

课程说明

- 了解什么是即时通信
- 了解探花交友的消息功能
- 了解即时通信的技术方案
- 了解环信的即时通讯
- 实现环信的用户体系集成
- 实现添加联系人、联系人列表功能
- 实现点赞列表
- 实现评论列表
- 实现喜欢列表
- 实现公告列表

1、即时通信

1.1、什么是即时通信？

即时通信(IM)是指能够即时发送和接收[互联网](#)消息等的业务。自1998年面世以来，特别是近几年的迅速发展，即时通信的功能日益丰富，逐渐集成了[电子邮件](#)、[博客](#)、[音乐](#)、电视、游戏和搜索等多种功能。即时通信不再是一个单纯的[聊天工具](#)，它已经发展成集交流、[资讯](#)、娱乐、搜索、[电子商务](#)、办公协作和企业客户服务等为一体的综合化信息平台。^[1]

[微软](#)、[腾讯](#)、[AOL](#)、[Yahoo](#)等重要即时通信提供商都提供通过手机接入[互联网](#)即时通信的业务，用户可以通过手机与其他已经安装了相应[客户端](#)软件的手机或[电脑](#)收发消息。

1.2、功能说明

在探花交友项目中也提供了类似微信的聊天功能，用户可以和好友或陌生人聊天。

如果是陌生人，通过《聊一下》功能进行打招呼，如果对方同意后，就成为了好友，可以进行聊天了。

陌生人之间如果相互喜欢，那么就会成为好友，也就可以聊天了。

在消息界面中也可以查看：点赞、评论、喜欢、公告等消息信息。

消息



点赞



评论



喜欢



公告



Wendy

10:07

黑马的作业你做了吗，一起来做作业吧~

99+



小小女子

昨天

黑马的作业你做了吗，一起来做作业吧~

99+



谁动了我的冬天

2019-7-03

黑马的作业你做了吗，一起来做作业吧~

12



交友



圈子



消息



视频



我的



黑马大鹏

更多



缘分值 98

年龄相仿 兴趣相仿 收入相仿



问：你喜欢什么？

答：我喜欢秋天的落叶，夏天的泉水，冬天的雪地，只要有你我一切皆可~

你要干嘛啊



7"



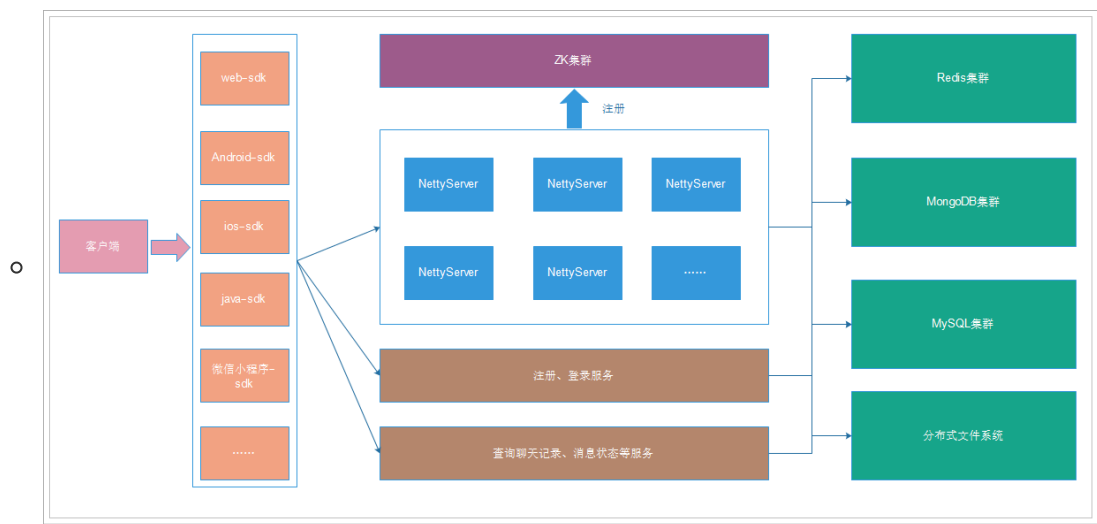


2、技术方案

对于高并发的即时通讯实现，还是很有挑战的，所需要考虑的点非常多，除了要实现功能，还要考虑并发、流量、负载、服务器、容灾等等。虽然有难度也并不是高不可攀。

对于现实即时通讯往往有两种方案：

- 方案一：
 - 自主实现，从设计到架构，再到实现。
 - 技术方面可以采用：Netty + WebSocket + RocketMQ + MongoDB + Redis + ZooKeeper + MySQL



- 方案二：
 - 对接第三方服务完成。
 - 这种方式简单，只需要按照第三方的api进行对接就可以了。
 - 如：环信、网易、容联云通讯等。

如何选择呢？

如果是中大型企业做项目可以选择自主研发，如果是中小型企业研发中小型的项目，选择第二种方案即可。方案一需要有大量的人力、物力的支持，开发周期长，成本高，但可控性强。方案二，成本低，开发周期短，能够快速的集成起来进行功能的开发，只是在可控性方面来说就差了一些。

探花交友项目选择方案二进行实现。

3、环信

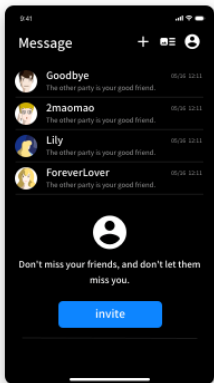
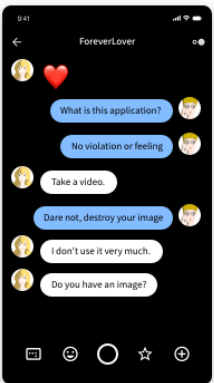
官网：<https://www.easemob.com/> 稳定健壮，消息必达，亿级并发的即时通讯云

IM 场景解决方案

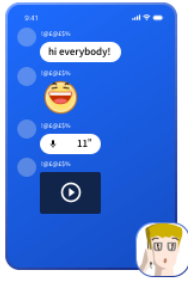
语音连麦聊天室 new
私密社交
群聊互动
视频会议
RTX企业门户
智能硬件

私密社交

- 支持全类型消息收发
- 消息撤回、阅后即焚、实时音视频等丰富的功能
- 反垃圾与敏感词保障内容合规
- 高并发与稳定性确保弱网络消息必达
- 多端多设备与消息漫游覆盖用户更多使用场景
- 7年研发积累，SLA 99.9%以上云服务集群保障

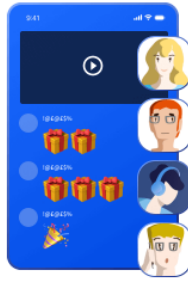
IM 核心服务功能



全类型消息

支持文字、表情、图片、语音、视频、附件、地理位置、扩展消息、透传消息、自定义消息等全类型消息收发；

[查看详情](#)



万人群聊互动

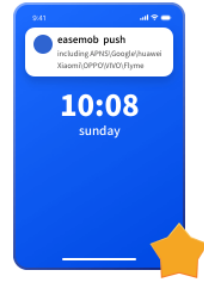
公有云支持5000人规模聊天室，专有化定制支持升级万人以上超级群，灵活满足各类应用场景的业务需求；



实时音视频

支持1对1、多对多音视频、音视频连麦等场景。低成本低延时、高品质、抗丢包抗抖动、百万级并发、全球多节点覆盖；

[查看详情](#)

