# 连接的IP地址为

public static final String BASE\_URL = "http://39.105.43.3:8080/";

# User相关

package com.hebtu.havefun.controller;  
  
import com.alibaba.fastjson.JSON;  
import com.hebtu.havefun.entity.Messages;  
import com.hebtu.havefun.service.UserService;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import javax.annotation.Resource;  
import java.util.List;  
  
/\*\*  
 \* @author PengHuAnZhi  
 \* @createTime 2020/11/19 11:15  
 \* @projectName HaveFun  
 \* @className UserController.java  
 \* @description TODO  
 \*/  
@RestController  
@RequestMapping("user")  
public class UserController {  
  
 @Resource  
 UserService userService;

## 判断是否注册

/\*\*
  
 \* @param phoneNum 电话号码
  
 \* @return 返回"true"或者"false"
  
 \* @description 判断是否注册
  
 \*/
  
 @RequestMapping("/judgeRegistered")
  
 public String judgeRegistered(String phoneNum) {
  
 if (phoneNum == null) {
  
 System.out.println("judgeRegistered Error");
  
 return "ErrorParameter";
  
 }
  
 if (userService.judgeRegistered(phoneNum)) {
  
 return "false";
  
 } else {
  
 return "true";
  
 }
  
 }

## 注册

/\*\*
  
 \* @param phoneNum 手机号码
  
 \* @param password 密码
  
 \* @return 返回"true"或者"false"
  
 \* @description 注册
  
 \*/
  
 @RequestMapping("/register")
  
 public String register(String phoneNum, String password) {
  
 if (phoneNum == null || password == null) {
  
 System.out.println("register Error");
  
 return "ErrorParameter";
  
 }
  
 if (userService.register(phoneNum, password)) {
  
 return "true";
  
 } else {
  
 return "false";
  
 }
  
 }

## 登录

/\*\*
  
 \* @param phoneNum 手机号
  
 \* @param password 密码
  
 \* @return 返回的是user的对象Json串
  
 \* @description 登录
  
 \*/
  
 @RequestMapping("/login")
  
 public String login(String phoneNum, String password) {
  
 if (phoneNum == null || password == null) {
  
 System.out.println("login Error");
  
 return "ErrorParameter";
  
 }
  
 String user = userService.login(phoneNum, password);
  
 return "".equals(user) ? "" : user;
  
 }

## 修改密码

/\*\*
  
 \* @param phoneNum 电话号码
  
 \* @param password 密码
  
 \* @return 返回"true"或"false"
  
 \* @description 修改密码
  
 \*/
  
 @RequestMapping("/modifyPassword")
  
 public String modifyPassword(String phoneNum, String password) {
  
 if (phoneNum == null || password == null) {
  
 System.out.println("modifyPassword Error");
  
 return "ErrorParameter";
  
 }
  
 if (userService.modifyPassword(phoneNum, password)) {
  
 return "true";
  
 } else {
  
 return "false";
  
 }
  
 }

## 报名活动

/\*\*
  
 \* @param activityId 活动id
  
 \* @param id 用户id,注意不是getUserId,是getId
  
 \* @return 返回报名成功"true"或者已经报名了"exists"
  
 \* @description 报名活动
  
 \*/
  
 @RequestMapping("/enrollActivity")
  
 public String enrollActivity(Integer activityId, Integer id) {
  
 if (activityId == null || id == null) {
  
 System.out.println("enrollActivity Error");
  
 return "ErrorParameter";
  
 }
  
 return userService.enrollActivity(activityId, id);
  
 }

## 取消报名活动

/\*\*
  
 \* @param activityId 活动id
  
 \* @param id 用户id,注意不是getUserId,是getId
  
 \* @return 返回取消报名成功"true"或者失败"false"
  
 \* @description 取消报名活动
  
 \*/
  
 @RequestMapping("/cancelEnrollActivity")
  
 public String cancelEnrollActivity(Integer activityId, Integer id) {
  
 if (activityId == null || id == null) {
  
 System.out.println("enrollActivity Error");
  
 return "ErrorParameter";
  
