**代码清单**

城院生态圈

——基于微信小程序的城院动植物交流论坛



项目名称： 城院生态圈

专业班级： 软件工程1903

小组成员：黄依豪，梁晓勇，李东泽

指导教师： 杨枨老师

软件代码清单说明

说明：

《软件代码清单》 描述了软件在开发结束后对所涉及的代码进行清单罗列以及检查相关规范。该文档可能对《软件测试报告》和《软件代码走查说明》以软件维护过程的阶段资料有所补充。

**版本记录**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 编号 | 修订日期 | 版本/状态 | 修订人 | 备注 |
| 01 | 2021-12-6 | 0.1 | 梁晓勇 | 报告的初始版本制作 |
| 02 | 2021-12-7 | 0.2 | 黄依豪 | 增加后端内容 |
| 03 | 2021-12-14 | 0.3 | 黄依豪 | 增加前后端内容 |
| 04 | 2021-12-26 | 0.4 | 黄依豪 | 增加前后端内容 |

目录

[代码清单 1](#_Toc367)

[软件代码清单说明 2](#_Toc26169)

[1引言 4](#_Toc12842)

[1.1标识 4](#_Toc13343)

[1.2系统概述 4](#_Toc16089)

[1.3文档概述 5](#_Toc26384)

[1.4基线 5](#_Toc26282)

[2引用文件 5](#_Toc19149)

[3前端代码 6](#_Toc21119)

[3.1主页界面： 6](#_Toc18511)

[3.2消息通知界面 9](#_Toc1178)

[3.3发帖界面 9](#_Toc26930)

[3.4我的界面 19](#_Toc22255)

[3.5动植物科普界面 27](#_Toc17242)

[3.6活动发帖界面 29](#_Toc1185)

[3.7活动详情界面 31](#_Toc30441)

[3.8活动界面 32](#_Toc24748)

[3.9我参与的活动 34](#_Toc30912)

[4后端代码 35](#_Toc10976)

[4.1DTO类 35](#_Toc30235)

[4.2filter 42](#_Toc15753)

[4.3service 46](#_Toc16912)

[4.4utils 49](#_Toc24953)

[4.5controller 50](#_Toc1011)

[4.4cofiguration 75](#_Toc17238)

[5注解 90](#_Toc21998)

[附录 92](#_Toc7250)

# 1引言

## 1.1标识

|  |  |  |
| --- | --- | --- |
| 文件状态：  [ ] 草稿  [ ] 正式发布  [√] 正在修改 | 文件标识： | SE2021-G005-代码清单 |
| 当前版本： | 0.4 |
| 作者： | 黄依豪，李东泽，梁晓勇 |
| 完成日期： | 2021-12-27 |

标题：《代码清单说明》

说明：本文档支持Office、WPS等word文档阅读软件

## 1.2系统概述

本文档用于说明对于“城院生态圈”的软件测试计划。

本项目是通过小程序实现用户的交流论坛——“城院生态圈”，以爱护动植物、科普相关知识为目的，作为一个论坛平台供校内学生们交流动植物。力争做到界面清新、充实功能、无广告、无收费，操作简单易上手，吸引用户加入。本项目实现的小程序最终会在微信小程序的搜索栏下找到并运行。

开发历史：

2021/9/15-2021/9/23 课题选择

2021/9/23-2021/10/10 项目计划制定

2021/10/10-2021/10/17 可行性分析报告制定

2021/10/18-2021/10/24 需求分析报告制定

2021/11/3-2021/11/7 系统设计说明报告制定

2021/11/8-2021/11/13 软件测试计划制定

2021/11/8-2021/11/13 用户手册制定

2021/11/8-2021/11/14 详细计划制定

相关文档：

SE2021-G005-项目想法1.0.docx

SE2021-G005-项目计划表0.1.docx

SE2021-G005-可行性分析报告0.1.docx

SE2021-G005-需求分析报告1.0.docx

SE2021-G005-系统设计说明报告1.0.docx

SE2021-G005-用户手册0.1.docx

投资方、开发方：

G005小组

支持机构：

浙大城市学院

## 1.3文档概述

软件项目的测试计划是描述测试目的、范围、[方法](https://baike.baidu.com/item/%E6%96%B9%E6%B3%95/2444" \t "_blank)和[软件测试](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E6%B5%8B%E8%AF%95/327953" \t "_blank)的重点等的文档。对于验证[软件产品](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E4%BA%A7%E5%93%81/6800028" \t "_blank)的可接受程度编写测试计划文档是一种有用的方式。

本系统设计说明用于指导开发“城院生态圈”小程序项目顺利进行并最终通过评审的项目产品。本系统设计说明面向项目组全体成员。

## 1.4基线

基线是软件生存期各个开发阶段的工作成果。

重要的检查点是里程碑，重要的需要客户确认的里程碑是基线。

主要基线：

SE2021-G005-项目想法1.0.docx

SE2021-G005-项目计划表0.1.docx

SE2021-G005-可行性分析报告1.1.docx

SE2021-G005-软件需求说明(SRS)1.1.docx

SE2021-G005-系统设计说明1.0.docx

# 2引用文件

本章应列出本文档引用的所有文档的编号、标题、修订版本和发行日期，也应标识不能通过正常的供货渠道获得的所有文档的来源。

[1]GB+T-8567-2006计算机软件文档编制规范 09 - 系统(子系统)设计(结构设计)说明(SSDD)

[2]SE2021-G005-项目想法1.0.docx

[3]SE2021-G005-项目计划表0.1.docx

[4]SE2021-G005-可行性分析报告0.1.docx

[5]SE2021-G005-需求分析报告1.0.docx

[6]SE2021-G005-系统设计说明报告1.0.docx

[7]SE2021-G005-用户手册1.0.docx

[8]张海藩，牟永敏.软件工程导论(第6版)[M]北京：清华大学出版社

[9]梁勇. JAVA语言程序设计(第10版)[M]北京：机械工业出版社

[10]王珊，萨师煊. 数据库系统概论(第5版)[M]北京：高等教育出版社

[11]覃征，李旭，王卫红. 软件体系结构(第4版)[M]北京：清华大学出版社

[12]李宁,李战怀.基于黑盒测试的软件测试策略研究与实践[J].计算机应用研究,2009,26(03):923-926.

## 3前端代码

### 3.1主页界面：

* 实现热帖的显示和点击跳转

<view class="white"></view>

    <view id="hots">

        <view id="hot\_left" bindtap="toTenHotPosts">

            <text id="hot\_text">十大热帖</text>

            <image id="hot\_pho" src="../../icon/hot2.png" />

        </view>

        <view id="hot\_right" >

            <view id="hr\_text" class="hr\_carousel" wx:for="{{postList}}"

                wx:key="post\_ID"

                wx:for-item="item1"

                data-index="{{index}}"

                bindtap="toPost">

                    0{{index+1}}

                    <text>{{item1.post\_Content}}</text>

            </view>

        </view>

    </view>

* 实现跳转到动物城友会界面和动植物科普界面

<view id="other\_acti">

        <button id="ani\_f" class="btn" type="primary" bindtap="tobiofri">

            动物城友会

            </button>

        <button id="ani\_info" class="btn" type="warn" disabled="{{disabled}}" bindtap="tobioinfo">

            动植物科普

        </button>

    </view>

* 实现瀑布流显示贴子

<view class="postWaterfall">

        <view class="post"

        wx:for="{{postList}}"

        wx:key="post\_ID"

        wx:for-item="item1"

        data-index="{{index}}"

        bindtap="toPost"

        >

wx-for区块化显示每个帖子的的内容

            <view class="postLine1">

第一行展示头像、名字、时间、是否置顶和是否精华

                <view class="headIcon" >

                    <image mode="aspectFill" src="{{item1.user\_head}}" />

                </view>

                <view class="NameAndPostTime">

                    <view class="postManName">{{item1.user\_ID}}</view>

                    <view class="postTime">发布于{{item1.post\_Date}}</view>

                </view>

                <view class="good" wx:if="{{item1.post\_isTop}}">

                    <view >顶置</view></view>

                <view class="good" wx:elif="{{item1.post\_isEssential}}">

                    <view >精华</view></view>

            </view>

第二行展示文字内容和图片内容

            <view class="postLine2">

                <view class="postContent">

                    <text >{{item1.post\_Content}}</text>

                </view>

                <view class="imageWithContent"

                wx:for="{{item1.post\_Image}}"

                wx:for-item="item2"

                >

                    <image src="{{item2.url}}" />

                </view>

            </view>

第三行展示点赞和点赞数，评论和评论数

            <view class="postLine3" >

                <view class="postOperate"

                catchtap="changeLike"

                data-index="{{index}}">

                    <image src="{{item1.post\_isLiked?'../../icon/dianzan.png':'../../icon/dianzan\_.png'}}" />

                    <view>{{item1.post\_likeNum?item1.post\_likeNum:''}}</view>

                </view>

                <view class="postOperate">

                    <image src="{{item1.post\_isStared?'../../icon/pinglun.png':'../../icon/pinglun\_.png'}}" />

                    <view>{{item1.post\_reportNum?item1.post\_reportNum:''}}</view>

                </view>

                <view class="postOperate">

                    <image src="../../icon/dashang2.png" />

                    <view>打赏</view>

                </view>

            </view>

        </view>

    </view>

</view>

### 3.2消息通知界面

实现通知和消息按钮

<view class="contain">

    <view >

        <view class="msgBlock">

            <view id="name\_notice">通知</view>

            <view id="name\_chat">私信</view>

        </view>

    </view>

    <view class="noticeBar">

        <view id="notice\_word">

            通知列

        </view>

        <view id="readAll">

            全部已读

        </view>

    </view>

    <view id="notice\_detail">

    </view>

</view>

### 3.3发帖界面

发帖主题选择

<view id="themeSelc">

        <view class="top">

            <view class="pic1">

                <image id="context1" src="../../icon/context.png" />

            </view>

            <view class="select" bindtap="showSelect"  id="{{index}}">{{postTheme.default}}

            </view>

            <view class="top2">

                <view class="option" hidden="{{postTheme.hidden}}" bindtap="SelectVal" >

                    <view wx:for="{{postTheme.txt}}" wx:for-item="test\_item" data-index="{{index}}"  wx:for-index="index">

                {{test\_item}}

                    </view>

                </view>

            </view>

            <view class="pic2">

                <image id="right1" src="{{postTheme.hidden?'../../icon/right2.png':'../../icon/down.png'}}" />

            </view>

    </view>

</view>

发帖正文内容

<view id="bodyText">

        <view id="textIn">

            <textarea placeholder="此处输入正文" name="" id="" cols="40" rows="20" bindinput="getPostContent"></textarea>

        </view>

        <view id="addPho">

            <image src="../../icon/addPho.png" />

        </view>

    </view>

发帖规则提示

<view id="textPrompt">

        <text >1、帖子需遵守法律法规以及社区条款

        </text>

        <text >2、目前只支持jpg,png格式图片上传

        </text>

        <text >3、请勿滥用帖子功能，违者将追究法律责任</text>

    </view>

发布按钮

<view id="lineButtom">

        <button class="btn" bindtap="release">发布</button>

    </view>

const app = getApp()

const api = app.globalData.api

Page({

    /\*\*

     \* 页面的初始数据

     \*/

    data: {

        textDown: "",

        postTheme: {

            isHidden: true,

            default: "选择主题",

            select: -1,

            txt: ["吐槽", "晒动物", "晒植物", "求助"]

        },

        images: [],

        imagesBase: [],

        tmpImages: [],

        maxImg: 3

    },

    /\*\*

     \* 展示或隐藏选项

     \*/

    showSelect: function (e) {

        //获取当前数据

        var that = this

        var data = this.data.postTheme;

        //变更isHidden属性

        data["isHidden"] = !data.isHidden

        //执行变更

        this.setData({

            postTheme: data,

        })

        if (data["isHidden"]){

            app.showToNone(that, 'selectAni');

            app.rotate(that, 'rotateAni', 0);

        }

        else{

            app.noneToShow(that, 'selectAni');

            app.rotate(that, 'rotateAni', 45);

            app.darkin(that, 'selectDarkAni', 0.9, '#fff', '#92c997');

        }

    },

    /\*\*

     \* 设置当前选项的值

     \*/

    SelectVal: function (e) {

        var that = this

        // 获取到点击的列表下标。

        var index = e.target.dataset.index;

        var data = this.data.postTheme

        //获取选中的选项的值

        var test\_name = data.txt[index];

        //设置区域默认和隐藏

        data["default"] = test\_name;

        data["select"] = index;

        data["isHidden"] = false;

        app.noneToShow(that, 'selectAni');

        app.rotate(that, 'rotateAni', 45);

        this.setData({

            postTheme: data,

            theme: test\_name

        })

    },

    /\* 获取帖子内容 \*/

    getPostContent: function (e) {

        this.setData({

            postContent: e.detail.value

        })