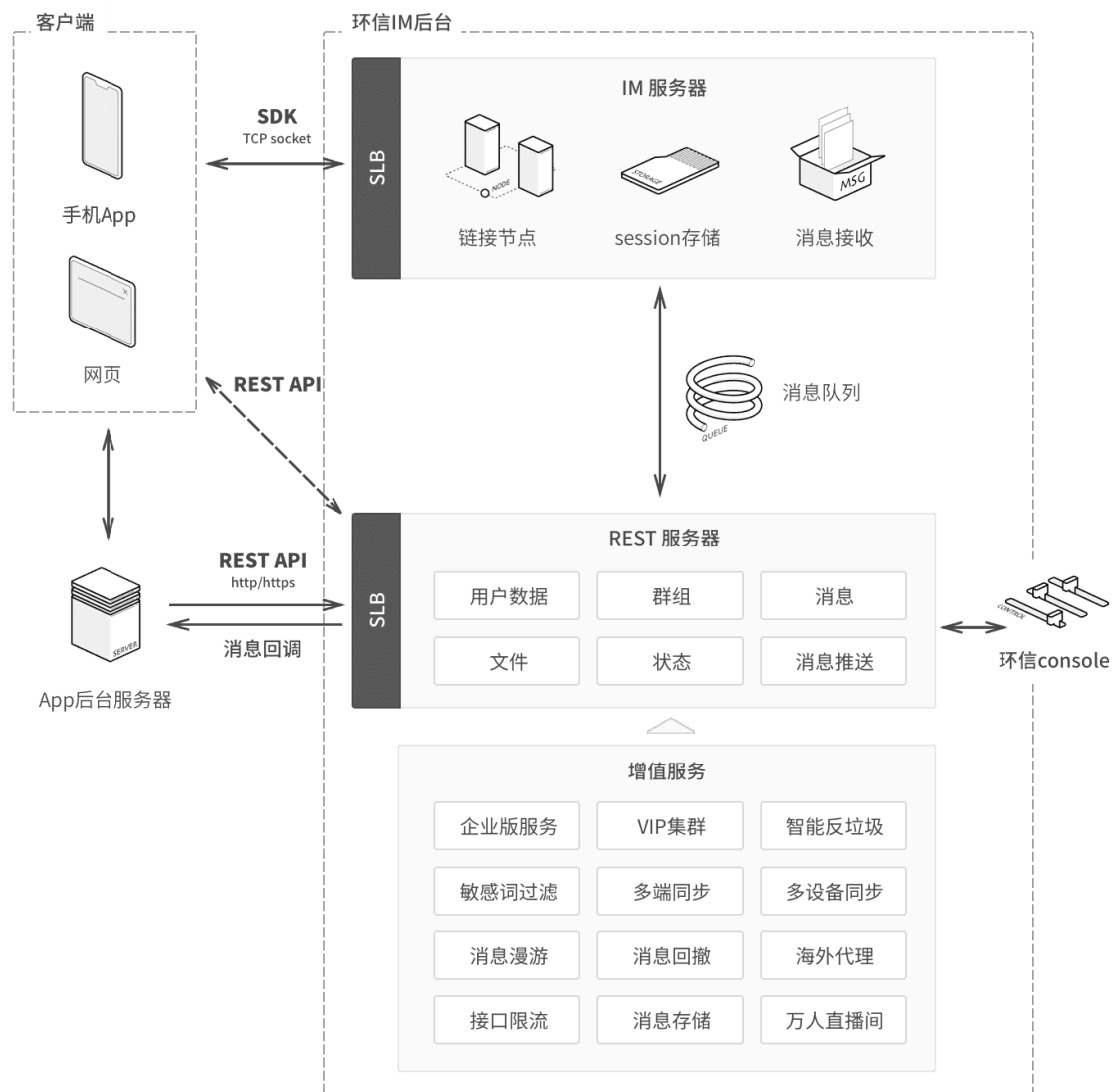


推送服务

服务端支持对接APNS（苹果）、Google、华为、小米、OPPO、VIVO、魅族等各大消息推送平台；

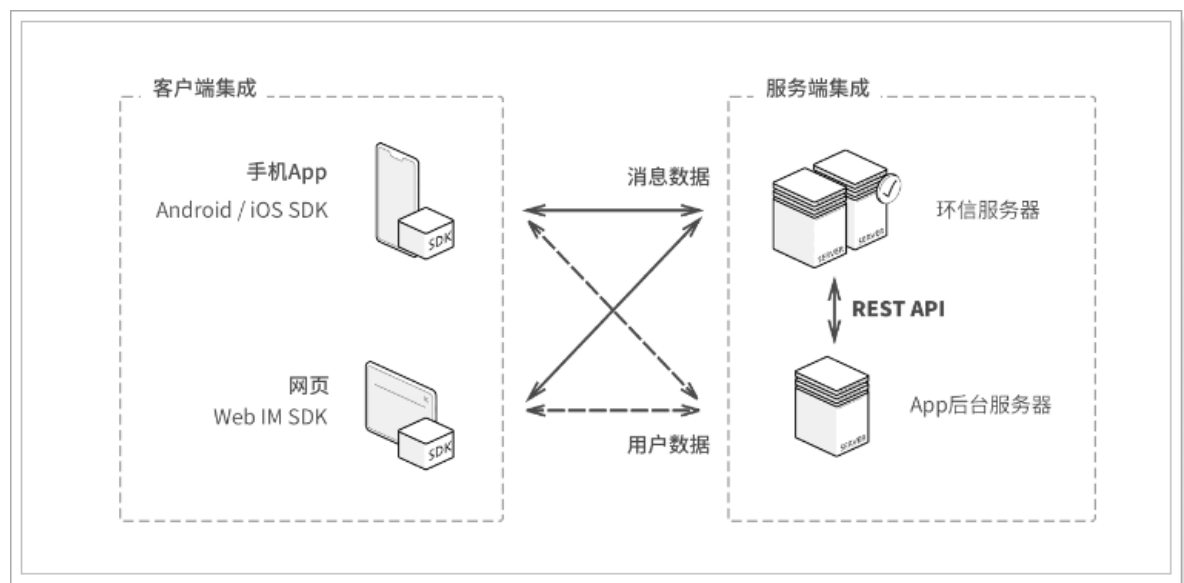
3.1、开发简介

平台架构：



集成：

环信和用户体系的集成主要发生在2个地方，服务器端集成和客户端集成。



探花集成：

- 探花前端使用AndroidSDK进行集成

- 文档：<http://docs-im.easemob.com/im/android/sdk/import>
- 后端集成用户体系
 - 文档：<http://docs-im.easemob.com/im/server/ready/user>

3.2、环信Console

需要使用环信平台，那么必须要进行注册，登录之后即可创建应用。环信100以内的用户免费使用，100以上就要注册企业版了。

企业版价格：

app当月第三高日活数范围 (单位：DAU)	当月服务扣费 (单位：¥)
500000-1000000	24000
100000-500000	14400
50000-100000	7500
10000-50000	3750
2000-10000	1500
0-2000	750

创建应用：

创建应用

appname: appname

appname是组成appkey的重要字段，只能是字母、数字、横线组合。长度不能超过32

Appkey: Appkey

Appkey是环信应用的唯一标识，由orgname和appname组成，生成后不允许更改

产品名称: 产品名称

方便管理员在管理后台范围内查找应用的代名词，长度32

描述: 描述

方便组织成员在管理后台理解产品的简单描述。长度不能超过512

注册模式: ☒ 开放注册

允许在该应用下自由注册新用户

☐ 授权注册

只有企业管理员或者应用管理员才能注册用户

取消

创建

创建完成：

探花 升级为企业版

IM用户

群组

聊天室

直播间

数据统计

证书

1105190515097562#tanhua

今日DAU

注册用户总数

用户注册模式 授权注册

0

3

强制推送 禁止

企业版服务

实时回调

音视频服务

最近更新 2019-06-25 11:11:10

<

1

>

4、用户体系集成

4.1、Appkey 数据结构

当您申请了 AppKey 后，会得到一个 **xxxx#xxxx** 格式的字符串，字符串只能由小写字母数字组成，AppKey是环信应用的唯一标识。前半部分 **org_name** 是在多租户体系下的唯一租户标识，后半部分 **app_name** 是租户下的app唯一标识（在环信后台创建一个app时填写的应用 id 即是 app_name）。下述的 REST API 中，**/org_name/app_name** 的请求，均是针对一个唯一的appkey进行的。目前环信注册的appkey暂不能由用户自己完成删除操作，如果对 APP 删除需要联系环信操作完成。

Appkey	xxxx	分隔符	xxxx
环信应用的唯一标识	org_name	#	app_name

4.2、环信 ID 数据结构

环信作为一个聊天通道，只需要提供环信 ID（也就是 IM 用户名）和密码就够了。

名称	字段名	数据类型	描述
环信 ID	username	String	在 AppKey 的范围内唯一用户名。
用户密码	password	String	用户登录环信使用的密码。

4.3、环信 ID 使用规则

当 APP 和环信集成的时候，需要把 APP 系统内的已有用户和新注册的用户和环信集成，为每个已有用户创建一个环信的账号（环信 ID），并且 APP 有新用户注册的时候，需要同步的在环信中注册。

在注册环信账户的时候，需要注意环信 ID 的规则：

- 使用英文字母和（或）数字的组合
- 不能使用中文
- 不能使用 email 地址
- 不能使用 UUID
- 用户ID的长度在255字节以内
- 中间不能有空格或者井号（#）等特殊字符
- 允许的用户名正则 “[a-zA-Z0-9_-]*”（a~z大小写字母/数字/下划线/横线/英文句号），其他都不允许 **如果是大写字母会自动转成小写**
- 不区分大小写。系统忽略大小写，认为 AA、Aa、aa、aA 都是一样的。如果系统已经存在了环信 ID 为 AA 的用户，再试图使用 aa 作为环信 ID 注册新用户，系统返回用户名重复，以此类推。但是请注意：环信 ID 在数据上的表现形式还是用户最初注册的形式，注册时候使用的大写就保存大写，是小写就保存小写。即：使用 AA 注册，环信保存的 ID 就是 AA；使用 Aa 注册，环信保存的 ID 就是 Aa，以此类推。

另：本文档中可能会交错使用“环信 ID”和“环信用户名”两个术语，但是请注意，这里两个的意思是一样的。

因为一个用户的环信 ID 和他的在 APP 中的用户名并不需要一致，只需要有一个明确的对应关系。例如，用户名是 [example@easemob.com](#)，当这个用户登录到 APP 的时候，可以登录成功之后，再登录环信的服务器，所以这时候，只需要能够从 [example@easemob.com](#) 推导出这个用户的环信 ID 即可。

4.4、获取管理员权限

环信提供的 REST API 需要权限才能访问，权限通过发送 HTTP 请求时携带 token 来体现，下面描述获取 token 的方式。说明：API 描述的时候使用到的 {APP 的 client_id} 之类的这种参数需要替换成具体的值。

