default {  data(){  return {  form:{  name:'',  password:''  }  }  },  methods: {  onSubmit() {  console.log('submit!');  }  }  }  </script>  <style>    .form-box{  width: 400px;  margin: 200px auto;  padding: 20px 50px 20px 20px;  border: 1px solid #409EFF;  border-radius: 20px;  box-shadow: 0px 0px 5px #409EFF;    }  .login\_title{  margin-bottom: 40px;  }  </style> </div>  </template>  <script>  export default {  data(){  return {  form:{  name:'',  password:''  }  }  },  methods: {  onSubmit() {  console.log('submit!');  }  }  }  </script>  <style>    .form-box{  width: 400px;  margin: 200px auto;  padding: 20px 50px 20px 20px;  border: 1px solid #409EFF;  border-radius: 20px;  box-shadow: 0px 0px 5px #409EFF;    }  .login\_title{  margin-bottom: 40px;  }  </style> |

# 二、使用axios发送ajax请求

## 1.安装axios

npm install --save axios vue-axios

## 2.使用

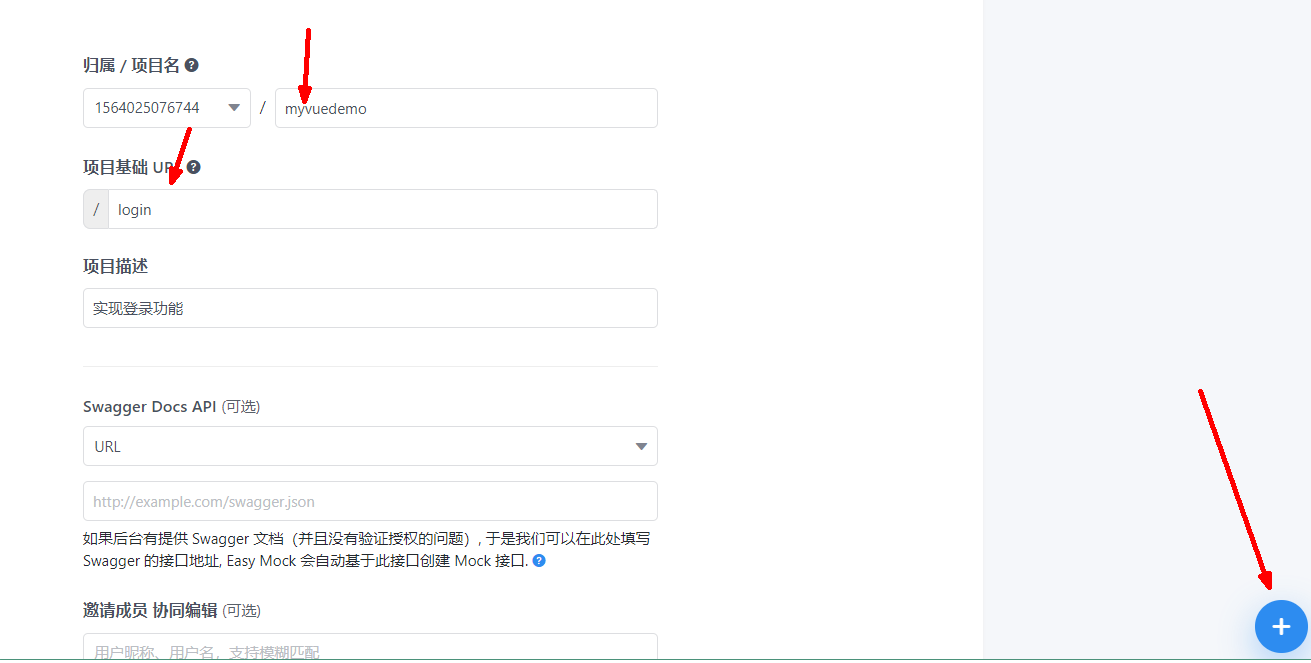
# 三、通过MOCK提供假数据

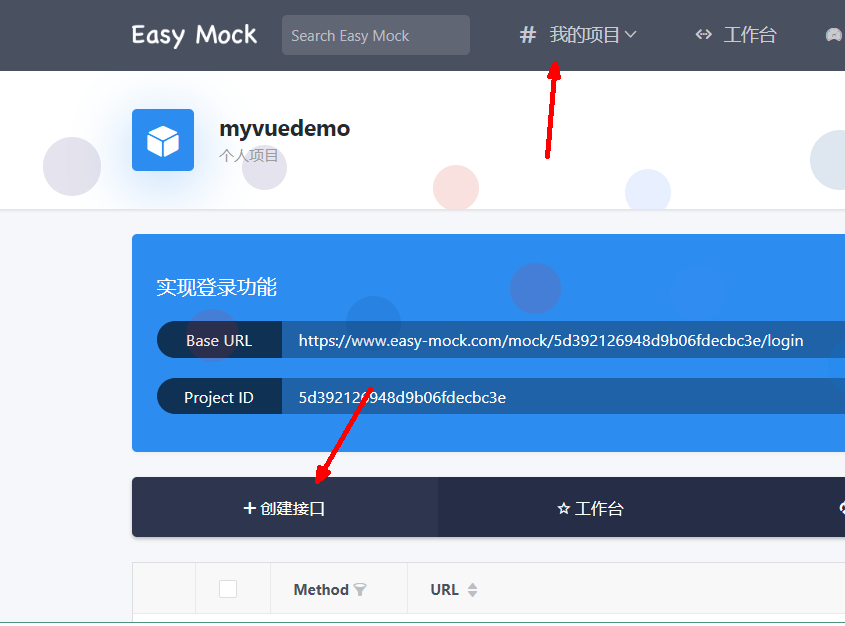
要开发实际的接口，但开发实际的接口是后端程序员做的事情，那前端程序员在开发过程中，不能总是等后端开发完后再得到接口再使用。这时可以通过MOCK提供的假数据来完成前端的开发。等后端接口开发完之后，再修改url即可。

