Axios的封装和接口调用

01、Aixos

- 基于http请求的异步处理的js库。vue本身没有异步处理框架。采用：axios作为基础。vue-axios作为插件进行全局的spa和组件调用。也就告诉你只要在vue-cli项目中，你安装了axios，vue-axios就可以在任何的spa和组件中（一句话：就是有vue生命周期的地方）你都可以使用 this.axios。
- vue-axios其实不能单独使用，必须要依赖安装axios。但是axios可以单独使用。你安装了vue-axios才可以在全局spa中使用组件。

02、安装和注册方式

安装

```
npm install -S axios vue-axios
```

注册，在main.js进行注册即可

```
import { createApp } from 'vue'
import axios from 'axios' //
import VueAxios from 'vue-axios'

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


const app = createApp(App);
app.use(router);
app.use(VueAxios,axios);//注册
app.mount('#app');

```

03、配置代理服务

因为现在的开发模式：前后端分离

- 开发阶段
  - 模式：
    - 本机开发：ip: 127.0.0.1 : 8081 —- 调用本机服务
    - 本机开发：ip: 127.0.0.1 : 8081 — 外网服务（阿里云服务）——-跨域
- 部署阶段
  - 前台项目云服务器——后台应用通服务器 —相同服务器
  - 前台项目云服务器——后台应用服务器 —不同服务器 ——-跨域

跨域存在的原因

跨域：是浏览器请求过程中，为了服务资源的安全性。进行一种防护手段。

如果请求域和服务的域（ip）不同。或者端口不同，协议不同，都将其限制。跨域。这种跨域一定是在服务被调用方去开启。

后台开启跨域：

```
package com.ksd.pug.config.webmvc;

import com.ksd.pug.config.interceptor.jwt.JwtInterceptor;
import com.ksd.pug.config.interceptor.repeat.RepeatSubmitInterceptor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.InterceptorRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

```

```java
/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2022/1/6 17:50
 */
@Configuration
public class WebMvcConfiguration implements WebMvcConfigurer {

    /**
     * jwt的token校验
     */
    @Autowired
    public JwtInterceptor jwtInterceptor;

    /**
     * 表单重复提交
     */
    @Autowired
    private RepeatSubmitInterceptor repeatSubmitInterceptor;


    /**
     * 拦截的注册
     * @param registry
     */
    @Override
```

```
40      public void addInterceptors(InterceptorRegistry
registry) {
41
 registry.addInterceptor(repeatSubmitInterceptor).addP
athPatterns("/admin/v1/**");
42
 registry.addInterceptor(jwtInterceptor).addPathPatter
ns("/admin/v1/**");
43      }
44
45
46      /**
47       * 解决跨域问题
48       * @param registry
49       */
50      @Override
51      public void addCorsMappings(CorsRegistry registry)
{
52          registry
53                  .addMapping("/**")
54                  //.allowedOrigins("http://yyy.com",
"http://xxx.com") //
55                  // 允许跨域的域名
56                  .allowedOriginPatterns("*") // 允许所有
域
57                  .allowedMethods("POST", "GET", "PUT",
"OPTIONS", "DELETE")
58                  //.allowedMethods("*") // 允许任何方法
（post、get等）
59                  .allowedHeaders("*") // 允许任何请求头
60                  .allowCredentials(true) // 允许证书、
cookie
61                  .maxAge(3600L); // maxAge(3600)表明在
3600秒内，不需要再发送预检验请求，可以缓存该结果
```

```
    }
}
```

> 原理就是：接口调用结束之后，会往浏览器的响应头中写一段告诉浏览器，这个调用接口的域可以获取数据。

前台跨域的开放：

在vue-cli项目中，因为考虑axios安全性，webpack在构建项目的时候，将其异步请求如果设置的 token或者auth头的时候，会将其限制服务访问，如何处理呢？在当前工程下根目录下新建一个文件：vue.config.js。如下：

```
module.exports = {


    // webpack-dev-server 相关配置
    devServer: {
        open: true,
        host: 'localhost',
        port: port,
        proxy: {
            ///api是后端数据接口的上下文路径
            '/admin': {
                //这里的地址是后端数据接口的地址
                target: 'http://120.77.34.190:8081/',
                //
                // pathRewrite: {
                //     '^/admin': ''// 这种接口配置出来
  http://XX.XX.XX.XX:8083/login
```

```
                //     //'^/api': '/' 这种接口配置出来
    http://xx.xx.xx.xx:8083/login
                // },
                //允许跨域
                changeOrigin: true,
                // 允许ws访问
                ws: true,
                // 允许https访问
                secure: false
            }

    }
}

```

如何理解路由代理

真实的访问地址：

```
http://120.77.34.190:8081/admin/v1/user/get/1
```

访问路径：

```
http://localhost:8081/index
```

异步请求地址

```
axios.post("/admin/v1/user/get/1").then(res=>{

})
```

根据相对路径的规则

```
axios.post("/admin/v1/user/get/1").then(res=>{

})
```

如果你在项目的根目录配置vue.config.js的proxy。那么就将其进行转发

```
axios.post("http://120.77.34.190:8081//admin/v1/user/get/1").then(res=>{

})
```

pathRewrite 是何物？

pathRewrite是一个路由伪造去除

真实地址： http://120.77.34.190:8081/v1/user/get/1

调用的时候有：

```
axios.post("/api/user/get/1").then(res=>{

})
```

第一个问题：为什么/api ，统一化，看到这个就代表的接口。

第二个问题：/api 这个路径是不存在，所有在最后转发应该去除。

为什么平时项目没有跨域问题呢？

飞哥：为什么平时开放项目中，我们不存在跨域

- springboot项目—/admin/v1/user/get/1
- vue调用

  http://locaclhost:8088/index

  ```
  1 axios.post("http://locaclhost:8088/admin/v1/user/get/1")
  ```