重要提醒：获取 token 时服务器会返回 token 有效期，具体值参考接口返回的 expires_in 字段值。由于网络延迟等原因，系统不保证 token 在此值表示的有效期内绝对有效，如果发现 token 使用异常请重新获取新的 token，比如“http response code”返回 401。另外，请不要频繁向服务器发送获取 token 的请求，同一账号发送此请求超过一定频率会被服务器封号，切记，切记！！

client_id 和 client_secret 可以在环信管理后台的 [APP 详情页面](#)看到。

HTTP Request

POST	/ {org_name} / {app_name} / token

Request Headers

参数	说明
Content-Type	application/json

Request Body

参数	说明
grant_type	client_credentials
client_id	App的client_id，可在 app详情页找到
client_secret	App的client_secret，可在 app详情页找到

Response Body

参数	说明
access_token	有效的token字符串
expires_in	token 有效时间，以秒为单位，在有效期内不需要重复获取
application	当前 App 的 UUID 值

请求示例

```
curl -X POST -H 'Content-Type: application/json' -H 'Accept: application/json' -d '{
  "grant_type": "client_credentials",
  "client_id": "YXA6i-Ak80l4Eei2l11ZjV-EAg",
  "client_secret": "YXA6VunqiNxoB7IwXHInk1cGiX00Jfc"
}' 'http://a1.easemob.com/easemob-demo/testapp/token'
```

可能返回的结果示例

返回值200，表示成功返回token

```
{
  "access_token": "YwMte3bGuOukEeiTkNP4grL7iwAAAAAAAAAAAAAAAAAGL4CTw6XgR6LaXXVmNX4QCAgHAAAFnKdc-ZgBPgGgBFT
rLhhyk8woMEI005emtrLJFJV6aoxsZSioSIzkr5kw",
  "expires_in": 5184000,
  "application": "8be024f0-e978-11e8-b697-5d598d5f8402"
}
```

返回值400，表示 client_id 或 client_secret 错误

```
{
  "error_description": "client_id does not match",
  "error": "invalid_grant"
}
```

如果返回结果是429、503或者其他5xx，有可能代表该接口被限流了，请稍微暂停一下并重试。详见[接口限流说明](#)

4.4.1、配置

将用户体系集成的逻辑写入到sso系统中。

huanxin.properties

```
1 tanhua.huanxin.url=http://a1.easemob.com/
2 tanhua.huanxin.orgName=1105190515097562
3 tanhua.huanxin.appName=tanhua
4 tanhua.huanxin.clientId=YXA67ZofwHb1Eems-_Fh-17T2g
5 tanhua.huanxin.clientSecret=YXA60r45rNy2Ux5wQ7YYoEPwynHmUZk
```

说明：这配置在控制台可以找到。

```
1 package com.tanhua.sso.config;
2
3 import lombok.Data;
4 import org.springframework.boot.context.properties.ConfigurationProperties;
5 import org.springframework.context.annotation.Configuration;
6 import org.springframework.context.annotation.PropertySource;
7
8 @Configuration
9 @PropertySource("classpath:huanxin.properties")
10 @ConfigurationProperties(prefix = "tanhua.huanxin")
11 @Data
12 public class HuanXinConfig {
13
14     private String url;
15     private String orgName;
16     private String appName;
17     private String clientId;
```

```

18     private String clientSecret;
19
20 }

```

4.4.2、获取token

```

1  package com.tanhua.sso.service;
2
3  import com.fasterxml.jackson.databind.JsonNode;
4  import com.fasterxml.jackson.databind.ObjectMapper;
5  import com.tanhua.sso.config.HuanXinConfig;
6  import org.apache.commons.lang3.StringUtils;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.boot.context.properties.ConfigurationProperties;
9  import org.springframework.data.redis.core.RedisTemplate;
10 import org.springframework.http.ResponseEntity;
11 import org.springframework.stereotype.Service;
12 import org.springframework.web.client.RestTemplate;
13
14 import java.time.Duration;
15 import java.util.HashMap;
16 import java.util.Map;
17
18 @Service
19 public class HuanxinTokenService {
20
21     private static final ObjectMapper MAPPER = new ObjectMapper();
22
23     @Autowired
24     private HuanXinConfig huanXinConfig;
25
26     @Autowired
27     private RestTemplate restTemplate;
28
29     public static final String REDIS_KEY = "HX_TOKEN";
30
31     @Autowired
32     private RedisTemplate<String, String> redisTemplate;
33
34     private String refreshToken() {
35         String targetUrl = this.huanXinConfig.getUrl() +
36             this.huanXinConfig.getOrgName() + "/" + this.huanXinConfig.getAppName() +
37             "/token";
38
39         Map<String, String> param = new HashMap<>();
40         param.put("grant_type", "client_credentials");
41         param.put("client_id", this.huanXinConfig.getClientId());
42         param.put("client_secret", this.huanXinConfig.getClientSecret());
43
44         //请求环信接口
45         ResponseEntity<String> responseEntity =
46             this.restTemplate.postForEntity(targetUrl, param,
47                 String.class);
48
49         if (responseEntity.getStatusCodeValue() != 200) {
50             return null;
51         }
52     }
53 }

```

```

49         String body = responseEntity.getBody();
50         try {
51             JsonNode jsonNode = MAPPER.readTree(body);
52             String accessToken = jsonNode.get("access_token").asText();
53             if (StringUtils.isNotBlank(accessToken)) {
54                 // 将token保存到redis，有效期为5天，环信接口返回的有效期为6天
55                 this.redisTemplate.opsForValue().set(REDIS_KEY,
56                 accessToken, Duration.ofDays(5));
57                 return accessToken;
58             } catch (Exception e) {
59                 e.printStackTrace();
60             }
61
62             return null;
63         }
64
65         public String getToken() {
66             String token = this.redisTemplate.opsForValue().get(REDIS_KEY);
67             if (StringUtils.isBlank(token)) {
68                 return this.refreshToken();
69             }
70             return token;
71         }
72     }
73

```

4.5、注册环信用户

注册环信用户分为2种，开放注册、授权注册，区别在于开发注册不需要token，授权注册需要token。

我们使用的授权注册：

```

1  package com.tanhua.sso.vo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  @Data
8  @NoArgsConstructor
9  @AllArgsConstructor
10 public class HuanXinUser {
11
12     private String username;
13     private String password;
14
15 }
16

```

```

1  package com.tanhua.sso.service;
2
3  import com.fasterxml.jackson.databind.ObjectMapper;
4  import com.tanhua.sso.config.HuanXinConfig;

```

```

5  import com.tanhua.sso.vo.HuanXinUser;
6  import org.apache.commons.codec.digest.DigestUtils;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.http.HttpEntity;
9  import org.springframework.http.HttpHeaders;
10 import org.springframework.http.ResponseEntity;
11 import org.springframework.stereotype.Service;
12 import org.springframework.web.client.RestTemplate;
13
14 import java.util.Arrays;
15
16 @Service
17 public class HuanXinService {
18
19     private static final ObjectMapper MAPPER = new ObjectMapper();
20
21     @Autowired
22     private HuanXinTokenService huanXinTokenService;
23
24     @Autowired
25     private RestTemplate restTemplate;
26
27     @Autowired
28     private HuanXinConfig huanXinConfig;
29
30     /**
31      * 注册环信用户
32      *
33      * @param userId
34      * @return
35      */
36     public boolean register(Long userId) {
37         String targetUrl = this.huanXinConfig.getUrl()
38             + this.huanXinConfig.getOrgName() + "/"
39             + this.huanXinConfig.getAppName() + "/users";
40
41         String token = this.huanXinTokenService.getToken();
42
43         try {
44             // 请求体
45             HuanXinUser huanXinUser = new
HuanXinUser(String.valueOf(userId), DigestUtils.md5Hex(userId +
"_itcast_tanhua"));
46             String body =
MAPPER.writeValueAsString(Arrays.asList(huanXinUser));
47
48             // 请求头
49             HttpHeaders headers = new HttpHeaders();
50             headers.add("Content-Type", "application/json");
51             headers.add("Authorization", "Bearer " + token);
52
53             HttpEntity<String> httpEntity = new HttpEntity<>(body,
headers);
54             ResponseEntity<String> responseEntity =
this.restTemplate.postForEntity(targetUrl, httpEntity, String.class);
55
56             return responseEntity.getStatusCodeValue() == 200;
57         } catch (Exception e) {

```

```

58         e.printStackTrace();
59     }
60
61     // 注册失败
62     return false;
63
64 }
65
66 }
67

```

加入到登录逻辑中：

```

queryWrapper.eq("column", "mobile", mobile);
User selectUser = this.userMapper.selectOne(queryWrapper);
if (selectUser == null) {
    // 该手机号未注册，进行注册操作
    User user = new User();
    user.setMobile(mobile);
    user.setPassword(DigestUtils.md5Hex(data: secret + "_123456")); //
    this.userMapper.insert(user);
    selectUser = user;
    isNew = true;

    //注册环信用户
    this.huanXinService.register(user.getId());
}

```

4.6、测试

+创建IM用户

搜索用户ID

Q

☐	用户ID	昵称	消息通知	免打扰	证书	操作
☐	113		仅通知	未开启	--	更多 v

上一页

下一页

可以看到已经注册到了环信。

4.7、查询环信用户信息

在app中，用户登录后需要根据用户名密码登录环信，由于用户名密码保存在后台，所以需要提供接口进行返回。

mock地址：<https://mock.boxuegu.com/project/164/interface/api/66955>

基本信息

接口名称：环信用户信息

创建人： 陶峙巍

状态：● 已完成

更新时间：2019-09-18 16:05:06

接口路径：GET /huanxin/user

Mock地址：<https://mock.boxuegu.com/mock/164/huanxin/user>

返回数据

名称	类型	是否必须	默认值	备注	其他信息
username	string	必须		用户名	枚举: taoshiwei
password	string	必须		用户密码	枚举: 123456