    },

    /\* 发布帖子 \*/

    release: function () {

        this.imageToBase();

        if (!this.data.postContent || this.data.postTheme == -1) {

            wx.showToast({

                title: '请输入完整!',

                icon: 'error',  // 图标类型，默认success

                duration: 1500  // 提示窗停留时间，默认1500ms

            })

        }

        else if (!getApp().globalData.userInfo) {

            wx.showToast({

                title: '请先登录!', // 标题

                icon: 'error',  // 图标类型，默认success

                duration: 1500  // 提示窗停留时间，默认1500ms

            })

        }

        else {

            var that = this;

            app.darkin(that, 'ani', 0.7, 'rgb(67, 196, 88)', 'rgb(36, 100, 47)');

            api.showLoading('上传中...')

            for (let i = 0; i < that.data.images.length; i++) {

                wx.getFileSystemManager().readFile({

                    filePath: this.data.images[i], //选择图片返回的相对路径

                    encoding: "base64", //这个是很重要的

                    success: res => {

                        //返回base64格式

                        that.data.imagesBase.push(res.data);

                        this.setData({})

                        if (that.data.imagesBase.length == that.data.images.length) {

                            console.log(this.data.imagesBase[0])

                            that.upCount();

                        }

                    }

                })

            }

            if (that.data.images.length == 0) {

                that.upCount();

            }

        }

    },

    upCount: function () {

        var that = this

        wx.request({

            url: getApp().globalData.urlHome + '/postBar/releasePost',

            method: 'POST',

            header: {

                'content-type': 'application/json;charset=utf-8',

                'x-auth-token': getApp().globalData.token

            },

            data: {

                postTheme: this.data.theme,

                postContent: this.data.postContent,

                userID: getApp().globalData.openid,

                userAvatarUrl: getApp().globalData.userInfo.avatarUrl,

                'picType1': that.data.tmpImages[0] ? that.data.tmpImages[0].split('.')[1] : null,

                'picType2': that.data.tmpImages[1] ? that.data.tmpImages[1].split('.')[1] : null,

                'picType3': that.data.tmpImages[2] ? that.data.tmpImages[2].split('.')[1] : null,

                'picName1': that.data.tmpImages[0] ? that.data.tmpImages[0] : null,

                'picName2': that.data.tmpImages[1] ? that.data.tmpImages[1] : null,

                'picName3': that.data.tmpImages[2] ? that.data.tmpImages[2] : null,

                'picture1': that.data.imagesBase[0] ? that.data.imagesBase[0] : null,

                'picture2': that.data.imagesBase[1] ? that.data.imagesBase[1] : null,

                'picture3': that.data.imagesBase[2] ? that.data.imagesBase[2] : null,

            },

            success(r) {

                wx.showToast({

                    title: '发布成功！', // 标题

                    icon: 'success',  // 图标类型，默认success

                    duration: 1500  // 提示窗停留时间，默认1500ms

                })

                api.hideLoading('上传中...')

                setTimeout(function () {

                    wx.switchTab({

                        url: '../home/home',

                    })

                }, 1500)

                that.clearData()

            }

        })

    },

    chooseImage: function () {

        const that = this

        console.log(this.data.images)

        if (that.data.images.length < that.data.maxImg) {

            let images = that.data.images

            wx.chooseImage({

                count: that.data.maxImg - that.data.images.length,

                sizeType: ['original', 'compressed'],  //可选择原图或压缩后的图片

                sourceType: ['album', 'camera'], //可选择性开放访问相册、相机

                success: res => {

                    console.log(res)

                    for (let i = 0; i < res.tempFiles.length; i++) {

                        images.push(res.tempFilePaths[i]);

                        this.setData({

                            images: images

                        })

                    }

                }

            });

        }

        else {

            wx.showToast({

                title: "最多上传" + that.data.maxImg + "张照片！"

            })

        }

    },

    removeImage: function (e) {

        var that = this;

        var images = that.data.images;

        var imagesBase = that.data.imagesBase;

        // 获取要删除的第几张图片的下标

        const idx = e.currentTarget.dataset.idx

        // splice  第一个参数是下表值  第二个参数是删除的数量

        images.splice(idx, 1)

        imagesBase.splice(idx, 1)

        this.setData({

            images: images,

            imagesBase: imagesBase

        });

    },

    handleImagePreview(e) {

        const idx = e.target.dataset.idx

        const images = this.data.images

        wx.previewImage({

            current: images[idx],  //当前预览的图片

            urls: images,  //所有要预览的图片

        })

    },

    // 图片转base64

    imageToBase() {

        const that = this

        this.setData({

            tmpImages: [],

        })

        this.data.images.forEach(element => {

            that.data.tmpImages.push(element.split('/')[3])

        });

    },

    clearData: function () {

        this.setData({

            postTheme: {

                isHidden: true,

                default: "选择主题",

                select: -1,

                txt: ["吐槽", "晒动物", "晒植物", "求助"]

            },

            images: [],

            imagesBase: [],

            tmpImages: [],

            textDown: "",

        })

    },

    /\*\*

     \* 生命周期函数--监听页面加载

     \*/

    onLoad: function (options) {

    },

    /\*\*

     \* 生命周期函数--监听页面初次渲染完成

     \*/

    onReady: function () {

    },

    /\*\*

     \* 生命周期函数--监听页面显示

     \*/

    onShow: function () {

    },

    /\*\*

     \* 生命周期函数--监听页面隐藏

     \*/

    onHide: function () {

    },

    /\*\*

     \* 生命周期函数--监听页面卸载

     \*/

    onUnload: function () {

    },

    /\*\*

     \* 页面相关事件处理函数--监听用户下拉动作

     \*/

    onPullDownRefresh: function () {

    },

    /\*\*

     \* 页面上拉触底事件的处理函数

     \*/

    onReachBottom: function () {

    },

    /\*\*

     \* 用户点击右上角分享

     \*/

    onShareAppMessage: function () {

    }

})

### 3.4我的界面

个人信息概述框

    <view class="white"></view>

        <view>

            <view  wx:if="{{!hasUserInfo}}">

             <button id="userLogin" bindtap="getUserProfile"> 登陆 </button>

            </view>

        <view id="infoAmost" wx:else="{{userInfo.avatarUrl}}">

            <view id="headNameInfo">

                <view >

                    <image id="headPho" mode="aspectFill" src="{{userInfo.avatarUrl}}" />

                </view>

                <view id="nameAndRP">

                    <view id="myName">{{userInfo.nickName}}</view>

                    <view id="myRP">

                        <view id="rpPho">

                            <image  src="../../icon/heart.png"/>

                        </view>

                        <view>

                            人品值：{{myrp}}

                        </view>

                        <view>

                        </view>

                    </view>

                </view>

                <view id="infoModify">

                    <image src="../../icon/modify.png" />

                </view>

            </view>

            <view id="pushStarCommit">

                <view id="myPush">

                    <view>{{myPush}}</view>

                    <view class="pSCword">贴子</view>

                </view>

                <view>

                    <view>{{myStar}}</view>

                    <view class="pSCword">收藏夹</view>

                </view>

                <view>

                    <view>{{myCommit}}</view>

                    <view class="pSCword">最近评论</view>

                </view>

            </view>

        </view>

    </view>

*<!-- <button id="user\_login" bindtap="login" wx:else="">登陆</button> -->*

    </view>

常用工具展示，包含四个功能

<view id="usualTools">

        <view id="uTTitle">常用工具</view>

        <view id="uTInfo">

            <view class="Tool">

                <view><image  src="../../icon/waiter.png"/></view>

                <view>客服中心</view>

            </view>

            <view class="Tool">

                <view><image  src="../../icon/discovery.png"/></view>

                <view>活动扫码</view>

            </view>

            <view class="Tool">

                <view><image  src="../../icon/invite.png"/></view>

                <view>邀请校友</view>

            </view>

            <view class="Tool">

                <view><image  src="../../icon/rule.png"/></view>

                <view>平台规则</view>

            </view>

        </view>

    </view>

其他功能，包括主题设置、关于我们和意见反馈

<view id="otherTool">

        <view class="otherLine">

            <view>

                <image src="../../icon/setting.png"/>

            </view>

            <view>主题设置</view>

            <view>

                <image class="afterWord" src="../../icon/right2.png"/>

            </view>

        </view>

        <view id="aboutUs" class="otherLine">

            <view>

                <image src="../../icon/abouUs.png"/>

            </view>

            <view>关于我们</view>

            <view>

                <image class="afterWord" src="../../icon/right2.png"/>

            </view>

        </view>

        <view class="otherLine">

            <view>

                <image src="../../icon/back.png"/>

            </view>

            <view>意见反馈</view>

            <view>

                <image class="afterWord" src="../../icon/right2.png"/>

            </view>

        </view>

    </view>

const app = getApp()

const api = app.globalData.api

Page({

    /\*\*

     \* 页面的初始数据

     \*/

    data: {

        userInfo: {},

        hasUserInfo: false,

        myrp: 0,

        myPush: 0,

        myComment: 0,

        myRewardRecord: 0,

        isAdmin: false,

        // commitList:[],

    },

    onLoad() {

        if (getApp().globalData.hasUserInfo) {

            this.setData({

                hasUserInfo: true,

                userInfo: getApp().globalData.userInfo

            })

            this.onRefreshUserInfo();

        }

    },

    onShow() {

        if (getApp().globalData.hasUserInfo) {

            this.onRefreshUserInfo();

        }

    },

    onPullDownRefresh: function () {

        if (getApp().globalData.hasUserInfo) {

            this.onRefreshUserInfo();

        }

        else{

            api.showToast('请登录后刷新!', 'error')

            setTimeout(() => {

                api.hideToast()

            }, 2000)

            wx.stopPullDownRefresh()

        }

    },

    //   跳转登录功能

    getUserProfile() { this.getUserProfile2() },

    // 用户登录并获取用户信息

    async getUserProfile2() {

        // 推荐使用wx.getUserProfile获取用户信息，开发者每次通过该接口获取用户个人信息均需用户确认

        // 开发者妥善保管用户快速填写的头像昵称，避免重复弹窗

        var that = this

        await that.userLogin()

        await wx.getUserProfile({

            desc: '用于完善会员资料', // 声明获取用户个人信息后的用途，后续会展示在弹窗中，请谨慎填写

            success: (res) => {

                app.globalData.userInfo = res.userInfo;

                app.globalData.hasUserInfo = true;

                this.setData({

                    hasUserInfo: true,

                    userInfo: res.userInfo

                })

                console.log("sendUserInfo Begin");

                this.sendUserInfo(res)

            },

        })

        await that.onRefreshUserInfo()

    },

    sendUserInfo(event) {

        var that = this

        return new Promise((resolve, reject) => {

            api.postData(app.globalData.urlHome + '/login/saveUserInfo', {

                userAvatarUrl: event.userInfo.avatarUrl,

                userName: event.userInfo.nickName,

                openId: app.globalData.openid,

            }).then(() => {

                console.log("sendUserInfo ok");

                that.onRefreshUserInfo()

                resolve()

            })

                .catch((err) => {

                    console.error(err)

                    reject(err)

                })

        })

    },

    userLogin() {

        // 登录

        wx.login({

            success: function (res) {

                // 发送 res.code 到后台换取 openId, sessionKey, unionId

                return new Promise((resolve, reject) => {

                    api.getHeader(app.globalData.urlHome + '/login/getUserInfo', {

                        code: res.code,

                        grant\_type: 'authorization\_code'

                    }).then((res2) => {

                        console.log(res2);

                        app.globalData.token = res2['x-auth-token'];

                        app.globalData.openid = res2['openid'];

                        console.log('获得了用户id');

                        wx.setStorageSync('openid', app.globalData.openid)

                        resolve()

                    })

                        .catch((err) => {

                            console.error(err)

                            reject(err)

                        })

                })

            }

        })

    },

    onRefreshUserInfo() {

        var that = this

        wx.request({

            url: getApp().globalData.urlHome + '/user/loadUserInfo',

            method: 'POST',

            header: {

                'content-type': 'application/json;charset=utf-8',

                'x-auth-token': getApp().globalData.token

            },

            data: {

                openid: getApp().globalData.openid

            },

            complete(r) {

                console.log("here are the deatail")

                console.log(r)

                that.setData({

                    myrp: r.data.myrp,

                    myPush: r.data.myPush,

                    myComment: r.data.myComment,

                    myRewardRecord: r.data.myRewardRecord,

                })

            },

        })

    },

    //   跳转到手册界面

    toRulesPage: function () {

        wx.navigateTo({

            url: '/pages/userRule/userRule'

        })

    },

    // 跳转到关于界面

    toAboutPage: function () {

        wx.navigateTo({

            url: '/pages/about/about'

        })

    },

    //人品值详细记录

    toMyRPRecord: function () {

        wx.navigateTo({

            url: '/pages/myRPRecord/myRPRecord'

