

课程说明

- 首页好友推荐的功能实现
- 谁看过我的功能实现
- 今日佳人的功能实现
- 聊一下的功能实现
- 上报地理位置的功能实现

1、首页好友推荐

1.1、默认推荐列表

如果查询不到推荐列表，那就需要给出默认的推荐列表。

1.1.1、配置默认推荐的用户

```
1 #application.properties
2 tanhua.sso.default.recommend.users=2,3,4,5,6,7,8,9,10,11,12,13
```

1.1.2、修改查询逻辑

```
1 //TodayBestService
2
3 /**
4  * 查询推荐用户列表
5  *
6  * @param queryParams
7  * @param token
8  * @return
9  */
10 public PageResult queryRecommendUserList(RecommendUserQueryParam
    queryParams, String token) {
11     //查询当前的登录信息
12     User user = this.userService.queryUserByToken(token);
13     if (null == user) {
14         return null;
15     }
16     PageResult pageResult = new PageResult();
17     PageInfo<RecommendUser> pageInfo =
18         this.recommendUserService.queryRecommendUserList(user.getId(),
19             queryParams.getPage(), queryParams.getPagesize());
20
21     pageResult.setCounts(0); //前端不参与计算，仅需要返回字段
22     pageResult.setPage(queryParams.getPage());
23     pageResult.setPagesize(queryParams.getPagesize());
24
25     List<RecommendUser> records = pageInfo.getRecords();
26
27     if(CollectionUtils.isEmpty(records)){
```

```

27         //默认推荐列表
28         String[] ss = StringUtils.split(defaultRecommendUsers, ',');
29         for (String s : ss) {
30             RecommendUser recommendUser = new RecommendUser();
31
32             recommendUser.setUserId(Long.valueOf(s));
33             recommendUser.setToUserId(user.getId());
34             recommendUser.setScore(RandomUtils.nextDouble(70, 99));
35
36             records.add(recommendUser);
37         }
38     }
39
40     List<Long> userIds = new ArrayList<>();
41     for (RecommendUser record : records) {
42         userIds.add(record.getUserId());
43     }
44
45     QueryWrapper<UserInfo> queryWrapper = new QueryWrapper<>();
46     queryWrapper.in("user_id", userIds);
47
48     if (StringUtils.isNotEmpty(queryParam.getGender())) { //性别条件
49         if (queryParam.getGender().equals("man")) {
50             queryWrapper.eq("sex", 1);
51         } else {
52             queryWrapper.eq("sex", 2);
53         }
54     }
55
56     if (StringUtils.isNotEmpty(queryParam.getCity())) { //居住城市
57         queryWrapper.eq("city", queryParam.getCity());
58     }
59
60     if (queryParam.getAge() != null) { //年龄
61         queryWrapper.lt("age", queryParam.getAge());
62     }
63
64     List<UserInfo> userInfos =
this.userService.queryList(queryWrapper);
65
66     List<TodayBest> todayBests = new ArrayList<>();
67     for (UserInfo userInfo : userInfos) {
68         TodayBest todayBest = new TodayBest();
69         todayBest.setId(userInfo.getUserId());
70         todayBest.setAge(userInfo.getAge());
71         todayBest.setAvatar(userInfo.getLogo());
72         todayBest.setGender(userInfo.getSex().name().toLowerCase());
73         todayBest.setNickname(userInfo.getNickName());
74         todayBest.setTags(StringUtils.split(userInfo.getTags(), ','));
75
76         //设置缘分值
77         for (RecommendUser record : records) {
78             if(userInfo.getUserId().longValue() ==
record.getUserId().longValue()){
79                 double score = Math.floor(record.getScore());
80
81                 todayBest.setFateValue(Double.valueOf(score).longValue());
82                 break;

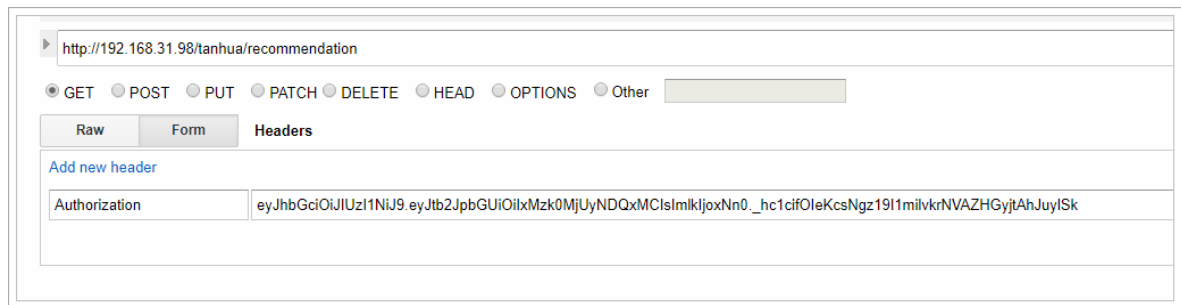
```

```

82         }
83     }
84
85     todayBests.add(todayBest);
86 }
87
88 //排序集合，按照score倒序排序
89 Collections.sort(todayBests, (o1, o2) -> new Long(o2.getFateValue()
- o1.getFateValue()).intValue());
90
91 pageResult.setItems(todayBests);
92
93 return pageResult;
94 }

```

1.1.3、测试



http://192.168.31.98/tanhua/recommendation

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJjb2JpbGUuOiIxMzk0MjUyNDQxMCIsImkiOiJoxNn0_hc1cifOIeKcsNgz19l1milvkrNVAZHGYjtAhJuyISk
---------------	--

响应结果：

```

1  {
2      "counts": 0,
3      "pagesize": 10,
4      "pages": 0,
5      "page": 1,
6      "items": [
7          {
8              "id": 4,
9              "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/4.jpg",
10             "nickname": "heima_4",
11             "gender": "man",
12             "age": 22,
13             "tags": [
14                 "单身",
15                 "本科",
16                 "年龄相仿"
17             ],
18             "fateValue": 95
19         },
20         {
21             "id": 9,
22             "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/8.jpg",
23             "nickname": "heima_9",
24             "gender": "man",
25             "age": 23,
26             "tags": [
27                 "单身",

```

```
28         "本科",
29         "年龄相仿"
30     ],
31     "fateValue": 95
32 },
33 {
34     "id": 5,
35     "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/5.jpg",
36     "nickname": "heima_5",
37     "gender": "man",
38     "age": 23,
39     "tags": [
40         "单身",
41         "本科",
42         "年龄相仿"
43     ],
44     "fateValue": 94
45 },
46 {
47     "id": 6,
48     "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/9.jpg",
49     "nickname": "heima_6",
50     "gender": "man",
51     "age": 23,
52     "tags": [
53         "单身",
54         "本科",
55         "年龄相仿"
56     ],
57     "fateValue": 94
58 },
59 {
60     "id": 2,
61     "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/22.jpg",
62     "nickname": "heima_2",
63     "gender": "man",
64     "age": 30,
65     "tags": [
66         "单身",
67         "本科",
68         "年龄相仿"
69     ],
70     "fateValue": 93
71 },
72 {
73     "id": 8,
74     "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/12.jpg",
75     "nickname": "heima_8",
76     "gender": "man",
77     "age": 26,
78     "tags": [
79         "单身",
80         "本科",
81         "年龄相仿"
```

```
82     ],
83     "fateValue": 92
84 },
85 {
86     "id": 11,
87     "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/11.jpg",
88     "nickname": "heima_11",
89     "gender": "man",
90     "age": 46,
91     "tags": [
92         "单身",
93         "本科",
94         "年龄相仿"
95     ],
96     "fateValue": 86
97 },
98 {
99     "id": 12,
100    "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/10.jpg",
101    "nickname": "heima_12",
102    "gender": "man",
103    "age": 40,
104    "tags": [
105        "单身",
106        "本科",
107        "年龄相仿"
108    ],
109    "fateValue": 86
110 },
111 {
112     "id": 13,
113     "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/5.jpg",
114     "nickname": "heima_13",
115     "gender": "man",
116     "age": 46,
117     "tags": [
118         "单身",
119         "本科",
120         "年龄相仿"
121     ],
122     "fateValue": 85
123 },
124 {
125     "id": 7,
126     "avatar": "https://itcast-tanhua.oss-cn-
shanghai.aliyuncs.com/images/logo/18.jpg",
127     "nickname": "heima_7",
128     "gender": "man",
129     "age": 42,
130     "tags": [
131         "单身",
132         "本科",
133         "年龄相仿"
134     ],
135     "fateValue": 78
```

```
136     },
137     {
138         "id": 3,
139         "avatar": "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/19.jpg",
140         "nickname": "heima_3",
141         "gender": "man",
142         "age": 45,
143         "tags": [
144             "单身",
145             "本科",
146             "年龄相仿"
147         ],
148         "fateValue": 77
149     },
150     {
151         "id": 10,
152         "avatar": "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/4.jpg",
153         "nickname": "heima_10",
154         "gender": "man",
155         "age": 46,
156         "tags": [
157             "单身",
158             "本科",
159             "年龄相仿"
160         ],
161         "fateValue": 76
162     }
163 ]
164 }
```