因为相对路径：是相对浏览器的访问路径，它们都是在一个域下面。当然就不存在跨域问题。

注意修改了**vue.config.js**

一定要记得重启

- 执行多次ctrl+c 关闭服务
- npm run serve

04、接口调用整个过程

测试后台（真实发布和部署）

- 创建一个springboot项目
- 定义接口，跑通
- 将其发布到服务器即可，
- 通过 postman测试通过没问题

前台

- axios—在前台通过axios将其调用，
- vue-axios—在前台通过axios将其调用
- 自定义axios对象-—在前台通过axios将其调用
- 然后使用vue将其渲染，呈现结果

05、实现登录获取token

后台部分：

- 创建一个springboot项目
- 定义接口，用户登录接口

Loginvo

```java
package com.ksd.pug.vo;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦!
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2022/1/20$ 19:02$
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class LoginVo implements java.io.Serializable {
    private String username;
    private String password;
    private String code;
}

```

LoginController.java

```java
  package com.ksd.pug.controller.login;

  import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
  import com.ksd.pug.commons.anno.PugLog;
```

```
5  import com.ksd.pug.commons.anno.PugRateLimiter;
6  import com.ksd.pug.commons.enums.ResultStatusEnumA;
7  import com.ksd.pug.commons.ex.BussinessException;
8  import
com.ksd.pug.commons.utils.fn.asserts.Vsserts;
9  import com.ksd.pug.commons.utils.jwt.JwtTokenUtils;
10 import com.ksd.pug.commons.utils.pwd.MD5Util;
11 import
com.ksd.pug.config.interceptor.repeat.RepeatSubmit;
12 import com.ksd.pug.pojo.SysLoginUser;
13 import
com.ksd.pug.service.jwt.JwtBlackStringService;
14 import
com.ksd.pug.service.user.ISysLoginUserService;
15 import com.ksd.pug.vo.LoginUserVo;
16 import com.ksd.pug.vo.LoginVo;
17 import com.pug.redis.config.PugRedisCacheTemplate;
18 import lombok.RequiredArgsConstructor;
19 import lombok.extern.slf4j.Slf4j;
20 import
org.springframework.web.bind.annotation.PostMapping;
21 import
org.springframework.web.bind.annotation.RequestBody;
22 import
org.springframework.web.bind.annotation.RestControlle
r;
23
24 import javax.servlet.http.HttpServletRequest;
25 import java.util.concurrent.TimeUnit;
26
27 /**
28  * @author 飞哥
29  * @Title: 学相伴出品
```

```
30     * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
31     * 记得关注和三连哦！
32     * @Description: 我们有一个学习网站：https://www.kuangstudy.com
33     * @date 2022/1/1 23:35
34     */
35   @RestController
36   @RequiredArgsConstructor
37   @Slf4j
38   public class LoginController {
39
40       private final ISysLoginUserService sysLoginUserService;
41       private final PugRedisCacheTemplate pugRedisCacheTemplate;
42       private final JwtBlackStringService jwtBlackListService2;
43
44
45       /**
46        * @param loginVo
47        * @return
48        */
49       @PostMapping("/auth/login/pwd")
50       @PugLog(title = "登录生成token")
51       @PugRateLimiter(timeout = 1, limit = 3)
52       public LoginUserVo loginbynamepwd(@RequestBody LoginVo loginVo) {
53           log.info("登录的用户是：{}", loginVo);
54           Vsserts.isEmptyEx(loginVo.getUsername(), new BussinessException(ResultStatusEnumA.USER_USERNAME_STATUS));
```

```
55          Vsserts.isEmptyEx(loginVo.getPassword(),
new
BussinessException(ResultStatusEnumA.USER_PWR_STATUS_
INPUT));
56          // 1: 根据username查询用户信息
57          LambdaQueryWrapper<SysLoginUser>
userLambdaQueryWrapper = new LambdaQueryWrapper<>();
58
 userLambdaQueryWrapper.eq(SysLoginUser::getUsername,
loginVo.getUsername());
59          SysLoginUser user =
sysLoginUserService.getOne(userLambdaQueryWrapper);
60          if (user != null) {
61              // 把用户输入的密码进行md5假面
62              String inputpwd =
MD5Util.md5slat(loginVo.getPassword());
63              // 然后把用户输入的加密的密码和数据库中
user.getPassword()进行比较
64              if
(!inputpwd.equalsIgnoreCase(user.getPassword())) {
65                  throw new
BussinessException(ResultStatusEnumA.USER_PWR_STATUS)
;
66              }
67
68              // 2: 生成token
69              String token =
JwtTokenUtils.createToken(user);
70              // 3: 创建vo,按需加载封装返回即可
71              LoginUserVo loginUserVo = new
LoginUserVo();
72              loginUserVo.setUserId(user.getId());
73              loginUserVo.setToken(token);
74
```

```
75                  // 写入信息到redis缓存中
76                  String tokenKey = "sys:user:token:" +
    token;
77
     pugRedisCacheTemplate.setCacheObject(tokenKey,
    token, JwtTokenUtils.REDIS_TOKEN_EXPIRATION,
    TimeUnit.MILLISECONDS);
78                  return loginUserVo;
79              } else {
80                  // 代表输入账号密码有误
81                  throw new
    BussinessException(ResultStatusEnumA.USER_PWR_STATUS)
    ;
82              }
83          }
84
85          /**
86           * 退出登录
87           *
88           * @param request
89           * @return
90           */
91          @PostMapping("/auth/logout")
92          @PugLog(title = "退出登录")
93          public String logout(HttpServletRequest
    request) {
94              // 获取请求toekn
95              String token =
    JwtTokenUtils.getJwtToken(request);
96              // 把当前token拉入黑名单中
97              jwtBlackListService2.addBlackList(token);
98              // 返回成功
99              return "success";
100         }
```

```
101     }
102
```