 }
  
 return "success".equals(userService.cancelEnrollActivity(activityId, id)) ? "true" : "false";
  
 }

## 收藏或者取消收藏

/\*\*
  
 \* @param activityId 活动的id
  
 \* @param id 用户id,注意不是getUserId,是getId
  
 \* @param collect 是否注册，发送给我的是"false"或者"true"
  
 \* @return
  
 \* @description 收藏或者取消收藏
  
 \*/
  
 @RequestMapping("/changeCollectActivity")
  
 public String changeCollectActivity(Integer activityId, Integer id, String collect) {
  
 if (activityId == null || id == null || collect == null) {
  
 System.out.println("changeCollectActivity Error");
  
 return "ErrorParameter";
  
 }
  
 boolean tag = Boolean.parseBoolean(collect);
  
 boolean flag = userService.changeCollectActivity(activityId, id, tag);
  
 return flag ? "true" : "false";
  
 }

## 修改个性签名

/\*\*
  
 \* @param id 用户id,注意不是getUserId,是getIds
  
 \* @param personalSignature 新的个性签名
  
 \* @return 返回"true"或者"false"
  
 \* @description 修改个性签名
  
 \*/
  
 @RequestMapping("/modifyPersonalSignature")
  
 public String modifyPersonalSignature(Integer id, String personalSignature) {
  
 if (id == null || personalSignature == null) {
  
 System.out.println("modifyPersonalSignature Error");
  
 return "ErrorParameter";
  
 }
  
 return userService.modifyPersonalSignature(id, personalSignature) ? "true" : "false";
  
 }

## 发送消息

/\*\*
  
 \* @param sender 发送者id,注意不是getUserId,是getId
  
 \* @param receiver 接收者id,注意不是getUserId,是getId
  
 \* @param msg 消息内容
  
 \* @return 返回这个消息对象Messages,出错返回"false"
  
 \* @description 发送消息，这边插入数据库
  
 \*/
  
 @RequestMapping("/addMsg")
  
 public String addMsg(Integer sender, Integer receiver, String msg) {
  
 if (sender == null || receiver == null || msg == null) {
  
 System.out.println("addMsg Error");
  
 return "ErrorParameter";
  
 }
  
 Messages messages = userService.addMsg(sender, receiver, msg);
  
 return messages != null ? JSON.toJSONString(messages) : "false";
  
 }

## 获取消息列表，分页显示

/\*\*
  
 \* @param sender 发送者id,注意不是getUserId,是getId
  
 \* @param receiver 接收者id,注意不是getUserId,是getId
  
 \* @param pageNum 当前页码
  
 \* @param pageSize 页面大小
  
 \* @return 返回List<Messages>集合Json,没有消息则为empty
  
 \* @description 获取消息列表，分页显示
  
 \*/
  
 @RequestMapping("/getMsg")
  
 public String getMsg(Integer sender, Integer receiver, Integer pageNum, Integer pageSize) {
  
 if (sender == null || receiver == null || pageNum == null || pageSize == null) {
  
 System.out.println("getMsg Error");
  
 return "ErrorParameter";
  
 }
  
 List<Messages> messagesList = userService.getMsg(sender, receiver, pageNum, pageSize);
  
 return messagesList != null ? JSON.toJSONString(messagesList) : "empty";
  
 }

## 获取两个人之间总消息数量

/\*\*
  
 \* @param sender 发送者id,注意不是getUserId,是getId
  
 \* @param receiver 接收者id,注意不是getUserId,是getId
  
 \* @return 返回的是一个数量的string类型
  
 \* @description 获取两个人之间总消息数量
  
 \*/
  
 @RequestMapping("/getMsgNum")
  
 public String getMsgNum(Integer sender, Integer receiver) {
  
 if (sender == null || receiver == null) {
  
 System.out.println("getMsgNum Error");
  
 return "ErrorParameter";
  
 }
  
 Long count = userService.getMsgNum(sender, receiver);
  
 return count + "";
  