1.2、好友推荐

对于好友的推荐，需要找出每个用户之间的相似性，具体规则如下：

字段	权重分			
年龄差	0-2岁 30分	3-5 20分	5-10岁 10分	10岁以上 0分
性别	异性 30分	同性 0分		
位置	同城 20分	不同 0分		
学历	相同 20分	不同 0分		

1.2.1、编写Spark程序

配置：

```

1 spark.mongodb.input.uri =
  mongodb://192.168.31.81:27017/tanhua.recommend_quanzi_20191001?
  readPreference=primaryPreferred
2 spark.video.mongodb.input.uri =
  mongodb://192.168.31.81:27017/tanhua.recommend_video_20191007?
  readPreference=primaryPreferred
3 spark.mongodb.output.user.uri =
  mongodb://192.168.31.81:27017/tanhua.recommend_user?
  readPreference=primaryPreferred
4
5 redis.cluster.nodes =
  192.168.31.81:6379,192.168.31.81:6380,192.168.31.81:6381
6
7 jdbc.driver-class-name=com.mysql.jdbc.Driver
8 jdbc.url=jdbc:mysql://127.0.0.1:3306/tanhua?
  useUnicode=true&characterEncoding=utf8&autoReconnect=true&allowMultiQueries
  =true&useSSL=false
9 jdbc.username=root
10 jdbc.password=root

```

添加MySQL驱动：

```

1 <dependency>
2   <groupId>mysql</groupId>
3   <artifactId>mysql-connector-java</artifactId>
4 </dependency>

```

```

1 package com.tanhua.spark.mongo;
2
3 import com.mongodb.spark.MongoSpark;
4 import org.apache.commons.lang.math.RandomUtils;
5 import org.apache.commons.lang3.StringUtils;
6 import org.apache.spark.SparkConf;
7 import org.apache.spark.api.java.JavaPairRDD;
8 import org.apache.spark.api.java.JavaRDD;
9 import org.apache.spark.api.java.JavaSparkContext;
10 import org.apache.spark.mllib.recommendation.MatrixFactorizationModel;
11 import org.apache.spark.mllib.recommendation.Rating;
12 import org.apache.spark.sql.Row;
13 import org.apache.spark.sql.SparkSession;
14 import org.bson.Document;
15 import org.bson.types.ObjectId;
16 import org.joda.time.DateTime;
17 import scala.Tuple2;
18
19 import java.io.InputStream;
20 import java.math.BigDecimal;
21 import java.util.Arrays;
22 import java.util.List;
23 import java.util.Properties;
24
25 public class SparkUserRecommend {
26
27     public static void main(String[] args) throws Exception {

```

```

28
29 //读取外部的配置文件
30 InputStream inputStream =
SparkUserRecommend.class.getClassLoader().getResourceAsStream("app.properties");
31 Properties properties = new Properties();
32 properties.load(inputStream);
33
34 //构建Spark配置
35 SparkConf sparkConf = new SparkConf()
36     .setAppName("SparkUserRecommend")
37     .setMaster("local[*]")
38     .set("spark.mongodb.output.uri",
properties.getProperty("spark.mongodb.output.user.uri"));
39
40 //构建Spark上下文环境
41 SparkSession sparkSession =
SparkSession.builder().config(sparkConf).getOrCreate();
42 String url = properties.getProperty("jdbc.url");
43
44 // 设置数据库连接信息
45 Properties connectionProperties = new Properties();
46 connectionProperties.put("driver",
properties.getProperty("jdbc.driver-class-name"));
47 connectionProperties.put("user",
properties.getProperty("jdbc.username"));
48 connectionProperties.put("password",
properties.getProperty("jdbc.password"));
49
50 //读取用户信息表数据
51 JavaRDD<Row> userInfoJavaRDD = sparkSession.read().
52     jdbc(url, "tb_user_info",
connectionProperties).toJavaRDD();
53
54 // 获取用户id列表
55 List<Long> userIdList = userInfoJavaRDD.map(v1 ->
v1.getLong(1)).collect();
56
57 //计算笛卡尔积
58 JavaPairRDD<Row, Row> cartesian =
userInfoJavaRDD.cartesian(userInfoJavaRDD);
59
60 JavaPairRDD<Long, Rating> userJavaPairRDD =
cartesian.mapToPair(rowRowTuple2 -> {
61     Row row1 = rowRowTuple2._1();
62     Row row2 = rowRowTuple2._2();
63
64     Long userId1 = row1.getLong(1);
65     Long userId2 = row2.getLong(1);
66
67     // 随机保证数据均匀分配
68     Long key = userId1 + userId2 + RandomUtils.nextLong();
69
70     if (userId1.longValue() == userId2.longValue()) {
71         // 相同数据
72         return new Tuple2<>(key % 10, new
Rating(userId1.intValue(), userId2.intValue(), 0));
73     }

```

```

74
75         double score = 0;
76
77         // 计算年龄差
78         int ageDiff = Math.abs(row1.getInt(6) - row2.getInt(6));
79         if (ageDiff <= 2) {
80             score += 30;
81         } else if (ageDiff >= 3 && ageDiff <= 5) {
82             score += 20;
83         } else if (ageDiff >= 6 && ageDiff <= 10) {
84             score += 10;
85         }
86
87         // 计算性别
88         if (row1.getInt(5) != row2.getInt(5)) {
89             score += 30;
90         }
91
92         // 计算城市
93         String city1 = StringUtils.substringBefore(row1.getString(8),
94 "-");
95         String city2 = StringUtils.substringBefore(row2.getString(8),
96 "-");
97
98         if (StringUtils.equals(city1, city2)) {
99             score += 20;
100         }
101
102         // 计算学历
103         String edu1 = row1.getString(7);
104         String edu2 = row2.getString(7);
105         if (StringUtils.equals(edu1, edu2)) {
106             score += 20;
107         }
108
109         Rating rating = new Rating(userId1.intValue(),
110 userId2.intValue(), score);
111         return new Tuple2<>(key % 10, rating);
112     });
113
114     //通过MLlib模型进行推荐，获取到最优的推荐模型
115     MLlibRecommend mllibRecommend = new MLlibRecommend();
116     MatrixFactorizationModel bestModel =
117 mllibRecommend.bestModel(userJavaPairRDD);
118
119     //用于MongoDB的写入
120     JavaSparkContext jsc = new
121 JavaSparkContext(sparkSession.sparkContext());
122     for (Long userId : userIdList) {
123         // 为用户推荐50个相似用户
124         Rating[] recommendProducts =
125 bestModel.recommendProducts(userId.intValue(), 50);
126
127         JavaRDD<Document> docs =
128 jsc.parallelize(Arrays.asList(recommendProducts)).map(v1 -> {
129             Document document = new Document();
130
131             document.put("_id", ObjectId.get());
132             document.put("userId", v1.product());
133         });
134     }