- 将其发布到服务器即可
- 通过 postman测试通过没问题

Postman测试脚本

```
{
    "info": {
        "_postman_id": "998487a0-b230-47c7-9840-2e8fa2dd0c74",
        "name": "旅游项目实战-接口集合",
        "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
    },
    "item": [
        {
            "name": "用户模块",
            "item": [
                {
                    "name": "1、查询用户明细",
                    "request": {
                        "method": "GET",
                        "header": [
                            {
                                "key": "Authorization",
                                "value": "Bearer xxxx11111",
                                "type": "default"
                            }
                        ],
                        "url": {
```

```
                            "raw":
"http://120.77.34.190:8081/admin/v1/user/get/1",
                            "protocol": "http",
                            "host": [
                                "120",
                                "77",
                                "34",
                                "190"
                            ],
                            "port": "8081",
                            "path": [
                                "admin",
                                "v1",
                                "user",
                                "get",
                                "1"
                            ]
                        }
                    },
                    "response": []
                },
                {
                    "name": "2、登录创建token",
                    "request": {
                        "method": "POST",
                        "header": [],
                        "body": {
                            "mode": "raw",
                            "raw": "
{\"username\":\"yykk\",\"password\":\"123456\"}",
                            "options": {
                                "raw": {
                                    "language":
"json"
```