        })

    },

    //跳转我的发帖界面

    toMyPost: function () {

        wx.navigateTo({

            url: '/pages/myPost/myPost'

        })

    },

    //我的评论

    toMyComment: function () {

        wx.navigateTo({

            url: '/pages/myComment/myComment'

        })

    },

    //我的打赏记录

    toRewardRecord: function () {

        wx.navigateTo({

            url: '/pages/myRewardRecord/myRewardRecord'

        })

    },

    // 扫一扫功能

    scan() {

        wx.scanCode({

            success: (res) => {

                console.log("扫码结果");

                console.log(res);

            },

            fail: (res) => {

                console.log(res);

            }

        })

    }

})

### 3.5动植物科普界面

动物科普和植物科普

<view id="searchBox">

搜索框

        <view id="searchLine">

            <view id="sl\_frame">

                <view id="sl\_image" class=".iconfont .icon-sousuo">

                </view>

                <view id="sl\_text">

                    请输入想要了解的动植物

                </view>

            </view>

            <view id="sl\_cancel">

                <button>

                    取消

                </button>

            </view>

        </view>

热门内容提示

        <view id="hotSearch">

            <view id="hs\_text">

                热门搜索：

            </view>

            <view id="hs\_content">

                <view class="hsc\_name">

                    {{hs\_name1}}

                </view>

                <view class="hsc\_name">

                    {{hs\_name2}}

                </view>

            </view>

        </view>

    </view>

<!-- 分类导航 -->

    <view id="typeNavigation">

        <view id="creatureType">

            <view id="ct\_animals" class="ct\_specific">

                动物

            </view>

            <view id="ct\_plants" class="ct\_specific">

                <button bindtap="tobioinfo\_plants" type="default">

                 植物

                </button>

            </view>

        </view>

</view>

科普更新按钮

    <view id="iWantUpdate">

        <button bindtap="toupdateInterface">

            我要科普

        </button>

    </view>

### 3.6活动发帖界面

活动主题选择

<view class="top">

            <view class="pic1">

                <image id="context1" src="../../icon/context.png" />

            </view>

            <view class="select" bindtap="showSelect"  id="{{index}}">{{ACFC\_startActivity.default}}

            </view>

            <view class="top2">

                <view class="option" hidden="{{ACFC\_startActivity.hidden}}" bindtap="SelectVal" >

                    <view wx:for="{{ACFC\_startActivity.txt}}" wx:for-item="test\_item" data-index="{{index}}"  wx:for-index="index">

                {{test\_item}}

                    </view>

                </view>

            </view>

            <view class="pic2">

                <image id="right1" src="../../icon/right2.png" />

            </view>

        </view>

活动具体信息填入

<view id="bodyText">

        <view id="textIn">

            <textarea placeholder="此处讲明活动时间、地点等信息" name="" id="" cols="40" rows="20" bindinput="getActivityDescription"></textarea>

*<!-- <input type="text" placeholder="此处讲明活动时间、地点等信息"/> -->*

        </view>

        <view id="addPho">

            <image src="../../icon/addPho.png" />

        </view>

    </view>

活动规则提示

<view id="textPrompt">

        <text >1、活动需遵守法律法规以及社区条款

        </text>

        <text >2、目前只支持jpg,png格式图片上传

        </text>

        <text >3、请勿滥用活动功能，违者将追究法律责任</text>

    </view>

发布按钮

<view id="lineButtom">

        <button class="btn" bindtap="release">发布</button>

    </view>

发帖功能函数

  release(){

    console.log(this.data.name+" "+this.data.description+" "+getApp().globalData.openid)

    if(!this.data.name || !this.data.description || !this.data.type){

      wx.showModal({

        title:'请输入完整！'

      })

    }else{

      wx.request({

        url: 'http://127.0.0.1:8080/api/sql/releaseActivity',

        method:'POST',

        header:{'content-type': 'application/json;charset=utf-8',

                'x-auth-token': getApp().globalData.token

      },

        data:{

          ActivityName:this.data.name,

          ActivityType:this.data.type,

          ActivityDescription:this.data.description,

          userId:getApp().globalData.openid,

        },

        success(*r*){

          wx.showToast({

            title: '操作成功！', *// 标题*

            icon: 'success',  *// 图标类型，默认success*

            duration: 1500  *// 提示窗停留时间，默认1500ms*

          })

          setTimeout(function(){

            wx.navigateBack({

              delta: 1,

            })

          },1500)

        }

      })

    }

### 3.7活动详情界面

<view class="contain">

    <view id="activityLine1">

        <view id="roughActivity">

            <view id="activityTitle">

*<!-- 　　这里复制了两个空白字符 -->*

                {{activityTitle}}

            </view>

            <view id="activityTheme">

                {{activityType}}

            </view>

        </view>

        <view id="relevantActivityInfo">

            <view id="location" >

                <view class=".iconfont .icon-dingwei">

                    {{activityLocation}}

                </view>

            </view>

            <view id="startTime">

                <view class=".iconfont .icon-miaobiao">

                    {{activityStartDate}}

                </view>

            </view>

        </view>

    </view>

    <view id="activityLine2">

        <view id="menberSituation">

            <view >

                参与人数：({{activityParticipantsNum}}/{{activityTotolParticipantsNum}})

            </view>

        </view>

    </view>

    <view id="activityLine3">

        <view >活动详情：</view>

        <view id="activityContent">

            　　{{activityContent}}

        </view>

    </view>

    <button id="participate">

        立即参与

    </button>

</view>

### 3.8活动界面

<view class="contain">

    <view id="actNavigation">

        <button id="organizeActivity" class="btn" bindtap="toStartActivity">发起活动</button>

        <button id="checkActivities" class="btn" bindtap="toMyActivity" >我加入的活动</button>

    </view>

    <view id="searchLine">

            <view id="sl\_frame">

                <view id="sl\_image"  class=".iconfont .icon-sousuo">

                </view>

                <input id="sl\_text" type="text" value='{{inputValue}}' placeholder="请输入想要查看的活动"/>

                </view>

            <view id="sl\_cancel">

                <button bindtap='clearInputEvent'>

                    取消

                </button>

            </view>

    </view>

    <view id="activityContent">

        <view class="singleActivity"

        wx:for="{{activitiesList}}"

        wx:key="{{activityId}}"

        wx:for-item="item"

        data-index="{{index}}"

        bindtap="toActivityDetail">

            <view class="actLeft" wx:if="{{item.activityType==1}}" style="background-color: aqua;">

                救助

            </view>

            <view class="actLeft" wx:if="{{item.activityType==2}}" style="background-color: rgb(250, 128, 91);">

                拍照

            </view>

            <view class="actLeft" wx:if="{{item.activityType==3}}" style="background-color: rgb(231, 241, 79);">

                喂食

            </view>

            <view class="actLeft" wx:if="{{item.activityType==4}}" style="background-color: rgb(194, 70, 224);">

                其他

            </view>

            <view class="actRight">

                <view class="actContent">

                     <view class="actRoughly">

                         <view >{{item.activityContent}}</view></view>

                </view>

                <view class="actStaterAndDate">

                    <view class="actStater">

                       发起人：{{item.activityStarter}}

                    </view>

                    <view class="actDate">

                        {{item.activityDate}}

                    </view>

                </view>

            </view>

        </view>

    </view>

</view>

### 3.9我参与的活动

*<!--pages/notice/notice.wxml-->*

<view class="">

    <view >

        <view class="msgBlock">

            <view id="name\_notice" class="nameTitle {{isNotice?'chooseOne':''}}" bindtap="toNotice" >通知</view>

            <view id="name\_chat" class="nameTitle {{!isNotice?'chooseOne':''}}"  bindtap="toChatLine">私信</view>

        </view>

    </view>

    <view class="noticeBar">

        <view id="notice\_word">

            通知列

        </view>

        <view id="readAll" >

            <view>

                全部已读

            </view>

            <view class=".iconfont icon-duihao">

            </view>

        </view>

    </view>

    <view id="notice\_detail">

        <view class="singleNotice"

        wx:for="{{isNotice?noticeList:chatList}}"

        wx:key="Id"

        >

            <view class="noticeLeft">

                <image mode="aspectFill" class="noticeImage" src="{{item.profilePicture}}"/>

            </view>

            <view class="noticeRight">

                <view class="noticeTime">

                    {{item.time}}

                </view>

                <view class="noticeName">

                    {{item.name}}

                </view>

                <view class="noticeText">

                    {{item.text}}

                </view>

            </view>

        </view>

    </view>

</view>

## 4后端代码

### 4.1DTO类

#### 4.1.1cardDto

public class cardDto {

    public String owner;

    public String classify="";

    public String name="";

    public Double latitude;

    public Double longitude;

    public String picUrl="";

    public String introduce="";

    public String picName="";

    public void setOwner(String *openid*){this.owner=openid;}

    public void setPicName(String *name*) {this.picName = name; }

    public void setClassify(String *cls*){

        this.classify=cls;

    }

    public void setName(String *name*){

        this.name=name;

    }

    public void setLatitude(double *a*){

        this.latitude=a;

    }

    public void setLongitude(double *a*){

        this.longitude=a;

    }

    public void setPicUrl(String *url*){

        this.picUrl=url;

    }

    public void setIntroduce(String *intro*){

        this.introduce=intro;

    }

}

#### 4.1.2postInfoDto

public class postInfoDto {

    public String userID;

    public String postContent;

    public String postDate;

    public Integer postLikeNum;

    public Integer postCommentNum;

    public String postUserAvatarUrl;

    public Integer postReportNum;

    public boolean postIsTop;

    public boolean postIsEssential;

    public String postTheme;

    public String postImage;

    public Long postID;

    public Long getPostID() {

        return postID;

    }

    public void setPostID(Long *postID*) {

        this.postID = postID;

    }

    public String getPostImage() {

        return postImage;

    }

    public void setPostImage(String *postImage*) {

        this.postImage = postImage;

    }

    public String getPostUserAvatarUrl() {

        return postUserAvatarUrl;

    }

    public void setPostUserAvatarUrl(String *postUserAvatarUrl*) {

        this.postUserAvatarUrl = postUserAvatarUrl;

    }

    public String getPostTheme() {

        return postTheme;

    }

    public void setPostTheme(String *postTheme*) {

        this.postTheme = postTheme;

    }

    public String getUserID() {

        return userID;

    }

    public void setUserID(String *userID*) {

        this.userID = userID;

    }

    public String getPostContent() {

        return postContent;

    }

    public void setPostContent(String *postContent*) {

        this.postContent = postContent;

    }

    public String getPostDate() {

        return postDate;

    }

    public void setPostDate(String *postDate*) {

        this.postDate = postDate;

    }

    public Integer getPostLikeNum() {

        return postLikeNum;

    }

    public void setPostLikeNum(Integer *postLikeNum*) {

        this.postLikeNum = postLikeNum;

    }

    public Integer getPostCommentNum() {

        return postCommentNum;

    }

    public void setPostCommentNum(Integer *postCommentNum*) {

        this.postCommentNum = postCommentNum;

    }

    public Integer getPostReportNum() {

        return postReportNum;

    }

    public void setPostReportNum(Integer *postReportNum*) {

        this.postReportNum = postReportNum;

    }

    public boolean isPostIsTop() {

        return postIsTop;

    }

    public void setPostIsTop(boolean *postIsTop*) {

        this.postIsTop = postIsTop;

    }

    public boolean isPostIsEssential() {

        return postIsEssential;

    }

    public void setPostIsEssential(boolean *postIsEssential*) {

        this.postIsEssential = postIsEssential;

    }

}

#### 4.1.3UserDto

public class UserDto implements Serializable {

    private static final long serialVersionUID = -9077975168976887742L;

    private String username;

    private char[] password;

    private String encryptPwd;

    private Long userId;

    private String salt;

    private List<String> roles;

    public String getUsername() {

        return username;

    }

    public void setUsername(String *username*) {

        this.username = username;

    }

    public Long getUserId() {

        return userId;

    }

    public void setUserId(Long *userId*) {

        this.userId = userId;

    }

    public char[] getPassword() {

        return password;

    }

    public void setPassword(char[] *password*) {

        this.password = password;

    }

    public String getSalt() {

        return salt;

    }

    public void setSalt(String *salt*) {

        this.salt = salt;

    }

    public List<String> getRoles() {

        return roles;

    }

    public void setRoles(List<String> *roles*) {

        this.roles = roles;

    }

    public String getEncryptPwd() {

        return encryptPwd;

    }

    public void setEncryptPwd(String *encryptPwd*) {

        this.encryptPwd = encryptPwd;

    }

}

#### 4.1.4userInfoDto

public class userInfoDto {

    public String Openid = "";

    public String Name = "";

    public String Password = "";

    public String Phone = "";

    public String Gender = "";

    public String AvatarUrl = "";

    public String getOpenid() {

        return Openid;