```

```

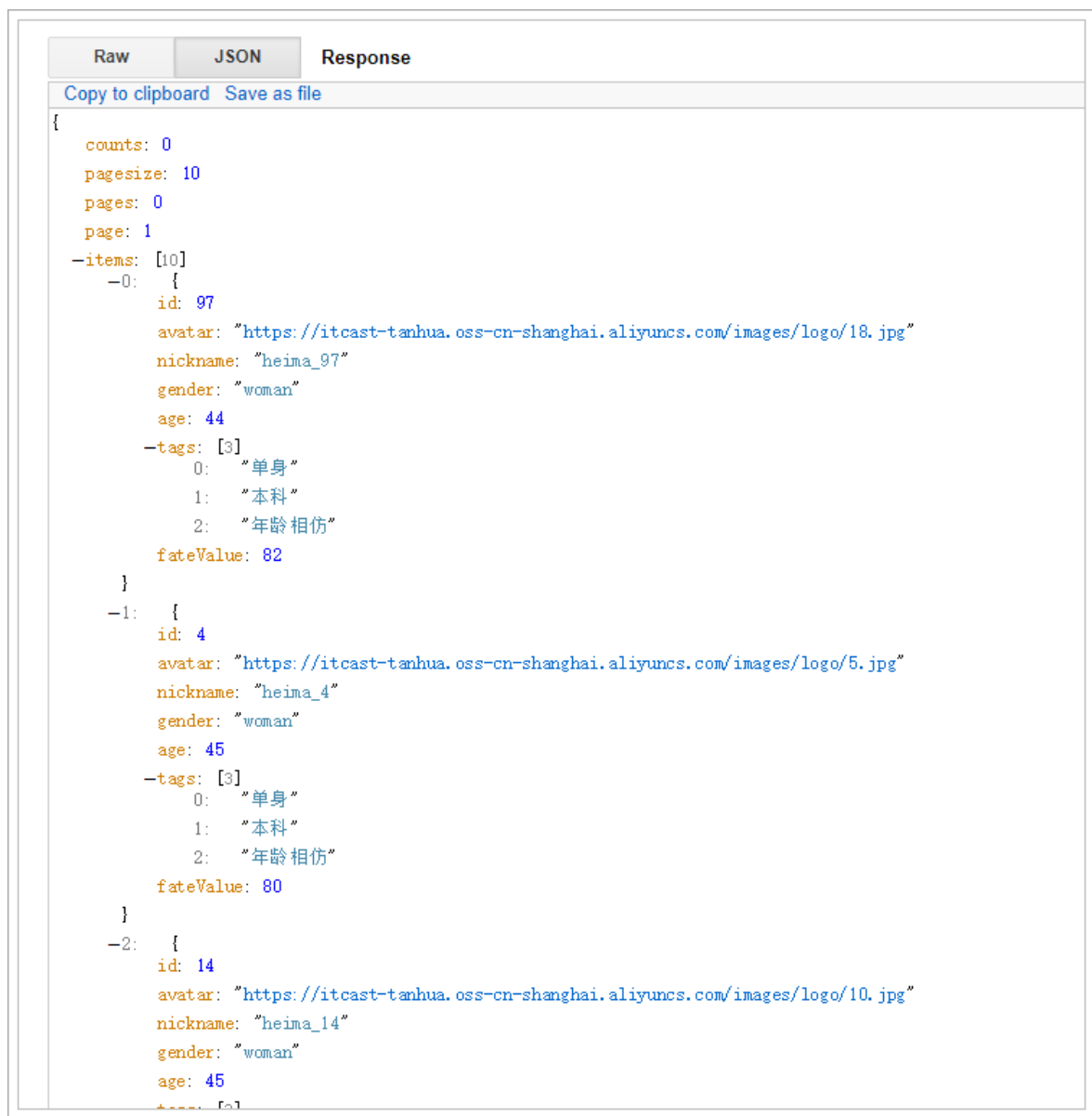
125         document.put("toUserId", v1.user());
126         //得分, 保留2位小数
127         double score = new BigDecimal(v1.rating()).setScale(2,
BigDecimal.ROUND_DOWN).doubleValue();
128         document.put("score", score);
129         document.put("date", new
DateTime().toString("yyyy/MM/dd"));
130
131         return document;
132     });
133
134     // 写数据到MongoDB
135     MongoSpark.save(docs);
136 }
137
138 }
139 }
140

```

1.2.2、测试

▲ (39) ObjectId("5daa706c77da9331843fa012")	{ 5 fields }	Object
_id	ObjectId("5daa706c77da9331843fa012")	ObjectId
userId	28	Int32
toUserId	10	Int32
score	43.45	Double
date	2019/10/19	String
▶ (40) ObjectId("5daa706c77da9331843fa016")	{ 5 fields }	Object
▶ (41) ObjectId("5daa706c77da9331843fa018")	{ 5 fields }	Object
▶ (42) ObjectId("5daa706c77da9331843fa01c")	{ 5 fields }	Object
▶ (43) ObjectId("5daa706c77da9331843fa021")	{ 5 fields }	Object
▲ (44) ObjectId("5daa706c77da9331843fa022")	{ 5 fields }	Object
_id	ObjectId("5daa706c77da9331843fa022")	ObjectId
userId	80	Int32
toUserId	10	Int32
score	40.15	Double
date	2019/10/19	String
▶ (45) ObjectId("5daa706c77da9331843fa025")	{ 5 fields }	Object
▲ (46) ObjectId("5daa706c77da9331843fa026")	{ 5 fields }	Object
_id	ObjectId("5daa706c77da9331843fa026")	ObjectId
userId	67	Int32
toUserId	10	Int32

可以看到数据已经写入到了MongoDB。



2、谁看过我

记录别人来访了我的主页的信息。

2.1、编写pojo

```
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 @Data
10 @NoArgsConstructor
11 @AllArgsConstructor
12 @Document(collection = "visitors")
13 public class Visitors implements java.io.Serializable{
14
15     private static final long serialVersionUID = 2811682148052386573L;
16 }
```

```

17     private ObjectId id;
18     private Long userId; //我的id
19     private Long visitorUserId; //来访用户id
20     private String from; //来源，如首页、圈子等
21     private Long date; //来访时间
22
23     private Double score; //得分
24
25 }
26

```

2.2、定义接口

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.tanhua.dubbo.server.pojo.Visitors;
4
5  import java.util.List;
6
7  public interface VisitorsApi {
8
9      /**
10       * 保存来访记录
11       *
12       * @param visitors
13       * @return
14       */
15     String saveVisitor(Visitors visitors);
16
17     /**
18      * 按照时间倒序排序，查询最近的访客信息
19      *
20      * @param userId
21      * @param num
22      * @return
23      */
24     List<Visitors> topVisitor(Long userId, Integer num);
25
26     /**
27      * 按照时间倒序排序，查询最近的访客信息
28      *
29      * @param userId
30      * @param date
31      * @return
32      */
33     List<Visitors> topVisitor(Long userId, Long date);
34 }
35

```

2.3、实现接口

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.alibaba.dubbo.config.annotation.Service;
4  import com.tanhua.dubbo.server.pojo.RecommendUser;
5  import com.tanhua.dubbo.server.pojo.Visitors;
6

```

```

6  import org.bson.types.ObjectId;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.data.domain.PageRequest;
9  import org.springframework.data.domain.Pageable;
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 VisitorsApiImpl implements VisitorsApi {
19
20     @Autowired
21     private MongoTemplate mongoTemplate;
22
23     @Override
24     public String saveVisitor(Visitors visitors) {
25
26         visitors.setId(ObjectId.get());
27         visitors.setDate(System.currentTimeMillis());
28
29         this.mongoTemplate.save(visitors);
30
31         return visitors.getId().toHexString();
32     }
33
34     @Override
35     public List<Visitors> topVisitor(Long userId, Integer num) {
36         Pageable pageable = PageRequest.of(0, num,
37             Sort.by(Sort.Order.desc("date")));
38         Query query =
39             Query.query(Criteria.where("userId").is(userId)).with(pageable);
40         return this.queryVisitorList(query);
41     }
42
43     @Override
44     public List<Visitors> topVisitor(Long userId, Long date) {
45         Query query = Query.query(Criteria
46             .where("userId").is(userId)
47             .and("date").lt(date));
48         return this.queryVisitorList(query);
49     }
50
51     private List<Visitors> queryVisitorList(Query query) {
52         List<Visitors> visitors = this.mongoTemplate.find(query,
53             Visitors.class);
54
55         // 查询得分
56         for (Visitors visitor : visitors) {
57             Query queryRecommend = Query.query(Criteria
58                 .where("toUserId").is(visitor.getUserId())
59                 .and("userId").is(visitor.getVisitorUserId()));
60             RecommendUser recommendUser =
61                 this.mongoTemplate.findOne(queryRecommend, RecommendUser.class);
62             if (null != recommendUser) {
63                 visitor.setScore(recommendUser.getScore());
64             }
65         }
66     }
67 }

```

```

60         } else {
61             visitor.setScore(30d);
62         }
63     }
64
65     return visitors;
66 }
67 }
68

```

2.4、测试

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.tanhua.dubbo.server.pojo.Visitors;
4  import org.apache.commons.lang3.RandomUtils;
5  import org.bson.types.ObjectId;
6  import org.junit.Test;
7  import org.junit.runner.RunWith;
8  import org.springframework.beans.factory.annotation.Autowired;
9  import org.springframework.boot.test.context.SpringBootTest;
10 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
11
12 @RunWith(SpringJUnit4ClassRunner.class)
13 @SpringBootTest
14 public class TestVisitors {
15
16     @Autowired
17     private VisitorsApi visitorsApi;
18
19     @Test
20     public void testSave(){
21         for (int i = 0; i < 100; i++) {
22             Visitors visitors = new Visitors();
23
24             visitors.setFrom("首页");
25             visitors.setUserId(RandomUtils.nextLong(1,10));
26             visitors.setVisitorUserId(RandomUtils.nextLong(11,50));
27
28             this.visitorsApi.saveVisitor(visitors);
29         }
30
31         System.out.println("ok");
32
33     }
34 }
35

```

2.5、mock接口

地址：<https://mock.bboxuegu.com/project/164/interface/api/64750>

接口名称： 谁看过我

创建人：  刘俊杰

状态： ● 已完成

更新时间： 2019-09-09 09:20:54

接口路径： GET /movements/visitors

Mock地址： <https://mock.bboxuegu.com/mock/164/movements/visitors>

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	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1Ni9.eyJleHAiOiJlE1NjI4MjkzMzYsInVzZXJfaWQiOiJlbn0.Mbzn6LzsLrkVWEbhexR3lTYDZjxqlcqW11rjxDQ6Ewk	是		令牌

名称	类型	是否必须	默认值	备注	其他信息
	object []	非必须			最小数量: 0 元素是否都不同: true 最大数量: 4 item 类型: object
id	integer	必须		用户id	最大值: 10000 最小值: 1
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: 黑马小妹, 米朵妹妹, 致远哥哥
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

2.6、MovementsController