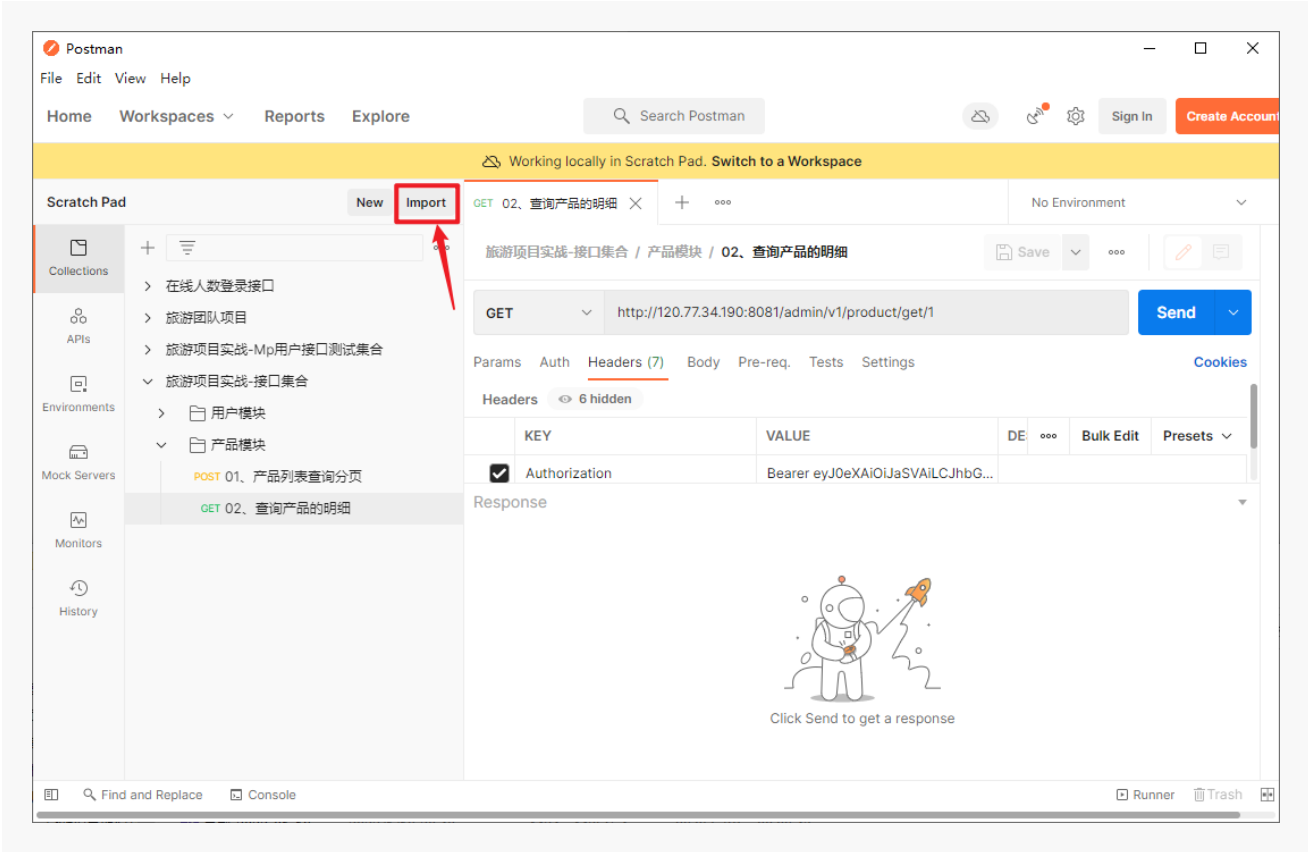
```
                                }
                            }
                        },
                        "url": {
                            "raw":
"http://120.77.34.190:8081/auth/login/pwd",
                            "protocol": "http",
                            "host": [
                                "120",
                                "77",
                                "34",
                                "190"
                            ],
                            "port": "8081",
                            "path": [
                                "auth",
                                "login",
                                "pwd"
                            ]
                        }
                    },
                    "response": []
                }
            ]
        },
        {
            "name": "产品模块",
            "item": [
                {
                    "name": "01、产品列表查询分页",
                    "request": {
                        "method": "POST",
                        "header": [
                            {
```

```
                                "key":
"Authorization",
                                "value": "Bearer
eyJ0eXAiOiJaSVAiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxIiwi
aXNzIjoieXlrayIsImV4cCI6MTY0MjY5MDIzMCwiaWF0IjoxNjQyN
jg4NDMwfQ.5oiELJ1RgiBqN5LCnsr-E-
5pZnEN6FTLmSqwr7sT854",
                                "type": "default"
                            }
                        ],
                        "body": {
                            "mode": "raw",
                            "raw": "{\r\n
\"pageNo\":\"1\",\r\n    \"pageSize\":\"10\",\r\n
\"keyword\":\"\"\r\n}",
                            "options": {
                                "raw": {
                                    "language":
"json"
                                }
                            }
                        },
                        "url": {
                            "raw":
"http://120.77.34.190:8081/admin/v1/product/list",
                            "protocol": "http",
                            "host": [
                                "120",
                                "77",
                                "34",
                                "190"
                            ],
                            "port": "8081",
                            "path": [
```

```
                                "admin",
                                "v1",
                                "product",
                                "list"
                            ]
                        }
                    },
                    "response": []
                },
                {
                    "name": "02、查询产品的明细",
                    "request": {
                        "method": "GET",
                        "header": [
                            {
                                "key":
"Authorization",
                                "value": "Bearer
eyJ0eXAiOiJaSVAiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxIiwi
aXNzIjoieXlrayIsImV4cCI6MTY0MjY5MDIzMCwiaWF0IjoxNjQyN
jg4NDMwfQ.5oiELJ1RgiBqN5LCnsr-E-
5pZnEN6FTLmSqwr7sT854",
                                "type": "default"
                            }
                        ],
                        "url": {
                            "raw":
"http://120.77.34.190:8081/admin/v1/product/get/1",
                            "protocol": "http",
                            "host": [
                                "120",
                                "77",
                                "34",
                                "190"
```

```
                ],
                "port": "8081",
                "path": [
                    "admin",
                    "v1",
                    "product",
                    "get",
                    "1"
                ]
            }
        },
        "response": []
    }
  ]
}
```

postman导入



06、使用Axios完成登录获取Token

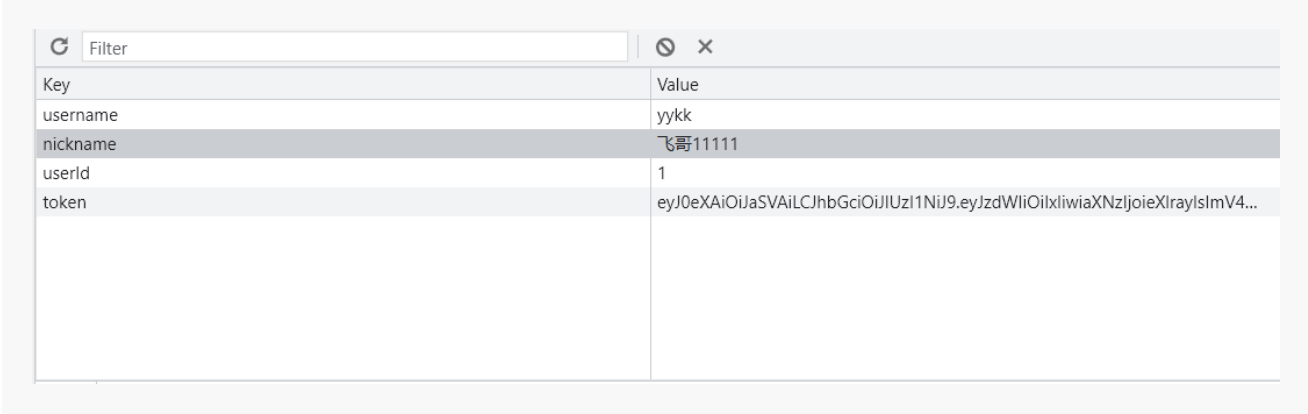
打开前台login.vue的spa页面如下：

```
<template>
  <h1>我是登录页面</h1>
  <div>
    <p>账号：<input type="text" v-model.trim="username"></p>
    <p>密码：<input type="text" v-model.trim="password"></p>
    <p>
      <button @click="logined">登录</button>
    </p>
  </div>
</template>

<script>

export default {
  name: "login.vue",

  data() {
    return {
      username: "yykk",
      password: "123456"
    }
  },