    }

    public void setOpenid(String *openid*) {

        this.Openid = openid;

    }

    public String getName() {

        return Name;

    }

    public void setName(String *name*) {

        Name = name;

    }

    public String getPassword() {

        return Password;

    }

    public void setPassword(String *password*) {

        Password = password;

    }

    public String getPhone() {

        return Phone;

    }

    public void setPhone(String *phone*) {

        Phone = phone;

    }

    public String getGender() {

        return Gender;

    }

    public void setGender(String *gender*) {

        Gender = gender;

    }

    public String getAvatarUrl() {

        return AvatarUrl;

    }

    public void setAvatarUrl(String *avatarUrl*) {

        AvatarUrl = avatarUrl;

    }

}

### 4.2filter

#### 4.2.1AnyRolesAuthorizationFilter

public class AnyRolesAuthorizationFilter  extends AuthorizationFilter {

    @Override

    protected void postHandle(ServletRequest *request*, ServletResponse *response*){

        request.setAttribute("anyRolesAuthFilter.FILTERED", true);

    }

    @Override

    protected boolean isAccessAllowed(ServletRequest *servletRequest*, ServletResponse *servletResponse*, Object *mappedValue*) throws Exception {

        Boolean afterFiltered = (Boolean)(servletRequest.getAttribute("anyRolesAuthFilter.FILTERED"));

        if( BooleanUtils.isTrue(afterFiltered))

            return true;

        Subject subject = getSubject(servletRequest, servletResponse);

        String[] rolesArray = (String[]) mappedValue;

        if (rolesArray == null || rolesArray.length == 0) { *//没有角色限制，有权限访问*

            return true;

        }

        for (String role : rolesArray) {

            if (subject.hasRole(role)) *//若当前用户是rolesArray中的任何一个，则有权限访问*

                return true;

        }

        return false;

    }

    @Override

    protected boolean onAccessDenied(ServletRequest *request*, ServletResponse *response*) throws IOException {

        HttpServletResponse httpResponse = WebUtils.toHttp(response);

        httpResponse.setCharacterEncoding("UTF-8");

        httpResponse.setContentType("application/json;charset=utf-8");

        httpResponse.setStatus(HttpStatus.SC\_UNAUTHORIZED);

        return false;

    }

}

#### 4.2.2JwtAuthFilter

package com.example.zuccbiosphere.filter;

import com.example.zuccbiosphere.configuration.JWTToken;

import com.example.zuccbiosphere.configuration.JwtUtils;

import com.example.zuccbiosphere.dto.UserDto;

import com.example.zuccbiosphere.service.UserService;

import org.apache.commons.lang3.BooleanUtils;

import org.apache.commons.lang3.StringUtils;

import org.apache.http.HttpStatus;

import org.apache.shiro.authc.AuthenticationException;

import org.apache.shiro.authc.AuthenticationToken;

import org.apache.shiro.subject.Subject;

import org.apache.shiro.web.filter.authc.AuthenticatingFilter;

import org.apache.shiro.web.util.WebUtils;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.web.bind.annotation.RequestMethod;

import javax.servlet.ServletRequest;

import javax.servlet.ServletResponse;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import java.time.LocalDateTime;

import java.time.ZoneId;

import java.util.Date;

public class JwtAuthFilter extends AuthenticatingFilter {

    private final Logger log = LoggerFactory.getLogger(JwtAuthFilter.class);

    private static final int tokenRefreshInterval = 300;

    private UserService userService;

    public JwtAuthFilter(UserService *userService*){

        this.userService = userService;

        this.setLoginUrl("/login");

    }

    @Override

    protected boolean preHandle(ServletRequest *request*, ServletResponse *response*) throws Exception {

        HttpServletRequest httpServletRequest = WebUtils.toHttp(request);

        if (httpServletRequest.getMethod().equals(RequestMethod.OPTIONS.name())) *//对于OPTION请求做拦截，不做token校验*

            return false;

        return super.preHandle(request, response);

    }

    @Override

    protected void postHandle(ServletRequest *request*, ServletResponse *response*){

        this.fillCorsHeader(WebUtils.toHttp(request), WebUtils.toHttp(response));

        request.setAttribute("jwtShiroFilter.FILTERED", true);

    }

    @Override

    protected boolean isAccessAllowed(ServletRequest *request*, ServletResponse *response*, Object *mappedValue*) {

        if(this.isLoginRequest(request, response))

            return true;

        Boolean afterFiltered = (Boolean)(request.getAttribute("jwtShiroFilter.FILTERED"));

        if( BooleanUtils.isTrue(afterFiltered))

            return true;

        boolean allowed = false;

        try {

            allowed = executeLogin(request, response);

        } catch(IllegalStateException *e*){ *//not found any token*

            log.error("Not found any token");

        }catch (Exception *e*) {

            log.error("Error occurs when login", e);

        }

        return allowed || super.isPermissive(mappedValue);

    }

    @Override

    protected AuthenticationToken createToken(ServletRequest *servletRequest*, ServletResponse *servletResponse*) {

        String jwtToken = getAuthzHeader(servletRequest);

        if(StringUtils.isNotBlank(jwtToken)&&!JwtUtils.isTokenExpired(jwtToken))

            return new JWTToken(jwtToken);

        return null;

    }

    @Override

    protected boolean onAccessDenied(ServletRequest *servletRequest*, ServletResponse *servletResponse*) throws Exception {

        HttpServletResponse httpResponse = WebUtils.toHttp(servletResponse);

        httpResponse.setCharacterEncoding("UTF-8");

        httpResponse.setContentType("application/json;charset=UTF-8");

        httpResponse.setStatus(HttpStatus.SC\_NON\_AUTHORITATIVE\_INFORMATION);

        fillCorsHeader(WebUtils.toHttp(servletRequest), httpResponse);

        return false;

    }

    @Override

    protected boolean onLoginSuccess(AuthenticationToken *token*, Subject *subject*, ServletRequest *request*, ServletResponse *response*) throws Exception {

        HttpServletResponse httpResponse = WebUtils.toHttp(response);

        String newToken = null;

        if(token instanceof JWTToken){

            JWTToken jwtToken = (JWTToken)token;

            UserDto user = (UserDto) subject.getPrincipal();

            boolean shouldRefresh = shouldTokenRefresh(JwtUtils.getIssuedAt(jwtToken.getToken()));

            if(shouldRefresh) {

                newToken = userService.generateJwtToken(user.getUsername());

            }

        }

        if(StringUtils.isNotBlank(newToken))

            httpResponse.setHeader("x-auth-token", newToken);

        return true;

    }

    @Override

    protected boolean onLoginFailure(AuthenticationToken *token*, AuthenticationException *e*, ServletRequest *request*, ServletResponse *response*) {

        log.error("Validate token fail, token:{}, error:{}", token.toString(), e.getMessage());

        return false;

    }

    protected String getAuthzHeader(ServletRequest *request*) {

        HttpServletRequest httpRequest = WebUtils.toHttp(request);

        String header = httpRequest.getHeader("x-auth-token");

        return StringUtils.removeStart(header, "Bearer ");

    }

    protected boolean shouldTokenRefresh(Date *issueAt*){

        LocalDateTime issueTime = LocalDateTime.ofInstant(issueAt.toInstant(), ZoneId.systemDefault());

        return LocalDateTime.now().minusSeconds(tokenRefreshInterval).isAfter(issueTime);

    }

    protected void fillCorsHeader(HttpServletRequest *httpServletRequest*, HttpServletResponse *httpServletResponse*){

        httpServletResponse.setHeader("Access-control-Allow-Origin", httpServletRequest.getHeader("Origin"));

        httpServletResponse.setHeader("Access-Control-Allow-Methods", "GET,POST,OPTIONS,HEAD");

        httpServletResponse.setHeader("Access-Control-Allow-Headers", httpServletRequest.getHeader("Access-Control-Request-Headers"));

    }

}

### 4.3service

#### 4.3.1UserService

package com.example.zuccbiosphere.service;

import com.example.zuccbiosphere.configuration.JwtUtils;

import com.example.zuccbiosphere.dto.UserDto;

import org.apache.shiro.crypto.hash.Sha256Hash;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.redis.core.StringRedisTemplate;

import org.springframework.stereotype.Service;

import java.util.Arrays;

import java.util.List;

import java.util.concurrent.TimeUnit;

*/\*\**

*\* 用户信息接口*

*\*/*

@Service

public class UserService {

    private static final String encryptSalt = "F12839WhsnnEV$#23b";

    @Autowired

    private StringRedisTemplate redisTemplate;

*/\*\**

*\* 保存user登录信息，返回token*

*\*/*

    public String generateJwtToken(String *username*) {

        String salt = JwtUtils.generateSalt();

*/\*\**

*\* @todo 将salt保存到数据库或者缓存中*

*\* redisTemplate.opsForValue().set("token:"+username, salt, 3600, TimeUnit.SECONDS);*

*\*/*

        redisTemplate.opsForValue().set("token:"+username, salt, 3600, TimeUnit.SECONDS);

        return JwtUtils.sign(username, salt, 3600); *//生成jwt token，设置过期时间为1小时*

    }

*/\*\**

*\* 获取上次token生成时的salt值和登录用户信息*

*\* @param username*

*\* @return*

*\*/*

    public UserDto getJwtTokenInfo(String *username*) {

        String salt = "";

*/\*\**

*\* @todo 从数据库或者缓存中取出jwt token生成时用的salt*

*\* salt = redisTemplate.opsForValue().get("token:"+username);*

*\*/*

        salt = redisTemplate.opsForValue().get("token:"+username);

*//      UserDto user = getUserInfo(username);*

        UserDto user = new UserDto();

        user.setSalt(salt);

        return user;

    }

*/\*\**

*\* 清除token信息*

*\* @param username 登录用户名*

*\*/*

    public void deleteLoginInfo(String *username*) {

*/\*\**

*\* @todo 删除数据库或者缓存中保存的salt*

*\* redisTemplate.delete("token:"+username);*

*\*/*

        redisTemplate.delete("token:"+username);

    }

*/\*\**

*\* 获取数据库中保存的用户信息，主要是加密后的密码*

*\* @param userName*

*\* @return*

*\*/*

    public UserDto getUserInfo(String *userName*) {

        UserDto user = new UserDto();

*//      String sql="select \* from user where username="+"\""+userName+"\"";*

*//      System.out.println(sql);*

*//      List<Map<String,Object>>maps=jdbcTemplate.queryForList(sql);*

*//      System.out.println(maps);*

*//*

*//      if(maps.size()==1){*

*//          Map<String,Object>usr=maps.get(0);*

*//          String name=usr.get("username").toString();*

*//          Long id=Long.valueOf(usr.get("id").toString());*

*//          String pwd=usr.get("password").toString();*

*//*

*//          user.setUserId(id);*

*//          user.setUsername(name);*

*//          user.setEncryptPwd(new Sha256Hash(pwd, encryptSalt).toHex());*

*//      }*

        user.setUsername(userName);

        user.setEncryptPwd(new Sha256Hash("123456",encryptSalt).toHex());

        return user;

    }

*/\*\**

*\* 获取用户角色列表，强烈建议从缓存中获取*

*\* @param userId*

*\* @return*

*\*/*

    public List<String> getUserRoles(Long *userId*){

        return Arrays.asList("admin");

    }

}

### 4.4utils

#### 4.4.1ZuccBiosphereApplication

package com.example.zuccbiosphere;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class ZuccBiosphereApplication {

    public static void main(String[] *args*) {

        SpringApplication.run(ZuccBiosphereApplication.class, args);

    }

}

### 4.5controller

#### 4.5.1ACFC\_startActivityController

package com.example.zuccbiosphere.controller;

import com.alibaba.fastjson.JSONObject;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

@Api(tags = "动物城友会模块")

@RestController

@RequestMapping("/api/sql")

public class ACFC\_startActivityController {

    @Autowired

    private JdbcTemplate jdbcTemplate;

*/\*\**

*\* 动物城友会发布活动并更新到数据库*

*\* @param req*

*\* @return*

*\*/*

    @ApiOperation(value = "发起活动", notes = "需要传入JSON，包括\n" +

            "openID:\*\*\*\n" +

            "ActivityName:\*\*\*\n" +

            "ActivityType:\*\*\*\n" +

            "ActivityDescription:\*\*\*")

    @PostMapping("/releaseActivity")

    public String ACFC\_startActivity(@RequestBody JSONObject *req*){

*//从前端发送的数据包中根据关键字获取对应信息*

        String ActivityName = req.getString("ActivityName");

        String ActivityType = req.getString("ActivityType");

        String ActivityDescription = req.getString("ActivityDescription");

        String userId = req.getString("userId");

*//        System.out.println(userId);*

        try{

*//上传到数据库*

            String sql = "insert into animalcityfriendsclub (user\_ID,activity\_Name,activity\_Content,activity\_Type) value (?,?,?,?)";

            jdbcTemplate.update(sql,userId,ActivityName,ActivityDescription,ActivityType);

            return "success";

