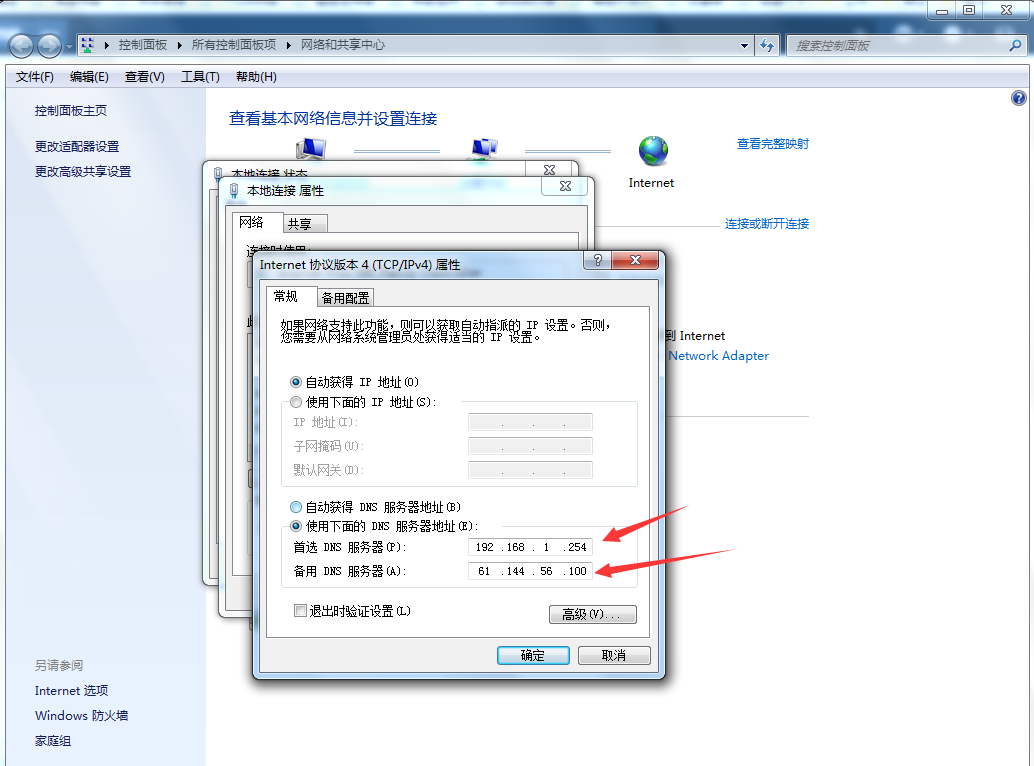
1. 前端开发环境准备
2. 办公室dns设置



1. 安装nodejs并设置

npm config set registry <http://nexus.toceansoft.com/repository/npm-tocean/>

**或**

**yarn config set registry**[**http://nexus.toceansoft.com/repository/npm-tocean/**](http://nexus.toceansoft.com/repository/npm-tocean/)