```
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 visitorsvo {  
11  
12     private Long id;  
13     private String avatar;
```

```

14     private String nickname;
15     private String gender;
16     private Integer age;
17     private String[] tags;
18     private Integer fateValue;
19
20 }
21

```

```

1     /**
2      * 谁看过我
3      *
4      * @return
5      */
6     @GetMapping("visitors")
7     public ResponseEntity<List<VisitorsVo>> queryVisitorsList(){
8         try {
9             List<VisitorsVo> list =
10 this.movementsService.queryVisitorsList();
11             return ResponseEntity.ok(list);
12         } catch (Exception e) {
13             e.printStackTrace();
14         }
15         return
16         ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
17     }
18

```

2.7、MovementsService

```

1     public List<VisitorsVo> queryVisitorsList() {
2         User user = UserThreadLocal.get();
3         String rediskey = "visitor_date_" + user.getId();
4
5         // 如果redis中存在上次查询的时间，就按照这个时间之后查询，如果没有就查询前5个
6         List<Visitors> visitors = null;
7         String value = this.redisTemplate.opsForValue().get(rediskey);
8         if(StringUtils.isEmpty(value)){
9             visitors = this.visitorsApi.topVisitor(user.getId(), 5);
10        }else{
11            visitors = this.visitorsApi.topVisitor(user.getId(),
12            Long.valueOf(value));
13        }
14
15        if(CollectionUtils.isEmpty(visitors)){
16            return Collections.emptyList();
17        }
18
19        List<Long> userIds = new ArrayList<>();
20        for (Visitors visitor : visitors) {
21            userIds.add(visitor.getVisitorUserId());
22        }
23
24        QueryWrapper<UserInfo> queryWrapper = new QueryWrapper<>();
25        queryWrapper.in("user_id", userIds);
26

```

```

25         List<UserInfo> userInfoList =
this.userInfoService.queryList(queryWrapper);
26
27         List<VisitorsVo> visitorsVoList = new ArrayList<>();
28
29         for (Visitors visitor : visitors) {
30             for (UserInfo userInfo : userInfoList) {
31                 if(visitor.getVisitorUserId().longValue() ==
userInfo.getUserId().longValue()){
32
33                     VisitorsVo visitorsVo = new VisitorsVo();
34                     visitorsVo.setAge(userInfo.getAge());
35                     visitorsVo.setAvatar(userInfo.getLogo());
36
37                     visitorsVo.setGender(userInfo.getSex().name().toLowerCase());
38                     visitorsVo.setId(userInfo.getUserId());
39                     visitorsVo.setNickname(userInfo.getNickName());
40
41                     visitorsVo.setTags(StringUtils.split(userInfo.getTags(), ','));
42                     visitorsVo.setFateValue(visitor.getScore().intValue());
43
44                     visitorsVoList.add(visitorsVo);
45                     break;
46                 }
47             }
48         }
49         return visitorsVoList;

```

2.8、测试

The screenshot shows a REST client interface with the following details:

- URL:** `http://192.168.31.98/movements/visitors`
- Method:** GET (selected from a dropdown menu)
- Headers:**
 - Authorization:** `eyJhbGciOiJIUzI1NiU9.eyJjb2JpbGUiOiJNzYwMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY`

数据：

```
1 [{ "id": 20, "avatar": "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/12.jpg", "nickname": "heima_20", "gender": "man", "age": 44, "tags": ["单身", "本科", "年龄相仿"], "fateValue": 30 },
  { "id": 19, "avatar": "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/4.jpg", "nickname": "heima_19", "gender": "woman", "age": 40, "tags": ["单身", "本科", "年龄相仿"], "fateValue": 76 },
  { "id": 36, "avatar": "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/8.jpg", "nickname": "heima_36", "gender": "woman", "age": 37, "tags": ["单身", "本科", "年龄相仿"], "fateValue": 64 },
  { "id": 33, "avatar": "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/18.jpg", "nickname": "heima_33", "gender": "man", "age": 49, "tags": ["单身", "本科", "年龄相仿"], "fateValue": 30 },
  { "id": 12, "avatar": "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/18.jpg", "nickname": "heima_12", "gender": "man", "age": 23, "tags": ["单身", "本科", "年龄相仿"], "fateValue": 30 } ]
```

3、佳人信息

首页显示的今日佳人，点击之后可以查看今日佳人的详情。

3.1、mock接口

地址：<https://mock.bboxuegu.com/project/164/interface/api/78008>

接口名称：佳人信息

创建人： 刘俊杰

状态： 已完成

更新时间：2019-10-11 16:55:38

接口路径： /tanhua/id/personalInfo

Mock地址：<https://mock.bboxuegu.com/mock/164/tanhua/id/personalInfo>

返回数据

名称	类型	是否必须	默认值	备注	其他信息
id	integer	必须		编号	最大值: 10000 最小值: 1
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: 黑马小妹
gender	string	必须		性别 man woman	枚举: man
age	integer	必须		年龄	最大值: 25 最小值: 25
tags	string []	必须		标签	最小数量: 1 元素是否都不同: true 最大数量: 3 item 类型: string
		非必须			枚举: 单身, 本科, 年龄相仿
fateValue	integer	必须		缘分值	最大值: 100 最小值: 80

3.2、TodayBestController

```
1  /**
2   * 查询今日佳人详情
3   *
4   * @param userId
5   * @return
6   */
7  @GetMapping("/{id}/personalInfo")
8  public ResponseEntity<TodayBest> queryTodayBest(@PathVariable("id")
9  Long userId) {
10     try {
11         TodayBest todayBest =
12         this.todayBestService.queryTodayBest(userId);
13         return ResponseEntity.ok(todayBest);
14     } catch (Exception e) {
15         e.printStackTrace();
16     }
17     return
18     ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
19 }
```

3.3、TodayBestService

```
1  public TodayBest queryTodayBest(Long userId) {
2
3     User user = UserThreadLocal.get();
4
5     TodayBest todayBest = new TodayBest();
6     //补全信息
7     UserInfo userInfo = this.userInfoService.queryById(userId);
```

```

8      todayBest.setId(userId);
9      todayBest.setAge(userInfo.getAge());
10     todayBest.setAvatar(userInfo.getLogo());
11     todayBest.setGender(userInfo.getSex().name().toLowerCase());
12     todayBest.setNickname(userInfo.getNickName());
13     todayBest.setTags(StringUtils.split(userInfo.getTags(), ','));
14
15     double score = this.recommendUserService.queryScore(userId,
16 user.getId());
17     if(score == 0){
18         score = 98; //默认分值
19     }
20
21     todayBest.setFateValue(Double.valueOf(score).longValue());
22
23     return todayBest;
24 }

```

3.4、查询缘分值

3.4.1、recommendUserApi

```

1  /**
2      * 查询推荐好友的缘分值
3      *
4      * @param userId
5      * @param toUserId
6      * @return
7      */
8  double queryScore(Long userId, Long toUserId);

```

3.4.2、RecommendUserApiImpl

```

1  @Override
2  public double queryScore(Long userId, Long toUserId) {
3      Query query = Query.query(Criteria
4          .where("toUserId").is(toUserId)
5          .and("userId").is(userId));
6      RecommendUser recommendUser = this.mongoTemplate.findOne(query,
7 RecommendUser.class);
8      if (null == recommendUser) {
9          return 0;
10     }
11     return recommendUser.getScore();
12 }

```

3.4.3、RecommendUserService

在server工程中完成。

```

1  /**
2   * 查询推荐好友的缘分值
3   *
4   * @param userId
5   * @param toUserId
6   * @return
7   */
8  double queryScore(Long userId, Long toUserId) {
9      return this.recommendUserApi.queryScore(userId, toUserId);
10 }

```

3.5、查询自己相册

3.5.1、mock接口

地址：<https://mock.bboxuegu.com/project/164/interface/api/77938>

接口路径：GET /movements/all

Mock地址：<https://mock.bboxuegu.com/mock/164/movements/all>

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	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1Ni9yJleHAiOiJlInj14MjkzMzYsInVzZXJfaWQiOiI0Ln0uMmZn6LzsLrkVWEbhexR3lTYDZjxqlcqW11rjxDQ6Ewk	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
userId	是	1	用户id

3.5.2、定义dubbo接口