        }catch (Exception *e*){

            e.printStackTrace();

            return "failed";

        }

    }

}

#### 4.5.2LoginController

package com.example.zuccbiosphere.controller;

import com.alibaba.fastjson.JSONObject;

import com.example.zuccbiosphere.dto.UserDto;

import com.example.zuccbiosphere.dto.userInfoDto;

import com.example.zuccbiosphere.service.UserService;

import com.example.zuccbiosphere.utils.HttpMethodUtil;

import com.example.zuccbiosphere.utils.encryptUtils;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import org.apache.shiro.SecurityUtils;

import org.apache.shiro.authc.AuthenticationException;

import org.apache.shiro.authc.UsernamePasswordToken;

import org.apache.shiro.subject.Subject;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.redis.core.StringRedisTemplate;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.web.bind.annotation.\*;

import javax.servlet.http.HttpServletResponse;

import java.util.HashMap;

import java.util.Map;

@Api(tags = "登录模块")

@RestController

@RequestMapping("/api/login")

public class LoginController {

    @Autowired

    private StringRedisTemplate redisTemplate;

    private final Logger logger = LoggerFactory.getLogger(LoginController.class);

    private UserService userService;

    public LoginController(UserService *userService*) {

        this.userService = userService;

    }

*/\*\**

*\* 用户第一次登录*

*\* @return token*

*\*/*

    @ApiOperation(value = "用户登录", notes = "wx.login上传加密字符串即可")

    @ResponseBody

    @RequestMapping(value="/getUserInfo",method = RequestMethod.GET)

    public ResponseEntity<Void> login(String *encryptedData*,String *iv*,String *code*, HttpServletResponse *response*){

        Map map = new HashMap();

*//获取用户openid*

        if(code == null || code.length() == 0){

            map.put("status",0);

            map.put("msg","code不能为空");

            return null;

        }

        String wxspAppid = "wxbe4d121aff0adbb7";

        String wxspSecret = "0dfe287f7d1088088217498b693ec061";

        String grant\_type = "authorization\_code";

*//获取openid*

        String params = "appid="+wxspAppid+"&secret="+wxspSecret+"&js\_code="+code+"&grant\_type="+grant\_type;

        String str= HttpMethodUtil.sendGet("https://api.weixin.qq.com/sns/jscode2session",params);

        JSONObject json=JSONObject.parseObject(str);

        String openid=json.get("openid").toString();

        logger.info(json.toJSONString());

*//获取access\_toekn*

*//        String tokenUrl="https://api.weixin.qq.com/cgi-bin/token";*

*//        String tokenParam="grant\_type=client\_credential&appid="+wxspAppid+"&secret="+wxspSecret;*

*//        String tokenStr=HttpMethodUtil.sendGet(tokenUrl,tokenParam);*

*//        logger.info(tokenStr);*

*//        JSONObject tokenJson=JSONObject.parseObject(tokenStr);*

*//        String access\_token=tokenJson.getString("access\_token");*

*//        String requestUrl="https://api.weixin.qq.com/cgi-bin/user/info";*

*//        String param="access\_token="+access\_token+"&openid="+openid+"&lang=zh\_CN";*

*//        String res=HttpMethodUtil.sendGet(requestUrl,param);*

*//*

*//        logger.info(res);*

*//*

*//        JSONObject userInfo=JSONObject.parseObject(res);*

*//        String headerUrl=userInfo.getString("headimgurl");*

*//        String nickName=userInfo.getString("nickname");*

*//为用户创建信息*

*//        Map<Object, Object> newUserMap=redisTemplate.opsForHash().entries("users:"+ encryptUtils.MD5EncryptMethod(openid));*

*/\*if(newUserMap.isEmpty()){*

*userInfoDto uData=new userInfoDto();*

*uData.setOpenid(encryptUtils.MD5EncryptMethod(openid));*

*uData.setOpenid("");*

*uData.setUserName("");*

*uData.setUserPhone("");*

*uData.setUserGender("");*

*Map<String, Object> newMap = new HashMap<String, Object>();*

*saveOpenId(uData);*

*newMap.put("key", JSON.toJSONString(uData));*

*redisTemplate.opsForHash().putAll("users:"+encryptUtils.MD5EncryptMethod(openid), newMap);*

*System.out.println("new user create success!");*

*}else{*

*System.out.println("user has existed!");*

*}\*/*

*//存储用户信息到数据库*

        userInfoDto uData=new userInfoDto();

        uData.setOpenid(encryptUtils.MD5EncryptMethod(openid));

        uData.setName("");

        uData.setPhone("");

        uData.setGender("");

        uData.setPassword("");

        saveOpenId(uData);

*//为用户生成token*

        Subject subject = SecurityUtils.getSubject();

        try {

            UsernamePasswordToken token = new UsernamePasswordToken(encryptUtils.MD5EncryptMethod(openid), "123456");

            subject.login(token);

            UserDto user = (UserDto) subject.getPrincipal();

            String newToken = userService.generateJwtToken(user.getUsername());

            response.setHeader("x-auth-token", newToken);

            response.setHeader("openid", encryptUtils.MD5EncryptMethod(openid));

            logger.info("User {} login success!",openid);

            return ResponseEntity.ok().build();

        } catch (AuthenticationException *e*) {

            logger.error("User {} login fail, Reason:{}",openid, e.getMessage());

            return ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();

        } catch (Exception *e*) {

            return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).build();

        }

    }

    @Autowired

    private JdbcTemplate jdbcTemplate;

    public void saveOpenId(*userInfoDto* *uDto*){

        String sql = "select \* from user where user\_ID = ?";

        String sql2 = "insert into user(user\_ID) value (?)";

*//如果从数据库中返回查询openid结果为空*

        if (jdbcTemplate.queryForList(sql,uDto.getOpenid()).isEmpty()){

            jdbcTemplate.update(sql2,uDto.getOpenid());

        }

    }

}

#### 4.5.3postBarController

package com.example.zuccbiosphere.controller;  
  
import com.alibaba.fastjson.JSON;  
import com.alibaba.fastjson.JSONArray;  
import com.alibaba.fastjson.JSONObject;  
import com.example.zuccbiosphere.dto.commentDto;  
import com.example.zuccbiosphere.dto.postInfoDto;  
import com.example.zuccbiosphere.dto.userInfoDto;  
import io.swagger.annotations.Api;  
import io.swagger.annotations.ApiOperation;  
import org.joda.time.DateTime;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.jdbc.core.JdbcTemplate;  
import org.springframework.web.bind.annotation.\*;  
  
import java.sql.Connection;  
import javax.annotation.Resource;  
import javax.sql.DataSource;  
import java.sql.\*;  
import java.text.SimpleDateFormat;  
import java.util.ArrayList;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
  
*/\*\*  
 \* 论坛功能模块  
 \*/*@Api(tags = "论坛模块")  
@RestController  
@RequestMapping("/api/postBar")  
public class postBarController {  
 @Autowired  
 private JdbcTemplate jdbcTemplate;  
 @ApiOperation(value = "发帖功能", notes = "需要传入JSON,其中包含:\n" +  
 "postTheme:\*\*\*\n" +  
 "postContent:\*\*\*\n" +  
 "userID:\*\*\*")  
 @PostMapping("/releasePost")  
 public String releasePost(@RequestBody JSONObject req){  
 postInfoDto postData = new postInfoDto();  
 postData.postTheme = req.getString("postTheme");  
 postData.postContent = req.getString("postContent");  
 postData.userID =req.getString("userID");  
 postData.postDate = new DateTime().toString("yyyy-MM-dd HH:mm:ss");  
 postData.postUserAvatarUrl = req.getString("userAvatarUrl");  
 postData.postLikeNum = 0;  
 postData.postReportNum = 0;  
 postData.postCommentNum = 0;  
 postData.postIsEssential = false;  
 postData.postIsTop = false;  
 System.out.println(postData.postTheme);  
  
 try{  
 String sql = "insert into post(user\_ID,post\_Theme,post\_Content," +  
 "post\_Date,post\_LikeNum,post\_CommentNum,post\_ReportNum," +  
 "post\_isTop,post\_isEssential,post\_UserAvatarUrl) value (?,?,?,?,?,?,?,?,?,?)";  
 jdbcTemplate.update(sql,postData.userID,postData.postTheme,postData.postContent,  
 postData.postDate,postData.postLikeNum,postData.postCommentNum,  
 postData.postReportNum,postData.postIsTop,postData.postIsEssential,  
 postData.postUserAvatarUrl);  
  
 return "Post release success";  
 }catch (Exception e){  
 e.printStackTrace();  
 return "Post release failed";  
 }  
  
 }  
  
 @Autowired  
 private DataSource dataSource;  
  
 @ApiOperation(value = "发送帖子信息到前端页面显示", notes = "不需要传入参数")  
  
 @PostMapping("/loadPost")  
 public JSONArray loadPost(){  
  
 //用于存储从数据库中获取的所有帖子  
 List<postInfoDto> postInfoList = new ArrayList<>();  
  
 //用于存储openid和用户名之间的对应关系  
 Map<String,String> userMap = new HashMap<>();  
  
  
 try{  
 //数据库连接  
 Connection conn = dataSource.getConnection();  
 String sql = "select \* from post ORDER BY post\_Date DESC";  
 java.sql.PreparedStatement pst = conn.prepareStatement(sql);  
 java.sql.ResultSet rs = pst.executeQuery();  
 //时间格式转换  
 SimpleDateFormat fmt = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  
  
  
 while(rs.next()){  
 postInfoDto post = new postInfoDto();  
 post.setPostID(rs.getLong(1));  
 post.setUserID(rs.getString(2));  
 post.setPostTheme(rs.getString(3));  
 post.setPostContent(rs.getString(4));  
 post.setPostDate(rs.getDate(5) + " " + rs.getTime(5));  
 post.setPostLikeNum(rs.getInt(6));  
 post.setPostCommentNum(rs.getInt(7));  
 post.setPostUserAvatarUrl(rs.getString(8));  
 post.setPostReportNum(rs.getInt(9));  
 post.setPostIsTop(rs.getBoolean(10));  
 post.setPostIsEssential(rs.getBoolean(11));  
 post.setPostImage(rs.getString(12));  
 postInfoList.add(post);  
 }  
  
 sql = "select user\_ID,user\_Name from user";  
 pst = conn.prepareStatement(sql);  
 rs = pst.executeQuery();  
  
 //建立openid和用户名之间的映射关系  
 while(rs.next()){  
 userMap.put(rs.getString(1),rs.getString(2));  
 }  
 pst.close();  
 conn.close();  
  
 }catch (SQLException e){  
 e.printStackTrace();  
 }  
  
 //返回给前端的json数据包  
 JSONArray arr = new JSONArray();  
  
 for(postInfoDto post: postInfoList){  
 JSONObject res = new JSONObject();  
 res.put("post\_ID", post.postID);  
 res.put("user\_ID", userMap.get(post.userID));  
 res.put("post\_Theme", post.postTheme);  
 res.put("post\_Content", post.postContent);  
 res.put("post\_Date", post.postDate);  
 res.put("post\_likeNum", post.postLikeNum);  
 res.put("post\_commentNum", post.postCommentNum);  
 res.put("user\_head", post.postUserAvatarUrl);  
 res.put("post\_isTop", post.postIsTop);  
 res.put("post\_isEssential", post.postIsEssential);  
 res.put("post\_Image",post.postImage);  
 arr.add(res);  
 }  
  
 return arr;  
  
  
 }  
  
 @ApiOperation(value = "从数据库中调取此帖子的详细信息", notes = "需要传入帖子的id")  
 @PostMapping("/loadPostDetail")  
 public JSONObject loadPostDetail(@RequestBody JSONObject req){  
  
 JSONObject result = new JSONObject();  
 Long postId = req.getLong("post\_ID");  
  
 //帖子信息实体数组存储取出的帖子信息  
 postInfoDto data = new postInfoDto();  
  
 //用于存储openid和用户名之间的对应关系  
 Map<String,String> userMap = new HashMap<>();  
  
 try{  
 Connection conn = dataSource.getConnection();  
 String sql = "select \* from post where post\_ID = ?";  
 PreparedStatement pst = conn.prepareStatement(sql);  
 pst.setLong(1,postId);  
 ResultSet rs = pst.executeQuery();  
 if (rs.next()) {  
 data.setPostID(postId);  
 data.setUserID(rs.getString("user\_ID"));  
 data.setPostTheme(rs.getString("post\_Theme"));  
 data.setPostContent(rs.getString("post\_Content"));  
 data.setPostDate(rs.getDate("post\_Date") + " " + rs.getTime("post\_Date"));  
 data.setPostLikeNum(rs.getInt("post\_LikeNum"));  
 data.setPostCommentNum(rs.getInt("post\_CommentNum"));  
 data.setPostUserAvatarUrl(rs.getString("post\_UserAvatarUrl"));  
 data.setPostIsTop(rs.getBoolean("post\_isTop"));  
 data.setPostIsEssential(rs.getBoolean("post\_isEssential"));  
 }  
  