实现：

```
1 package com.tanhua.server.controller;
2
3 import com.tanhua.server.pojo.User;
4 import com.tanhua.server.utils.UserThreadLocal;
5 import com.tanhua.server.vo.HuanXinUser;
6 import org.apache.commons.codec.digest.DigestUtils;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.web.bind.annotation.GetMapping;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 @RestController
13 @RequestMapping("huanxin")
14 public class HuanXinController {
15
16     @GetMapping("user")
17     public ResponseEntity<HuanXinUser> queryHuanXinUser(){
18         User user = UserThreadLocal.get();
19
20         HuanXinUser huanXinUser = new HuanXinUser();
21         huanXinUser.setUsername(user.getId().toString());
22         huanXinUser.setPassword(DigestUtils.md5Hex(user.getId() +
23             "_itcast_tnhua"));
24
25         return ResponseEntity.ok(huanXinUser);
26     }
27 }
```

测试：

http://192.168.31.98/huanxin/user

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

Authorization eyJhbGciOiJIUzI1NiU9.eyJjb2JpbGUiOiNzYwMjAyNjg2OSIsImkiOiJoxMTR9.IhbcYZz_sxTKzDnNsOsuW82lgemh0EqCnjoPbM7Ixbc

Status 200 Loading time: 1101 ms

Request headers Authorization: eyJhbGciOiJIUzI1NiU9.eyJjb2JpbGUiOiNzYwMjAyNjg2OSIsImkiOiJoxMTR9.IhbcYZz_sxTKzDnNsOsuW82lgemh0EqCnjoPbM7Ixbc
User-Agent: Mozilla/5.0 (Windows NT 6.3; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.103 Safari/537.36
Accept: */*

Response headers Server: nginx/1.17.3
Date: Sat, 12 Oct 2019 02:42:19 GMT
Content-Type: application/json; charset=UTF-8
Transfer-Encoding: chunked
Connection: keep-alive

Raw JSON Response

Copy to clipboard Save as file

```
{
  "username": "114",
  "password": "0330b0245266869aa2fdc61124e199e3"
}
```

4.8、发送消息给客户端

目前已经完成了用户体系的对接，下面我们进行测试发送消息，场景是这样的：



点击“聊一下”，就会给对方发送一条陌生人信息，这个消息由系统发送完成。

我们暂时通过环信的控制台进行发送：



消息内容：

```
1 {"userId": "1", "nickname": "黑马小妹", "strangerQuestion": "你喜欢去看蔚蓝的大海还是去爬巍峨的高山？", "reply": "我喜欢秋天的落叶，夏天的泉水，冬天的雪地，只要有你我一切皆可~"}

```





可以看到已经接收到了消息。

5、添加联系人

点击“聊一下”，就会成为联系人（好友）。

实现：

- 将好友写入到MongoDB中
- 将好友关系注册到环信

5.1、mock接口

地址：<https://mock.bxuegu.com/project/164/interface/api/77980>

接口路径： **POST** /messages/contacts

Mock地址： <https://mock.bboxuegu.com/mock/164/messages/contacts>

备注

"聊一下"一键添加好友

请求参数

Headers：

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		
Authorization	eyJ0eXAiOiKV1QilCJhbGciOiJIUz11NiJ9.eyJleHAiOiJlNj14MjkzMzYslVzZUfaWQiOiXln0.Mbzn6LzsLrkVWEbhexR3lTYDZjxqlcqW11rJxDQ6Ewk	是		令牌

Body:

名称	类型	是否必须	默认值	备注	其他信息
userid	integer	必须		用户id	最大值: 10000 最小值: 1

5.2、定义dubbo服务

```
1 package com.tanhua.dubbo.server.pojo;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6 import org.bson.types.ObjectId;
7 import org.springframework.data.mongodb.core.mapping.Document;
8
9 import java.util.Date;
10
11 @Data
12 @NoArgsConstructor
13 @AllArgsConstructor
14 @Document(collection = "tanhua_users")
15 public class Users implements java.io.Serializable{
16
17     private static final long serialVersionUID = 6003135946820874230L;
18     private ObjectId id;
19     private Long userId; //用户id
20     private Long friendId; //好友id
21     private Long date; //时间
22 }
23
```

```
1 package com.tanhua.dubbo.server.api;
2
3 import com.tanhua.dubbo.server.pojo.Users;
4
5 public interface UsersApi {
6
```

```

7      /**
8       * 保存好友
9       *
10      * @param users
11      * @return
12      */
13      String saveUsers(Users users);
14  }
15

```

实现：

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.alibaba.dubbo.config.annotation.Service;
4  import com.tanhua.dubbo.server.pojo.Users;
5  import org.bson.types.ObjectId;
6  import org.springframework.beans.factory.annotation.Autowired;
7  import org.springframework.data.mongodb.core.MongoTemplate;
8  import org.springframework.data.mongodb.core.query.Criteria;
9  import org.springframework.data.mongodb.core.query.Query;
10
11  @Service(version = "1.0.0")
12  public class UsersApiImpl implements UsersApi {
13
14      @Autowired
15      private MongoTemplate mongoTemplate;
16
17
18      @Override
19      public String saveUsers(Users users) {
20
21          if (users.getFriendId() == null || users.getUserId() == null) {
22              return null;
23          }
24
25          // 检测是否该好友关系是否存在
26          Query query =
27              Query.query(Criteria.where("userId").is(users.getUserId()).and("friendId").
28                  is(users.getFriendId()));
29          Users oldUsers = this.mongoTemplate.findOne(query, Users.class);
30          if (null != oldUsers) {
31              return null;
32          }
33
34          users.setId(ObjectId.get());
35          users.setDate(System.currentTimeMillis());
36
37          this.mongoTemplate.save(users);
38          return users.getId().toHexString();
39      }
40  }

```

5.3、注册好友到环信

对接环信api的操作在sso工程中完成。

```
1 package com.tanhua.sso.controller;
2
3 import com.tanhua.sso.service.HuanXinService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.http.HttpStatus;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.PathVariable;
8 import org.springframework.web.bind.annotation.PostMapping;
9 import org.springframework.web.bind.annotation.RequestMapping;
10 import org.springframework.web.bind.annotation.RestController;
11
12 @RestController
13 @RequestMapping("user/huanxin")
14 public class HuanXinController {
15
16     @Autowired
17     private HuanXinService huanXinService;
18
19     /**
20      * 添加联系人
21      *
22      * @param userId
23      * @param friendId
24      * @return
25      */
26     @PostMapping("contacts/{owner_username}/{friend_username}")
27     public ResponseEntity<Void>
28     contactUsers(@PathVariable("owner_username") Long userId,
29
30     @PathVariable("friend_username") Long friendId) {
31         try {
32             boolean result = this.huanXinService.contactUsers(userId,
33             friendId);
34             if (result) {
35                 return ResponseEntity.ok(null);
36             }
37         } catch (Exception e) {
38             e.printStackTrace();
39         }
40
41         return
42         ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
43     }
44 }
```

```
1 package com.tanhua.sso.service;
2
3 import com.fasterxml.jackson.databind.ObjectMapper;
4 import com.tanhua.sso.config.HuanXinConfig;
5 import com.tanhua.sso.vo.HuanXinUser;
6 import org.apache.commons.codec.digest.DigestUtils;
7 import org.springframework.beans.factory.annotation.Autowired;
8 import org.springframework.http.HttpEntity;
```

```

 9 import org.springframework.http.HttpHeaders;
10 import org.springframework.http.ResponseEntity;
11 import org.springframework.stereotype.Service;
12 import org.springframework.web.client.RestTemplate;
13
14 import java.util.Arrays;
15
16 @Service
17 public class HuanXinService {
18
19     private static final ObjectMapper MAPPER = new ObjectMapper();
20
21     @Autowired
22     private HuanXinTokenService huanXinTokenService;
23
24     @Autowired
25     private RestTemplate restTemplate;
26
27     @Autowired
28     private HuanXinConfig huanXinConfig;
29
30     /**
31      * 添加好友
32      *
33      * @param userId
34      * @param friendId
35      * @return
36      */
37     public boolean contactUsers(Long userId, Long friendId) {
38         String targetUrl = this.huanXinConfig.getUrl()
39             + this.huanXinConfig.getOrgName() + "/"
40             + this.huanXinConfig.getAppName() + "/users/" +
41             userId + "/contacts/users/" + friendId;
42         try {
43             String token = this.huanXinTokenService.getToken();
44             // 请求头
45             HttpHeaders headers = new HttpHeaders();
46             headers.add("Authorization", "Bearer " + token);
47
48             HttpEntity<String> httpEntity = new HttpEntity<>(headers);
49             ResponseEntity<String> responseEntity =
this.restTemplate.postForEntity(targetUrl, httpEntity, String.class);
50
51             return responseEntity.getStatusCodeValue() == 200;
52         } catch (Exception e) {
53             e.printStackTrace();
54         }
55
56         // 添加失败
57         return false;
58     }
59 }
60

```

5.4、编写服务

在itcast-tanhua-server中完成。