```

```
24    methods: {
25
26      logined() {
27        // 1: 获取用户输入的账号和密码
28        let username = this.username;
29        let password = this.password;
30        // 2: 发起异步请求
31        this.axios.post("/auth/login/pwd", {username,
password}).then(res => {
32          if (res.data.status == 200) {
33            // 3: 获取服务器段返回的数据信息
34            const {token, userId,username,nickname} =
res.data.data;
35            // 4: 把服务器段信息放入到session中  $store
36            sessionStorage.setItem("token", token);
37            sessionStorage.setItem("nickname",
nickname);
38            sessionStorage.setItem("username",
username);
39            sessionStorage.setItem("userId", userId);
40            // 5: 跳转后台首页
41            this.$router.push("/")
42          }
43        }).catch(err => {
44          console.error("登录失败!!!", err)
45        })
46      }
47    }
48
49  }
50  </script>
51
52  <style scoped>
53
```

```
54 </style>
```

浏览器获取服务器段的数据进行存储。



但是sessionStorage是无状态的，后续我们肯定会通过vuex进行状态管理存储，让这种数据变得有状态。

07、后台产品接口的定义

bean

```
1 package com.ksd.pug.pojo;
2
3 import com.baomidou.mybatisplus.annotation.*;
4 import lombok.AllArgsConstructor;
5 import lombok.Data;
6 import lombok.NoArgsConstructor;
7
8 import java.util.Date;
```

```java

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦！
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2022/1/15$ 22:52$
 */
@Data
@NoArgsConstructor
@AllArgsConstructor
@TableName("kss_product")
public class Product implements java.io.Serializable {
    /**
     * 主键
     */
    @TableId(type = IdType.ASSIGN_ID)
    private Long id;
    /**
     * 产品名称
     */
    private String name;
    /**
     * 产品封面
     */
    private String img;
    /**
     * 产品标签
     */
    private String tags;
    /**

```

```
41      * 产品价格
42      */
43     private String price;
44     /**
45      * 产品交易价格
46      */
47     private String realprice;
48     /**
49      * 产品描述
50      */
51     private String description;
52     /**
53      * 产品组图
54      */
55     private String imgjson;
56     /**
57      * 产品浏览数
58      */
59     private Integer views;
60     /**
61      * 产品收藏数
62      */
63     private Integer collections;
64     /**
65      * 产品分类id
66      */
67     private Integer categoryId;
68     /**
69      * 产品分类标题
70      */
71     private String categoryName;
72     /**
73      * 产品的商家
74      */
```

```
75      private Long userId;
76      /**
77       * 产品的删除
78       */
79      private Integer isdelete;
80      /**
81       * 发布状态
82       */
83      private Integer status;
84      /**
85       * 创建时间
86       */
87      @TableField(fill = FieldFill.INSERT)
88      private Date createTime;
89      /**
90       * 更新时间
91       */
92      @TableField(fill = FieldFill.INSERT_UPDATE)
93      private Date updateTime;
94  }
95
```

Mapper

```
1  package com.ksd.orm.pug.mapper;
2
3  import
   com.baomidou.mybatisplus.core.mapper.BaseMapper;
4  import com.ksd.pug.pojo.Product;
5
6  /**
7   * @author 飞哥
8   * @SysLoginUseritle: 学相伴出品
```

```
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2022/1/2 12:42
 */
public interface ProductMapper extends BaseMapper<Product> {
}

```

IService

```
package com.ksd.pug.service.product;

import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.service.IService;
import com.ksd.pug.pojo.Product;
import com.ksd.pug.vo.ProductVo;

/**
 * @author 飞哥
 * @SysLoginUseritle: 学相伴出品
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2022/1/2 12:42
 */
public interface IProductService extends IService<Product> {

    /**
     * 查询产品列表信息并分页
     *
     * @return
     */
```

```
21      IPage<Product> findProductPage(ProductVo
    productVo);
22
23
24      /**
25       * 保存和修改产品
26       *
27       * @param product
28       */
29      Product saveUpdateProduct(Product product);
30
31
32      /**
33       * 根据产品id删除产品
34       *
35       * @param id
36       * @return
37       */
38      boolean delProduct(Long id);
39
40      /**
41       * 根据产品id查询产品
42       *
43       * @param id
44       * @return
45       */
46      Product getProduct(Long id);
47
48
49  }
50
```