  
  
 sql = "select user\_ID,user\_Name from user";  
 pst = conn.prepareStatement(sql);  
 rs = pst.executeQuery();  
  
 while(rs.next()){  
 userMap.put(rs.getString("user\_ID"),rs.getString("user\_Name"));  
 }  
 pst.close();  
 conn.close();  
  
 }catch (SQLException e){  
 e.printStackTrace();  
 }  
  
 result.put("user\_ID",userMap.get(data.getUserID()));  
 result.put("user\_icon",data.getPostUserAvatarUrl());  
 result.put("user\_content",data.getPostContent());  
 result.put("post\_Date",data.getPostDate());  
 result.put("post\_likeNum",data.getPostLikeNum());  
 result.put("post\_isLiked",data.postIsTop);  
 result.put("post\_isEssential",data.postIsEssential);  
  
 return result;  
 }  
  
  
  
 @ApiOperation(value = "从数据库中调取此帖子的评论", notes = "需要传入帖子的id")  
 @PostMapping("/loadPostComment")  
 public JSONArray loadPostComment(@RequestBody JSONObject req){  
  
 JSONArray result = new JSONArray();  
 Long postId = req.getLong("post\_ID");  
  
 //评论信息实体数组存储数据库中取出的评论信息  
 List<commentDto> commentDtoList = new ArrayList<>();  
  
 //用于存储openid和用户名、用户头像之间的对应关系  
 Map<String,List<String>> userMap = new HashMap<>();  
  
 try{  
 Connection conn = dataSource.getConnection();  
 String sql = "select \* from commentrecord where post\_ID = ? order by Comment\_Date";  
 PreparedStatement pst = conn.prepareStatement(sql);  
 pst.setLong(1,postId);  
 ResultSet rs = pst.executeQuery();  
 while(rs.next()){  
 commentDto temp = new commentDto();  
 temp.setUserId(rs.getString("user\_ID"));  
 temp.setPostId(rs.getLong("post\_ID"));  
 temp.setCommentContent(rs.getString("Comment\_Con"));  
 temp.setCommentWho(rs.getString("Comment\_Acced"));  
 temp.setCommentDate(rs.getDate("Comment\_Date") + " " + rs.getTime("Comment\_Date"));  
 commentDtoList.add(temp);  
 }  
  
 sql = "select user\_ID,user\_Name,user\_AvatarUrl from user";  
 pst = conn.prepareStatement(sql);  
 rs = pst.executeQuery();  
  
 //建立openid和用户名之间的映射关系  
 while(rs.next()){  
 List<String> list = new ArrayList<>();  
 list.add(0,rs.getString("user\_Name"));  
 list.add(1,rs.getString("user\_AvatarUrl"));  
 userMap.put(rs.getString("user\_ID"),list);  
 }  
 pst.close();  
 conn.close();  
  
 }catch (SQLException e){  
 e.printStackTrace();  
 }  
  
 for (commentDto data: commentDtoList) {  
 JSONObject temp = new JSONObject();  
 List<String> userInfoList = userMap.get(data.getUserId());  
 temp.put("user\_ID",userInfoList.get(0));  
 temp.put("user\_icon",userInfoList.get(1));  
 temp.put("user\_Content",data.getCommentContent());  
 temp.put("post\_Date",data.getCommentDate());  
 result.add(temp);  
 }  
  
  
 return result;  
 }  
  
  
}

#### 4.5.4UserInfoController

package com.example.zuccbiosphere.controller;

import com.alibaba.fastjson.JSONObject;

import com.example.zuccbiosphere.dto.userInfoDto;

import io.swagger.annotations.ApiOperation;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

*/\*\**

*\* 登录模块*

*\*/*

@RestController

@RequestMapping("/api/login")

public class UserInfoController {

    @Autowired

    private JdbcTemplate jdbcTemplate;

    @ApiOperation(value = "保存用户信息到数据库", notes = "需要传入JSON，其中包括\n" +

            "openId:\*\*\*\n" +

            "userName:\*\*\*\n" +

            "avatarUrl:\*\*\*")

    @PostMapping("/saveUserInfo")

    public String saveUserInfo(@RequestBody JSONObject *req*){

        userInfoDto curUser = new userInfoDto();

        curUser.Name = req.getString("userName");

        curUser.Openid = req.getString("openId");

        curUser.AvatarUrl = req.getString("userAvatarUrl");

        try{

            String sql = "update user set user\_Name = ?, user\_AvatarUrl = ? where user\_ID = ?";

            jdbcTemplate.update(sql,curUser.Name,curUser.AvatarUrl,curUser.Openid);

            return "Save done!";

        }catch (Exception *e*){

            e.printStackTrace();

            return "Save failed";

        }

    }

}

#### 4.5.5usersController

package com.example.zuccbiosphere.controller;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.example.zuccbiosphere.utils.DistanceUtil;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import io.swagger.annotations.ApiResponse;

import io.swagger.annotations.ApiResponses;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.redis.core.StringRedisTemplate;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;

import java.util.Map;

@Api(tags = "用户模块")

@RestController

@RequestMapping("/api/user")

public class usersController {

    @Autowired

    private StringRedisTemplate redisTemplate;

    private final Logger logger = LoggerFactory.getLogger(usersController.class);

    @ApiOperation(value = "获取用户信息", notes = "需要传入JSON,其中包含:\n" +

            "openid:\*\*\*")

    @ApiResponses({

            @ApiResponse(code = 200, message = "得到Map格式用户数据\n" +

                    "得到null则openid有误"),

    })

    @PostMapping("/getUserData")

    public Map<Object, Object> getUser(@RequestBody JSONObject *req*) {

        String openid=req.getString("openid");

        String key = "users:"+openid;

        try{

            return redisTemplate.opsForHash().entries(key);

        }catch (Exception *e*){

            logger.info(e.toString());

        }

        return null;

    }

    @ApiOperation(value = "添加用户足迹", notes = "需要传入JSON,其中包含:\n" +

            "openid:\*\*\*\n" +

            "cardId:\*\*\*\n" +

            "latitude:\*\*\*\n" +

            "longitude:\*\*\*")

    @ApiResponses({

            @ApiResponse(code = 200, message = "得到add foot success!表示添加成功\n" +

                    "得到distance not enough!表示用户距离打卡点超过1公里\n" +

                    "得到Unknown Error!表示更新失败，可能是网络问题或者传入参数有误"),

    })

    @PostMapping("/addUserFoot")

    public String addCollect(@RequestBody JSONObject *req*){

        String openid=req.getString("openid");

        String cardId=req.getString("cardId");

        Double latitude=req.getDouble("latitude");

        Double longitude=req.getDouble("longitude");

        try {

            String cardKey="cards-info:"+cardId;

            Map<Object, Object> map=redisTemplate.opsForHash().entries(cardKey);

            JSONObject card = JSONObject.parseObject(map.get("key").toString());

            Double cardLatitude=card.getDouble("latitude");

            Double cardLongitude=card.getDouble("longitude");

            Double dis= DistanceUtil.getDistance(longitude,latitude,cardLongitude,cardLatitude);

            if(dis<=1000){

*//为用户添加足迹*

                String userKey="users:"+openid;

                Map<Object, Object> userMap=redisTemplate.opsForHash().entries(userKey);

                JSONObject user = JSONObject.parseObject(userMap.get("key").toString());

                String myFoot = user.getString("myFoot");

*//检验是否重复*

                String[] tmp=myFoot.split("\\\*");

                for(int i=0;i<tmp.length;i++){

                    if(tmp[i].length()==0)continue;

                    if(tmp[i].equals(cardKey)){

                        return "repeated!";

                    }

                }

                myFoot+=cardKey+"\*";

                user.replace("myFoot",myFoot);

                Map<String, Object> newUserMap = new HashMap<>();

                newUserMap.put("key", JSON.toJSONString(user));

                redisTemplate.opsForHash().putAll(userKey, newUserMap);

                return "add foot success!";

            }else{

                return "distance not enough!";

            }

        }catch (Exception *e*){

            logger.info(e.toString());

            return "Unknown Error!";

        }

    }

    @ApiOperation(value = "添加用户收藏", notes = "需要传入JSON,其中包含:\n" +

            "openid:\*\*\*\n" +

            "cardId:\*\*\*\n")

    @ApiResponses({

            @ApiResponse(code = 200, message = "得到add collect success!表示添加成功\n" +

                    "得到add collect failed!可能网络不畅或者传入参数有误"),

    })

    @PostMapping("/addUserCollect")

    public String addFoot(@RequestBody JSONObject *req*){

        String openid=req.getString("openid");

        String cardId=req.getString("cardId");

        String[] check=cardId.split("&");

        if(check.length!=3){

            return "PE";

        }

        if(check[0].equals("undefined")||check[1].equals("undefined")||check[2].equals("undefined")){

            return "PE";

        }

        try {

*//为用户添加收藏*

            String cardKey="cards-info:"+cardId;

            String userKey="users:"+openid;

            Map<Object, Object> userMap=redisTemplate.opsForHash().entries(userKey);

            JSONObject user = JSONObject.parseObject(userMap.get("key").toString());

            String myCollect = user.getString("myCollect");

*//检验是否重复*

            String[] tmp=myCollect.split("\\\*");

            for(int i=0;i<tmp.length;i++){

                if(tmp[i].length()==0)continue;

                if(tmp[i].equals(cardKey)){

                    return "repeated!";

                }

            }

            myCollect+=cardKey+"\*";

            user.replace("myCollect",myCollect);

            Map<String, Object> newUserMap = new HashMap<>();

            newUserMap.put("key", JSON.toJSONString(user));

            redisTemplate.opsForHash().putAll(userKey, newUserMap);

            return "add collect success!";

        }catch (Exception *e*){

            logger.info(e.toString());

            return "add collect failed!";

        }

    }

    @ApiOperation(value = "得到用户收藏/足迹列表", notes = "需要传入JSON,其中包含:\n" +

            "cards:\*\*\*\n" +

            "注:只需把user信息中的collect传上来即可")

    @ApiResponses({

            @ApiResponse(code = 200, message = "得到Map格式的卡片列表"),

    })

    @PostMapping("/getUserCollectOrFoot")

    public Map<Object, Object> getUserCollect(@RequestBody JSONObject *req*){

        String tmp=req.getString("cards");

        String[] cards=tmp.split("\\\*");

        Map<Object, Object> res= new HashMap<>();

        int cnt=1;

        for(int i=1;i<=cards.length;i++){

            if(cards[i-1].length()==0)continue;

            Map<Object, Object> map=redisTemplate.opsForHash().entries(cards[i-1]);

            if(map!=null&&!map.isEmpty()){

                res.put("key"+cnt,map);

                cnt++;

            }

        }

        return res;

    }

    @ApiOperation(value = "删除用户收藏", notes = "需要传入JSON,其中包含:\n" +

            "openid:\*\*\*\n" +

            "cardId:\*\*\*\n")

    @ApiResponses({

            @ApiResponse(code = 200, message = "得到delete collect success!表示添加成功\n" +

                    "得到delete collect failed!可能网络不畅或者传入参数有误"),

    })

    @PostMapping("deleteCollect")

    public String deleteCollect(@RequestBody JSONObject *req*){

        String openid=req.getString("openid");

        String cardId=req.getString("cardId");

        String[] check=cardId.split("&");

        if(check.length!=3){

            return "PE";

        }

        if(check[0].equals("undefined")||check[1].equals("undefined")||check[2].equals("undefined")){

            return "PE";

        }

        try {

*//为用户删除收藏*

            String cardKey="cards-info:"+cardId;

            String userKey="users:"+openid;

            Map<Object, Object> userMap=redisTemplate.opsForHash().entries(userKey);

            JSONObject user = JSONObject.parseObject(userMap.get("key").toString());

            String myCollect = user.getString("myCollect");

            String[] list=myCollect.split("\\\*");

            String res="";

            for(int i=0;i<list.length;i++){

                if(list[i].length()!=0 && !list[i].equals(cardKey)){

                    res+=list[i]+"\*";

                }

            }

            user.replace("myCollect",res);

            Map<String, Object> newUserMap = new HashMap<>();

            newUserMap.put("key", JSON.toJSONString(user));

            redisTemplate.opsForHash().putAll(userKey, newUserMap);

            return "delete collect success!";

        }catch (Exception *e*){

            logger.info(e.toString());

            return "delete collect failed!";