 }

## 刷新消息

/\*\*
  
 \* @param otherId 对方的id，注意不是getUserId,是getId
  
 \* @param mineId 自己的id，注意不是getUserId,是getId
  
 \* @return 返回依然是List<Messages>集合Json,没有新消息则是"empty"
  
 \* @description 每隔若干秒刷新消息
  
 \*/
  
 @RequestMapping("/freshMsg")
  
 public String freshMsg(Integer otherId, Integer mineId) {
  
 if (otherId == null || mineId == null) {
  
 System.out.println("freshMsg Error");
  
 return "ErrorParameter";
  
 }
  
 List<Messages> messagesList = userService.freshMsg(otherId, mineId);
  
 return messagesList.size() != 0 ? JSON.toJSONString(messagesList) : "empty";
  
 }

## 判断是否关注

/\*\*
  
 \* @param followId 当前用户的id，即关注者id，注意不是getUserId,是getId
  
 \* @param followedId 对方id，即被关注者id，注意不是getUserId,是getId
  
 \* @return 返回是否关注 "true"或者"false"
  
 \* @description 判断是否关注
  
 \*/
  
 @RequestMapping("/judgeFollow")
  
 public String judgeFollow(Integer followId, Integer followedId) {
  
 if (followId == null || followedId == null) {
  
 System.out.println("judgeFollow Error");
  
 return "ErrorParameter";
  
 }
  
 return userService.judgeFollow(followId, followedId) ? "true" : "false";
  
 }

## 关注用户

/\*\*
  
 \* @param followId 当前用户的id，即关注者id，注意不是getUserId,是getId
  
 \* @param followedId 对方id，即被关注者id，注意不是getUserId,是getId
  
 \* @return 返回关注成功还是失败true或者false
  
 \* @description 关注用户
  
 \*/
  
 @RequestMapping("/followUser")
  
 public String followUser(Integer followId, Integer followedId) {
  
 if (followId == null || followedId == null) {
  
 System.out.println("followUser Error");
  
 return "ErrorParameter";
  
 }
  
 return userService.followUser(followId, followedId) ? "true" : "false";
  
 }

## 取消关注用户

/\*\*
  
 \* @param followId 当前用户的id，即关注者id，注意不是getUserId,是getId
  
 \* @param followedId 对方id，即被关注者id，注意不是getUserId,是getId
  
 \* @return 返回取消关注成功还是失败true或者false
  
 \* @description 取消关注用户
  
 \*/
  
 @RequestMapping("/unFollowUser")
  
 public String unFollowUser(Integer followId, Integer followedId) {
  
 if (followId == null || followedId == null) {
  
 System.out.println("unFollowUser Error");
  
 return "ErrorParameter";
  
 }
  
 return userService.unFollowUser(followId, followedId) ? "true" : "false";
  
 }

## 获取粉丝数量

/\*\*
  
 \* @param id 当前用户的id
  
 \* @return 返回粉丝数量
  
 \* @description 获取用户被多少人关注，即粉丝数量
  
 \*/
  
 @RequestMapping("/getFollowedCount")
  
 public String getFollowedCount(Integer id) {
  
 if (id == null) {
  
 System.out.println("getFollowedCount Error");
  
 return "ErrorParameter";
  
 }
  
 Long count = userService.getFollowedCount(id);
  
 return count + "";
  
 }

## 获取关注人的数量

/\*\*
  
 \* @param id 当前用户的id
  
 \* @return 返回关注人数
  
 \* @description 获取用户关注了多少人
  
 \*/
  
 @RequestMapping("/getFollowCount")
  
 public String getFollowCount(Integer id) {
  
 if (id == null) {
  
 System.out.println("getFollowCount Error");
  
 return "ErrorParameter";
  
 }
  
 Long count = userService.getFollowCount(id);
  
 return count + "";
  
 }

## 获取用户信息

/\*\*
  
 \* @param id 用户的id
  
 \* @return 返回user对象的JSON串,没找到对应的用户则是"false"
  