ServiceImpl

```
```

```java
package com.ksd.pug.service.product;

import com.baomidou.mybatisplus.core.conditions.query.LambdaQueryWrapper;
import com.baomidou.mybatisplus.core.metadata.IPage;
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
import com.baomidou.mybatisplus.extension.service.impl.ServiceImpl;
import com.ksd.orm.pug.mapper.ProductMapper;
import com.ksd.pug.commons.utils.fn.asserts.Vsserts;
import com.ksd.pug.pojo.Product;
import com.ksd.pug.vo.ProductVo;
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;

/**
 * @author 飞哥
 * @SysLoginUseritle: 学相伴出品
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2022/1/2 12:42
 */
@Service
@Slf4j
public class ProductServiceImpl extends ServiceImpl<ProductMapper, Product> implements IProductService {

    /**
     * 查询产品列表信息并分页
```

```
26          *
27          * @return
28          */
29         @Override
30         public IPage<Product> findProductPage(ProductVo
productVo) {
31             // 1: 设置分页
32             Page<Product> page = new Page<>
(productVo.getPageNo(), productVo.getPageSize());
33             // 2: 设置条件
34             LambdaQueryWrapper<Product> lambdaQueryWrapper
= new LambdaQueryWrapper<>();
35             // 3: 查询过滤一些字段，为啥要这样做呢？因为在开放中很
多时候，我们一些表存在大字段列，而这些返回的过程非常非常查询性
能。
36             lambdaQueryWrapper.select(Product.class,column
-> !column.getColumn().equals("description"));
37             // 4: 把未删除的查询出来，那些删除的数据可以做一个回收
站去回收即可
38             lambdaQueryWrapper.eq(Product::getIsdelete,
0);
39             // 5:根据名字进行模糊匹配
40
 lambdaQueryWrapper.like(Vsserts.isNotEmpty(productVo.
getKeyword()), Product::getName,
productVo.getKeyword());
41             // 6: 设置排序 根据产品排降序，代表，最新的放在最前
42
 lambdaQueryWrapper.orderByDesc(Product::getCreateTime
);
43             // 7: 最后返回
44             return this.page(page, lambdaQueryWrapper);
45         }
46
```

```
47
48      /**
49       * 保存和修改产品
50       *
51       * @param product
52       */
53      @Override
54      public Product saveUpdateProduct(Product product)
    {
55          return this.saveOrUpdate(product) ? product :
    null;
56      }
57
58
59      /**
60       * 根据产品id删除产品
61       *
62       * @param id
63       * @return
64       */
65      @Override
66      public boolean delProduct(Long id) {
67          return this.removeById(id);
68      }
69
70      /**
71       * 根据产品id查询产品
72       *
73       * @param id
74       * @return
75       */
76      @Override
77      public Product getProduct(Long id) {
78          return this.getById(id);
```

```
    }
}

```

Controller

```
package com.ksd.pug.controller.product;

import com.baomidou.mybatisplus.core.metadata.IPage;
import com.ksd.pug.controller.common.BaseController;
import com.ksd.pug.pojo.Product;
import com.ksd.pug.service.product.IProductService;
import com.ksd.pug.vo.ProductVo;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.web.bind.annotation.*;

/**
 * @author 飞哥
 * @Title: 学相伴出品
 * @Description: 飞哥B站地址：https://space.bilibili.com/490711252
 * 记得关注和三连哦！
 * @Description: 我们有一个学习网站：https://www.kuangstudy.com
 * @date 2022/1/15$ 23:05$
 */
@RestController
@RequiredArgsConstructor
@Slf4j
public class ProductController extends BaseController{

    private final IProductService productService;
```

```
26
27
28      /**
29       * 查询产品列表信息并分页
30       *
31       * @return
32       */
33      @GetMapping("/product/list")
34      public IPage<Product> findProductPage(@RequestBody
    ProductVo productVo) {
35          return
    productService.findProductPage(productVo);
36      }
37
38      /**
39       * 保存和修改产品
40       *
41       * @param product
42       */
43      @PostMapping("/product/saveupdate")
44      public Product saveUpdateProduct(@RequestBody
    Product product) {
45          return
    productService.saveUpdateProduct(product);
46      }
47
48
49      /**
50       * 根据产品id删除产品
51       *
52       * @param id
53       * @return
54       */
55      @PostMapping("/product/del/{id}")
```

```
    public boolean delProduct(@PathVariable("id") Long id) {
        return productService.delProduct(id);
    }

    /**
     * 根据产品id查询产品
     *
     * @param id
     * @return
     */
    @GetMapping("/product/get/{id}")
    public Product getProduct(@PathVariable("id") Long id) {
        return productService.getProduct(id);
    }


}

```

Postman测试脚本

```
{
    "info": {
        "_postman_id": "998487a0-b230-47c7-9840-2e8fa2dd0c74",
        "name": "旅游项目实战-接口集合",
        "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
    },
    "item": [
        {
```

```
            "name": "用户模块",
            "item": [
                {
                    "name": "1、查询用户明细",
                    "request": {
                        "method": "GET",
                        "header": [
                            {
                                "key": "Authorization",
                                "value": "Bearer xxxx11111",
                                "type": "default"
                            }
                        ],
                        "url": {
                            "raw": "http://120.77.34.190:8081/admin/v1/user/get/1",
                            "protocol": "http",
                            "host": [
                                "120",
                                "77",
                                "34",
                                "190"
                            ],
                            "port": "8081",
                            "path": [
                                "admin",
                                "v1",
                                "user",
                                "get",
                                "1"
                            ]
                        }
```