        }

    }

}

### 4.4cofiguration

#### 4.6.1DbShiroRealm

package com.example.zuccbiosphere.configuration;

import com.example.zuccbiosphere.dto.UserDto;

import com.example.zuccbiosphere.service.UserService;

import org.apache.shiro.authc.\*;

import org.apache.shiro.authc.credential.HashedCredentialsMatcher;

import org.apache.shiro.authz.AuthorizationInfo;

import org.apache.shiro.authz.SimpleAuthorizationInfo;

import org.apache.shiro.crypto.hash.Sha256Hash;

import org.apache.shiro.realm.AuthorizingRealm;

import org.apache.shiro.subject.PrincipalCollection;

import org.apache.shiro.util.ByteSource;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.List;

public class DbShiroRealm extends AuthorizingRealm {

    private final Logger log = LoggerFactory.getLogger(DbShiroRealm.class);

    private static final String encryptSalt = "F12839WhsnnEV$#23b";

    private UserService userService;

    public DbShiroRealm(UserService *userService*) {

        this.userService = userService;

        this.setCredentialsMatcher(new HashedCredentialsMatcher(Sha256Hash.ALGORITHM\_NAME));

    }

    @Override

    public boolean supports(AuthenticationToken *token*) {

        return token instanceof UsernamePasswordToken;

    }

    @Override

    protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken *token*) throws AuthenticationException {

        UsernamePasswordToken userpasswordToken = (UsernamePasswordToken)token;

        String username = userpasswordToken.getUsername();

        UserDto user = userService.getUserInfo(username);

        if(user == null)

            throw new AuthenticationException("用户名或者密码错误");

        return new SimpleAuthenticationInfo(user, user.getEncryptPwd(), ByteSource.Util.bytes(encryptSalt), "dbRealm");

    }

    @Override

    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection *principals*) {

        SimpleAuthorizationInfo simpleAuthorizationInfo = new SimpleAuthorizationInfo();

        UserDto user = (UserDto) principals.getPrimaryPrincipal();

        List<String> roles = user.getRoles();

        if(roles == null) {

            roles = userService.getUserRoles(user.getUserId());

            user.setRoles(roles);

        }

        if (roles != null)

            simpleAuthorizationInfo.addRoles(roles);

        return simpleAuthorizationInfo;

    }

}

#### 4.6.2JWTCredentialsMatcher

package com.example.zuccbiosphere.configuration;

import com.auth0.jwt.JWT;

import com.auth0.jwt.JWTVerifier;

import com.auth0.jwt.algorithms.Algorithm;

import com.auth0.jwt.exceptions.JWTVerificationException;

import com.example.zuccbiosphere.dto.UserDto;

import org.apache.shiro.authc.AuthenticationInfo;

import org.apache.shiro.authc.AuthenticationToken;

import org.apache.shiro.authc.credential.CredentialsMatcher;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.io.UnsupportedEncodingException;

public class JWTCredentialsMatcher implements CredentialsMatcher {

    private final Logger log = LoggerFactory.getLogger(JWTCredentialsMatcher.class);

    @Override

    public boolean doCredentialsMatch(AuthenticationToken *authenticationToken*, AuthenticationInfo *authenticationInfo*) {

        String token = (String) authenticationToken.getCredentials();

        Object stored = authenticationInfo.getCredentials();

        String salt = stored.toString();

        UserDto user = (UserDto)authenticationInfo.getPrincipals().getPrimaryPrincipal();

        try {

            Algorithm algorithm = Algorithm.HMAC256(salt);

            JWTVerifier verifier = JWT.require(algorithm)

                    .withClaim("username", user.getUsername())

                    .build();

            verifier.verify(token);

            return true;

        } catch (JWTVerificationException *e*) {

            log.error("Token Error:{}", e.getMessage());

        }

        return false;

    }

}

#### 4.6.3JWTShiroRealm

package com.example.zuccbiosphere.configuration;

import com.example.zuccbiosphere.dto.UserDto;

import com.example.zuccbiosphere.service.UserService;

import org.apache.shiro.authc.AuthenticationException;

import org.apache.shiro.authc.AuthenticationInfo;

import org.apache.shiro.authc.AuthenticationToken;

import org.apache.shiro.authc.SimpleAuthenticationInfo;

import org.apache.shiro.authz.AuthorizationInfo;

import org.apache.shiro.authz.SimpleAuthorizationInfo;

import org.apache.shiro.realm.AuthorizingRealm;

import org.apache.shiro.subject.PrincipalCollection;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

*/\*\**

*\* 自定义身份认证*

*\* 基于HMAC（ 散列消息认证码）的控制域*

*\*/*

public class JWTShiroRealm extends AuthorizingRealm {

    private final Logger log = LoggerFactory.getLogger(JWTShiroRealm.class);

    protected UserService userService;

    public JWTShiroRealm(UserService *userService*){

        this.userService = userService;

        this.setCredentialsMatcher(new JWTCredentialsMatcher());

    }

    @Override

    public boolean supports(AuthenticationToken *token*) {

        return token instanceof JWTToken;

    }

*/\*\**

*\* 认证信息.(身份验证) : Authentication 是用来验证用户身份*

*\* 默认使用此方法进行用户名正确与否验证，错误抛出异常即可。*

*\*/*

    @Override

    protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken *authcToken*) throws AuthenticationException {

        JWTToken jwtToken = (JWTToken) authcToken;

        String token = jwtToken.getToken();

        UserDto user = userService.getJwtTokenInfo(JwtUtils.getUsername(token));

        if(user == null)

            throw new AuthenticationException("token过期，请重新登录");

        SimpleAuthenticationInfo authenticationInfo = new SimpleAuthenticationInfo(user, user.getSalt(), "jwtRealm");

        return authenticationInfo;

    }

    @Override

    protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection *principals*) {

        return new SimpleAuthorizationInfo();

    }

}

#### 4.6.4JWTToken

package com.example.zuccbiosphere.configuration;

import org.apache.shiro.authc.HostAuthenticationToken;

public class JWTToken implements HostAuthenticationToken {

    private static final long serialVersionUID = 9217639903967592166L;

    private String token;

    private String host;

    public JWTToken(String *token*) {

        this(token, null);

    }

    public JWTToken(String *token*, String *host*) {

        this.token = token;

        this.host = host;

    }

    public String getToken(){

        return this.token;

    }

    public String getHost() {

        return host;

    }

    @Override

    public Object getPrincipal() {

        return token;

    }

    @Override

    public Object getCredentials() {

        return token;

    }

    @Override

    public String toString(){

        return token + ':' + host;

    }

}

#### 4.6.5JwtUtils

package com.example.zuccbiosphere.configuration;

import com.auth0.jwt.JWT;

import com.auth0.jwt.algorithms.Algorithm;

import com.auth0.jwt.exceptions.JWTDecodeException;

import com.auth0.jwt.interfaces.DecodedJWT;

import org.apache.shiro.crypto.SecureRandomNumberGenerator;

import java.io.UnsupportedEncodingException;

import java.util.Calendar;

import java.util.Date;

public class JwtUtils {

*/\*\**

*\* 获得token中的信息无需secret解密也能获得*

*\* @return token中包含的签发时间*

*\*/*

    public static Date getIssuedAt(String *token*) {

        try {

            DecodedJWT jwt = JWT.decode(token);

            return jwt.getIssuedAt();

        } catch (JWTDecodeException *e*) {

            return null;

        }

    }

*/\*\**

*\* 获得token中的信息无需secret解密也能获得*

*\* @return token中包含的用户名*

*\*/*

    public static String getUsername(String *token*) {

        try {

            DecodedJWT jwt = JWT.decode(token);

            return jwt.getClaim("username").asString();

        } catch (JWTDecodeException *e*) {

            return null;

        }

    }

*/\*\**

*\* 生成签名,expireTime后过期*

*\* @param username 用户名*

*\* @param time 过期时间s*

*\* @return 加密的token*

*\*/*

    public static String sign(String *username*, String *salt*, long *time*) {

        Date date = new Date(System.currentTimeMillis()+time\*1000);

        Algorithm algorithm = Algorithm.HMAC256(salt);

*// 附带username信息*

        return JWT.create()

                .withClaim("username", username)

                .withExpiresAt(date)

                .withIssuedAt(new Date())

                .sign(algorithm);

    }

*/\*\**

*\* token是否过期*

*\* @return true：过期*

*\*/*

    public static boolean isTokenExpired(String *token*) {

        Date now = Calendar.getInstance().getTime();

        DecodedJWT jwt = JWT.decode(token);

        return jwt.getExpiresAt().before(now);

    }

*/\*\**

*\* 生成随机盐,长度32位*

*\* @return*

*\*/*

    public static String generateSalt(){

        SecureRandomNumberGenerator secureRandom = new SecureRandomNumberGenerator();

        String hex = secureRandom.nextBytes(16).toHex();

        return hex;

    }

}

#### 4.6.6ResponseHeaderAdvice

package com.example.zuccbiosphere.configuration;

import org.springframework.core.MethodParameter;

import org.springframework.http.MediaType;

import org.springframework.http.converter.HttpMessageConverter;

import org.springframework.http.server.ServerHttpRequest;

import org.springframework.http.server.ServerHttpResponse;

import org.springframework.http.server.ServletServerHttpRequest;

import org.springframework.http.server.ServletServerHttpResponse;

import org.springframework.web.bind.annotation.ControllerAdvice;

import org.springframework.web.servlet.mvc.method.annotation.ResponseBodyAdvice;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

@ControllerAdvice

public class ResponseHeaderAdvice implements ResponseBodyAdvice<Object> {

    @Override

    public boolean supports(MethodParameter *methodParameter*, Class<? extends HttpMessageConverter<?>> *aClass*) {

        return true;

    }

    @Override

    public Object beforeBodyWrite(Object *o*, MethodParameter *methodParameter*, MediaType *mediaType*, Class<? extends HttpMessageConverter<?>> *aClass*,

                                  ServerHttpRequest *serverHttpRequest*, ServerHttpResponse *serverHttpResponse*) {

        ServletServerHttpRequest serverRequest = (ServletServerHttpRequest)serverHttpRequest;

        ServletServerHttpResponse serverResponse = (ServletServerHttpResponse)serverHttpResponse;

        if(serverRequest == null || serverResponse == null

                || serverRequest.getServletRequest() == null || serverResponse.getServletResponse() == null) {

            return o;

        }

*// 对于未添加跨域消息头的响应进行处理*

        HttpServletRequest request = serverRequest.getServletRequest();

        HttpServletResponse response = serverResponse.getServletResponse();

        String originHeader = "Access-Control-Allow-Origin";

        if(!response.containsHeader(originHeader)) {

            String origin = request.getHeader("Origin");

            if(origin == null) {

                String referer = request.getHeader("Referer");

                if(referer != null)

                    origin = referer.substring(0, referer.indexOf("/", 7));

            }

            response.setHeader("Access-Control-Allow-Origin", origin);

        }

        String allowHeaders = "Access-Control-Allow-Headers";

        if(!response.containsHeader(allowHeaders))

            response.setHeader(allowHeaders, request.getHeader(allowHeaders));

        String allowMethods = "Access-Control-Allow-Methods";

        if(!response.containsHeader(allowMethods))

            response.setHeader(allowMethods, "GET,POST,OPTIONS,HEAD");

        String exposeHeaders = "access-control-expose-headers";

        if(!response.containsHeader(exposeHeaders))

            response.setHeader(exposeHeaders, "x-auth-token");

        return o;

    }

}

#### 4.6.7ShiroConfig

package com.example.zuccbiosphere.configuration;

import com.example.zuccbiosphere.filter.AnyRolesAuthorizationFilter;

import com.example.zuccbiosphere.filter.JwtAuthFilter;

import com.example.zuccbiosphere.service.UserService;

import org.apache.shiro.authc.Authenticator;

import org.apache.shiro.authc.pam.FirstSuccessfulStrategy;

import org.apache.shiro.authc.pam.ModularRealmAuthenticator;

import org.apache.shiro.mgt.SessionStorageEvaluator;

import org.apache.shiro.realm.Realm;

import org.apache.shiro.spring.security.interceptor.AuthorizationAttributeSourceAdvisor;

import org.apache.shiro.spring.web.ShiroFilterFactoryBean;

import org.apache.shiro.spring.web.config.DefaultShiroFilterChainDefinition;

import org.apache.shiro.spring.web.config.ShiroFilterChainDefinition;

import org.apache.shiro.web.mgt.DefaultWebSessionStorageEvaluator;

import org.springframework.beans.factory.annotation.Qualifier;

import org.springframework.boot.web.servlet.FilterRegistrationBean;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import javax.servlet.DispatcherType;

import javax.servlet.Filter;

import java.util.Arrays;

import java.util.Map;

import org.apache.shiro.mgt.SecurityManager;

*/\*\**

*\* shiro配置类*

*\*/*

@Configuration

public class ShiroConfig {

    @Bean

    public AuthorizationAttributeSourceAdvisor authorizationAttributeSourceAdvisor(@Qualifier("securityManager") SecurityManager *securityManager*) {