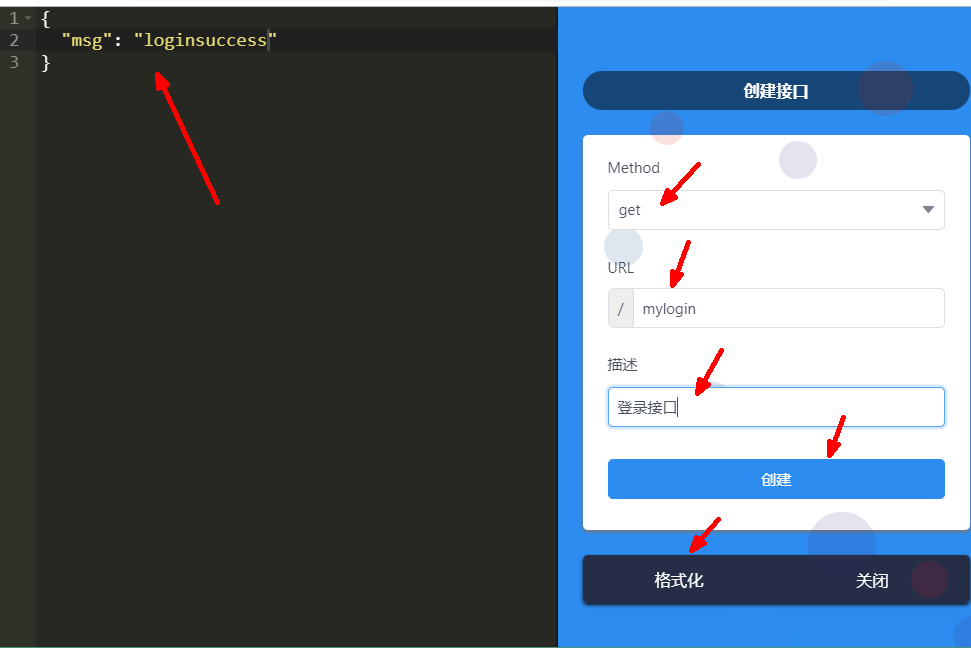
1. 使用easy-mock提供假数据

<https://www.easy-mock.com/>

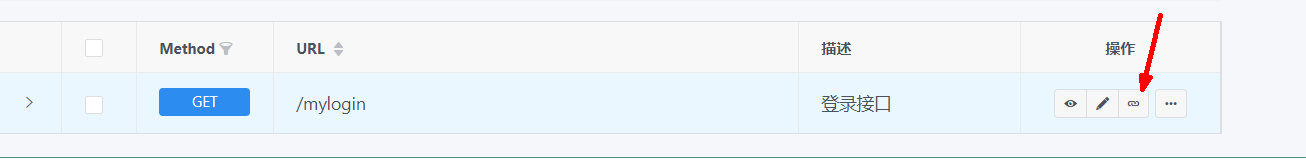
2.创建项目







复制接口地址



就在axios中使用接口地址

|  |
| --- |
| this.axios({  //请求方式  methods:'get',  //请求的url  url:'https://www.easy-mock.com/mock/5d392126948d9b06fdecbc3e/login/mylogin',    }).then(function(resp){  //当请求成功以后会调用then里的这个function  //此时，resp里面封装了响应消息，响应消息中的data返回的数据对象（json）  console.log(resp.data);  }).catch(function(error){  //如果请求失败，则会调用catch中的function  }) |