```

1  //QuanZiApi
2  /**
3   * 查询相册表
4   *
5   * @param userId
6   * @param page
7   * @param pageSize
8   * @return
9   */
10 PageInfo<Publish> queryAlbumList(Long userId, Integer page, Integer
    pageSize);
11

```

3.5.3、dubbo接口实现

```

1 //QuanZiApiImpl
2 @Override
3     public PageInfo<Publish> queryAlbumList(Long userId, Integer page,
4 Integer pageSize) {
5         PageInfo<Publish> pageInfo = new PageInfo<>();
6         pageInfo.setPageNum(page);
7         pageInfo.setPageSize(pageSize);
8         pageInfo.setTotal(0); //不提供总数
9
10        PageRequest pageRequest = PageRequest.of(page - 1, pageSize,
11 Sort.by(Sort.Order.desc("created")));
12        Query query = new Query().with(pageRequest);
13        List<Album> albumList = this.mongoTemplate.find(query, Album.class,
14 "quanzi_album_" + userId);
15
16        if(CollectionUtils.isEmpty(albumList)){
17            return pageInfo;
18        }
19
20        List<ObjectId> publishIds = new ArrayList<>();
21        for (Album album : albumList) {
22            publishIds.add(album.getPublishId());
23        }
24
25        //查询发布信息
26        Query queryPublish =
27 Query.query(Criteria.where("id").in(publishIds)).with(Sort.by(Sort.Order.de
28 sc("created")));
29        List<Publish> publishList = this.mongoTemplate.find(queryPublish,
30 Publish.class);
31
32        pageInfo.setRecords(publishList);
33
34        return pageInfo;
35    }

```

3.5.3、MovementsController

```

1 /**
2  * 自己的所有动态
3  *
4  * @return
5  */
6 @GetMapping("all")
7     public ResponseEntity<PageResult> queryAlbumList(@RequestParam(value =
8 "page", defaultValue = "1") Integer page,
9                                                         @RequestParam(value =
10 "pagesize", defaultValue = "10") Integer pageSize,
11                                                         @RequestParam(value =
12 "userId") Long userId) {
13     try {
14         PageResult pageResult =
15 this.movementsService.queryAlbumList(userId, page, pageSize);
16         return ResponseEntity.ok(pageResult);
17     } catch (Exception e) {
18         e.printStackTrace();
19     }
20 }

```

```

15         }
16         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
17     }

```

3.5.4、MovementsService

```

1     public PageResult queryAlbumList(Long userId, Integer page, Integer
pageSize) {
2         PageResult pageResult = new PageResult();
3         pageResult.setPage(page);
4         pageResult.setPagesize(pageSize);
5
6         PageInfo<Publish> albumPageInfo =
this.quanZiApi.queryAlbumList(userId, page, pageSize);
7         List<Publish> records = albumPageInfo.getRecords();
8
9         if(CollectionUtils.isEmpty(records)){
10             return pageResult;
11         }
12
13         List<Movements> movementsList = new ArrayList<>();
14         for (Publish record : records) {
15             Movements movements = new Movements();
16
17             movements.setId(record.getId().toHexString());
18             movements.setImageContent(record.getMedias().toArray(new
String[]{}));
19             movements.setTextContent(record.getText());
20             movements.setUserId(record.getUserId());
21             movements.setCreateDate(RelativeDateFormat.format(new
Date(record.getCreated())));
22
23             movementsList.add(movements);
24         }
25
26         List<Long> userIds = new ArrayList<>();
27         for (Movements movements : movementsList) {
28             if (!userIds.contains(movements.getUserId())) {
29                 userIds.add(movements.getUserId());
30             }
31         }
32
33         QueryWrapper<UserInfo> queryWrapper = new QueryWrapper<>();
34         queryWrapper.in("user_id", userIds);
35         List<UserInfo> userInfos =
this.userInfoService.queryList(queryWrapper);
36         for (Movements movements : movementsList) {
37             for (UserInfo userInfo : userInfos) {
38                 if (movements.getUserId().longValue() ==
userInfo.getUserId().longValue()) {
39                     this.fillValueToMovements(movements, userInfo);
40                     break;
41                 }
42             }
43         }
44     }

```

```

45
46         pageResult.setItems(movementsList);
47
48         return pageResult;
49     }

```

3.5.5、测试

http://192.168.31.98/movements/all?userId=57

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

Raw Form Headers

Add new header

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUiOiIxNzYwMjAyNjg2OCIsImkiOiJjoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffmcoptl3lhY
---------------	--

Raw JSON Response

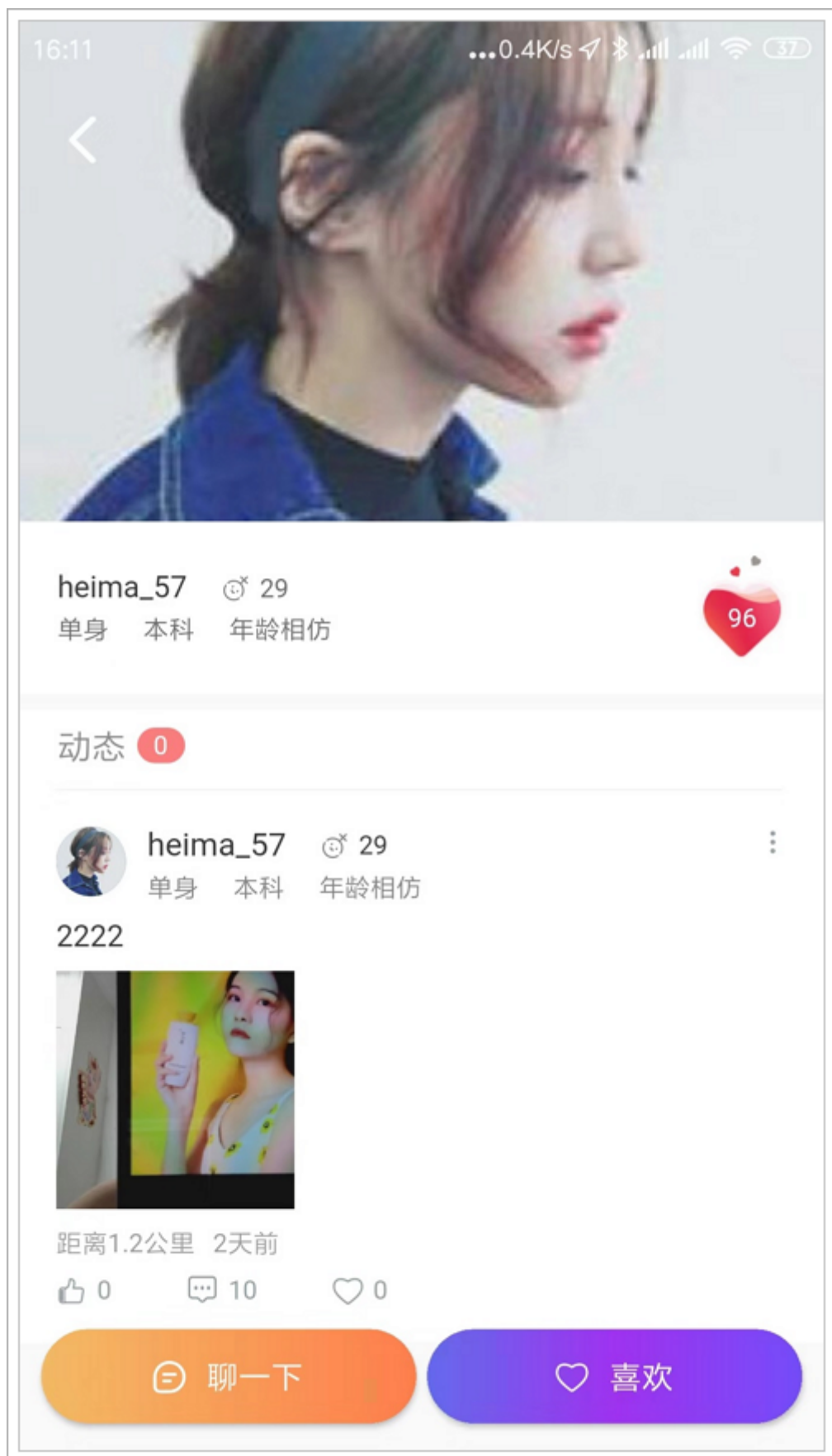
Copy to clipboard Save as file