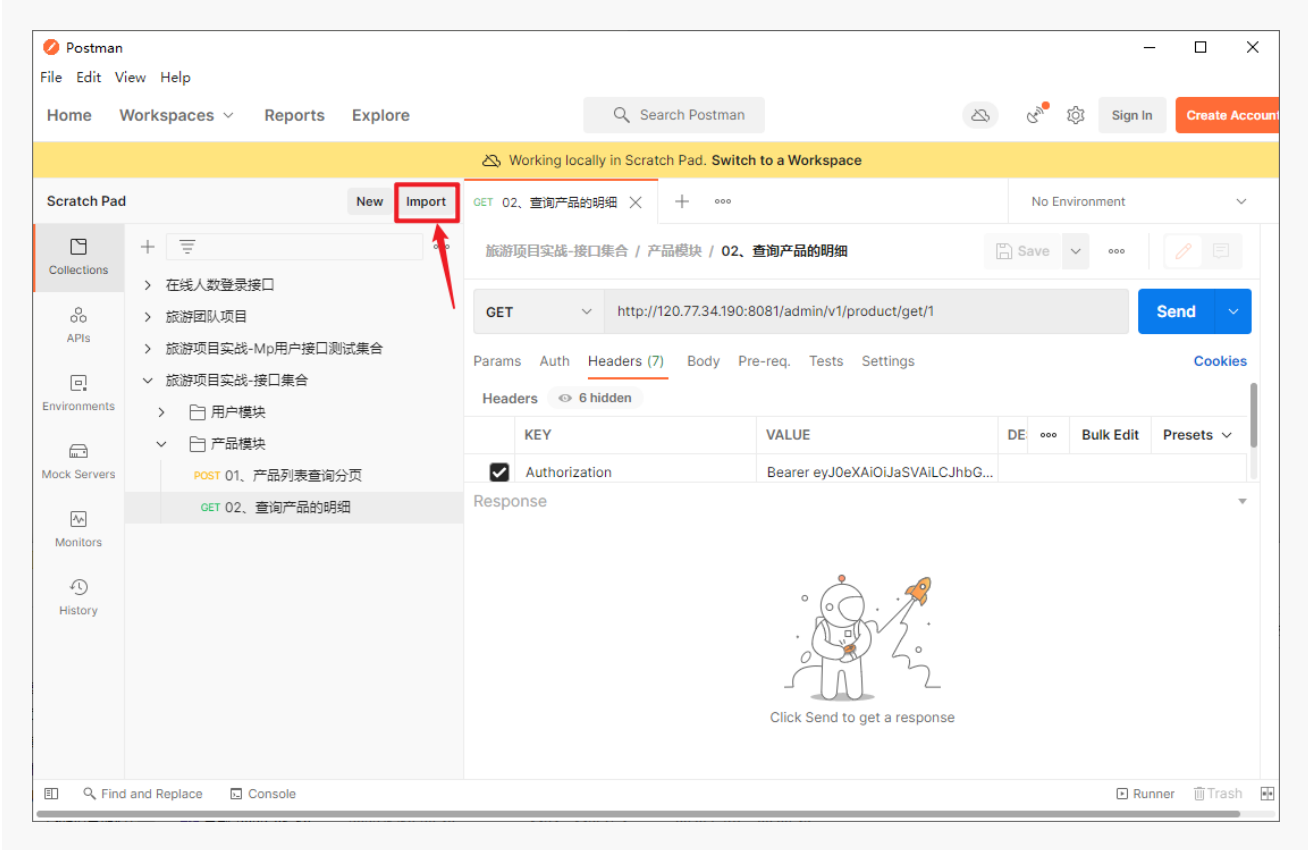
```
                    },
                    "response": []
                },
                {
                    "name": "2、登录创建token",
                    "request": {
                        "method": "POST",
                        "header": [],
                        "body": {
                            "mode": "raw",
                            "raw": "{\"username\":\"yykk\",\"password\":\"123456\"}",
                            "options": {
                                "raw": {
                                    "language": "json"
                                }
                            }
                        },
                        "url": {
                            "raw": "http://120.77.34.190:8081/auth/login/pwd",
                            "protocol": "http",
                            "host": [
                                "120",
                                "77",
                                "34",
                                "190"
                            ],
                            "port": "8081",
                            "path": [
                                "auth",
                                "login",
                                "pwd"
```

```
                            ]
                        }
                    },
                    "response": []
                }
            ]
        },
        {
            "name": "产品模块",
            "item": [
                {
                    "name": "01、产品列表查询分页",
                    "request": {
                        "method": "POST",
                        "header": [
                            {
                                "key":
"Authorization",
                                "value": "Bearer
eyJ0eXAiOiJaSVAiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxIiwi
aXNzIjoieXlrayIsImV4cCI6MTY0MjY5MDIzMCwiaWF0IjoxNjQyN
jg4NDMwfQ.5oiELJ1RgiBqN5LCnsr-E-
5pZnEN6FTLmSqwr7sT854",
                                "type": "default"
                            }
                        ],
                        "body": {
                            "mode": "raw",
                            "raw": "{\r\n
\"pageNo\":\"1\",\r\n    \"pageSize\":\"10\",\r\n
\"keyword\":\"\"\r\n}",
                            "options": {
                                "raw": {
```

```
                                "language":
"json"
                            }
                        }
                    },
                    "url": {
                        "raw":
"http://120.77.34.190:8081/admin/v1/product/list",
                        "protocol": "http",
                        "host": [
                            "120",
                            "77",
                            "34",
                            "190"
                        ],
                        "port": "8081",
                        "path": [
                            "admin",
                            "v1",
                            "product",
                            "list"
                        ]
                    }
                },
                "response": []
            },
            {
                "name": "02、查询产品的明细",
                "request": {
                    "method": "GET",
                    "header": [
                        {
                            "key":
"Authorization",
```

```
                                "value": "Bearer
eyJ0eXAiOiJaSVAiLCJhbGciOiJIUzI1NiJ9.eyJzdWIiOiIxIiwi
aXNzIjoieXlrayIsImV4cCI6MTY0MjY5MDIzMCwiaWF0IjoxNjQyN
jg4NDMwfQ.5oiELJ1RgiBqN5LCnsr-E-
5pZnEN6FTLmSqwr7sT854",
                                "type": "default"
                            }
                        ],
                        "url": {
                            "raw":
"http://120.77.34.190:8081/admin/v1/product/get/1",
                            "protocol": "http",
                            "host": [
                                "120",
                                "77",
                                "34",
                                "190"
                            ],
                            "port": "8081",
                            "path": [
                                "admin",
                                "v1",
                                "product",
                                "get",
                                "1"
                            ]
                        }
                    },
                    "response": []
                }
            ]
        }
    ]
}
```