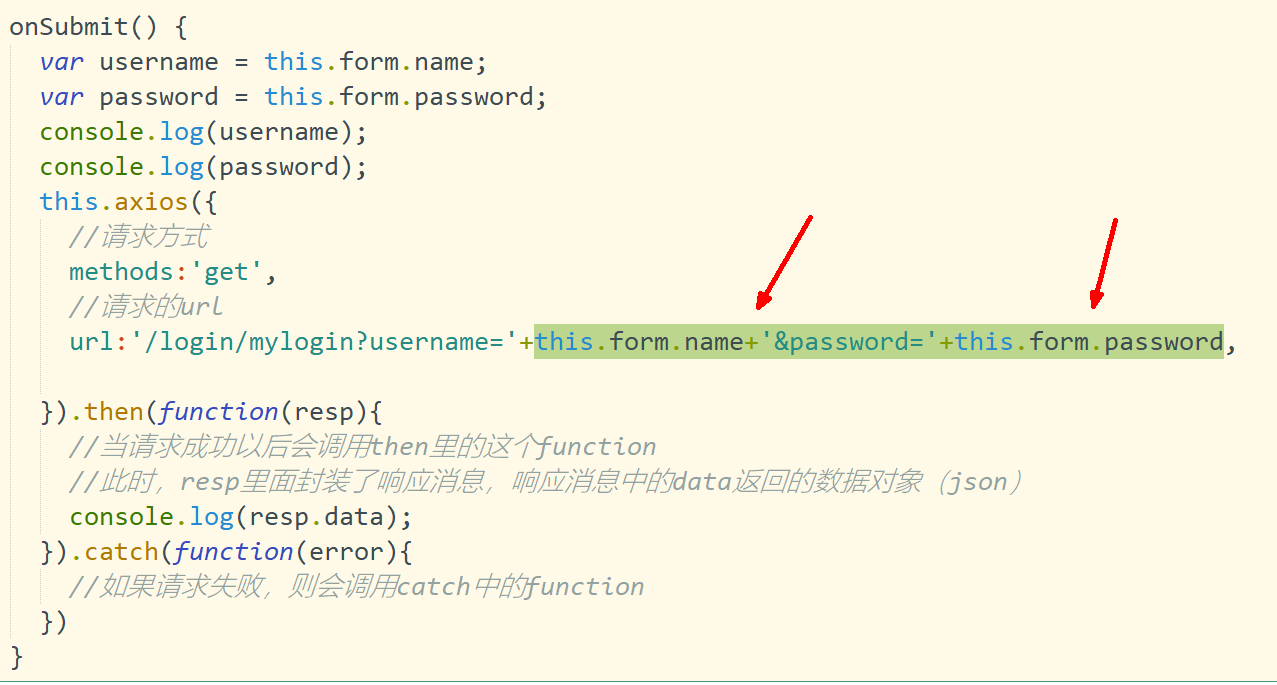
# 四、axios请求方式及携带参数问题

## 1. get请求携带参数

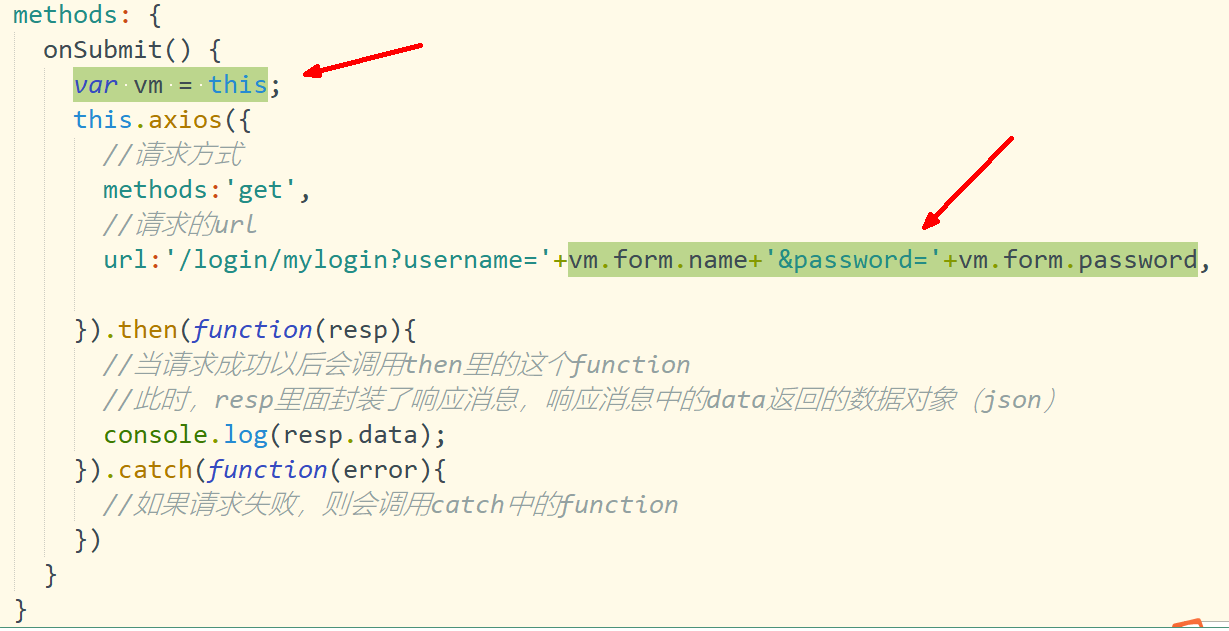
以下这种方式是正确的：

|  |
| --- |
| methods: {  onSubmit() {  var username = this.form.name;  var password = this.form.password;  console.log(username);  console.log(password);  this.axios({  //请求方式  methods:'get',  //请求的url  url:'/login/mylogin?username='+username+'&password='+password    }).then(function(resp){  //当请求成功以后会调用then里的这个function  //此时，resp里面封装了响应消息，响应消息中的data返回的数据对象（json）  console.log(resp.data);  }).catch(function(error){  //如果请求失败，则会调用catch中的function  })  }  } |