```

{
  counts: 0
  pagesize: 10
  pages: 0
  page: 1
  -items: [1]
    -0: {
      id: "5da830d977da9301984f1173"
      userId: 57
      avatar: "https://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/logo/1.jpg"
      nickname: "heima_57"
      gender: "woman"
      age: 29
      -tags: [3]
        0: "单身"
        1: "本科"
        2: "年龄相仿"
      textContent: "2222"
      -imageContent: [1]
        0: "http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/10/17/15713036417518278.jpg"
      distance: "1.2公里"
      createDate: "2天前"
      likeCount: 0
      commentCount: 10
      loveCount: 0
      hasLiked: 0
      hasLoved: 0
    }
  }

```

3.5、整合测试



4、聊一下

聊一下功能包含了2个接口：

- 查询对方设置的问题
- 提交答案

- 提交答案后要发送消息给对方

mock接口：<https://mock.boxuegu.com/project/164/interface/api/77973>

接口路径：GET /tanhua/strangerQuestions
Mock地址：<https://mock.boxuegu.com/mock/164/tanhua/strangerQuestions>

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	eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1Ni9.eyJleHAiOiJlNjI4MjkzMzYsinVzZXJfaWQiOiIiLn0.Mbzn6LzsLrkVWEbhexR3ITYDZjxqlcqW11rjxDQ6Ewk	是		令牌

Query :

参数名称	是否必须	示例	备注
userId	是	1	用户id

返回数据

名称	类型	是否必须	默认值	备注	其他信息
	string	非必须		陌生人问题	枚举: 你喜欢去看蔚蓝的大海还是去爬巍峨的高山?

4.1、表结构

```

1 CREATE TABLE `tb_question` (
2   `id` bigint(20) NOT NULL AUTO_INCREMENT,
3   `user_id` bigint(20) DEFAULT NULL COMMENT '用户id',
4   `txt` varchar(200) DEFAULT NULL COMMENT '问题内容',
5   `created` datetime DEFAULT NULL,
6   `updated` datetime DEFAULT NULL,
7   PRIMARY KEY (`id`),
8   KEY `user_id` (`user_id`)
9 ) ENGINE=InnoDB DEFAULT CHARSET=utf8;
10
11 -- 插入数据
12 INSERT INTO `tb_question` (`id`, `user_id`, `txt`, `created`, `updated`)
13 VALUES ('1', '1', '你喜欢去看蔚蓝的大海还是去爬巍峨的高山?', '2019-10-20
14 17:25:58', '2019-10-20 17:26:01');
15
16 INSERT INTO `tb_question` (`id`, `user_id`, `txt`, `created`, `updated`)
17 VALUES ('2', '57', '你喜欢什么颜色?', '2019-10-20 23:17:39', '2019-10-20
18 23:17:41');

```

4.2、编写Question

```

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 Question extends BasePojo {
11
12     private Long id;
13     private Long userId;
14     //问题内容
15     private String txt;
16
17 }
18

```

4.3、QuestionMapper

```

1 package com.tanhua.server.mapper;
2
3 import com.baomidou.mybatisplus.core.mapper.BaseMapper;
4 import com.tanhua.server.pojo.Question;
5
6 public interface QuestionMapper extends BaseMapper<Question> {
7
8 }
9

```

4.4、QuestionService

```

1 package com.tanhua.server.service;
2
3 import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
4 import com.tanhua.server.mapper.QuestionMapper;
5 import com.tanhua.server.pojo.Question;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.stereotype.Service;
8
9 @Service
10 public class QuestionService {
11
12     @Autowired
13     private QuestionMapper questionMapper;
14
15
16     public Question queryQuestion(Long userId) {
17         QueryWrapper queryWrapper = new QueryWrapper();
18         queryWrapper.eq("user_id", userId);
19         return this.questionMapper.selectOne(queryWrapper);
20     }
21 }
22

```

4.5、查询陌生人问题

4.5.1、TodayBestController

```

1      /**
2       * 查询陌生人问题
3       *
4       * @param userId
5       * @return
6       */
7      @GetMapping("strangerQuestions")
8      public ResponseEntity<String> queryQuestion(@RequestParam("userId")
Long userId) {
9          try {
10             String question = this.todayBestService.queryQuestion(userId);
11             return ResponseEntity.ok(question);
12         } catch (Exception e) {
13             e.printStackTrace();
14         }
15         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
16     }

```

4.5.2、TodayBestService

```

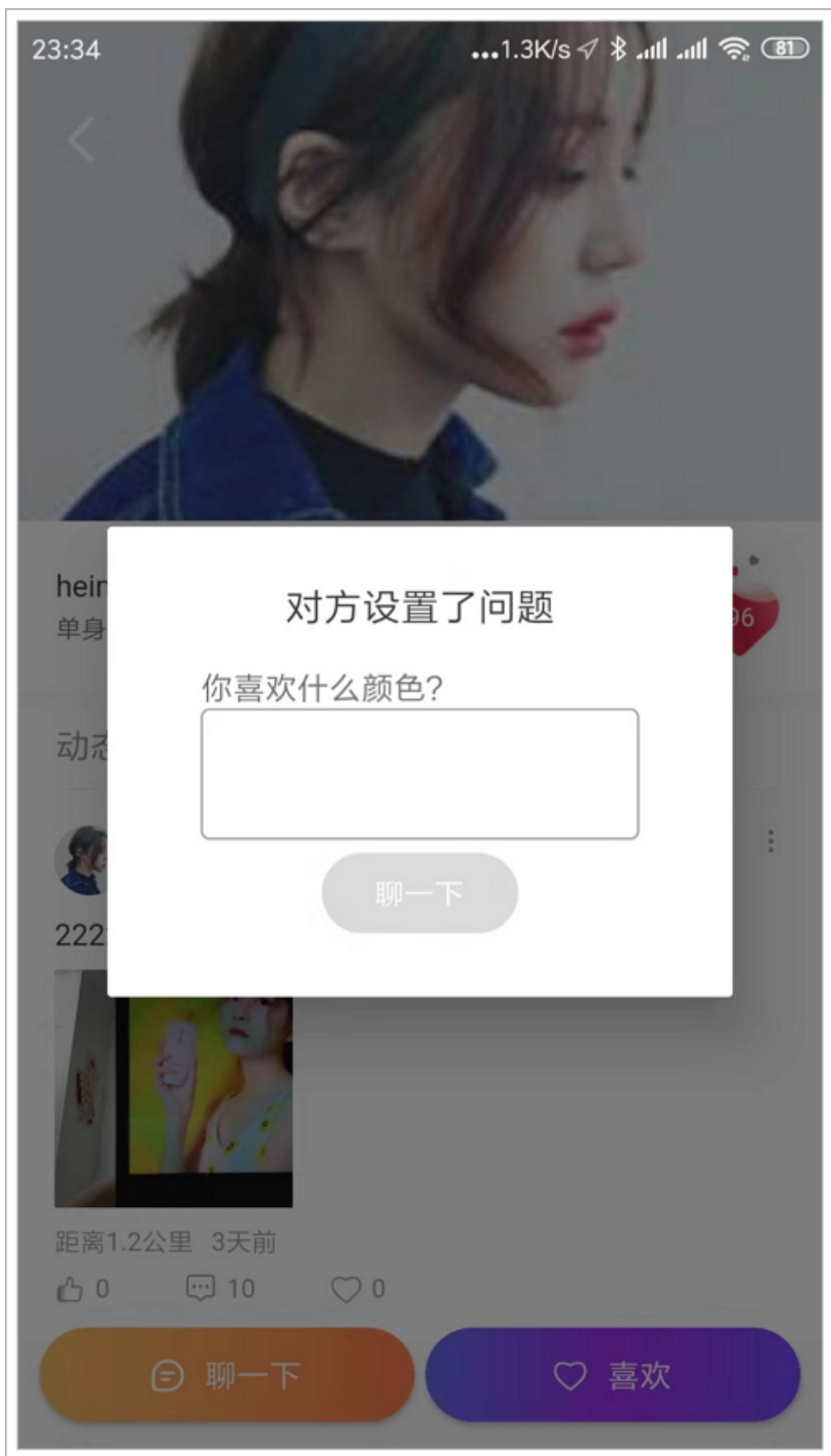
1      public String queryQuestion(Long userId) {
2          Question question = this.questionService.queryQuestion(userId);
3          if (null != question) {
4              return question.getTxt();
5          }
6          return "";
7      }

```

4.5.3、测试

The screenshot shows a REST client interface with the following details:

- URL:** `http://192.168.31.98/tanhua/strangerQuestions?userId=57`
- Method:** GET (selected)
- Headers:**
 - Authorization: eyJhbGciOiJIUzI1NiJ9.eyJjb2JpbGUlOiNzYmMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
- Status:** 200 (OK), Loading time: 1266 ms
- Request headers:**
 - Authorization: eyJhbGciOiJIUzI1NiJ9.eyJjb2JpbGUlOiNzYmMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
 - 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: Sun, 20 Oct 2019 15:33:34 GMT
 - Content-Type: text/plain; charset=UTF-8
 - Content-Length: 24
 - Connection: keep-alive
- Response body:** 你喜欢什么颜色?



4.6、回复陌生人问题

4.6.1、mock接口

接口路径：[POST](#) /tanhua/strangerQuestions

Mock地址：<https://mock.bboxuegu.com/mock/164/tanhua/strangerQuestions>

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	eyJ0eXAiOiKV1QjLCJhbGciOiJIUzI1Ni9.eyJleHAiOiJlNjI4MjZMzYsinVzZXJfaWQiOiixin0.Mbzn6LzsLrkVWEbhexR3ITYDZjxqlcqW11rjxDQ6Ewk	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	
reply	string	必须		回复	

4.6.2、发送消息给环信

在sso系统中完成。

4.6.2.1、HuanXinController