postman导入



08、异步请求封装**request.js**和调用接口

request.js

```
// 1: 导入异步处理
import axios from "axios";


// 2: 创建axios异步实列对象
const request = axios.create({
```

```
    //`baseURL` 将自动加在 `url` 前面，除非 `url` 是一个绝对 URL。
    baseURL: "/admin/v1/",
    // 请求服务器最大时长，如果请求超过这个时间直接终止
    timeout: 10000
    // 设置请求头信息
    //headers: {'X-Custom-Header': 'foobar'}  // 不建议：因为太死了
});

// 添加请求拦截器
// 在执行axios.get/post 在来进行处理加工
request.interceptors.request.use(function (config) {
    // 获取toekn
    var token = sessionStorage.getItem("token");

    if (token) {
        config.headers['Authorization'] = 'Bearer ' + token // 让每个请求携带自定义token 请根据实际情况自行修改
    }

    // get请求映射params参数
    if (config.method === 'get' && config.params) {
        let url = config.url + '?' + tansParams(config.params);
        url = url.slice(0, -1);
        config.params = {};
        config.url = url;
    }

    // 这里处理post逻辑
    if (config.method === 'post' || config.method === 'put') {
        if (!config.data) {
```

```
            config.data = {};
        }
    }

    // 在发送请求之前做些什么
    return config;
}, function (error) {
    // 对请求错误做些什么
    return Promise.reject(error);
});

// 添加响应拦截器
request.interceptors.response.use(function (response)
{
    if(response.data.status === 500){
        console.log("1")
    }else if(response.data.status === 600){
        console.log("2")
    }else {
        // 对响应数据做点什么
        return response.data;
    }
}, function (error) {
    // 对响应错误做点什么
    return Promise.reject(error);
});


function tansParams(params) {
    let result = ''
    for (const propName of Object.keys(params)) {
        const value = params[propName];
        var part = encodeURIComponent(propName) + "=";
```

```
68          if (value !== null && typeof (value) !==
    "undefined") {
69              if (typeof value === 'object') {
70                  for (const key of Object.keys(value))
    {
71                      if (value[key] !== null && typeof
    (value[key]) !== 'undefined') {
72                          let params = propName + '[' +
    key + ']';
73                          var subPart =
    encodeURIComponent(params) + "=";
74                          result += subPart +
    encodeURIComponent(value[key]) + "&";
75                      }
76                  }
77
78              } else {
79                  result += part +
    encodeURIComponent(value) + "&";
80              }
81          }
82      }
83      return result
84  }
85
86
87  // 3:  导出
88  export default request;
```

UserService.js

```
import request from "@/utils/request";
export default {
    // 1: 查询产品列表
    loadProduct() {
        return request.post("/product/list");
    },

    // 2: 根据产品id查询产品明细
    getProduct(pid) {
        return request.get(`/product/get/${pid}`);
    }
}
```

Product/index.vue

```
<template>
  <h1>我是产品列表</h1>

  <el-table :data="productList" style="width: 100%">
    <el-table-column prop="id" label="主键" width="80"
/>
    <el-table-column prop="name" label="标题"  />
    <el-table-column prop="createTime" label="创建时间"
width="180"/>
  </el-table>

  <table>
    <thead>
    <tr>
      <th>id</th>
      <th>标题</th>
      <th>价格</th>
      <th>操作</th>
    </tr>
```

```
    </thead>
    <tbody>
    <tr v-for="(product,index) in productList" :key="product.id">
      <td>{{ product.id }}</td>
      <td>{{ product.name }}</td>
      <td><a href="" @click.prevent="getProduct(index)">查询产品明细</a></td>
    </tr>
    </tbody>
  </table>
</template>

<script>
import productService from '@/services/ProductService'

export default {
  name: "index.vue",
  data() {
    return {
      productList: [],
      keyword: "",
      total: 0,
      pageNo: 1,
      pageSize: 15,
      pages: 0
    }
  },

  created() {
    // 初始化调用，去执行异步查询产品信息
    this.loadProduct();
  },

```

```
  methods: {
    // 产品查询
    async loadProduct() {
      const res = await  productService.loadProduct();
      if (res.status == 200) {
        const {total, pages, current, size, records} = res.data;
        this.productList = records;
        this.total = total;
        this.pageNo = current;
        this.pageSize = size;
        this.pages = pages;
      }
    },

    async getProduct(index) {
      const pid = this.productList[index].id;
      const res = await productService.getProduct(pid);
      console.log(res);
    }
  }
}
</script>

<style scoped>

</style>
```

08、整合Element-Plus

官网：https://element-plus.gitee.io/zh-CN/guide/design.html

安装

```
# 选择一个你喜欢的包管理器

# NPM
$ npm install element-plus --save

# Yarn
$ yarn add element-plus

# pnpm
$ pnpm install element-plus
```