```

1 package com.tanhua.server.controller;
2
3 import com.tanhua.server.service.IMService;
4 import org.slf4j.Logger;
5 import org.slf4j.LoggerFactory;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.http.HttpStatus;
8 import org.springframework.http.ResponseEntity;
9 import org.springframework.web.bind.annotation.PostMapping;
10 import org.springframework.web.bind.annotation.RequestBody;
11 import org.springframework.web.bind.annotation.RequestMapping;
12 import org.springframework.web.bind.annotation.RestController;
13
14 import java.util.Map;
15
16 @RestController
17 @RequestMapping("messages")
18 public class IMController {
19
20     private static final Logger LOGGER =
21         LoggerFactory.getLogger(IMController.class);
22
23     @Autowired
24     private IMService imService;
25
26     /**
27      * 添加好友
28      *
29      * @param param
30      * @return
31      */
32     @PostMapping("contacts")
33     public ResponseEntity<Void> contactUser(@RequestBody Map<String,
34         Object> param) {
35         try {
36             Long userId = Long.valueOf(param.get("userId").toString());
37             boolean result = this.imService.contactUser(userId);
38             if (result) {
39                 return ResponseEntity.ok(null);
40             }
41         } catch (Exception e) {
42             LOGGER.error("添加联系人失败! param = " + param, e);
43         }
44         return
45             ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
46     }
47 }

```

```

1 package com.tanhua.server.service;
2
3 import com.alibaba.dubbo.common.utils.StringUtils;
4 import com.alibaba.dubbo.config.annotation.Reference;
5 import com.tanhua.dubbo.server.api.UsersApi;
6 import com.tanhua.dubbo.server.pojo.Users;
7 import com.tanhua.server.pojo.User;

```

```

8  import com.tanhua.server.utils.UserThreadLocal;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.beans.factory.annotation.Value;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.stereotype.Service;
13 import org.springframework.web.client.RestTemplate;
14
15 @Service
16 public class IMService {
17
18     @Reference(version = "1.0.0")
19     private UsersApi usersApi;
20
21     @Autowired
22     private RestTemplate restTemplate;
23
24     @Value("${tanhua.sso.url}")
25     private String url;
26
27     /**
28      * 添加好友
29      *
30      * @param userId
31      */
32     public boolean contactUser(Long userId) {
33         User user = UserThreadLocal.get();
34
35         Users users = new Users();
36         users.setUserId(user.getId());
37         users.setFriendId(userId);
38
39         String id = this.usersApi.saveUsers(users);
40         if (StringUtils.isEmpty(id)) {
41             //注册好友关系到环信
42             String targetUrl = url + "/user/huanxin/contacts/" +
users.getUserId() + "/" + users.getFriendId();
43             ResponseEntity<Void> responseEntity =
this.restTemplate.postForEntity(targetUrl, null, Void.class);
44             if (responseEntity.getStatusCode().is2xxSuccessful()) {
45                 return true;
46             }
47             return false;
48         }
49
50         return false;
51     }
52 }
53

```

5.5、测试

http://192.168.31.98/messages/contacts

☐ GET ☒ POST ☐ PUT ☐ PATCH ☐ DELETE ☐ HEAD ☐ OPTIONS ☐ Other

Raw Form Headers

Add new header

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJjb2JpbGUioiixNzYwMjAyNjg2OSIsImkiOiJjoxMTR9.LhbcYZz_sxTKzDnNsOsuW82lgemh0EqCnjoPbM7txbc
---------------	---

Raw Form Files (0) Payload

Encode payload Decode payload

```
{ "userId": 115 }
```

application/json Set "Content-Type" header to overwrite this value.

查看IM用户好友

X

添加好友

用户ID

添加好友

115好友列表-共1项

序号	用户ID	操作
1	114	解除好友

可以看到好友已经添加成功。

6、联系人列表

6.1、mock接口

接口路径：GET /messages/contacts

Mock地址：<https://mock.boxuegu.com/mock/164/messages/contacts>

请求参数

Headers :

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		
Authorization	eyJ0eXAiOiKV1QiLCJhbGciOiJIUz11NiJ9.eyJleHAiOjE1NjI4MjkzMzYsInVzZXJfaWQiOiOilxln0.Mbzn6LzsLrkVWEbhexR3ITYDZjxqIcqW11rJxDQ6Ewk	是		令牌

Query :

参数名称	是否必须	示例	备注
page	是	1	当前页数
pagesize	是	10	页尺寸
keyword	是		关键字

响应数据结构：

items	object []	必须		列表	最小数量: 10 元素是否都不同: true 最大数量: 10 item 类型: object
id	integer	非必须		编号	最大值: 10000 最小值: 1
userid	string	必须		用户id	枚举: liujunjie
avatar	string	必须		头像	枚举: https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_1.png, https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_2.png, https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_3.png, https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_4.png
nickname	string	必须		昵称	枚举: Wendy,谁动了我的冬天, 卿本笨笨
gender	string	必须		性别 man woman	枚举: man,woman
age	integer	必须		年龄	最大值: 30 最小值: 20
city	string	必须		城市	枚举: 上海,北京,广州

6.2、定义Contacts

```
1 package com.tanhua.server.vo;  
2  
3 import lombok.AllArgsConstructor;  
4 import lombok.Data;  
5 import lombok.NoArgsConstructor;  
6  
7 @Data
```

```

8  @NoArgsConstructor
9  @AllArgsConstructor
10 public class Contacts {
11
12     private Long id;
13     private String userId;
14     private String avatar;
15     private String nickname;
16     private String gender;
17     private Integer age;
18     private String city;
19
20 }
21

```

6.4、dubbo接口

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.tanhua.dubbo.server.pojo.Users;
4  import com.tanhua.dubbo.server.vo.PageInfo;
5
6  import java.util.List;
7
8  public interface UsersApi {
9
10     /**
11      * 保存好友
12      *
13      * @param users
14      * @return
15      */
16     String saveUsers(Users users);
17
18     /**
19      * 根据用户id查询Users列表
20      *
21      * @param userId
22      * @return
23      */
24     List<Users> queryAllUsersList(Long userId);
25
26     /**
27      * 根据用户id查询Users列表(分页查询)
28      *
29      * @param userId
30      * @return
31      */
32     PageInfo<Users> queryUsersList(Long userId, Integer page, Integer
33     pageSize);
34 }

```

实现：

```

1  package com.tanhua.dubbo.server.api;

```

```

2
3 import com.alibaba.dubbo.config.annotation.Service;
4 import com.tanhua.dubbo.server.pojo.Publish;
5 import com.tanhua.dubbo.server.pojo.Users;
6 import com.tanhua.dubbo.server.vo.PageInfo;
7 import org.bson.types.ObjectId;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.data.domain.PageRequest;
10 import org.springframework.data.domain.Sort;
11 import org.springframework.data.mongodb.core.MongoTemplate;
12 import org.springframework.data.mongodb.core.query.Criteria;
13 import org.springframework.data.mongodb.core.query.Query;
14
15 import java.util.List;
16
17 @Service(version = "1.0.0")
18 public class UsersApiImpl implements UsersApi {
19
20     @Autowired
21     private MongoTemplate mongoTemplate;
22
23     @Override
24     public List<Users> queryAllUsersList(Long userId) {
25         Query query = Query.query(Criteria.where("userId").is(userId));
26         return this.mongoTemplate.find(query, Users.class);
27     }
28
29     @Override
30     public PageInfo<Users> queryUsersList(Long userId, Integer page,
31 Integer pageSize) {
32         PageRequest pageRequest = PageRequest.of(page - 1, pageSize,
33 Sort.by(Sort.Order.desc("created")));
34         Query query =
35 Query.query(Criteria.where("userId").is(userId)).with(pageRequest);
36
37         List<Users> usersList = this.mongoTemplate.find(query,
38 Users.class);
39
40         PageInfo<Users> pageInfo = new PageInfo<>();
41         pageInfo.setPageNum(page);
42         pageInfo.setPageSize(pageSize);
43         pageInfo.setRecords(usersList);
44         pageInfo.setTotal(0); //不提供总数
45         return pageInfo;
46     }
47 }
48

```

6.5、编写接口服务

```

1 package com.tanhua.server.controller;
2
3 import com.tanhua.dubbo.server.vo.PageInfo;
4 import com.tanhua.server.service.IMService;
5 import com.tanhua.server.vo.Contacts;
6 import com.tanhua.server.vo.PageResult;
7 import org.slf4j.Logger;

```

```

8  import org.slf4j.LoggerFactory;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.http.HttpStatus;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.web.bind.annotation.*;
13
14 import java.util.Map;
15
16 @RestController
17 @RequestMapping("messages")
18 public class IMController {
19
20     private static final Logger LOGGER =
21     LoggerFactory.getLogger(IMController.class);
22
23     @Autowired
24     private IMService imService;
25
26     /**
27      * 查询联系人列表
28      *
29      * @param page
30      * @param pageSize
31      * @param keyword
32      * @return
33      */
34     @GetMapping("contacts")
35     public ResponseEntity<PageResult> queryContactsList(@RequestParam(value
36     = "page", defaultValue = "1") Integer page,
37
38     @RequestParam(value = "pagesize", defaultValue = "10") Integer pageSize,
39
40     @RequestParam(value = "keyword", required = false) String keyword) {
41         PageResult pageResult = this.imService.queryContactsList(page,
42         pageSize, keyword);
43         return ResponseEntity.ok(pageResult);
44     }
45 }

