

连接的IP地址为

```
1 public static final String BASE_URL = "http://39.105.43.3:8080/";
```

User相关

```
1 package com.hebtu.havefun.controller;
2
3 import com.alibaba.fastjson.JSON;
4 import com.hebtu.havefun.entity.Messages;
5 import com.hebtu.havefun.service.UserService;
6 import org.springframework.web.bind.annotation.RequestMapping;
7 import org.springframework.web.bind.annotation.RestController;
8
9 import javax.annotation.Resource;
10 import java.util.List;
11
12 /**
13  * @author PengHuAnZhi
14  * @createTime 2020/11/19 11:15
15  * @projectName HaveFun
16  * @className UserController.java
17  * @description TODO
18  */
19 @RestController
20 @RequestMapping("user")
21 public class UserController {
22
23     @Resource
24     UserService userService;
```

判断是否注册

```
1 /**
2  * @param phoneNum 电话号码
3  * @return 返回"true"或者"false"
4  * @description 判断是否注册
5  */
6 @RequestMapping("/judgeRegistered")
7 public String judgeRegistered(String phoneNum) {
8     if (phoneNum == null) {
9         System.out.println("judgeRegistered Error");
10        return "ErrorParameter";
11    }
12    if (userService.judgeRegistered(phoneNum)) {
13        return "false";
14    } else {
15        return "true";
16    }
17 }
```

注册

```
1  /**
2   * @param phoneNum 手机号码
3   * @param password 密码
4   * @return 返回"true"或者"false"
5   * @description 注册
6   */
7  @RequestMapping("/register")
8  public String register(String phoneNum, String password) {
9      if (phoneNum == null || password == null) {
10         System.out.println("register Error");
11         return "ErrorParameter";
12     }
13     if (userService.register(phoneNum, password)) {
14         return "true";
15     } else {
16         return "false";
17     }
18 }
```

登录

```
1  /**
2   * @param phoneNum 手机号
3   * @param password 密码
4   * @return 返回的是user的对象Json串
5   * @description 登录
6   */
7  @RequestMapping("/login")
8  public String login(String phoneNum, String password) {
9      if (phoneNum == null || password == null) {
10         System.out.println("login Error");
11         return "ErrorParameter";
12     }
13     String user = userService.login(phoneNum, password);
14     return "".equals(user) ? "" : user;
15 }
16 }
```

修改密码

```
1  /**
2   * @param phoneNum 电话号码
3   * @param password 密码
4   * @return 返回"true"或"false"
5   * @description 修改密码
6   */
7  @RequestMapping("/modifyPassword")
8  public String modifyPassword(String phoneNum, String password) {
9      if (phoneNum == null || password == null) {
10         System.out.println("modifyPassword Error");
11         return "ErrorParameter";
12     }
13     if (userService.modifyPassword(phoneNum, password)) {
```

```

14         return "true";
15     } else {
16         return "false";
17     }
18 }

```

报名活动

```

1  /**
2   * @param activityId 活动id
3   * @param id          用户id,注意不是getUserId,是getId
4   * @return 返回报名成功"true"或者已经报名了"exists"
5   * @description 报名活动
6   */
7  @RequestMapping("/enrollActivity")
8  public String enrollActivity(Integer activityId, Integer id) {
9      if (activityId == null || id == null) {
10         System.out.println("enrollActivity Error");
11         return "ErrorParameter";
12     }
13     return userService.enrollActivity(activityId, id);
14 }

```

修改个性签名

```

1  /**
2   * @param id          用户id,注意不是getUserId,是getIds
3   * @param personalSignature 新的个性签名
4   * @return 返回"true"或者"false"
5   * @description 修改个性签名
6   */
7  /**
8   * @RequestMapping("/modifyPersonalSignature")
9   * public String modifyPersonalSignature(Integer id, String
personalSignature) {
10     if (id == null || personalSignature == null) {
11         System.out.println("modifyPersonalSignature Error");
12         return "ErrorParameter";
13     }
14     return userService.modifyPersonalSignature(id, personalSignature) ?
"true" : "false";
15 }

```

发送消息

```

1  /**
2   * @param sender 发送者id,注意不是getUserId,是getId
3   * @param receiver 接收者id,注意不是getUserId,是getId
4   * @param msg      消息内容
5   * @return 返回这个消息对象Messages
6   * @description 发送消息, 这边插入数据库
7   */
8  @RequestMapping("/addMsg")
9  public String addMsg(Integer sender, Integer receiver, String msg) {