 \* @description 根据id获取用户的信息，用于点击某个用户头像进入个人主页
  
 \*/
  
 @RequestMapping("/getUser")
  
 public String getUser(Integer id) {
  
 User user = userService.getUser(id);
  
 if (id == null) {
  
 System.out.println("getUser Error");
  
 return "ErrorParameter";
  
 }
  
 return user != null ? JSON.toJSONString(user) : "false";
  
 }

## 获取和用户交流过的消息列表

/\*\*
  
 \* @param id 当前user的id
  
 \* @return 返回一个List<Map < Messages, Integer>>,messages的数据表示用于展示聊天列表中最新的一条消
  
 \* 息，Integer的数据为未读消息数量，业务逻辑为我是接收者，发送者为对方，且消息未读,没有聊天信息则是"empty"
  
 \* @description 获取和用户交流过的消息列表
  
 \*/
  
 @RequestMapping("/getMessageList")
  
 public String getMessageList(Integer id) {
  
 if (id == null) {
  
 System.out.println("getMessageList Error");
  
 return "ErrorParameter";
  
 }
  
 List<Map<Messages, Integer>> messagesList = userService.getMessageList(id);
  
 return messagesList.size() != 0 ? JSON.toJSONString(messagesList) : "empty";
  
 }

## 获取当前用户的关注列表

/\*\*
  
 \* @param id 当前用户的id
  
 \* @return 返回一个用户列表的Json串，如果为空则返回empty
  
 \* @description 获取当前用户的关注列表
  
 \*/
  
 @RequestMapping("/getFollowUsers")
  
 public String getFollowUsers(Integer id) {
  
 List<User> userList = userService.getFollowUsers(id);
  
 return userList.size() != 0 ? JSON.toJSONString(userList) : "empty";
  
 }

## 获取当前用户的粉丝列表

/\*\*
  
 \* @param id 当前用户的id
  
 \* @return 返回一个用户列表的Json串，如果为空则返回empty
  
 \* @description 获取当前用户的粉丝列表
  
 \*/
  
 @RequestMapping("/getFollowedUsers")
  
 public String getFollowedUsers(Integer id) {
  
 List<User> userList = userService.getFollowedUsers(id);
  
 return userList.size() != 0 ? JSON.toJSONString(userList) : "empty";
  
 }

## 更改用户头像

/\*\*
  
 \* @param headPortrait 使用OkHttp3传输图片，类型为MultipartFile，名称为headPortrait
  
 \* @param id 用户id
  
 \* @return 返回修改成功还是失败"true"或者"false"
  
 \* @description 更改用户头像
  
 \*/
  
 @RequestMapping("/modifyUserHeadPortrait")
  
 public String modifyUserHeadPortrait(MultipartFile headPortrait, Integer id) {
  
 return userService.modifyUserHeadPortrait(headPortrait, id) ? "true" : "false";
  
 }

## 更改用户昵称

/\*\*
  
 \* @param userName 用户新昵称，请求参数名称为userName
  
 \* @param id 用户id
  
 \* @return 返回修改成功还是失败"true"或者"false"
  
 \* @description 更改用户昵称
  
 \*/
  
 @RequestMapping("/modifyUserName")
  
 public String modifyUserName(String userName, Integer id) {
  
 return userService.modifyUserName(userName, id) ? "true" : "false";
  
 }

## 更改用户性别

/\*\*
  
 \* @param sex 用户性别字段，1或者0
  
 \* @param id 用户id
  
 \* @return 返回修改成功还是失败"true"或者"false"
  
 \* @description 修改用户性别
  
 \*/
  
 @RequestMapping("/modifyUserSex")
  
 public String modifyUserSex(Integer sex, Integer id) {
  
 return userService.modifyUserSex(sex, id) ? "true" : "false";
  
 }
  
}

# Activity相关

package com.hebtu.havefun.controller;  
  
import com.alibaba.fastjson.JSON;  
import com.google.gson.Gson;  
import com.hebtu.havefun.entity.activity.Activity;  
import com.hebtu.havefun.entity.activity.ActivityDetail;  
import com.hebtu.havefun.s ervice.ActivityService;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
  
import javax.annotation.Resource;  
import java.util.List;  
  