```

```

1  package com.tanhua.server.service;
2
3  import com.alibaba.dubbo.config.annotation.Reference;
4  import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
5  import com.tanhua.dubbo.server.api.UsersApi;
6  import com.tanhua.dubbo.server.pojo.Users;
7  import com.tanhua.dubbo.server.vo.PageInfo;
8  import com.tanhua.server.pojo.User;
9  import com.tanhua.server.pojo.UserInfo;
10 import com.tanhua.server.utils.UserThreadLocal;
11 import com.tanhua.server.vo.Contacts;
12 import com.tanhua.server.vo.PageResult;
13 import org.apache.commons.lang3.StringUtils;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.beans.factory.annotation.Value;
16 import org.springframework.http.ResponseEntity;
17 import org.springframework.stereotype.Service;

```

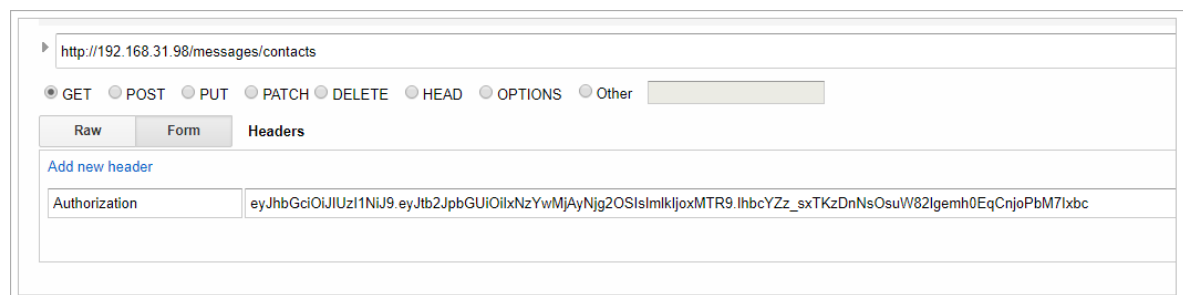
```
18 import org.springframework.web.client.RestTemplate;
19
20 import java.util.ArrayList;
21 import java.util.List;
22
23 @Service
24 public class IMService {
25
26     @Reference(version = "1.0.0")
27     private UsersApi usersApi;
28
29     @Autowired
30     private RestTemplate restTemplate;
31
32     @Value("${tanhua.sso.url}")
33     private String url;
34
35     @Autowired
36     private UserInfoService userInfoService;
37
38     public PageResult queryContactsList(Integer page, Integer pageSize,
39 String keyword) {
40         User user = UserThreadLocal.get();
41
42         List<Users> usersList = null;
43         if (StringUtils.isEmpty(keyword)) {
44             usersList = this.usersApi.queryAllUsersList(user.getId());
45         } else {
46             PageInfo<Users> usersPageInfo =
47 this.usersApi.queryUsersList(user.getId(), page, pageSize);
48             usersList = usersPageInfo.getRecords();
49         }
50
51         List<Long> userIds = new ArrayList<>();
52         for (Users users : usersList) {
53             userIds.add(users.getFriendId());
54         }
55
56         QueryWrapper<UserInfo> queryWrapper = new QueryWrapper<>();
57         queryWrapper.in("user_id", userIds);
58         if (StringUtils.isEmpty(keyword)) {
59             queryWrapper.like("nick_name", keyword);
60         }
61
62         List<UserInfo> userInfoList =
63 this.userInfoService.queryList(queryWrapper);
64
65         List<Contacts> contactsList = new ArrayList<>();
66
67         if (StringUtils.isEmpty(keyword)) {
68             for (Users users : usersList) {
69                 for (UserInfo userInfo : userInfoList) {
70                     if (users.getFriendId().longValue() ==
71 userInfo.getUserId().longValue()) {
72                         Contacts contacts = new Contacts();
73                         contacts.setAge(userInfo.getAge());
74                         contacts.setAvatar(userInfo.getLogo());
75                     }
76                 }
77             }
78         }
79     }
80 }
```

```

71 contacts.setGender(userInfo.getSex().name().toLowerCase());
72         contacts.setNickname(userInfo.getNickName());
73
74 contacts.setUserId(String.valueOf(userInfo.getUserId()));
75
76 contacts.setCity(StringUtils.substringBefore(userInfo.getCity(), "-"));
77
78         contactsList.add(contacts);
79         break;
80     }
81 } else {
82     for (UserInfo userInfo : userInfoList) {
83         Contacts contacts = new Contacts();
84         contacts.setAge(userInfo.getAge());
85         contacts.setAvatar(userInfo.getLogo());
86
87         contacts.setGender(userInfo.getSex().name().toLowerCase());
88         contacts.setNickname(userInfo.getNickName());
89         contacts.setUserId(String.valueOf(userInfo.getUserId()));
90
91         contacts.setCity(StringUtils.substringBefore(userInfo.getCity(), "-"));
92
93         contactsList.add(contacts);
94     }
95
96     PageResult pageResult = new PageResult();
97     pageResult.setPage(page);
98     pageResult.setPages(0);
99     pageResult.setCounts(0);
100    pageResult.setPagesize(pageSize);
101    pageResult.setItems(contactsList);
102
103    return pageResult;
104 }
105 }

```

6.6、测试







7、点赞列表

7.1、mock接口

地址：<https://mock.boxuegu.com/project/164/interface/api/63966>

接口路径：`GET` /messages/likes

Mock地址：<https://mock.bboxuegu.com/mock/164/messages/likes>

请求参数

Headers：

参数名称	参数值	是否必须	示例	备注
Content-Type	application/json	是		
Authorization	eyJ0eXAiOiKV1QilCJhbGciOiJIUzI1Ni9.eyJleHAiOjE1NjI4MjkzMzYsInVzZXJfaWQiOiIxln0.Mbzn6LzsLrkVWEbhexR3ITYDZjxlcqW11rJxDQ6Ewk	是		令牌

Query：

参数名称	是否必须	示例	备注
page	是	1	当前页数
pagesize	是	10	页尺寸

名称	类型	是否必须	默认值	备注	其他信息
counts	integer	必须		总记录数	最大值: 5000 最小值: 100
pagesize	integer	必须		页大小	最大值: 50 最小值: 5
pages	integer	必须		总页数	最大值: 100 最小值: 1
page	integer	必须		当前页码	最大值: 100 最小值: 1
☐ items	object []	必须		列表	最小数量: 10 元素是否都不同: true 最大数量: 10 item 类型: object
id	string	必须		编号	
avatar	string	必须		头像	枚举: https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_1.png , https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_2.png , https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_3.png , https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/tanhua/avatar_4.png
nickname	string	必须		昵称	枚举: Wendy,谁动了我的冬天,卿本笨笨
createDate	string	必须		点赞时间	枚举: 2019-09-08 10:07

7.2、MessageLike

根据接口定义vo对象。

```
1 package com.tanhua.server.vo;  
2  
3 import lombok.AllArgsConstructor;  
4 import lombok.Data;  
5 import lombok.NoArgsConstructor;  
6  
7 @Data  
8 @NoArgsConstructor
```

```

9  @AllArgsConstructor
10 public class MessageLike {
11
12     private String id;
13     private String avatar;
14     private String nickname;
15     private String createDate;
16
17 }
18

```

7.3、dubbo接口

```

1  //QuanZiApi
2  /**
3   * 查询用户的评论数据
4   *
5   * @return
6   */
7  PageInfo<Comment> queryCommentListByUser(Long userId, Integer type,
Integer page, Integer pageSize);

```

实现：

```

1  @Override
2  public PageInfo<Comment> queryCommentListByUser(Long userId, Integer
type, Integer page, Integer pageSize) {
3
4      PageRequest pageRequest = PageRequest.of(page - 1, pageSize,
Sort.by(Sort.Order.desc("created")));
5      Query query = new Query(Criteria
6          .where("userId").is(userId)
7          .and("commentType").is(type)).with(pageRequest);
8
9      List<Comment> commentList = this.mongoTemplate.find(query,
Comment.class);
10
11      PageInfo<Comment> pageInfo = new PageInfo<>();
12      pageInfo.setPageNum(page);
13      pageInfo.setPageSize(pageSize);
14      pageInfo.setRecords(commentList);
15      pageInfo.setTotal(0); //不提供总数
16      return pageInfo;
17  }

```

7.4、编写接口服务

```

1  package com.tanhua.server.controller;
2
3  import com.tanhua.dubbo.server.vo.PageInfo;
4  import com.tanhua.server.service.IMService;
5  import com.tanhua.server.vo.Contacts;
6  import com.tanhua.server.vo.PageResult;
7  import org.slf4j.Logger;
8  import org.slf4j.LoggerFactory;