        AuthorizationAttributeSourceAdvisor advisor = new AuthorizationAttributeSourceAdvisor();

        advisor.setSecurityManager(securityManager);

        return advisor;

    }

    @Bean

    public FilterRegistrationBean<Filter> filterRegistrationBean(SecurityManager *securityManager*, UserService *userService*) throws Exception{

        FilterRegistrationBean<Filter> filterRegistration = new FilterRegistrationBean<Filter>();

        filterRegistration.setFilter((Filter)shiroFilter(securityManager, userService).getObject());

        filterRegistration.addInitParameter("targetFilterLifecycle", "true");

        filterRegistration.setAsyncSupported(true);

        filterRegistration.setEnabled(true);

        filterRegistration.setDispatcherTypes(DispatcherType.REQUEST,DispatcherType.ASYNC);

        return filterRegistration;

    }

    @Bean

    public Authenticator authenticator(UserService *userService*) {

        ModularRealmAuthenticator authenticator = new ModularRealmAuthenticator();

        authenticator.setRealms(Arrays.asList(jwtShiroRealm(userService), dbShiroRealm(userService)));

        authenticator.setAuthenticationStrategy(new FirstSuccessfulStrategy());

        return authenticator;

    }

    @Bean

    protected SessionStorageEvaluator sessionStorageEvaluator(){

        DefaultWebSessionStorageEvaluator sessionStorageEvaluator = new DefaultWebSessionStorageEvaluator();

        sessionStorageEvaluator.setSessionStorageEnabled(false);

        return sessionStorageEvaluator;

    }

    @Bean("dbRealm")

    public Realm dbShiroRealm(UserService *userService*) {

        DbShiroRealm myShiroRealm = new DbShiroRealm(userService);

        return myShiroRealm;

    }

    @Bean("jwtRealm")

    public Realm jwtShiroRealm(UserService *userService*) {

        JWTShiroRealm myShiroRealm = new JWTShiroRealm(userService);

        return myShiroRealm;

    }

*/\*\**

*\* 设置过滤器*

*\*/*

    @Bean("shiroFilter")

    public ShiroFilterFactoryBean shiroFilter(SecurityManager *securityManager*, UserService *userService*) {

        ShiroFilterFactoryBean factoryBean = new ShiroFilterFactoryBean();

        factoryBean.setSecurityManager(securityManager);

        Map<String, Filter> filterMap = factoryBean.getFilters();

        filterMap.put("authcToken", createAuthFilter(userService));

        filterMap.put("anyRole", createRolesFilter());

        factoryBean.setFilters(filterMap);

        factoryBean.setFilterChainDefinitionMap(shiroFilterChainDefinition().getFilterChainMap());

        return factoryBean;

    }

    @Bean

    protected ShiroFilterChainDefinition shiroFilterChainDefinition() {

        DefaultShiroFilterChainDefinition chainDefinition = new DefaultShiroFilterChainDefinition();

*//swagger2免拦截*

        chainDefinition.addPathDefinition("/swagger-ui.html\*\*", "anon");

        chainDefinition.addPathDefinition("/v2/api-docs", "anon");

        chainDefinition.addPathDefinition("/swagger-resources/\*\*", "anon");

        chainDefinition.addPathDefinition("/webjars/\*\*", "anon");

*//接口拦截*

*//任务相关*

        chainDefinition.addPathDefinition("/api/task/addUserScore", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/task/getTodayTask", "noSessionCreation,anon");

        chainDefinition.addPathDefinition("/api/task/finishTask", "noSessionCreation,anon");

*//用户相关*

        chainDefinition.addPathDefinition("/api/user/getUserData", "noSessionCreation,anon");

        chainDefinition.addPathDefinition("/api/user/addUserFoot", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/user/addUserCollect", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/user/getUserCollectOrFoot", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/user/deleteCollect", "noSessionCreation,authcToken");

*//卡片相关*

        chainDefinition.addPathDefinition("/api/card/uploadCard", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/card/checkCard", "noSessionCreation,anon");

        chainDefinition.addPathDefinition("/api/card/addCard", "noSessionCreation,anon");

        chainDefinition.addPathDefinition("/api/card/deleteCard", "noSessionCreation,anon");

        chainDefinition.addPathDefinition("/api/card/getCardStatus", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/card/getCards", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/card/cardsPopular", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/card/addCardsChat", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/api/card/addCardsScore", "noSessionCreation,authcToken");

*//排行榜相关*

        chainDefinition.addPathDefinition("/api/rank/getUserRank", "noSessionCreation,anon");

        chainDefinition.addPathDefinition("/api/rank/getCardRank", "noSessionCreation,anon");

*//token*

        chainDefinition.addPathDefinition("/api/login/getUserInfo", "noSessionCreation,anon");

*//测试*

        chainDefinition.addPathDefinition("/api/test/hello", "noSessionCreation,anon");

        chainDefinition.addPathDefinition("/api/test/upload", "noSessionCreation,authcToken");

        chainDefinition.addPathDefinition("/\*\*", "noSessionCreation,authcToken");

        return chainDefinition;

    }

    protected JwtAuthFilter createAuthFilter(UserService *userService*){

        return new JwtAuthFilter(userService);

    }

    protected AnyRolesAuthorizationFilter createRolesFilter(){

        return new AnyRolesAuthorizationFilter();

    }

}

#### 4.6.8swaggerConfig

package com.example.zuccbiosphere.configuration;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;

import org.springframework.web.servlet.config.annotation.WebMvcConfigurationSupport;

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 extends WebMvcConfigurationSupport {

    @Bean

    public Docket createRestApi() {

        return new Docket(DocumentationType.SWAGGER\_2)

                .apiInfo(apiInfo())

                .select()

*//                //为当前包下controller生成API文档*

*//                .apis(RequestHandlerSelectors.basePackage("com.hogeon"))*

*//                //为有@Api注解的Controller生成API文档*

*//                .apis(RequestHandlerSelectors.withClassAnnotation(Api.class))*

*//                //为有@ApiOperation注解的方法生成API文档*

*//                .apis(RequestHandlerSelectors.withMethodAnnotation(ApiOperation.class))*

*//为任何接口生成API文档*

               .apis(RequestHandlerSelectors.any())

               .paths(PathSelectors.any())

                .build();

    }

    private ApiInfo apiInfo() {

        Contact contact = new Contact("hogeon", "", "1981712456@qq.com");

        return new ApiInfoBuilder()

                .title("Swagger文档")

                .description("微信小程序——TEST 接口文档")

                .contact(contact)

                .version("1.0")

                .build();

    }

    @Override

    protected void addResourceHandlers(ResourceHandlerRegistry *registry*) {

*// 解决静态资源无法访问*

        registry.addResourceHandler("/\*\*")

                .addResourceLocations("classpath:/static/");

*// 解决swagger无法访问*

        registry.addResourceHandler("/swagger-ui.html")

                .addResourceLocations("classpath:/META-INF/resources/");

*// 解决swagger的js文件无法访问*

        registry.addResourceHandler("/webjars/\*\*")

                .addResourceLocations("classpath:/META-INF/resources/webjars/");

    }

}

# 5注解

**1、软件开发背景**

（TBD）

**2、软件测试相关补充**

在软件实现中包括编码和测试两个阶段。按照传统的软件工程方法学，编码是在对软件进行了总体设计和详细设计之后进行的，它只不过是把软件设计的结果翻译成用某种程序设计语言书写的程序，因此，程序的质量基本上取决于设计的质量。程序内部的良好文档资料，有规律的数据说明格式，简单清晰的语句构造和输人输出格式等，都对提高程序的可读性有很大作用，也在相当大的程度上改进了程序的可维护性。

目前软件测试仍然是保证软件可靠性的主要手段。测试阶段的根本任务是发现并改正软件中的错误，软件测试是软件开发过程中最艰巨最繁重的任务，大型软件的测试应该分阶段地进行，通常至少分为单元测试、集成测试和验收测试3个基本阶段。设计测试方案是测试阶段的关键技术问题，基本目标是选用最少量的高效测试数据，做到尽可能完善的测试，从而尽可能多地发现软件中的问题。

白盒测式和黑盒测试是软件测试的两类基本方法,这两类方法各有所长，相互补充。通常，在测试过程的早期阶段主要使用白盒方法，而在测试过程的后期阶段主要使用黑盒方法。为了设计出有效的测试方案.软件工程师应该深人理解并坚持运用关于软件测试的基本准则。设计白盒测试方案的技术主要有，逻辑覆盖和控制结构测试；设计黑盒测试方案的技术主要有，等价划分、边界值分析和错误推测。在测试过程中发现的软件错误必须及时改正，这就是调试的任务。为了改正错误，首先必须确定错误的准确位置，这是调试过程中最困难的工作，需要审慎周密的思考和推理。为了改正错误往往需要修正原来的设计，必须通盘考虑统筹兼顾，而不能“头疼医头、脚疼医脚”,应该尽量避免在调试过程中引进新错误。

测试和调试是软件测试阶段中的两个关系非常密切的过程，它们往往交替进行。程序中潜藏的错误的数目，直接决定了软件的可靠性。通过测试可以估算出程序中剩余的错误数。根据测试和调试过程中已经发现和改正的错误数，可以估算软件的平均无故障时间；反之，根据要求达到的软件平均无故障时间，可以估算出应该改正的错误数，从而能够判断测试阶段何时可以结束。

**3、相关术语解释**

**单元测试**：对软件组成进行的测试。其目的是检验软件基本组成单位的正确性。测试对象是软件设计的最小单元：模块，又称为模块测试。

**集成测试**：也成联合测试，联调、组装测试，将程序模块采用适当的集成策略组装起来。

**系统测试**：将软件系统看成一个系统测试。包括对功能、性能以及软件所运行的硬软件环境进行测试。时间大部分在系统测试执行阶段，包括了回归测试和冒烟测试。

**验收测试**：部署软件之前的最后一个测试操作，它是技术测试室的最后一个阶段，也叫做交付测试，验收测试的目的是保证软件的准备就绪，按照项目合同、任务书、双方约定的验收依据文档，向软件的购买者展示该软件的原始的需求。

**α测试**：主要是由一个用户在开发环境进行的测试，也可以是公司内部的用户在模拟实际操作环境下进行的测试。主要的目的是：评价软件产品的FLURPS（即功能、局域化、可使用性、可靠性、性能和支持）。

**β测试**：由软件的最终的用户们在一个或者多个客户场所进行的测试。

**第三方测试**：介于开发方和用户之间的组织测试。

**静态测试**：静态测试值的是不运行程序本身，仅通过分析和检查源程序的语法、结构、过程、接口来检查程序的正确性。对需求规格说明书、软件设计说明书、流程图分析、符号执行来进行找错。

**动态测试**：指的就是运行被测的程序。检查运行结果与预期结果的差异，并分析运行效率、正确性和健壮性的等性能，这种方法主要是由三部分进行组成的：测试用例、执行程序、分析程序运行输出的结果。

**手工测试**：是由人一个一个的输入测试用例，然后观察结果、和机器测试相对应，属于比较原始，大事需要一个一个步骤进行测试。

自动化测试：在预设条件下运行系统或应用程序，评估运行结果、预先条件应该包括正常的条件和异常条件。简单的说自动化测试是把人为驱动的测试行为转化为机器执行的一种过程。

黑盒测试（Black-box-Testing）：黑盒测试也称为功能测试，测试中把被测的软件当成一个黑盒子，不关心盒子的内部结构是什么，指关心软件的输入数据和输出数据。

白盒测试（White-box-Testing）：白盒测试又称结构测试，透明盒测试、逻辑驱动测试或基于代码的测试。白盒值的是打开的盒子，去研究里面的源代码和程序结果。

灰盒测试（White-box-Testing）：灰盒测试是介于白盒测试与黑盒测试之间的一种测试，主要用于集成测试阶段。不仅观念朱输入输出的正确性。同时也关注程序内部的情况。

# 附录

参考资料：

[1]项目计划书模板（GB/T8567-2006）

[2]Spring Boot官网：https://spring.io/projects/spring-boot 2021/11/13

[3]Vue中文官网：<https://cn.vuejs.org/v2/guide/> 2021/11/13

[4]GB+T-8567-2006计算机软件文档编制规范

[5]凯西·施瓦尔贝.IT项目管理（第八版）[M]：机械工业出版社

[6]张海藩，牟永敏.软件工程导论(第6版)[M]北京：清华大学出版社

[7]梁勇. JAVA语言程序设计(第10版)[M]北京：机械工业出版社

[8]王珊，萨师煊. 数据库系统概论(第5版)[M]北京：高等教育出版社

[9]覃征，李旭，王卫红. 软件体系结构(第4版)[M]北京：清华大学出版社

[10]李宁,李战怀.基于黑盒测试的软件测试策略研究与实践[J].计算机应用研究,2009,26(03):923-926.

[11]胡静.浅析黑盒测试与白盒测试[J].衡水学院学报,2008(01):30-32.