```

```

10         if (sender == null || receiver == null || msg == null) {
11             System.out.println("addMsg Error");
12             return "ErrorParameter";
13         }
14         Messages messages = userService.addMsg(sender, receiver, msg);
15         return messages != null ? JSON.toJSONString(messages) : "false";
16     }

```

获取消息列表，分页显示

```

1  /**
2   * @param sender 发送者id,注意不是getUserId,是getId
3   * @param receiver 接收者id,注意不是getUserId,是getId
4   * @param pageNum 当前页码
5   * @param pageSize 页面大小
6   * @return 返回List<Messages>集合Json
7   * @description 获取消息列表，分页显示
8   */
9   @RequestMapping("/getMsg")
10   public String getMsg(Integer sender, Integer receiver, Integer pageNum,
11   Integer pageSize) {
12       if (sender == null || receiver == null || pageNum == null ||
13       pageSize == null) {
14           System.out.println("getMsg Error");
15           return "ErrorParameter";
16       }
17       List<Messages> messagesList = userService.getMsg(sender, receiver,
18       pageNum, pageSize);
19       return messagesList != null ? JSON.toJSONString(messagesList) :
20       "empty";
21   }

```

获取两个人之间总消息数量

```

1  /**
2   * @param sender 发送者id,注意不是getUserId,是getId
3   * @param receiver 接收者id,注意不是getUserId,是getId
4   * @return 返回的是一个数量的string类型
5   * @description 获取两个人之间总消息数量
6   */
7   @RequestMapping("/getMsgNum")
8   public String getMsgNum(Integer sender, Integer receiver) {
9       if (sender == null || receiver == null) {
10           System.out.println("getMsgNum Error");
11           return "ErrorParameter";
12       }
13       return userService.getMsgNum(sender, receiver) + "";
14   }

```

刷新消息

```

1  /**
2   * @param otherId 对方的id, 注意不是getUserId,是getId

```

```

4      * @param mineId 自己的id, 注意不是getUserId,是getId
5      * @return 返回依然是List<Messages>集合Json
6      * @description 每隔若干秒刷新消息
7      */
8      @RequestMapping("/freshMsg")
9      public String freshMsg(Integer otherId, Integer mineId) {
10         if (otherId == null || mineId == null) {
11             System.out.println("freshMsg Error");
12             return "ErrorParameter";
13         }
14         List<Messages> messagesList = userService.freshMsg(otherId, mineId);
15         return messagesList != null ? JSON.toJSONString(messagesList) :
"empty";
16     }

```

判断是否关注

```

1      /**
2      * @description 判断是否关注
3      * @param followId 当前用户的id, 即关注者id, 注意不是getUserId,是getId
4      * @param followedId 对方id, 即被关注者id, 注意不是getUserId,是getId
5      * @return 返回是否关注 "true"或者"false"
6      */
7      @RequestMapping("/judgeFollow")
8      public String judgeFollow(Integer followId, Integer followedId) {
9         if (followId == null || followedId == null) {
10             System.out.println("judgeFollow Error");
11             return "ErrorParameter";
12         }
13         return userService.judgeFollow(followId, followedId) ? "true" :
"false";
14     }

```

关注用户

```

1      /**
2      * @param followId 当前用户的id, 即关注者id, 注意不是getUserId,是getId
3      * @param followedId 对方id, 即被关注者id, 注意不是getUserId,是getId
4      * @return 返回关注成功还是失败true或者false
5      * @description 关注用户
6      */
7      @RequestMapping("/followUser")
8      public String followUser(Integer followId, Integer followedId) {
9         if (followId == null || followedId == null) {
10             System.out.println("followUser Error");
11             return "ErrorParameter";
12         }
13         return userService.followUser(followId, followedId) ? "true" :
"false";
14     }

```

取消关注用户

```

1      /**

```

```

2      * @param followId 当前用户的id, 即关注者id, 注意不是getUserId,是getId
3      * @param followedId 对方id, 即被关注者id, 注意不是getUserId,是getId
4      * @return 返回取消关注成功还是失败true或者false
5      * @description 取消关注用户
6      */
7      @RequestMapping("/unFollowUser")
8      public String unFollowUser(Integer followId, Integer followedId) {
9          if (followId == null || followedId == null) {
10             System.out.println("unFollowUser Error");
11             return "ErrorParameter";
12         }
13         return userService.unFollowUser(followId, followedId) ? "true" :
"false";
14     }