/\*\*  
 \* @author PengHuAnZhi  
 \* @createTime 2020/11/21 15:23  
 \* @projectName HaveFun  
 \* @className ActivityController.java  
 \* @description TODO  
 \*/  
@RestController  
@RequestMapping("activity")  
public class ActivityController {  
  
 @Resource  
 ActivityService activityService;

## 获取轮播图在服务器上的地址

/\*\*
  
 \* @return 返回的是一个json串，解析出来是一个list<String>,字符串分别是图片在服务器上的地址
  
 \* @description 获取轮播图在服务器上的地址
  
 \*/
  
 @RequestMapping("/getRotationChartPictures")
  
 public String getRotationChartPictures() {
  
 return new Gson().toJson(activityService.getRotationChartPictures());
  
 }

## 获取活动列表

/\*\*
  
 \* @param activityKind 热门或者近期，1或者0
  
 \* @param pageNum 页码
  
 \* @param pageSize 页大小
  
 \* @return List<Activity>集合的JSON串,没有活动则是"empty"
  
 \* @description 获取活动列表
  
 \*/
  
 @RequestMapping("/getActivityList")
  
 public String getActivityList(Integer activityKind, Integer pageNum, Integer pageSize) {
  
 if (activityKind == null || pageNum == null || pageSize == null) {
  
 System.out.println("getActivityList Error");
  
 return "ErrorParameter";
  
 }
  
 String activities = activityService.getActivityList(activityKind, pageNum, pageSize);
  
 return "empty".equals(activities) ? "" : activities;
  
 }

## 获取活动详细信息

/\*\*
  
 \* @param activityId 活动的id
  
 \* @return 返回ActivityDetail的json串,如果没有找到对应的活动详细信息则返回空串""
  
 \* @description 获取活动详细信息
  
 \*/
  
 @RequestMapping("/getActivityDetail")
  
 public String getActivityDetail(Integer activityId) {
  
 if (activityId == null) {
  
 System.out.println("getActivityDetail Error");
  
 return "ErrorParameter";
  
 }
  
 String activityDetail = activityService.getActivityDetail(activityId);
  
 return activityDetail != null ? activityDetail : "";
  
 }

## 判断是否收藏

/\*\*
  
 \* @param id 用户id,注意不是getUserId,是getId
  
 \* @param activityId 活动的id
  
 \* @return 返回true或者false的字符串
  
 \* @description 判断是否收藏
  
 \*/
  
 @RequestMapping("/judgeCollected")
  
 public String judgeCollected(Integer id, Integer activityId) {
  
 if (id == null || activityId == null) {
  
 System.out.println("judgeCollected Error");
  
 return "ErrorParameter";
  
 }
  
 boolean flag = activityService.judgeCollected(id, activityId);
  
 return flag ? "true" : "false";
  
 }

## 获取收藏的活动列表

/\*\*
  
 \* @param id 用户id,注意不是getUserId,是getId
  
 \* @param pageNum 页码
  
 \* @param pageSize 页大小
  
 \* @return 返回List<UserCollectActivity>集合,如果没有就返回"empty"
  
 \* @description 获取收藏的活动列表
  
 \*/
  
 @RequestMapping("/getCollectedActivities")
  
 public String getCollectedActivities(Integer id, Integer pageNum, Integer pageSize) {
  
 if (id == null || pageNum == null || pageSize == null) {
  
 System.out.println("getCollectedActivities Error");
  
 return "ErrorParameter";
  
 }
  
 return activityService.getCollectedActivities(id, pageNum, pageSize);
  
 }

## 添加活动

/\*\*
  
 \* @param files 客户端传递若干MultipartFile文件，要求请求的参数名字都
  
 \* 是file（就是要重复），这边接收就是自动加入List<MultipartFile>中
  
 \* @param activityDetailJson 客户端将封装好的ActivityDetail类转换为Json串发送过来，
  
 \* 注意ActivityDetail里面的Activity类也得把数据都封装好，
  
 \* 添加操作，不需要设置id值，因为数据库id是自增的
  
 \* @return 返回一个添加成功或者失败"true"或者"false"
  