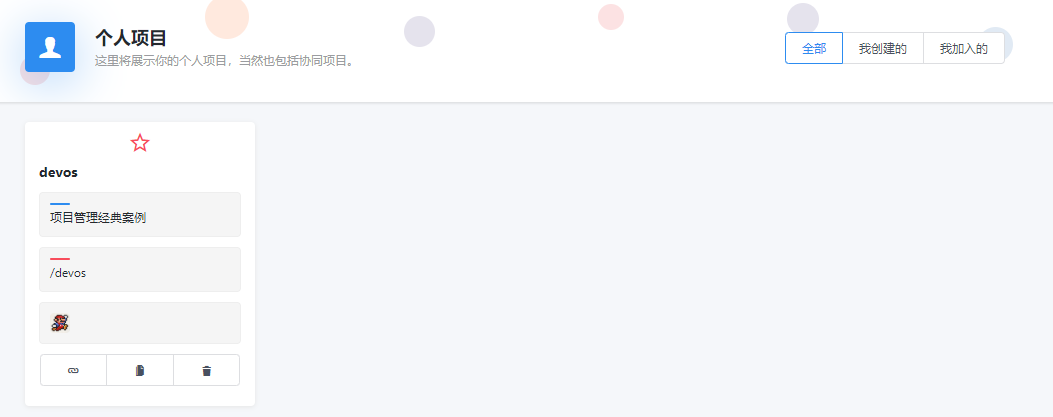
1. easy mock配置

登陆

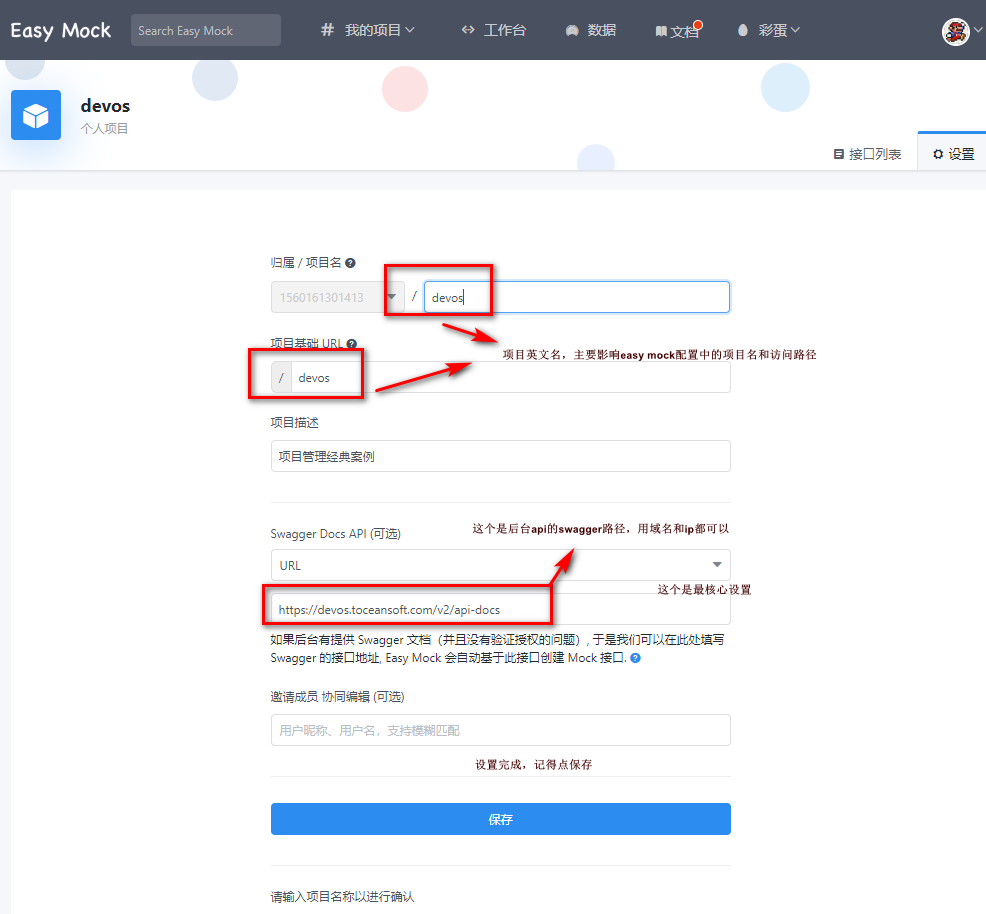
<http://devos.toceansoft.com:7300/>

账号 narci 密码 tocean788

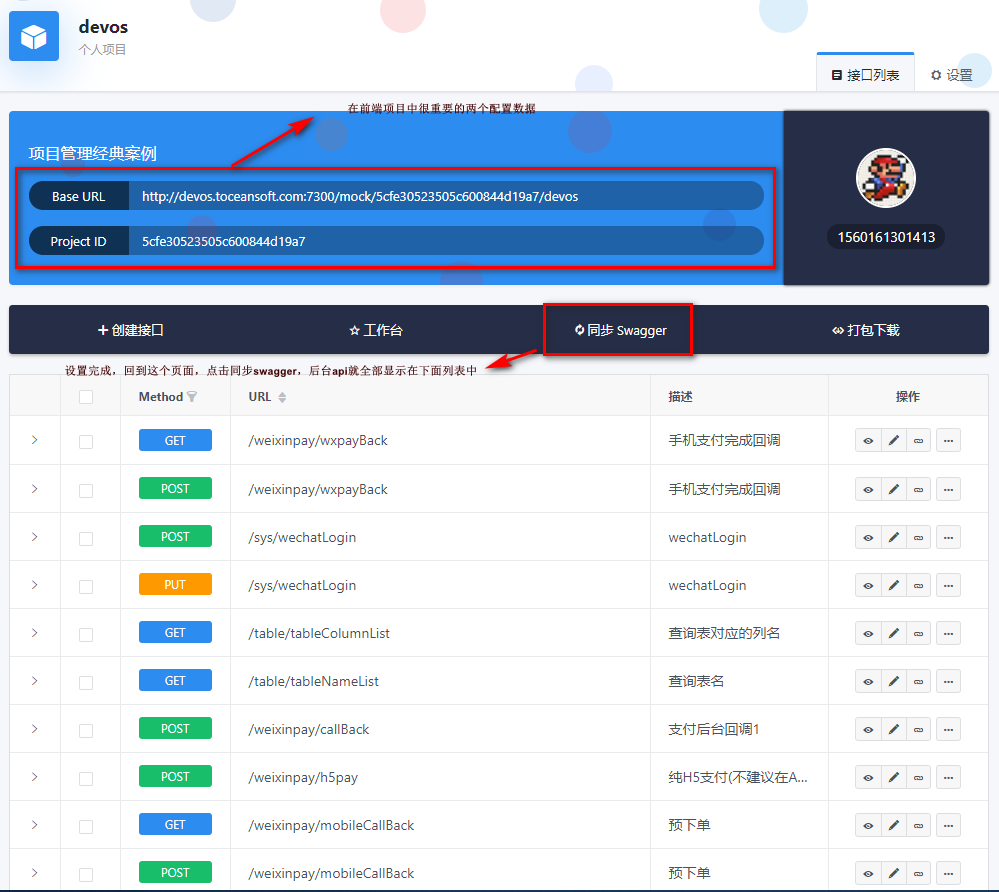
创建项目



配置项目



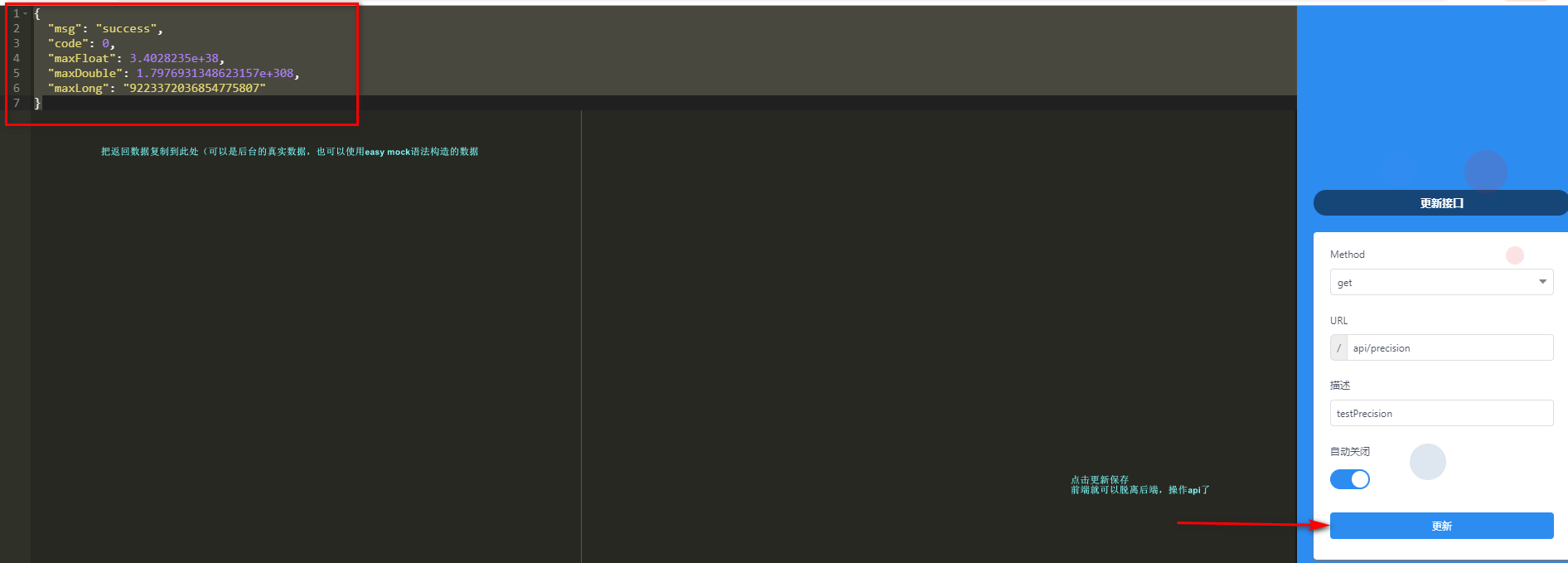
同步api



修改api的返回值（自己根据需要编写）



进入修改页面



1. API返回值问题

不管API正常执行，还是出现异常，统一返回R对象，R对象一定包含以下两个属性

(少数需求除外，如下载文件等，直接写respone对象，可以返回void)

code:0和2xx表示成功返回，其他值表示出现异常或错误

msg:错误的信息提示

如果后台返回值不含这两个信息，说明后台有问题，前端应该反馈后台排查错误。

返回值实例：正确

{

"msg": "success",

"code": 0,

"user": {

"userId": 1,

"username": "admin",

"password": "2f561551901025c04b601e86fd9f19b0",

"email": "admin@toceansoft.com",

"mobile": "13888888888",

"status": 1,

"roleIdList": null,

"createUserId": 1,

"createTime": "2018-01-18 11:11:11"