这种方式是不行的



可以这么做：



## 2.POST携带参数的问题

|  |
| --- |
| <script>  import Qs from 'qs'    export default {  data(){  return {  form:{  name:'',  password:''  }  }  },  //后端使用ssm 拿请求体中的东西 需要引入qs内置模块  methods: {  onSubmit() {  var vm = this;  this.axios({  //请求方式  methods:'post',  //请求的url  url:'/login/mylogin',  transformRequest: [function (data) {  return Qs.stringify(data)  }],  data:{  username:vm.form.name,  password:vm.form.password  }    }).then(function(resp){  //当请求成功以后会调用then里的这个function  //此时，resp里面封装了响应消息，响应消息中的data返回的数据对象（json）  console.log(resp.data);  }).catch(function(error){  //如果请求失败，则会调用catch中的function  })  }  }  }  </script> |

3.完成前台的登录验证（假数据的方式）

|  |
| --- |
| methods: {  onSubmit() {  var vm = this;  this.axios({  //请求方式  methods:'get',  //请求的url  url:'https://www.easy-mock.com/mock/5d392126948d9b06fdecbc3e/login/loginfail'  }).then(function(resp){  //当请求成功以后会调用then里的这个function  //此时，resp里面封装了响应消息，响应消息中的data返回的数据对象（json）  // console.log(resp.data);  var result = resp.data.resultdata.result;  if(result){  //登录成功  vm.$message({  message: resp.data.resultdata.msg,  type: 'success'  });  }else{  vm.$message.error(resp.data.resultdata.msg);  }  }).catch(function(error){  //如果请求失败，则会调用catch中的function  })  }  }  } |

# 五、路由

## 1.路由的介绍

路由在vue里是一个模块，要实现页面的跳转。

一个页面==一个组件

一个组件： template script style

一个vue项目中只有一个主组件App，在主组件里切换显示其他组件

通过路由，在这个App.vue中显示出不同的组件

## 2.安装路由模块

npm install vue-router -s

## 3.创建路由表

|  |
| --- |
| import Home from '/src/components/Home.vue'  import Product from '/src/components/Product.vue'  export const routes = [  {  path:'/home',  component:Home  },  {  path:'/product',  component:Product  }  ] |

## 4.在main.js中使用路由表

|  |
| --- |
| import Vue from 'vue'  import ElementUI from 'element-ui';  import 'element-ui/lib/theme-chalk/index.css';  import axios from 'axios'  import VueAxios from 'vue-axios'  import App from './App.vue'  import VueRouter from 'vue-router' //1.引入路由模块  import {routes} from './routes' //2.引入静态路由表  Vue.config.productionTip = false  Vue.use(ElementUI);  Vue.use(VueAxios, axios);  Vue.use(VueRouter); //3.使用路由模块  //4.创建一个VueRouter模块的实例  const router = new VueRouter({  routes:routes  });  /\* eslint-disable no-new \*/  new Vue({  el: '#app',  router,//5.把router实例放入到vue实例中  components: { App },  template: '<App/>'  }) |

## 5.使用路由，实现组件之间的跳转效果

1）在要展现路由的地方 使用标签： <router-view></router-view>

2）通过<router-link to="路径"></router-link>来实现路由的触发，此时，to后的路径是会去路由表中找到匹配的path,在找到相应的component，然后显示在router-view.

## 6.路由的跳转两种形式

1）通过<router-link>实现跳转

2）this.$router.push("/home");