```

获取粉丝数量

```

1      /**
2      * @description 获取用户被多少人关注, 即粉丝数量
3      * @param id 当前用户的id
4      * @return 返回粉丝数量
5      */
6      @RequestMapping("/getFollowedCount")
7      public String getFollowedCount(Integer id) {
8          if (id == null) {
9             System.out.println("getFollowedCount Error");
10             return "ErrorParameter";
11         }
12         Long count = userService.getFollowedCount(id);
13         return count + "";
14     }

```

获取关注人的数量

```

1      /**
2      * @description 获取用户关注了多少人
3      * @param id 当前用户的id
4      * @return 返回关注人数
5      */
6      @RequestMapping("/getFollowCount")
7      public String getFollowCount(Integer id) {
8          if (id == null) {
9             System.out.println("getFollowCount Error");
10             return "ErrorParameter";
11         }
12         Long count = userService.getFollowCount(id);
13         return count + "";
14     }

```

获取用户信息

```

1      /**
2      * @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
3      * @param id 用户的id

```

```

4      * @return 返回user对象的JSON串
5      */
6      @RequestMapping("/getUser")
7      public String getUser(Integer id){
8          User user = userService.getUser(id);
9          if(id == null){
10             System.out.println("getUser Error");
11             return "ErrorParameter";
12         }
13         return user != null ? JSON.toJSONString(user) : "false";
14     }

```

获取和用户交流过的消息列表

```

1      /**
2      * @param id 当前user的id
3      * @return 返回一个List<Map < Messages, Integer>>,messages的数据表示用于展示
聊天列表中
4      * 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
5      * @description 获取和用户交流过的消息列表
6      */
7      @RequestMapping("/getMessageList")
8      public String getMessageList(Integer id) {
9          if (id == null) {
10             System.out.println("getMessageList Error");
11             return "ErrorParameter";
12         }
13         List<Map<Messages, Integer>> messagesList =
userService.getMessageList(id);
14         return messagesList.size() != 0 ? JSON.toJSONString(messagesList) :
"empty";
15     }

```

获取当前用户的关注列表

```

1      /**
2      * @param id 当前用户的id
3      * @return 返回一个用户列表的Json串，如果为空则返回empty
4      * @description 获取当前用户的关注列表
5      */
6      @RequestMapping("/getFollowUsers")
7      public String getFollowUsers(Integer id) {
8          List<User> userList = userService.getFollowUsers(id);
9          return userList.size() != 0 ? JSON.toJSONString(userList) : "empty";
10     }

```

获取当前用户的粉丝列表

```

1  /**
2   * @param id 当前用户的id
3   * @return 返回一个用户列表的Json串，如果为空则返回empty
4   * @description 获取当前用户的粉丝列表
5   */
6  @RequestMapping("/getFollowedUsers")
7  public String getFollowedUsers(Integer id) {
8      List<User> userList = userService.getFollowedUsers(id);
9      return userList.size() != 0 ? JSON.toJSONString(userList) : "empty";
10 }

```

更改用户头像

```

1  /**
2   * @param headPortrait 使用OkHttp3传输图片，类型为MultipartFile，名称为
   headPortrait
3   * @param id 用户id
4   * @return 返回修改成功还是失败
5   * @description 更改用户头像
6   */
7  @RequestMapping("/modifyUserHeadPortrait")
8  public String modifyUserHeadPortrait(MultipartFile headPortrait, Integer
   id) {
9      return userService.modifyUserHeadPortrait(headPortrait, id) ? "true"
   : "false";
10 }

```

更改用户昵称

```

1  /**
2   * @param userName 用户新昵称，请求参数名称为userName
3   * @param id 用户id
4   * @return 返回修改成功还是失败
5   * @description 更改用户昵称
6   */
7  @RequestMapping("/modifyUserName")
8  public String modifyUserName(String userName, Integer id) {
9      return userService.modifyUserName(userName, id) ? "true" : "false";
10 }
11