}

}

返回值实例：错误

{

"msg": "该记录不存在：1",

"code": 500

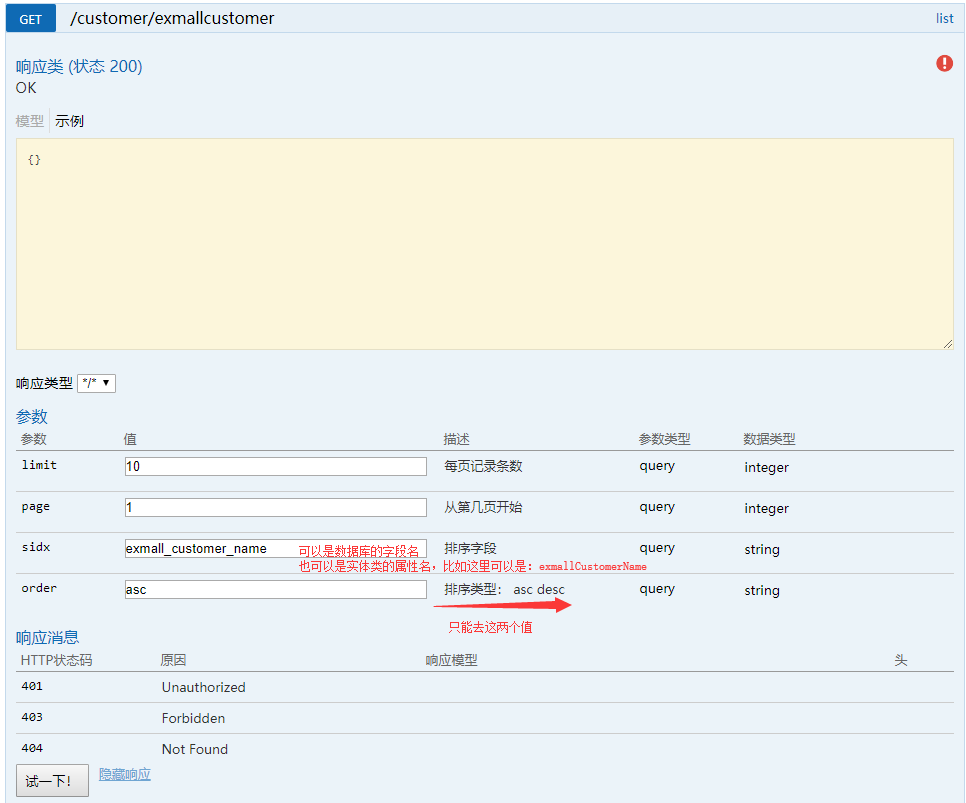
}

1. 拓胜开发框架自动生成接口默认支持以下8个API

以上8个api基本上比较简单，自己看Swagger 文档即可。

这里重点解释2个稍微复杂的查询，特别是动态条件列表查询

1. 列表查询



* 1. 前端项目规范

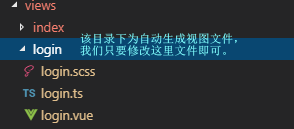
1. 目录结构说明

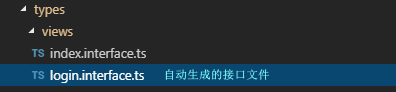


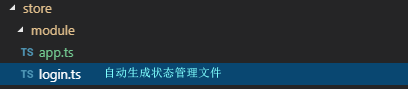
1. 增加一个页面举例说明
2. 增加登陆Login页面

只要运行以下指令

npm run tep login

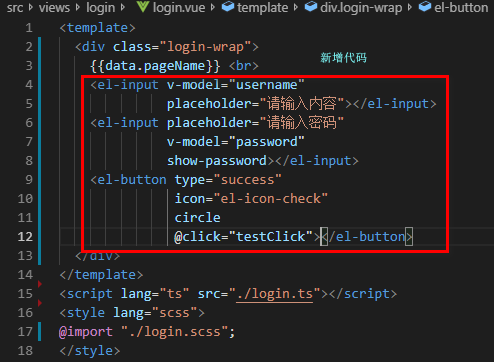




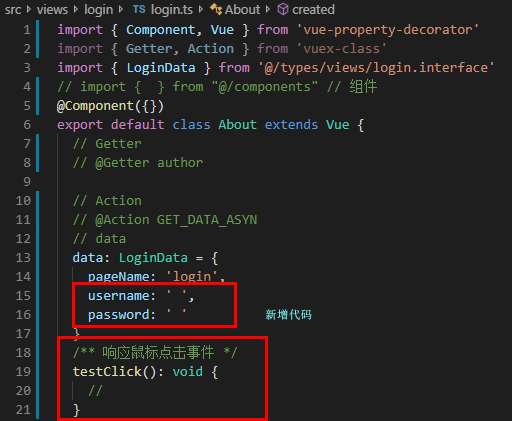


手工修改

/views/login/login.vue

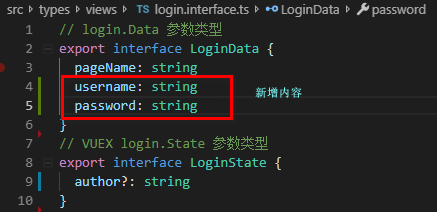


/views/login/login.ts 增加如下内容



同时修改login模块的接口文件，增加两个数据属性username和password（上图输入框中用到的两个变量）

/types/views/login.interfaces.ts



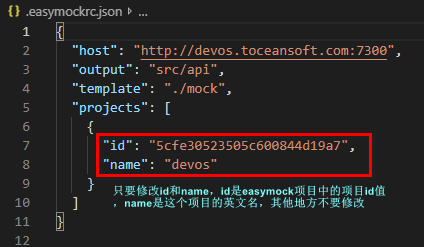
后台api调用

## step1:安装 CLI

npm install -g easy-mock-module-cli (第一次安装)

npm update easy-mock-module-cli –g （如果有新版本，则升级）

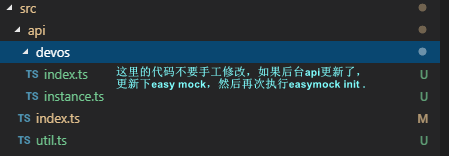
##step2 修改配置



## step3:初始化api

toceanmock init .

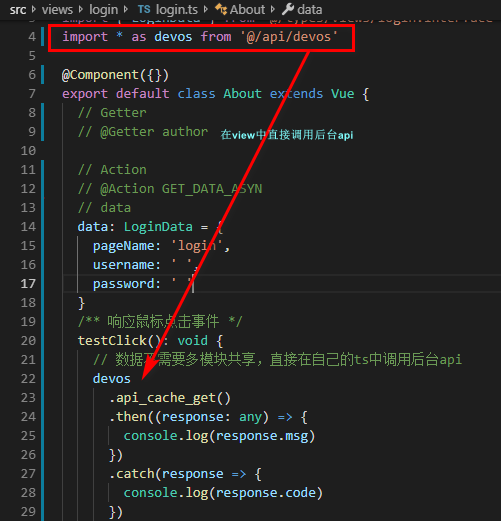
执行之后生成下图代码



前端调用后台api可以在两个地方

1. 直接在/views/login/login.ts

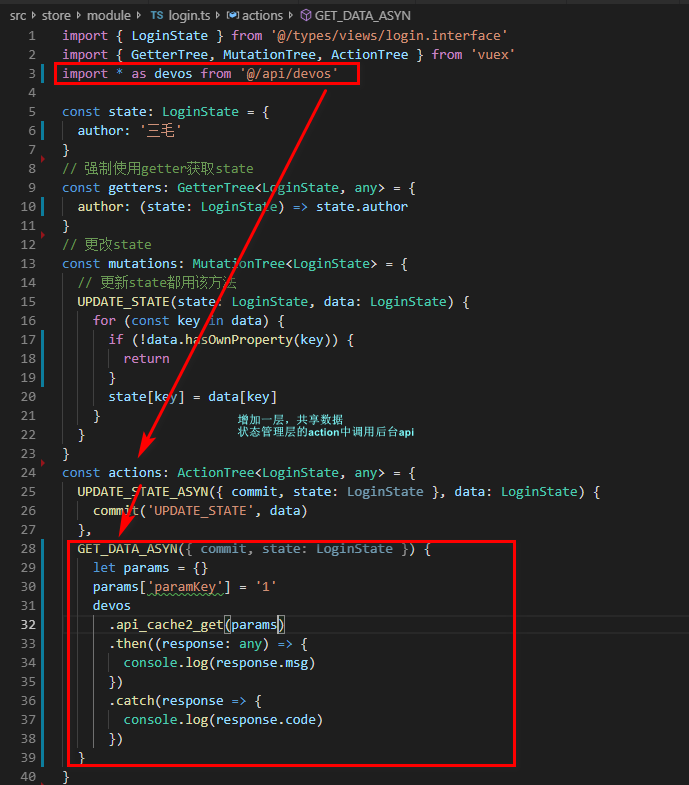
如果数据是login页面需要的，不需要跟其他模块交互，则直接在login.ts访问api。