```
1  /**
2      * 发送系统消息
3      *
4      * @param target
5      * @param msg
6      * @param type
7      * @return
8      */
9      @PostMapping("messages")
10     public ResponseEntity<Void> sendMsg(@RequestParam("target") String
target,
11                                         @RequestParam("msg") String msg,
12                                         @RequestParam(value = "type",
defaultValue = "txt") String type) {
13         try {
14             boolean result = this.huanXinService.sendMsg(target, type,
msg);
15             if (result) {
16                 return ResponseEntity.ok(null);
17             }
18         } catch (Exception e) {
19             e.printStackTrace();
20         }
21
22         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
23     }
```

4.6.2.2、HuanXinService

```
1  public boolean sendMsg(String target, String type, String msg) {
2      String targetUrl = this.huanXinConfig.getUrl()
```

```

3          + this.huanXinConfig.getOrgName() + "/"
4          + this.huanXinConfig.getAppname() + "/messages";
5      try {
6          String token = this.huanXinTokenService.getToken();
7          // 请求头
8          HttpHeaders headers = new HttpHeaders();
9          headers.add("Authorization", "Bearer " + token);
10
11         Map<String, Object> paramMap = new HashMap<>();
12         paramMap.put("target_type", "users");
13         paramMap.put("target", Arrays.asList(target));
14
15         Map<String, Object> msgMap = new HashMap<>();
16         msgMap.put("type", type);
17         msgMap.put("msg", msg);
18
19         paramMap.put("msg", msgMap);
20
21         //表示消息发送者;无此字段Server会默认设置为“from”:“admin”，有from字段但
22         //值为空串(“”)时请求失败
23         msgMap.put("from", type);
24
25         HttpEntity<String> httpEntity = new HttpEntity<>
26             (MAPPER.writeValueAsString(paramMap), headers);
27         ResponseEntity<String> responseEntity =
28             this.restTemplate.postForEntity(targetUrl, httpEntity, String.class);
29
30         return responseEntity.getStatusCodeValue() == 200;
31     } catch (Exception e) {
32         e.printStackTrace();
33     }
34     return false;
35 }

```

4.6.2.3、测试

The screenshot shows a REST client interface with the following details:

- URL:** http://192.168.31.98/user/huanxin/messages
- Method:** POST
- Headers:**
 - Authorization: eyJhbGciOiJIUzI1NiJ9.eyJjb2JpbGU0IjoxNzYwMjAyNjg2OCIsImkljoxQ.OzoVY9yavYS-aibcNGIYg1kKK2pp3Qffrmcopf3lhY
- Payload:**
 - target: 1
 - msg: {"userId": "1", "nickname": "黑马小妹", "strangerQuestion": "你最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, 夏天的露水, 冬天的雪地, 只要有你一切皆可"} (Note: The screenshot shows a simplified version of this JSON object)
- Content-Type:** application/x-www-form-urlencoded
- Status:** 200 (Loading time: 3600 ms)
- Request Headers:**
 - Authorization: eyJhbGciOiJIUzI1NiJ9.eyJjb2JpbGU0IjoxNzYwMjAyNjg2OCIsImkljoxQ.OzoVY9yavYS-aibcNGIYg1kKK2pp3Qffrmcopf3lhY
 - Origin: chrome-extension://mhaabkcmplalpkmbnlpemmhjndigk
 - 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
 - Content-Type: application/x-www-form-urlencoded
 - Accept: */*

4.6.3、实现服务接口

4.6.3.1、TodayBestController

1 / **

```

2      * 回复陌生人问题
3      *
4      * @return
5      */
6      @PostMapping("strangerQuestions")
7      public ResponseEntity<Void> replyQuestion(@RequestBody Map<String,
8      Object> param) {
9          try {
10              Long userId = Long.valueOf(param.get("userId").toString());
11              String reply = param.get("reply").toString();
12              Boolean result = this.todayBestService.replyQuestion(userId,
13              reply);
14              if (result) {
15                  return ResponseEntity.ok(null);
16              }
17              } catch (Exception e) {
18                  e.printStackTrace();
19              }
20          return
21          ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
22      }

```

4.6.3.2、TodayBestService

```

1  /**
2      * 回复陌生人问题，发送消息给对方
3      *
4      * @param userId
5      * @param reply
6      * @return
7      */
8      public Boolean replyQuestion(Long userId, String reply) {
9          User user = UserThreadLocal.get();
10         UserInfo userInfo = this.userInfoService.queryById(user.getId());
11
12         //构建消息内容
13         Map<String, Object> msg = new HashMap<>();
14         msg.put("userId", user.getId().toString());
15         msg.put("nickname", this.queryQuestion(userId));
16         msg.put("strangerQuestion", userInfo.getNickName());
17         msg.put("reply", reply);
18
19         try {
20             String msgStr = MAPPER.writeValueAsString(msg);
21
22             String targetUrl = this.url + "/user/huanxin/messages";
23
24             HttpHeaders headers = new HttpHeaders();
25             headers.setContentType(MediaType.APPLICATION_FORM_URLENCODED);
26
27             MultivalueMap<String, String> params = new
28             LinkedMultivalueMap<>();
29             params.add("target", userId.toString());
30             params.add("msg", msgStr);
31
32             HttpEntity<MultivalueMap<String, String>> httpEntity = new
33             HttpEntity<>(params, headers);

```

```
32
33         ResponseEntity<Void> responseEntity =
this.restTemplate.postForEntity(targetUrl, httpEntity, Void.class);
34
35         return responseEntity.getStatusCodeValue() == 200;
36     } catch (Exception e) {
37         e.printStackTrace();
38     }
39
40
41     return false;
42 }
```

4.6.4、测试

11:31

...0.3K/s 4G 蓝牙 闹钟 信号 49

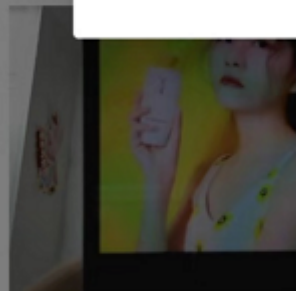


heir
单身

动态



222



距离1.2公里 3天前

👍 0

💬 10

❤️ 0

🗨️ 聊一下

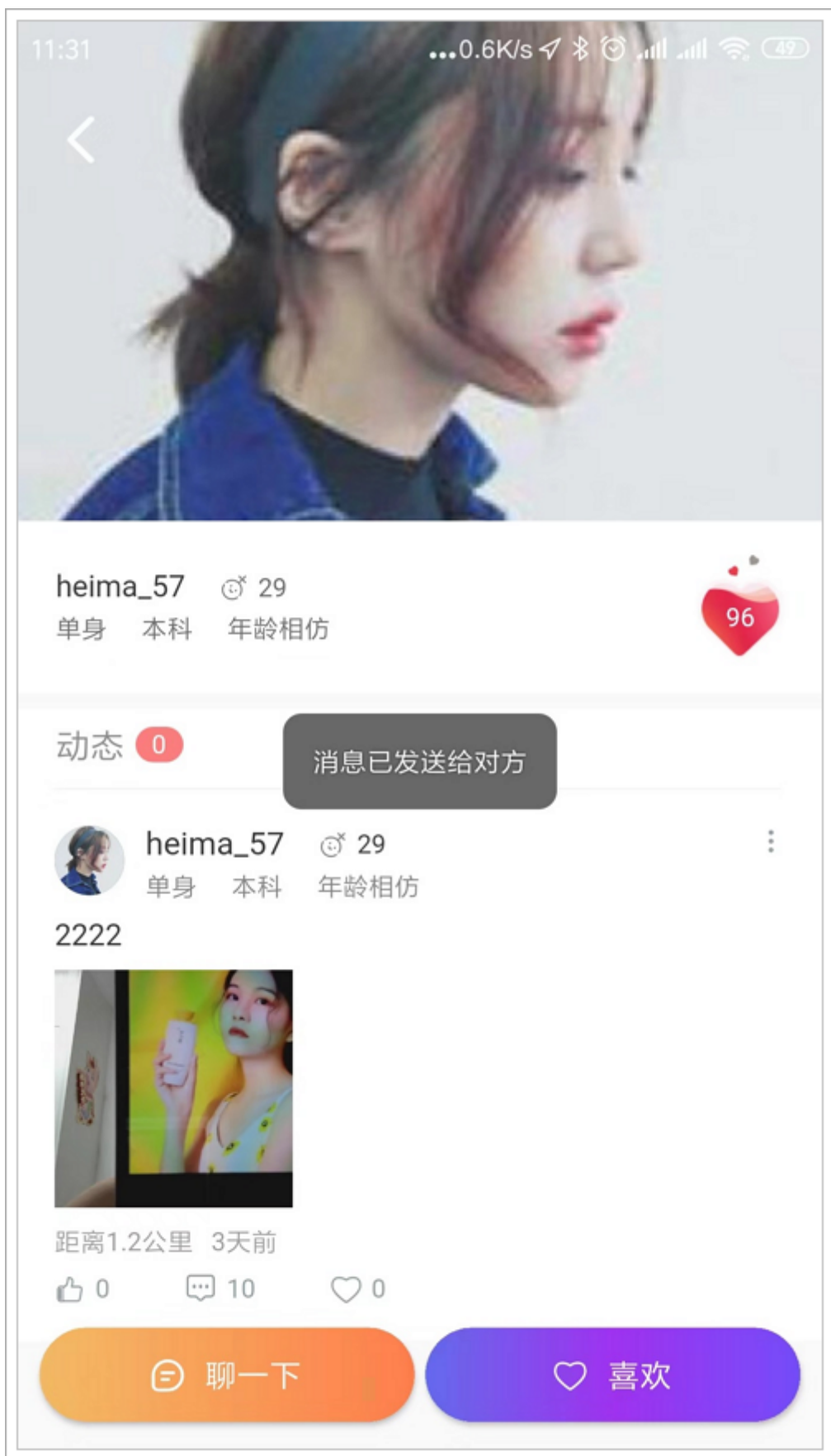
❤️ 喜欢

对方设置了问题

你喜欢什么颜色？

红色

聊一下



5、地理位置

客户端检测用户的地理位置，当变化大于500米时或每隔5分钟，向服务端发送地理位置。

6.1、定义pojo