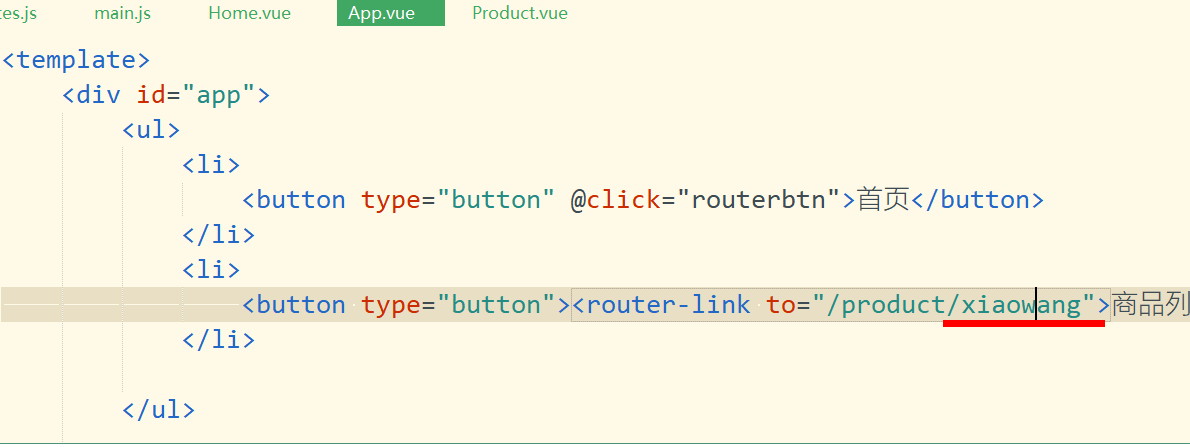
## 7.路由跳转时如何携带参数

### 通过router-link的方式实现跳转

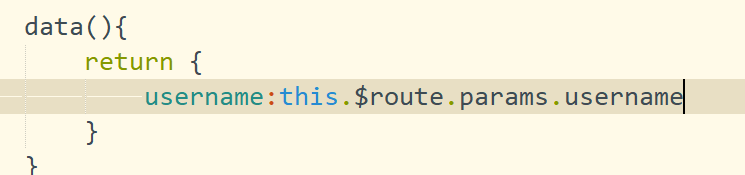
设参：



传参：

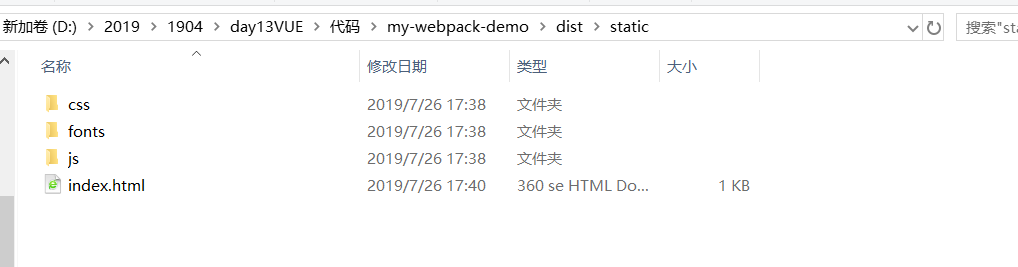


接参：



# 六、项目的部署

使用npm build命令进行构建，将dist文件夹内的东西放到服务器上即可。



# 七、vue-element-admin

是一个后台系统的前端解决方案

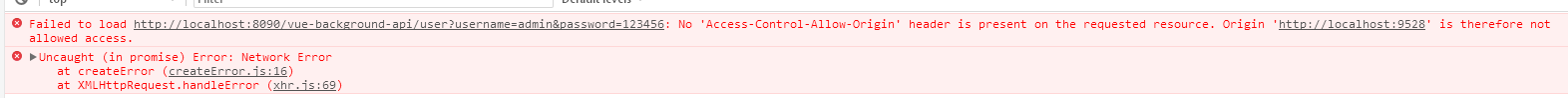
1.如何使用

从github上拉取源代码，使用基础模板做二次开发，需要的东西可以从集成方案（全面）中获取

* 集成方案: [vue-element-admin](https://github.com/PanJiaChen/vue-element-admin" \t "_blank)
* 基础模板: [vue-admin-template](https://github.com/PanJiaChen/vue-admin-template" \t "_blank)