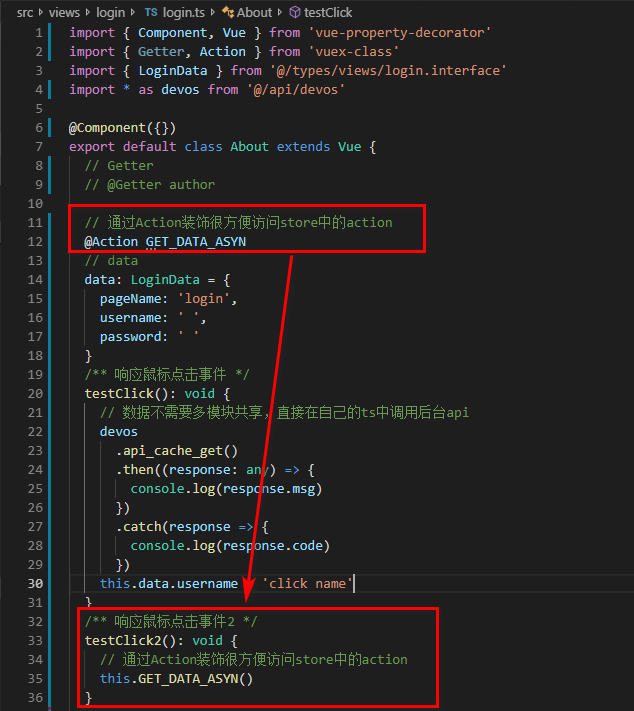
1. 在/store/module/login.ts

如果数据除了login页面外，还有其他模块的页面需要共享数据，则应该在增加一层，/views/login/login.ts调用/store/module/login.ts再访问api

/store/module/login.ts 访问后台api

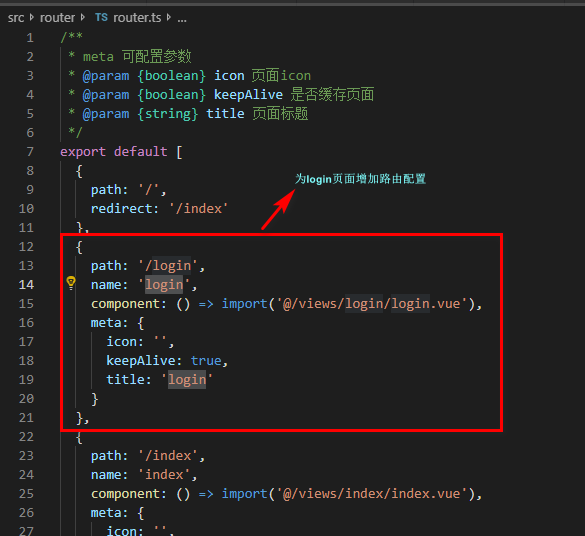


/views/login/login.ts调用/store/module/login.ts



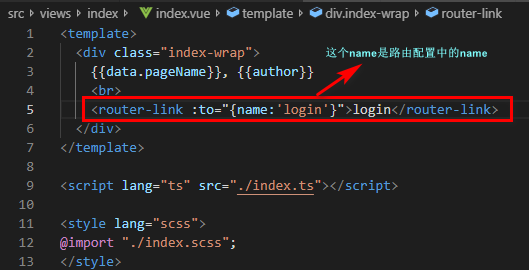
1. 为login页面增加路由

/router/router.ts



首页上增加链接

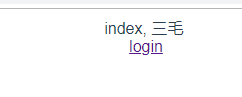
/views/index/index.vue

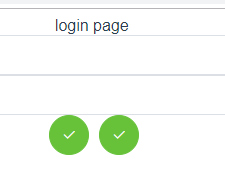


运行：

npm install

npm run serve





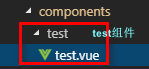
1. 增加组件

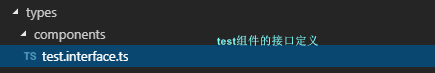
什么时候开发组件？一个项目里面，如果很多个页面都存在重复代码，就可以把这些代码提炼到一个组件里面。本框架要增加一个组件很简单，只要运行脚本即可。

1. 创建组件。

npm run com test

自动生成代码如下

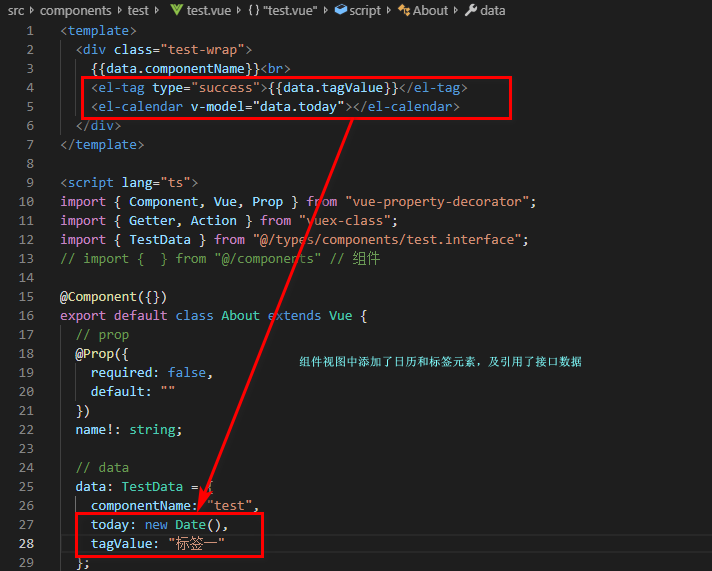




编写组件（UI在/components/test/test.vue中添加，暴露数据则在接口/types/components/test.interface.ts中定义）