```

更改用户性别


```

1      /**
2       * @param sex 用户性别字段, 1或者0
3       * @param id 用户id
4       * @return 返回修改成功还是失败
5       * @description 修改用户性别
6       */
7      @RequestMapping("/modifyUserSex")
8      public String modifyUserSex(Integer sex, Integer id) {
9          return userService.modifyUserSex(sex, id) ? "true" : "false";
10     }
11 }

```

Activity相关

```

1 package com.hebtu.havefun.controller;
2
3 import com.alibaba.fastjson.JSON;
4 import com.google.gson.Gson;
5 import com.hebtu.havefun.entity.activity.Activity;
6 import com.hebtu.havefun.entity.activity.ActivityDetail;
7 import com.hebtu.havefun.service.ActivityService;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RestController;
10
11 import javax.annotation.Resource;
12 import java.util.List;
13
14 /**
15  * @author PengHuAnZhi
16  * @createTime 2020/11/21 15:23
17  * @projectName HaveFun
18  * @className ActivityController.java
19  * @description TODO
20  */
21 @RestController
22 @RequestMapping("activity")
23 public class ActivityController {
24
25     @Resource
26     ActivityService activityService;

```

获取轮播图在服务器上的地址

```

1      /**
2       * @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
3       * @description 获取轮播图在服务器上的地址
4       */
5      @RequestMapping("/getRotationChartPictures")
6      public String getRotationChartPictures() {
7          return new Gson().toJson(activityService.getRotationChartPictures());
8      }

```

获取活动列表

```

1  /**
2   * @param activityKind 热门或者近期, 1或者0
3   * @param pageNum      页码
4   * @param pageSize     页大小
5   * @return List<Activity>集合
6   * @description 获取活动列表
7   */
8  @RequestMapping("/getActivityList")
9  public String getActivityList(Integer activityKind, Integer pageNum,
10 Integer pageSize) {
11      if (activityKind == null || pageNum == null || pageSize == null) {
12          System.out.println("getActivityList Error");
13          return "ErrorParameter";
14      }
15      List<Activity> activities =
16      activityService.getActivityList(activityKind, pageNum, pageSize);
17      return activities != null ? JSON.toJSONString(activities) : "";
18  }

```

获取活动详细信息

```

1  /**
2   * @param activityId 活动的id
3   * @return 返回ActivityDetail的json串
4   * @description 获取活动详细信息
5   */
6  @RequestMapping("/getActivityDetail")
7  public String getActivityDetail(Integer activityId) {
8      if (activityId == null) {
9          System.out.println("getActivityDetail Error");
10         return "ErrorParameter";
11     }
12     ActivityDetail activityDetail =
13     activityService.getActivityDetail(activityId);
14     return activityDetail != null ? JSON.toJSONString(activityDetail) :
15     "";
16 }

```

判断是否收藏

```

1  /**
2   * @param id          用户id,注意不是getUserId,是getId
3   * @param activityId 活动的id
4   * @return 返回true或者false的字符串
5   * @description 判断是否收藏
6   */
7  @RequestMapping("/judgeCollected")
8  public String judgeCollected(String id, String activityId) {
9      if (id == null || activityId == null) {
10         System.out.println("judgeCollected Error");
11         return "ErrorParameter";
12     }
13     boolean flag = activityService.judgeCollected(id, activityId);
14     return flag ? "true" : "false";
15 }

```

收藏或者取消收藏

```
1  /**
2   * @param activityId 活动的id
3   * @param id 用户id,注意不是getUserId,是getId
4   * @param collect 是否注册,发送给我的是"false"或者"true"
5   * @return
6   * @description 收藏或者取消收藏
7   */
8  @RequestMapping("/changeCollectActivity")
9  public String changeCollectActivity(Integer activityId, Integer id,
10 String collect) {
11      if (activityId == null || id == null || collect == null) {
12          System.out.println("changeCollectActivity Error");
13          return "ErrorParameter";
14      }
15      boolean tag = Boolean.parseBoolean(collect);
16      boolean flag = activityService.changeCollectActivity(activityId, id,
17 tag);
18      return flag ? "true" : "false";
19  }
```

获取收藏的活动列表

```
1  /**
2   * @param id 用户id,注意不是getUserId,是getId
3   * @param pageNum 页码
4   * @param pageSize 页大小
5   * @return 返回List<UserCollectActivity>集合
6   * @description 获取收藏的活动列表
7   */
8  @RequestMapping("/getCollectedActivities")
9  public String getCollectedActivities(Integer id, Integer pageNum,
10 Integer pageSize) {
11      if (id == null || pageNum == null || pageSize == null) {
12          System.out.println("getCollectedActivities Error");
13          return "ErrorParameter";
14      }
15      return activityService.getCollectedActivities(id, pageNum,
16 pageSize);
17  }
```