# 八、跨域问题

当我们的前端服务器去访问后端服务器的接口时，那么就有可能出现跨域问题。



解决方案：

在响应头里加上：

|  |
| --- |
| response.setHeader("Access-Control-Allow-Origin", "\*"); |

# 九、提供前端需要的json对象

## 1.gson来提供

## 2.封装响应消息

## 3.解决中文乱码

## 4.解决跨域问题

|  |
| --- |
| **protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {  String username = request.getParameter("username");  String password = request.getParameter("password");  System.***out***.println("username="+username);  System.***out***.println("password="+password);    /\*  \* {  \* resultdata:{result:false,msg:''}  \* }  \*  \* gson  \*/  //创建一个封装了验证结果的数据传输对象  ResultDTO resultDTO = **new** ResultDTO();    **if**("admin".equals(username)&&"123456".equals(password)) {  //如果登录成功，那么就把相应的消息封装进dto中  resultDTO.setResult(**true**);  resultDTO.setMsg("登录成功");  }**else** {  //登录失败  resultDTO.setResult(**false**);  resultDTO.setMsg("用户名或密码错误");  }    //使用gson来将dto对象转换成json字符串  Gson gson = **new** Gson();  String json = gson.toJson(resultDTO);  //解决响应的中文乱码问题  response.setContentType("text/json;charset=utf8");  //将json字符串响应给前端  response.getWriter().write(json);  //解决跨域问题  response.setHeader("Access-Control-Allow-Origin", "\*");    } |

# 十、编写侧边栏

通过读vue-element-admin官方文档得知侧边栏的开发得通过路由表。

|  |
| --- |
| {  path: '/example',  component: Layout,  redirect: '/example/table',  name: 'Example',  meta: { title: '商品管理', icon: 'example' },  children: [  {  path: 'table',  name: 'Table',  component: () => import('@/views/products/index'),  meta: { title: '商品列表', icon: 'table' }  },  {  path: 'tree',  name: 'Tree',  component: () => import('@/views/tree/index'),  meta: { title: 'Tree', icon: 'tree' }  }  ]  }, |

vue-element-admin项目 使用vue框架 基于element-ui组件而提供的一套半成品。

# 十一、VUE页面加载完后就要有数据的问题

也就是当页面加载完以后，要向后端请求数据，并将数据展现在页面中。要向在vue里面实现这个功能，就必须通过vue生命周期钩子函数 created()

|  |
| --- |
| export default {    data() {  return {    }  },  created() {  this.fetchData()  },  methods: {  fetchData() {  this.axios({  methods:'get',  url:'https://www.easy-mock.com/mock/5d392126948d9b06fdecbc3e/login/products'  }).then(function(resp){  var d = resp.data;  console.log(d);  })  }  }  } |

# 十二、商品列表展示

|  |
| --- |
| <template>  <div class="app-container">    <el-table  :data="list"  style="width: 100%">  <el-table-column  label="序号"  prop="">  <template slot-scope="scope">  {{ scope.$index }}  </template>  </el-table-column>  <el-table-column  label="商品名称"  prop="">  <template slot-scope="scope">  {{ scope.row.pname }}  </template>  </el-table-column>  <el-table-column  label="商品价格"  prop="">  <template slot-scope="scope">  {{ scope.row.price }}  </template>  </el-table-column>  <el-table-column  label="商品图片"  prop="">  <template slot-scope="scope">  {{ scope.row.pimage }}  </template>  </el-table-column>  <el-table-column  label="操作"  prop="">  </el-table-column>        </el-table>    </div>  </template>  <script>  import { getList } from '@/api/table'  export default {    data() {  return {  list:''  }  },  created() {  this.fetchData()  },  methods: {  fetchData() {  var vm = this;  this.axios({  methods:'get',  url:'https://www.easy-mock.com/mock/5d392126948d9b06fdecbc3e/login/products'  }).then(function(resp){  vm.list = resp.data.products;  console.log(vm.list);    })  }  }  }  </script> |

# 十三、实现数据的增删改查

1.查： 商品列表的展示

2. 添加商品

3.参数传递的问题

路由表里 设参：



传参



接参

