Day02周

1.本周任务安排

1. springboot前后端分离完成锋迷商城项目
2. swagger接口文档、登录验证token等等
3. vue element ui 页面

2.《锋迷商城》业务流程设计--接口规范

在企业项目开发中，当完成项目的需求分析、功能分析、数据库分析和设计之后，项目组就会按照项目中的功能进行开发任务的分配。

2.1 前后端分离与单体架构流程实现区别

- 单体架构：页面和控制器之间可以进行 跳转，同步请求控制器，流程控制由控制器控制完成。
- 前后端分离架构：前端和后端分离开发和部署，前端只能通过异步向后端发送请求，后端只负责接收请求及参数、处理请求、返回处理结果，但是后端并不负责流程控制，流程控制由前端完成。

2.2 接口介绍

2.2.1 接口概念

控制器中可以接收用户请求的某个方法

应用程序编程接口，简称API ,就是软件系统不同组成部分链接的约定

2.2.2 接口规范

> 作为一个后端开发者，我们不仅要完成接口程序的开发，还需要编写接口的说明文档----接口规范。

接口规范示例

参考：《锋迷商城》后端接口说明

2.3 swagger

> 前后端分离开发，后端需要编写接口说明文档，会耗费比较多的时间
>
> swagger是一个用于生成服务器接口的规范性文档、并且能够对接口进行测试的工具

2.3.1 作用

- 生成接口说明文档
- 对接口进行 测试

2.3.2 Swagger整合

1. 添加依赖

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>3.0.0</version>
</dependency>

<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>3.0.0</version>
</dependency>

<dependency>
    <groupId>com.github.xiaoymin</groupId>
    <artifactId>swagger-bootstrap-ui</artifactId>
    <version>1.9.6</version>
</dependency>
```

2. 在项目中添加swagger配置（java配置）

```
package org.qf.config;

import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.PathSelectors;
import
springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import
springfox.documentation.swagger2.annotations.EnableSwagger2;

@Configuration
@EnableSwagger2
public class SwaggerConfig {

    /**
     * swagger帮助我们生成接口文档
     * 1. 配置生成的文档信息
     * 2. 配置生成规则
     * @return
     */
    @Bean
    public Docket getDocket(){
        //创建封面信息对象
        ApiInfoBuilder apiInfoBuilder=new ApiInfoBuilder();
        apiInfoBuilder.title("《锋迷商城》后端接口说明")
                .description("此文档详细说明了锋迷商城项目后端接口
规范....")
                .version("v 2.0.1")
                .contact(new Contact("建
哥","www.liujian.com","2414561093@qq.com"));
        ApiInfo apiInfo=apiInfoBuilder.build();
        Docket docket=new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo)  //指定生成的文档中的封面信息：
文档标题  版本  作者
```

```
                .select()

.apis(RequestHandlerSelectors.basePackage("org.qf.web"))
                .paths(PathSelectors.any())
                .build();

        return docket;
    }

}
```

3. 测试

   启动springboot应用：访问： http://localhost:8090/doc.html

2.3.3 swagger注解说明

> swagger提供了一套注解，可以对每个接口进行详细说明

`@Api`类注解，在控制器类添加此注解，可以对控制器类进行功能说明

```
@Api(value = "提供了用户登录、注册相关的接口",tags = "用户管理")
```

`@ApiOperation` 方法注解:说明接口方法的作用

`@ApiImplicitParams`和`@ApiImplicitParam`方法注解，说明接口方法的参数

```
  @ApiOperation("用户登录接口")
    @ApiImplicitParams({
            @ApiImplicitParam(dataType = "string",name =
"username",value = "用户登录账号",required = true),
            @ApiImplicitParam(dataType = "string",name =
"password",value = "用户登录密码",required = false,defaultValue
= "123456")
    })
    @RequestMapping(value = "/login",method =
RequestMethod.GET)
    public ResultVO login(@RequestParam("username") String
name, @RequestParam("password") String pwd){
        System.out.println("------------login----------------
--------------");
        ResultVO resultVO=usersService.checkLogin(name, pwd);
        return resultVO;
    }
```

`@ApiModel`和`@ApiModelProperty`当接口参数和返回值为对象类型时，在实体类中添加注解说明

`@ApiIgnore`接口方法注解，添加此注解的方法将不会生成到接口文档中

3. 《锋迷商城》设计及实现——用户管理

3.1 后端接口开发

3.1.1 数据库设计

```
CREATE TABLE `users` (
  `user_id` int NOT NULL AUTO_INCREMENT COMMENT '主键id，用户
id',
  `username` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
COMMENT '用户名',
  `password` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
COMMENT '密码',
  `nickname` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
COMMENT '昵称',
  `realname` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
COMMENT '真实姓名',
```

```
  `user_img` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL
COMMENT '头像',
  `user_mobile` varchar(255) COLLATE utf8_unicode_ci DEFAULT
NULL COMMENT '手机号',
  `user_email` varchar(255) COLLATE utf8_unicode_ci DEFAULT
NULL COMMENT '邮箱地址',
  `user_sex` char(1) COLLATE utf8_unicode_ci DEFAULT NULL
COMMENT '性别(M 男   or  F  女)',
  `user_birth` date DEFAULT NULL COMMENT '生日',
  `user_regtime` datetime DEFAULT NULL COMMENT '注册时间、创建时
间',
  `user_modtime` datetime DEFAULT NULL COMMENT '更新时间',
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3
COLLATE=utf8_unicode_ci;
```

3.1.2 实体类