添加活动

```
1  /**
2   * @param files 客户端传递若干MultipartFile文件, 要求请求的参数名
   字都
3   * 是file（就是要重复），这边接收就是自动加入
   List<MultipartFile>中
4   * @param activityDetailJson 客户端将封装好的ActivityDetail类转换为Json串发送
   过来,
5   * 注意ActivityDetail里面的Activity类也得把数据都
   封装好,
6   * 添加操作, 不需要设置id值, 因为数据库id是自增的
7   * @return 返回一个添加成功或者失败"true"或者"false"
```

```

8      * @description 添加活动, 接受一个ActivityDetail类的Json串数据, 且参数名称为
activityJson
9      */
10     @RequestMapping("/addActivity")
11     public String addActivity(@RequestParam("file") List<MultipartFile>
files, String activityDetailJson) {
12         if (activityDetailJson == null) {
13             System.out.println("addActivity Error");
14             return "ErrorParameter";
15         }
16         return activityService.addActivity(files, activityDetailJson) ?
"true" : "false";
17     }

```

获取报名的活动列表

```

1      /**
2      * @param id      用户id,注意不是getUserId,是getId
3      * @param pageNum 页码
4      * @param pageSize 页大小
5      * @return 返回List<UserCollectActivity>集合
6      * @description 获取报名的活动列表
7      */
8      @RequestMapping("/getEnterActivities")
9      public String getEnterActivities(Integer id, Integer pageNum, Integer
pageSize) {
10         if (id == null || pageNum == null || pageSize == null) {
11             System.out.println("getEnterActivities Error");
12             return "ErrorParameter";
13         }
14         return activityService.getEnterActivities(id, pageNum, pageSize);
15     }

```

获取发布的活动列表

```

1      /**
2      * @param id      用户id,注意不是getUserId,是getId
3      * @param pageNum 页码
4      * @param pageSize 页大小
5      * @return 返回List<UserCollectActivity>集合
6      * @description 获取发布的活动列表
7      */
8      @RequestMapping("/getPublishActivities")
9      public String getPublishActivities(Integer id, Integer pageNum, Integer
pageSize) {
10         if (id == null || pageNum == null || pageSize == null) {
11             System.out.println("getPublishActivities Error");
12             return "ErrorParameter";
13         }
14         return activityService.getPublishActivities(id, pageNum, pageSize);
15     }

```

根据时间筛选活动

```

1      /**
2       * @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
3       * @return 返回一个List<Activity>集合
4       * @description 根据时间筛选活动
5       */
6      @RequestMapping("/screenTimeActivities")
7      public String screenTimeActivities(Integer howManyDays, Integer pageNum,
8      Integer pageSize) {
9          List<Activity> activityList =
10         activityService.screenTimeActivities(howManyDays, pageNum, pageSize);
11         return activityList.size() != 0 ? JSON.toJSONString(activityList) :
12         "empty";
13     }

```

根据花费筛选活动

```

1      /**
2       * @param lowCost 价格区间的小值
3       * @param highCost 价格区间的大值
4       * @return 返回一个List<Activity>集合
5       * @description 根据花费筛选活动
6       */
7      @RequestMapping("/screenTypeActivities")
8      public String screenTypeActivities(Integer lowCost, Integer highCost,
9      Integer pageNum, Integer pageSize) {
10         List<Activity> activityList =
11         activityService.screenCostActivities(lowCost, highCost, pageNum, pageSize);
12         return activityList.size() != 0 ? JSON.toJSONString(activityList) :
13         "empty";
14     }

```

根据活动种类筛选活动

```

1      /**
2       * @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
3       * @return 返回一个List<Activity>集合
4       * @description 根据活动种类筛选活动
5       */
6      @RequestMapping("/screenCostActivities")
7      public String screenCostActivities(Integer tag, Integer pageNum, Integer
8      pageSize) {
9          List<Activity> activityList =
10         activityService.screenTypeActivities(tag, pageNum, pageSize);
11         return activityList.size() != 0 ? JSON.toJSONString(activityList) :
12         "empty";
13     }

```

