vue-cli中如何配置mock

1.1. 什么 mock

- **是什么?**

  是一个生成随机数据，拦截 Ajax 请求的工具

  文档地址：http://mockjs.com/

- **运用场景**

  当我们想开发某个模块的功能，但是后台接口没有开发好，我们可以事先约定好和后端的接口规则，然后自己使用mock来模拟数据，直接进行项目开发，这样不会受后端的限制，加快前端开发的速度，更多的时间学习（摸鱼）

1.2. 基础使用

这里以在node.js中为例进行简单讲解，掌握核心方法，稍后下一节在具体讲解在`vue-cli`中使用。

```
const Mock = require('mockjs')
/*
模拟:
  {
    code:200,
    data:{
      nickname:'用户昵称',
      sex:'0或1',
      avatar:'头像地址'
    }
  }
*/
// Mock.mock(语法模板对象)
let res = Mock.mock({
  "nickname":"@cname",  //随机生成中文名字
  "sex|1":[0,1],  // 在数组中取随机某个值
  "avatar": "@image('200x200')"  //随机生成图片地址200*200
})
console.log(res);
```

```
D:\Desktop\test>node index.js
{ nickname: '贺霞', avatar: 'http://dummyimage.com/200x100' }

D:\Desktop\test>node index.js
{ nickname: '董桂英', sex: 0, avatar: 'http://dummyimage.com/200x200' }

D:\Desktop\test>node index.js
{ nickname: '段秀英', sex: 1, avatar: 'http://dummyimage.com/200x200' }
```

具体语法参见：http://mockjs.com/examples.html

1.3. vue-cli中如何使用mock

1.3.1. 基础搭建

- 创建项目

  创建一个vue项目，使用脚手架是 `vue-cli` 新版本脚手架，版本号4.4.4

  ```
  vue create vuedemo
  ```

- 梳理相关内容，将`App.vue`里面的内容清空，设置初始模板样式

  ```
  <template>
    <div id="app">
    </div>
  </template>
  <script>
  export default {
    name: 'App'
  }
  </script>
  <style>
  </style>
  ```

- 安装`axios`
  - 安装`axios`

    ```
    npm i  axios
    ```

  - 创建`axios`实例对象，挂载到Vue 原型上面

    ```
    import Vue from 'vue'
    ```

```
import App from './App.vue'
// 1.导入axios
import axios from 'axios'

// 2.创建axios实例对象
const http = axios.create({
  baseURL:'/api',
  timeout:6000
})
// 3.挂载到vue原型，给所有组件使用
Vue.prototype.$http = http

Vue.config.productionTip = false

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

1.3.2. 使用mockjs

- 安装`mockjs`

```
npm i mockjs
```

- 在根目录（`package.json`同级）创建`mock`目录，在里面创建`index.js`文件，书写以下内容

```
// 导入mockjs
const Mock = require('mockjs');

// 核心方法
// Mock.mock(请求地址,请求方式,数据模板)

// 拦截登录请求，返回数据
Mock.mock('/api/login','post',{
  code:200,
  data:{
    userinfo:{
      username:"@cname",  // 返回随机中文名
      "sex|1":[0,1],
      avatar: "@image('200x200')"
    },
    token:"@word(50)"  // 返回长度为50的英文字符串
```

```
  }
})
```

- 在`main.js`中配置导入 我们定义的`mock`，注意我们一般只需要在开发环境下使用，所以用环境变量做一下判断

```
// 其他代码...

// 【如果在开发环境下则使用mock】
if(process.env.NODE_ENV === 'development'){
  require("../mock");
}

// 其他代码...
```

- `App.vue`中测试

```
<template>
  <div id="app">
  </div>
</template>
<script>
export default {
  name: 'App',
  data() {
    return {
      // 1. 定义临时数据
      loginInfo:{
        username:'admin',
        password:'admin888'
      }
    }
  },
  created () {
    // 3. 调用测试方法
    this.loginTest()
  },
  methods: {
    // 2. 登录测试方法
    async loginTest() {
     const res =  await
this.$http.post('/login',this.loginInfo)
     console.log(res.data);  // 神奇的事情发生了，这里会输出返回的数
据内容！！！
    }
```

```
  },
}
</script>
<style>
</style>
```

通过观察控制台 发现会返回数据，而且刷新一下，每次返回的都不一样，成功，注意不要看`network`因为不会发送请求，mock本身原理就是拦截请求，直接返回数据。

1.3.3. 使用升级

- `Mock.mock(请求地址,请求方式,函数)`

  第三个参数可以是函数，且该函数有个形参，形参为请求对象信息，该函数的返回值就作为拦截请求返回的数据，这样我们就可以在这里做一些业务逻辑判断，如下面的用户账户密码判断

```
// 导入mockjs
const Mock = require('mockjs');

// 核心方法
// Mock.mock(请求地址,请求方式,数据模板)

// 模拟登录请求
// Mock.mock('/api/login','post',{
//   code:200,
//   data:{
//     userinfo:{
//       username:"@cname",
//       "sex|1":[0,1],
//       avatar: "@image('200x200')"
//     },
//     token:"@word(50)"
//   }
// })

//  Mock.mock(请求地址,请求方式,函数)
//  形参3 为函数
//  1.函数的形参为 请求对象信息（地址/方式/数据）
//  2.函数的返回值将作为拦截请求返回的数据
Mock.mock('/api/login','post',function(request){
  // 请求提交的数据
```

```
  let data = JSON.parse(request.body)
  // 判断账号密码
  if(data.username && data.password){
    if(data.username !== 'admin') return { code:401, data:'账号
不正确' }
    if(data.password !== 'admin888') return { code:401,
data:'密码不正确' }
    return {
      code:200,
      data:{
        userinfo:{
          username:"@cname",
          "sex|1":[0,1],
          avatar: "@image('200x200')"
        },
        token:"@word(50)"
      }
    }
  }else{
    return { code:403,data:'请提交账号和密码'}
  }
})
```

- 延迟拦截

```
// 导入mockjs
const Mock = require('mockjs');

//延时400s请求到数据
Mock.setup({
  timeout: 400
})
```

- 更多细节，应该就是掌握他的那些有意思的语法了