```

```

9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.http.HttpStatus;
11 import org.springframework.http.ResponseEntity;
12 import org.springframework.web.bind.annotation.*;
13
14 import java.util.Map;
15
16 @RestController
17 @RequestMapping("messages")
18 public class IMController {
19
20     private static final Logger LOGGER =
21     LoggerFactory.getLogger(IMController.class);
22
23     @Autowired
24     private IMService imService;
25
26     /**
27      * 查询点赞列表
28      *
29      * @param page
30      * @param pageSize
31      * @return
32      */
33     @GetMapping("likes")
34     public ResponseEntity<PageResult>
35     queryMessageLikeList(@RequestParam(value = "page", defaultValue = "1")
36     Integer page,
37
38     @RequestParam(value = "pagesize", defaultValue = "10") Integer pageSize) {
39         PageResult pageResult = this.imService.queryMessageLikeList(page,
40         pageSize);
41         return ResponseEntity.ok(pageResult);
42     }
43 }

```

```

1  package com.tanhua.server.service;
2
3  import com.alibaba.dubbo.config.annotation.Reference;
4  import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
5  import com.tanhua.dubbo.server.api.QuanZiApi;
6  import com.tanhua.dubbo.server.api.UsersApi;
7  import com.tanhua.dubbo.server.pojo.Comment;
8  import com.tanhua.dubbo.server.pojo.Users;
9  import com.tanhua.dubbo.server.vo.PageInfo;
10 import com.tanhua.server.pojo.User;
11 import com.tanhua.server.pojo.UserInfo;
12 import com.tanhua.server.utils.UserThreadLocal;
13 import com.tanhua.server.vo.Contacts;
14 import com.tanhua.server.vo.MessageLike;
15 import com.tanhua.server.vo.PageResult;
16 import org.apache.commons.lang3.StringUtils;
17 import org.joda.time.DateTime;
18 import org.springframework.beans.factory.annotation.Autowired;

```

```

19 import org.springframework.beans.factory.annotation.Value;
20 import org.springframework.http.ResponseEntity;
21 import org.springframework.stereotype.Service;
22 import org.springframework.web.client.RestTemplate;
23
24 import java.util.ArrayList;
25 import java.util.List;
26
27 @Service
28 public class IMService {
29
30     @Reference(version = "1.0.0")
31     private UsersApi usersApi;
32
33     @Autowired
34     private RestTemplate restTemplate;
35
36     @Value("${tanhua.sso.url}")
37     private String url;
38
39     @Autowired
40     private UserInfoService userInfoService;
41
42     @Reference(version = "1.0.0")
43     private QuanZiApi quanZiApi;
44
45     public PageResult queryMessageLikeList(Integer page, Integer pageSize)
46     {
47         User user = UserThreadLocal.get();
48         PageInfo<Comment> pageInfo =
49         this.quanZiApi.queryCommentListByUser(user.getId(), 1, page, pageSize);
50
51         PageResult pageResult = new PageResult();
52         pageResult.setPage(page);
53         pageResult.setPages(0);
54         pageResult.setCounts(0);
55         pageResult.setPagesize(pageSize);
56
57         List<Comment> records = pageInfo.getRecords();
58
59         List<Long> userIds = new ArrayList<>();
60         for (Comment comment : records) {
61             userIds.add(comment.getUserId());
62         }
63
64         QueryWrapper<UserInfo> querywrapper = new QueryWrapper<>();
65         querywrapper.in("user_id", userIds);
66         List<UserInfo> userInfoList =
67         this.userInfoService.queryList(querywrapper);
68
69         List<MessageLike> messageLikeList = new ArrayList<>();
70         for (Comment record : records) {
71             for (UserInfo userInfo : userInfoList) {
72                 if(userInfo.getUserId().longValue() ==
73                 record.getUserId().longValue()){
74
75                     MessageLike messageLike = new MessageLike();
76                     messageLike.setId(record.getId().toHexString());

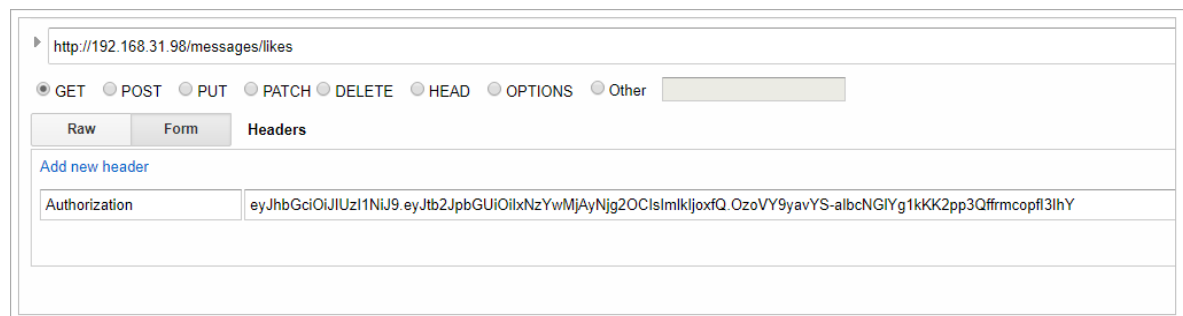
```

```

73         messageLike.setAvatar(userInfo.getLogo());
74         messageLike.setNickname(userInfo.getNickName());
75         messageLike.setCreateDate(new
DateTime(record.getCreated()).toString("yyyy-MM-dd HH:mm"));
76
77         messageLikeList.add(messageLike);
78         break;
79     }
80 }
81 }
82
83     pageResult.setItems(messageLikeList);
84     return pageResult;
85 }
86 }
87

```

7.5、测试



8、评论、喜欢列表

8.1、IMController

```

1  /**
2      * 查询评论列表
3      *
4      * @param page
5      * @param pageSize
6      * @return
7      */
8  @GetMapping("comments")

```

```

9      public ResponseEntity<PageResult>
queryMessageCommentList(@RequestParam(value = "page", defaultValue = "1")
Integer page,
10
    @RequestParam(value = "pagesize", defaultValue = "10") Integer pageSize) {
11          PageResult pageResult =
this.imService.queryMessageCommentList(page, pageSize);
12          return ResponseEntity.ok(pageResult);
13      }
14
15      /**
16       * 查询喜欢列表
17       *
18       * @param page
19       * @param pageSize
20       * @return
21       */
22      @GetMapping("loves")
23      public ResponseEntity<PageResult>
queryMessageLoveList(@RequestParam(value = "page", defaultValue = "1")
Integer page,
24
    @RequestParam(value = "pagesize", defaultValue = "10") Integer pageSize) {
25          PageResult pageResult = this.imService.queryMessageLoveList(page,
pageSize);
26          return ResponseEntity.ok(pageResult);
27      }

```

8.2、IMService

```

1  public PageResult queryMessageLikeList(Integer page, Integer pageSize) {
2      return this.messageCommentList(1, page, pageSize);
3  }
4
5  public PageResult queryMessageCommentList(Integer page, Integer
pageSize) {
6      return this.messageCommentList(2, page, pageSize);
7  }
8
9  public PageResult queryMessageLoveList(Integer page, Integer pageSize)
{
10     return this.messageCommentList(3, page, pageSize);
11 }
12
13 private PageResult messageCommentList(Integer type, Integer page,
Integer pageSize) {
14     User user = UserThreadLocal.get();
15     PageInfo<Comment> pageInfo =
this.quanZiApi.queryCommentListByUser(user.getId(), type, page, pageSize);
16
17     PageResult pageResult = new PageResult();
18     pageResult.setPage(page);
19     pageResult.setPages(0);
20     pageResult.setCounts(0);
21     pageResult.setPagesize(pageSize);
22
23     List<Comment> records = pageInfo.getRecords();

```

```

24
25     List<Long> userIds = new ArrayList<>();
26     for (Comment comment : records) {
27         userIds.add(comment.getUserId());
28     }
29
30     QueryWrapper<UserInfo> queryWrapper = new QueryWrapper<>();
31     queryWrapper.in("user_id", userIds);
32     List<UserInfo> userInfoList =
this.userInfoService.queryList(queryWrapper);
33
34     List<MessageLike> messageLikeList = new ArrayList<>();
35     for (Comment record : records) {
36         for (UserInfo userInfo : userInfoList) {
37             if (userInfo.getUserId().longValue() ==
record.getUserId().longValue()) {
38
39                 MessageLike messageLike = new MessageLike();
40                 messageLike.setId(record.getId().toHexString());
41                 messageLike.setAvatar(userInfo.getLogo());
42                 messageLike.setNickname(userInfo.getNickName());
43                 messageLike.setCreateDate(new
DateTime(record.getCreated()).toString("yyyy-MM-dd HH:mm"));
44
45                 messageLikeList.add(messageLike);
46                 break;
47             }
48         }
49     }
50
51     pageResult.setItems(messageLikeList);
52     return pageResult;
53 }

```

8.3、解决bug

点赞、评论、喜欢列表应该是别人对我发布的信息做了操作之后显示的数据，所以查询条件是发布人的id作为查询条件。

第一步：修改Comment对象，增加 publishUserId 字段。

```

1  package com.tanhua.dubbo.server.pojo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6  import org.bson.types.ObjectId;
7  import org.springframework.data.mongodb.core.mapping.Document;
8
9  import java.util.Date;
10
11  /**
12   * 评论表
13   */
14  @Data
15  @NoArgsConstructor
16  @AllArgsConstructor