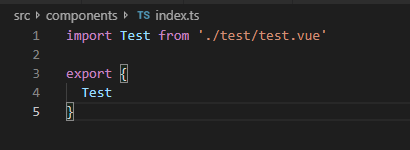
我们新增的组件包含一个标签元素和一个日历元素





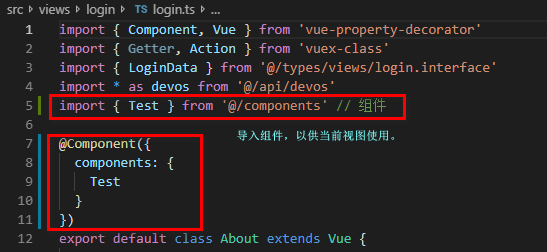
导出组件，方便其他视图引用

/components/index.ts添加代码，导出组件

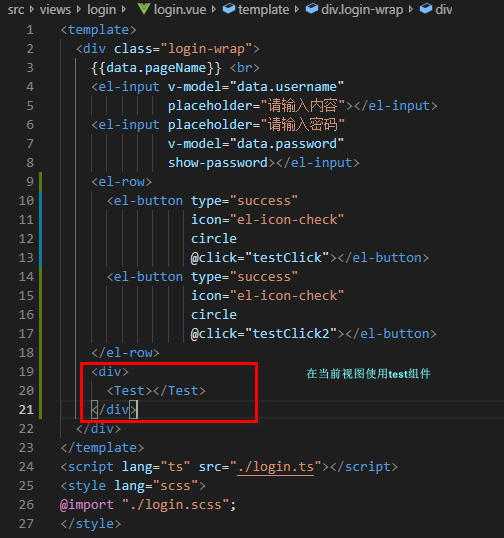


1. 在login视图中引用新创建的test组件

/views/login/login.ts



/views/login/login.vue



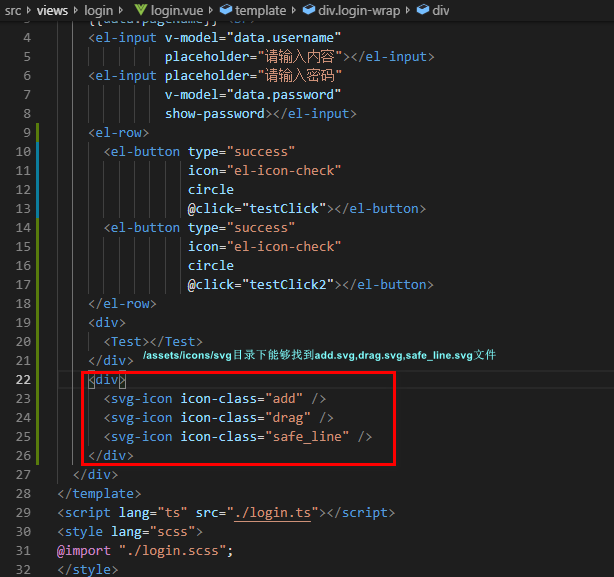
1. 运行效果



1. UI组件及ICON说明

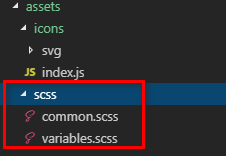
本框架UI组件默认使用饿了么UI组件。如果[饿了么UI组件](https://element.eleme.cn/#/zh-CN/component/installation)不能胜任需求，则再寻找其他第三方组件。

项目中的图标应该统一使用svg格式图片。默认使用饿了么的图片，具体使用参加饿了么官方文档：[图标使用大全](https://element.eleme.cn/#/zh-CN/component/icon)。如果饿了么中图标不够用，可以网上查找其他开源的svg格式图标库（如[阿里的开源svg库](https://www.iconfont.cn/)），下载svg格式文件，并拷贝到项目的/assets/svg目录下。使用这些图片的方法见下图。