```
package org.qf.entity;

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

import java.util.Date;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Users {

    private int userId;
    private String username;
    private String password;
    private String nickname;
    private String realname;
    private String userImg;
    private String userMobile;
    private String userEmail;
    private String userSex;
    private Date userBirth;
```

```
    private Date userRegtime;
    private Date userModtime;

}
```

3.1.2 dao接口

```
package org.qf.dao;

import org.qf.entity.Users;
import org.springframework.stereotype.Component;

@Component
public interface UsersMapper {

    /**
     * 注册功能
     * @param users
     * @return
     */
    public int insert(Users users);

    /**
     * 根据用户名查询用户信息
     * @param name
     * @return
     */
    public Users query(String name);
}
```

3.1.3 mapper映射文件

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
        PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
        "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="org.qf.dao.UsersMapper">

    <insert id="insert">
```

```
        insert into
users(username,password,user_regtime,user_modtime)values(#
{username},#{password},#{userRegtime},#{userModtime})
    </insert>

    <resultMap id="userMap" type="users">
        <id column="user_id" property="userId"></id>
        <result column="username" property="username">
</result>
        <result column="password" property="password">
</result>
        <result column="nickname" property="nickname">
</result>
        <result column="realname" property="realname">
</result>
        <result column="user_img" property="userImg"></result>
        <result column="user_mobile" property="userMobile">
</result>
        <result column="user_email" property="userEmail">
</result>
        <result column="user_sex" property="userSex"></result>
        <result column="user_birth" property="userBirth">
</result>
        <result column="user_regtime" property="userRegtime">
</result>
        <result column="user_modtime" property="userModtime">
</result>

    </resultMap>
    <select id="query" resultMap="userMap">
        select * from users where username=#{username}
    </select>

</mapper>
```

3.1.4 业务层接口

```
package org.qf.service;

import org.qf.utils.ResultVO;

public interface UsersService {

    public ResultVO userResgit(String name,String pwd);


    public ResultVO checkLogin(String name, String pwd);

}
```

3.1.5 业务层实现类

```
package org.qf.service.impl;

import org.qf.dao.UsersMapper;
import org.qf.entity.Users;
import org.qf.service.UsersService;
import org.qf.utils.MD5Utils;
import org.qf.utils.ResultVO;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.Date;

@Service
public class UsersServiceImpl implements UsersService {

    @Autowired
    private UsersMapper usersMapper;

    //注册
    @Override
    public ResultVO userResgit(String name, String pwd) {
        //1.根据用户查询，这个用户是否已经注册
        Users user = usersMapper.query(name);
        //2. 如果没有注册则进行保存操作
        if(user==null){
            //密码加密  md5
```

```
            String md5Pwd=MD5Utils.md5(pwd);
            user=new Users();
            user.setUsername(name);
            user.setPassword(md5Pwd);
            user.setUserRegtime(new Date());
            user.setUserModtime(new Date());
            int i = usersMapper.insert(user);
            if(i>0){
                return new ResultVO(10000,"注册成功！",null);
            }else{
                return new ResultVO(10002,"注册失败！",null);
            }
        }else{
            return new ResultVO(10001,"用户名已经注册！",null);
        }
    }

    //登录
    @Override
    public ResultVO checkLogin(String name, String pwd) {
        Users user = usersMapper.query(name);
        if(user==null){
            return new ResultVO(10001,"登录失败，用户名不存
在！",null);
        }else{
            String md5Pwd=MD5Utils.md5(pwd);
            if(md5Pwd.equals(user.getPassword())){
                return new ResultVO(10000,"登录成功!",user);
            }else{
                return new ResultVO(10001,"登录失败，密码错
误！",null);
            }
        }
    }
}
```

3.1.6 控制器

```
package org.qf.web;
```

```
import org.qf.service.UsersService;
import org.qf.utils.ResultVO;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import javax.annotation.Resource;


@RestController
@RequestMapping("/user")
public class UsersController {

    @Resource
    private UsersService usersService;

    //登录
    @RequestMapping(value = "/login",method =
RequestMethod.GET)
    public ResultVO login(@RequestParam("username") String
name, @RequestParam("password") String pwd){
        System.out.println("------------login----------------
--------------");
        ResultVO resultVO=usersService.checkLogin(name, pwd);
        return resultVO;
    }

    //注册
    @RequestMapping("/regist")
    public ResultVO regist(String username,String password){
        ResultVO resultVO=usersService.userResgit(username,
password);
        return resultVO;
    }

}
```

3.1.7 ResultVO工具类

```java
package org.qf.utils;

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

/**
 * 封装接口返回给前端的数据
 */
@Data
@AllArgsConstructor
@NoArgsConstructor
public class ResultVO {

    //响应给前端的状态码
    private int code;

    //响应给前端的提示信息
    private String msg;

    //响应给前端的数据
    private Object data;

}
```

3.2 前端跨域访问

3.2.1 跨域访问概念

- 什么是跨域访问?

> ajax跨域访问时用户访问A网站时所产生的对B网站的跨域请求均提交到 A网站的指定页面

3.2.2 如何解决跨域访问?

- 前端使用JSONP配置
- 后端使用@CrossOrign-----就是设置 响应头允许跨域