```

```

17 @Document(collection = "quanzi_comment")
18 public class Comment implements java.io.Serializable{
19
20     private static final long serialVersionUID = -291788258125767614L;
21
22     private ObjectId id;
23
24     private ObjectId publishId; //发布id
25     private Integer commentType; //评论类型, 1-点赞, 2-评论, 3-喜欢
26     private String content; //评论内容
27     private Long userId; //评论人
28     private Long publishUserId; //发布人的用户id
29
30     private Boolean isParent = false; //是否为父节点, 默认是否
31     private ObjectId parentId; // 父节点id
32
33     private Long created; //发表时间
34
35 }
36

```

第二步：修改保存Comment逻辑

```

1  @Override
2      public boolean saveComment(Long userId, String publishId, Integer type,
3      String content) {
4
5          try {
6              Comment comment = new Comment();
7              comment.setContent(content);
8              comment.setIsParent(true);
9              comment.setCommentType(type);
10             comment.setPublishId(new ObjectId(publishId));
11             comment.setUserId(userId);
12             comment.setId(ObjectId.get());
13             comment.setCreated(System.currentTimeMillis());
14
15             // 设置发布人的id
16             Publish publish =
17             this.mongoTemplate.findById(comment.getPublishId(), Publish.class);
18             if (null != publish) {
19                 comment.setPublishUserId(publish.getUserId());
20             } else {
21                 Video video =
22                 this.mongoTemplate.findById(comment.getPublishId(), Video.class);
23                 if (null != video) {
24                     comment.setPublishUserId(video.getUserId());
25                 }
26             }
27
28             this.mongoTemplate.save(comment);
29
30             return true;
31         } catch (Exception e) {
32             e.printStackTrace();
33         }
34     }
35

```

```

32         return false;
33     }

```

第三步，修改查询条件

```

1  @Override
2      public PageInfo<Comment> queryCommentListByUser(Long userId, Integer
type, Integer page, Integer pageSize) {
3
4      PageRequest pageRequest = PageRequest.of(page - 1, pageSize,
Sort.by(Sort.Order.desc("created")));
5      Query query = new Query(Criteria
6          .where("publishUserId").is(userId)
7          .and("commentType").is(type)).with(pageRequest);
8
9      List<Comment> commentList = this.mongoTemplate.find(query,
Comment.class);
10
11      PageInfo<Comment> pageInfo = new PageInfo<>();
12      pageInfo.setPageNum(page);
13      pageInfo.setPageSize(pageSize);
14      pageInfo.setRecords(commentList);
15      pageInfo.setTotal(0); //不提供总数
16      return pageInfo;
17  }

```

9、公告列表

公告是后台系统对所有用户发布的公告消息。

9.1、表结构

```

1  CREATE TABLE `tb_announcement` (
2      `id` bigint(20) NOT NULL AUTO_INCREMENT,
3      `title` varchar(200) DEFAULT NULL COMMENT '标题',
4      `description` text COMMENT '描述',
5      `created` datetime DEFAULT NULL,
6      `updated` datetime DEFAULT NULL,
7      PRIMARY KEY (`id`),
8      KEY `created` (`created`)
9  ) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COMMENT='公告表';

```

```

1  --插入数据
2  INSERT INTO `tb_announcement` (`id`, `title`, `description`, `created`,
`updated`) VALUES ('1', '探花新版本上线发布啦~,盛夏high趴开始了,赶紧来报名吧!',
'探花App2019年7月23日起在苹果商店...,浓情夏日,清爽一聚,探花将吧大家聚...', '2019-10-14
11:06:34', '2019-10-14 11:06:37');
3  INSERT INTO `tb_announcement` (`id`, `title`, `description`, `created`,
`updated`) VALUES ('2', '探花交友的圈子功能正式上线啦~~', '探花交友的圈子功能正式上线
啦,欢迎使用~', '2019-10-14 11:09:31', '2019-10-14 11:09:33');
4  INSERT INTO `tb_announcement` (`id`, `title`, `description`, `created`,
`updated`) VALUES ('3', '国庆放假期间,探花交友正常使用~', '国庆放假期间,探花交友正常
使用~', '2019-10-14 11:10:01', '2019-10-14 11:10:04');
5

```

9.2、 pojo

```
1 package com.tanhua.server.pojo;
2
3 import lombok.AllArgsConstructor;
4 import lombok.Data;
5 import lombok.NoArgsConstructor;
6
7 @Data
8 @NoArgsConstructor
9 @AllArgsConstructor
10 public class Announcement extends BasePojo {
11
12     private Long id;
13     private String title;
14     private String description;
15
16 }
17
```

9.3、 AnnouncementMapper

```
1 package com.tanhua.server.mapper;
2
3 import com.baomidou.mybatisplus.core.mapper.BaseMapper;
4 import com.tanhua.server.pojo.Announcement;
5
6 public interface AnnouncementMapper extends BaseMapper<Announcement> {
7 }
8
```

9.4、 AnnouncementService

```
1 package com.tanhua.server.service;
2
3 import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
4 import com.baomidou.mybatisplus.core.metadata.IPage;
5 import com.baomidou.mybatisplus.extension.plugins.pagination.Page;
6 import com.tanhua.server.mapper.AnnouncementMapper;
7 import com.tanhua.server.pojo.Announcement;
8 import org.springframework.beans.factory.annotation.Autowired;
9 import org.springframework.stereotype.Service;
10
11 @Service
12 public class AnnouncementService {
13
14     @Autowired
15     private AnnouncementMapper announcementMapper;
16
17
18     public IPage<Announcement> queryList(Integer page, Integer pageSize) {
19         QueryWrapper queryWrapper = new QueryWrapper();
20         queryWrapper.orderByDesc("created");
21     }
22 }

```

```

21         return this.announcementMapper.selectPage(new Page<Announcement>
22             (page, pageSize), queryWrapper);
23     }
24 }

```

9.5、定义vo对象

```

1  package com.tanhua.server.vo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  @Data
8  @NoArgsConstructor
9  @AllArgsConstructor
10 public class MessageAnnouncement {
11
12     private String id;
13     private String title;
14     private String description;
15     private String createDate;
16
17 }

```

9.6、IMController

```

1  /**
2   * 查询公告列表
3   *
4   * @param page
5   * @param pageSize
6   * @return
7   */
8  @GetMapping("announcements")
9  @NoAuthorization //优化, 无需进行token校验
10 public ResponseEntity<PageResult>
11 queryMessageAnnouncementList(@RequestParam(value = "page", defaultValue =
12     "1") Integer page,
13 @RequestParam(value = "pagesize", defaultValue = "10") Integer pageSize) {
14     PageResult pageResult =
15     this.imService.queryMessageAnnouncementList(page, pageSize);
16     return ResponseEntity.ok(pageResult);
17 }

```

9.7、IMService

```

1  public PageResult queryMessageAnnouncementList(Integer page, Integer
2     pageSize) {
3     IPage<Announcement> announcementPage =
4     this.announcementService.queryList(page, pageSize);
5
6 }

```

```

4      List<MessageAnnouncement> messageAnnouncementList = new ArrayList<>
      ();
5
6      for (Announcement record : announcementPage.getRecords()) {
7          MessageAnnouncement messageAnnouncement = new
      MessageAnnouncement();
8          messageAnnouncement.setId(record.getId().toString());
9          messageAnnouncement.setTitle(record.getTitle());
10         messageAnnouncement.setDescription(record.getDescription());
11         messageAnnouncement.setCreateDate(new
      DateTime(record.getCreated()).toString("yyyy-MM-dd HH:mm"));
12
13         messageAnnouncementList.add(messageAnnouncement);
14     }
15
16     PageResult<PageResult> pageResult = new PageResult();
17     pageResult.setPage(page);
18     pageResult.setPages(0);
19     pageResult.setCounts(0);
20     pageResult.setPagesize(pageSize);
21     pageResult.setItems(messageAnnouncementList);
22
23     return pageResult;
24 }

```

9.8、测试

http://192.168.31.98/messages/announcements

☒ GET
 ☐ POST
 ☐ PUT
 ☐ PATCH
 ☐ DELETE
 ☐ HEAD
 ☐ OPTIONS
 ☐ Other

Raw Form Headers

[Add new header](#)

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJjb2JpbGUuOmxNzYwMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
---------------	---

Raw	JSON	Response
Copy to clipboard Save as file		
<pre>{ counts: 0 pagesize: 10 pages: 0 page: 1 -items: [3] -0: { id: "3" title: "国庆放假期间，探花交友正常使用~" description: "国庆放假期间，探花交友正常使用~" createDate: "2019-10-14 11:10" } -1: { id: "2" title: "探花交友的圈子功能正式上线啦~~" description: "探花交友的圈子功能正式上线啦，欢迎使用~" createDate: "2019-10-14 11:09" } -2: { id: "1" title: "探花新版本上线发布啦~, 盛夏high趴开始了，赶紧来报名吧！" description: "探花App2019年7月23日起在苹果商店..., 浓情夏日，清爽一聚，探花将吧大家聚..." createDate: "2019-10-14 11:06" } } }</pre>		

11:35

...0.0K/s 4G 蓝牙 闹钟 信号 79



公告



国庆放假期间，探花交友正常使用~

国庆放假期间，探花交友正常使用~

2019-10-14 11:10



探花交友的圈子功能正式上线啦~~

探花交友的圈子功能正式上线啦，欢迎使用~

2019-10-14 11:09



探花新版本上线发布啦~，盛夏high趴开始了

探花App2019年7月23日起在苹果商店...，浓情夏日

2019-10-14 11:06



T OVERFLOWED BY 122 PIXELS