1. css说明

多个模块公用的css放到/assets/scss目录下

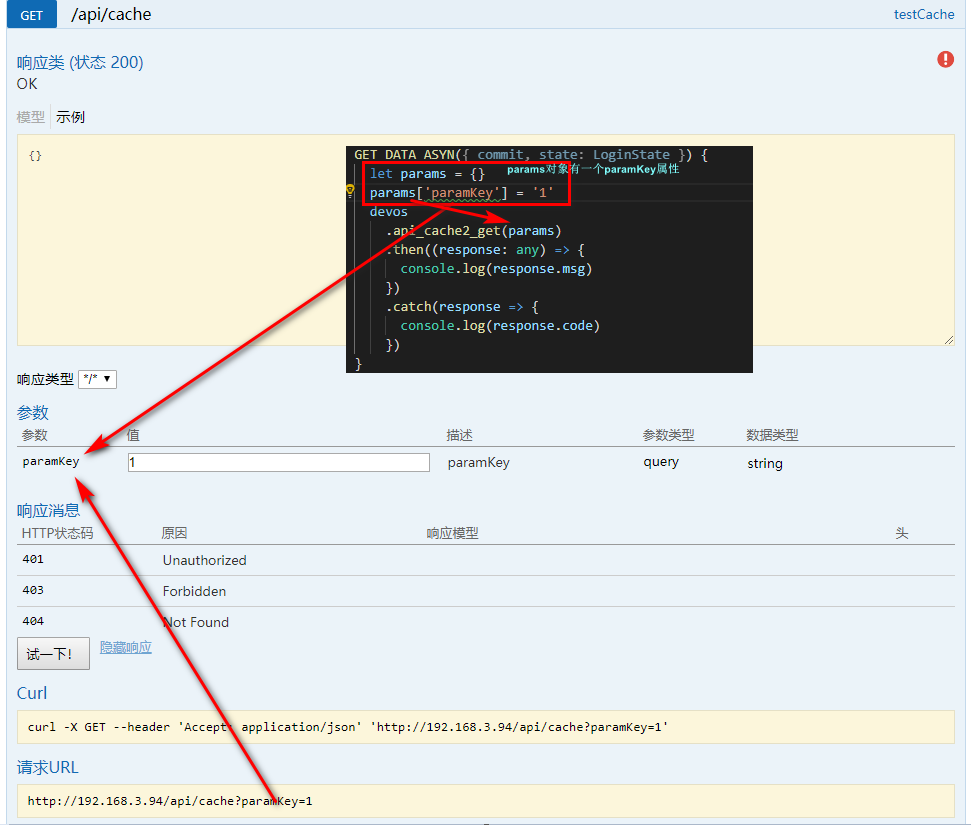


某个页面视图私有的css，放到/views/视图目录下

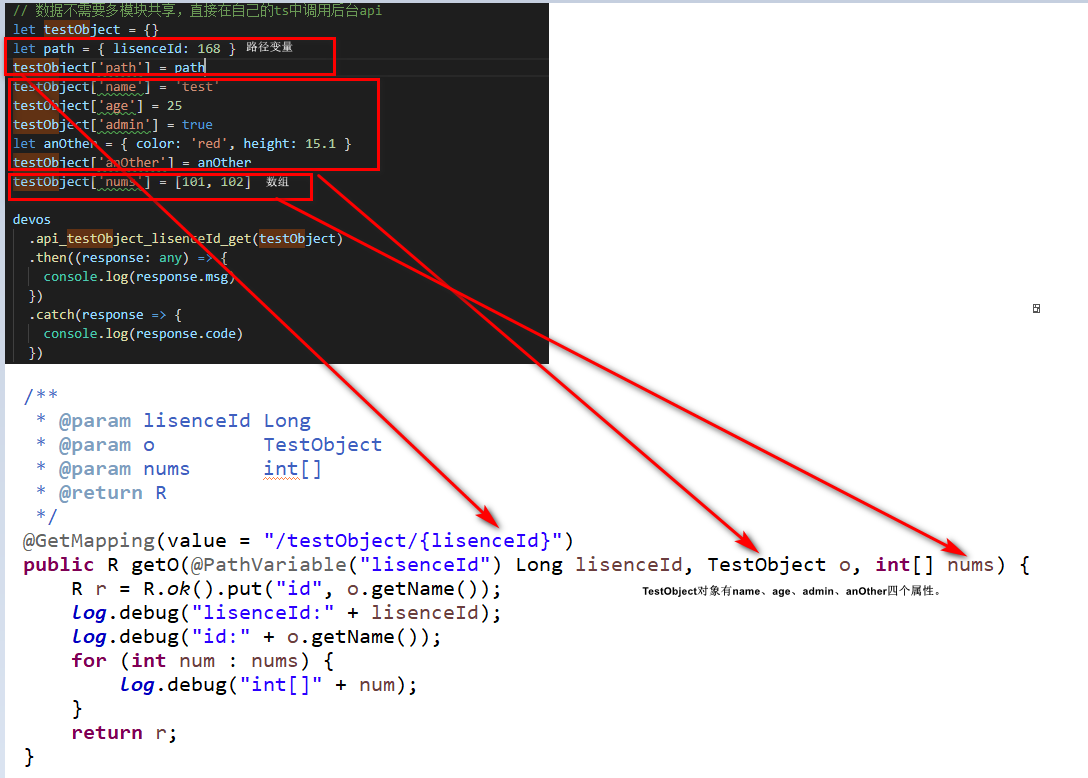


1. 后台api参数传递说明

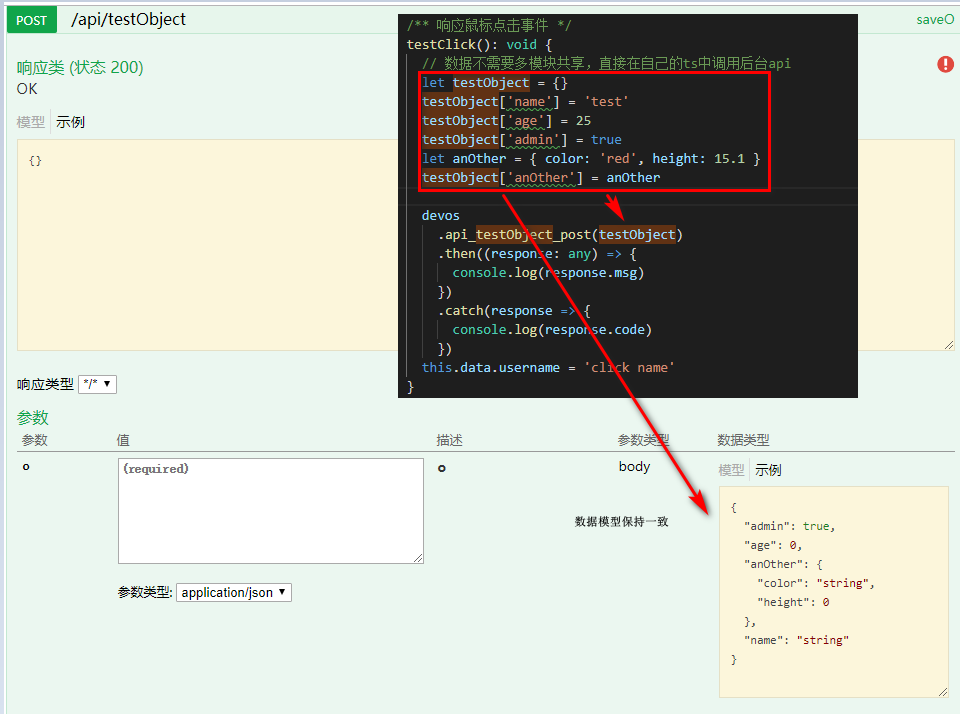
get、delete方法调用（参数不可以加@RequestBody修饰），如下图