 \* @description 添加活动，接受一个ActivityDetail类的Json串数据，且参数名称为activityJson
  
 \*/
  
 @RequestMapping("/addActivity")
  
 public String addActivity(@RequestParam("file") List<MultipartFile> files, String activityDetailJson) {
  
 if (activityDetailJson == null) {
  
 System.out.println("addActivity Error");
  
 return "ErrorParameter";
  
 }
  
 return activityService.addActivity(files, activityDetailJson) ? "true" : "false";
  
 }

## 获取报名的活动列表

/\*\*
  
 \* @param id 用户id,注意不是getUserId,是getId
  
 \* @param pageNum 页码
  
 \* @param pageSize 页大小
  
 \* @return 返回List<UserCollectActivity>集合,如果没有就返回"empty"
  
 \* @description 获取报名的活动列表
  
 \*/
  
 @RequestMapping("/getEnterActivities")
  
 public String getEnterActivities(Integer id, Integer pageNum, Integer pageSize) {
  
 if (id == null || pageNum == null || pageSize == null) {
  
 System.out.println("getEnterActivities Error");
  
 return "ErrorParameter";
  
 }
  
 return activityService.getEnterActivities(id, pageNum, pageSize);
  
 }

## 获取发布的活动列表

/\*\*
  
 \* @param id 用户id,注意不是getUserId,是getId
  
 \* @param pageNum 页码
  
 \* @param pageSize 页大小
  
 \* @return 返回List<UserCollectActivity>集合,如果没有就返回"empty"
  
 \* @description 获取发布的活动列表
  
 \*/
  
 @RequestMapping("/getPublishActivities")
  
 public String getPublishActivities(Integer id, Integer pageNum, Integer pageSize) {
  
 if (id == null || pageNum == null || pageSize == null) {
  
 System.out.println("getPublishActivities Error");
  
 return "ErrorParameter";
  
 }
  
 return activityService.getPublishActivities(id, pageNum, pageSize);
  
 }

## 根据时间筛选活动

/\*\*
  
 \* @param howManyDays 参数为近多少天，用于筛选符合要求的时间内开始的活动
  
 \* @return 返回一个List<Activity>集合,如果没有就返回"empty"
  
 \* @description 根据时间筛选活动
  
 \*/
  
 @RequestMapping("/screenTimeActivities")
  
 public String screenTimeActivities(Integer howManyDays, Integer pageNum, Integer pageSize) {
  
 String activityList = activityService.screenTimeActivities(howManyDays, pageNum, pageSize);
  
 return "empty".equals(activityList) ? JSON.toJSONString(activityList) : "empty";
  
 }

## 根据花费筛选活动

/\*\*
  
 \* @param lowCost 价格区间的小值
  
 \* @param highCost 价格区间的大值
  
 \* @return 返回一个List<Activity>集合,如果没有就返回"empty"
  
 \* @description 根据花费筛选活动
  
 \*/
  
 @RequestMapping("/screenTypeActivities")
  
 public String screenTypeActivities(Integer lowCost, Integer highCost, Integer pageNum, Integer pageSize) {
  
 String activityList = activityService.screenCostActivities(lowCost, highCost, pageNum, pageSize);
  
 return "empty".equals(activityList) ? JSON.toJSONString(activityList) : "empty";
  
 }

## 根据活动种类筛选活动

/\*\*
  
 \* @param tag tag为活动种类的最小划分，即小类而不是大类，值为种类的id
  
 \* @return 返回一个List<Activity>集合,如果没有就返回"empty"
  
 \* @description 根据活动种类筛选活动
  
 \*/
  
 @RequestMapping("/screenCostActivities")
  
 public String screenCostActivities(Integer tag, Integer pageNum, Integer pageSize) {
  
 String activityList = activityService.screenTypeActivities(tag, pageNum, pageSize);
  
 return "empty".equals(activityList) ? JSON.toJSONString(activityList) : "empty";
  
 }