有路径变量和数组的传参例子



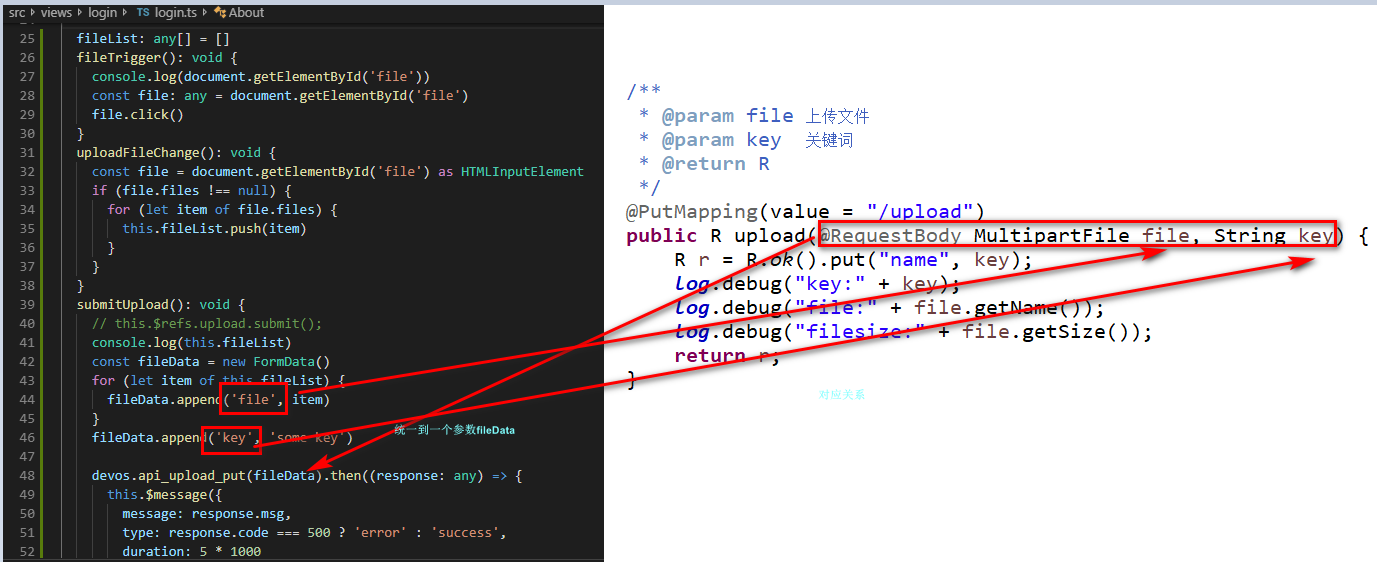
post、put（在Spring MVC中参数要加@RequestBody修饰）等方法调用，如下图



如果第一个参数是@RequestBody修饰，后面还有其他参数，后面的参数可以使用api的第二个参数传递，如下图



上传文件



1. 非常方便就可以切换后台环境

# 

# /utils/config.ts 全局设置（本前端框架支持同时访问多个后台系统api，此处设置参数对全部后台url路径生效。

# 

# 除了上面的全局设计，在多个后台系统的情况下，也支持局部设置，局部设置会覆盖全局设置。

# /api/backend/config.ts

# 

# 例子

# /api/devos/config.ts

# 

# /api/devos2/config.ts

# 

1. 日志对象this.$log有一组方法：['debug', 'info', 'warn', 'error', 'fatal']

开发期间方便调试，多用debug、info、warn输出调试信息。程序一些错误信息，则使用error和fatal。

调试信息：

this.$log.debug('test', 'this.a', 123)

this.$log.info('test', 'this.b')

this.$log.warn('test')

错误信息：

this.$log.error('test')

this.$log.fatal('test')

在this关键字失效的地方，写日志可以使用全局变量logger对象

logger.debug('test', 'this.a', 123)

logger.info('test', 'this.b')

logger.warn('test')

logger.error('test')

logger.fatal('test')

1. 运行前端

npm run serve // 后台连接dev环境api

npm run mock // 后台连接mock环境api

1. 编译发布前端代码

npm run build //发布到生成环境，后台连接生产环境api

npm run pre //发布到预发环境，后台连接预发环境api

# 参考文章

# [《vue-cli3.0 搭建项目模版教程》](https://www.gotofullstack.com/#/article/26874?type=expert)

# 《vuex-module-decorators 使用记录 typescript vuex》

# [《Vue-Class-Component 使用》](https://www.gotofullstack.com/#/article/26946?type=expert)