```

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.annotation.Id;
8 import org.springframework.data.mongodb.core.geo.GeoJsonPoint;
9 import org.springframework.data.mongodb.core.index.CompoundIndex;
10 import org.springframework.data.mongodb.core.index.Indexed;
11 import org.springframework.data.mongodb.core.mapping.Document;
12
13 @Data
14 @NoArgsConstructor
15 @AllArgsConstructor
16 @Document(collection = "user_location")
17 @CompoundIndex(name = "location_index", def = "{ 'location': '2dsphere' }")
18 public class UserLocation implements java.io.Serializable{
19
20     private static final long serialVersionUID = 4508868382007529970L;
21
22     @Id
23     private ObjectId id;
24     @Indexed
25     private Long userId; //用户id
26     private GeoJsonPoint location; //x:经度 y:纬度
27     private String address; //位置描述
28     private Long created; //创建时间
29     private Long updated; //更新时间
30     private Long lastUpdated; //上次更新时间
31
32 }
33

```

6.2、定义dubbo接口

```

1 package com.tanhua.dubbo.server.api;
2
3 public interface UserLocationApi {
4
5     /**
6      * 更新用户地理位置
7      *
8      * @return
9      */
10     String updateUserLocation(Long userId, Double longitude, Double
11 latitude, String address);
12 }
13

```

6.3、编写实现

```

1 package com.tanhua.dubbo.server.api;
2
3 import com.alibaba.dubbo.config.annotation.Service;

```

```

4  import com.tanhua.dubbo.server.pojo.UserLocation;
5  import com.tanhua.dubbo.server.vo.UserLocationVo;
6  import org.bson.types.ObjectId;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.data.mongodb.core.MongoTemplate;
9  import org.springframework.data.mongodb.core.geo.GeoJsonPoint;
10 import org.springframework.data.mongodb.core.query.Criteria;
11 import org.springframework.data.mongodb.core.query.Query;
12 import org.springframework.data.mongodb.core.query.Update;
13
14 @Service(version = "1.0.0")
15 public class UserLocationApiImpl implements UserLocationApi {
16
17     @Autowired
18     private MongoTemplate mongoTemplate;
19
20     @Override
21     public String updateUserLocation(Long userId, Double longitude, Double
latitude, String address) {
22
23         UserLocation userLocation = new UserLocation();
24         userLocation.setAddress(address);
25         userLocation.setLocation(new GeoJsonPoint(longitude, latitude));
26         userLocation.setUserId(userId);
27
28         Query query =
Query.query(Criteria.where("userId").is(userLocation.getUserId()));
29         UserLocation u1 = this.mongoTemplate.findOne(query,
UserLocation.class);
30         if (u1 == null) {
31             //新增
32             userLocation.setId(ObjectId.get());
33             userLocation.setCreated(System.currentTimeMillis());
34             userLocation.setUpdated(userLocation.getCreated());
35             userLocation.setLastUpdated(userLocation.getCreated());
36
37             this.mongoTemplate.save(userLocation);
38
39             return userLocation.getId().toHexString();
40         } else {
41             //更新
42             Update update = Update
43                 .update("location", userLocation.getLocation())
44                 .set("updated", System.currentTimeMillis())
45                 .set("lastUpdated", u1.getUpdated());
46             this.mongoTemplate.updateFirst(query, update,
UserLocation.class);
47         }
48
49         return u1.getId().toHexString();
50     }
51
52 }
53

```

6.4、mock接口

接口路径： **POST** /baidu/location

Mock地址： <https://mock.bxuegu.com/mock/164/baidu/location>

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		

Body:

名称	类型	是否必须	默认值	备注	其他信息
latitude	number	必须		纬度	
longitude	number	必须		经度	
addrStr	string	必须		位置描述	

6.5、实现服务接口

```
1 package com.tanhua.server.controller;
2
3 import com.tanhua.dubbo.server.pojo.UserLocation;
4 import com.tanhua.dubbo.server.vo.UserLocationVo;
5 import com.tanhua.server.service.BaiduService;
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.*;
10
11 import java.util.Map;
12
13 @RestController
14 @RequestMapping("baidu")
15 public class BaiduController {
16
17     @Autowired
18     private BaiduService baiduService;
19
20     /**
21      * 更新位置
22      *
23      * @param param
24      * @return
25      */
26     @PostMapping("location")
27     public ResponseEntity<Void> updateLocation(@RequestBody Map<String,
28     Object> param) {
29         try {
30             Double longitude =
31             Double.valueOf(param.get("longitude").toString());
32             Double latitude =
33             Double.valueOf(param.get("latitude").toString());
34             String address = param.get("addrStr").toString();
```

```

33         Boolean bool = this.baiduService.updateLocation(longitude,
latitude, address);
34         if (bool) {
35             return ResponseEntity.ok(null);
36         }
37     } catch (Exception e) {
38         e.printStackTrace();
39     }
40
41     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
42 }
43 }
44

```

```

1  package com.tanhua.server.service;
2
3  import com.alibaba.dubbo.config.annotation.Reference;
4  import com.tanhua.dubbo.server.api.UserLocationApi;
5  import com.tanhua.dubbo.server.pojo.UserLocation;
6  import com.tanhua.server.pojo.User;
7  import com.tanhua.server.utils.UserThreadLocal;
8  import org.springframework.data.mongodb.core.geo.GeoJsonPoint;
9  import org.springframework.stereotype.Service;
10
11  @Service
12  public class BaiduService {
13
14      @Reference(version = "1.0.0")
15      private UserLocationApi userLocationApi;
16
17      public Boolean updateLocation(Double longitude, Double latitude, String
address) {
18          try {
19              User user = UserThreadLocal.get();
20              this.userLocationApi.updateUserLocation(user.getId(),
longitude, latitude, address);
21              return true;
22          } catch (Exception e) {
23              e.printStackTrace();
24          }
25          return false;
26      }
27
28  }
29

```

6.6、测试

http://192.168.31.98/baidu/location

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

Raw Form Headers

Add new header

Authorization	eyJhbGciOiJIUzI1NiU9_eyJjb2JpbGUOIiwzYmMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
---------------	--

Raw Form Files (0) Payload

Encode payload Decode payload

```
{
  "longitude": 121.5168758203125,
  "latitude": 31.12354322438761,
  "addrStr": "上海"
}
```

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

Status: 200 Loading time: 7082 ms

Request headers: Authorization: eyJhbGciOiJIUzI1NiU9_eyJjb2JpbGUOIiwzYmMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
 Origin: chrome-extension://mhaabkcmplagpkembnlplemmhndkgk
 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
 Content-Type: application/json
 Accept: */*

(1) ObjectId("5dadbc1c677da9336682a63d8")	{ 8 fields }	Object
_id	ObjectId("5dadbc1c677da9336682a63d8")	ObjectId
userId	1	Int64
location	{ 2 fields }	Object
type	Point	String
coordinates	[2 elements]	Array
[0]	121.516875820313	Double
[1]	31.1235432243876	Double
address	上海	String
created	1571664326134	Int64
updated	1571670421239	Int64
lastUpdated	1571666665681	Int64
_class	com.tanhua.dubbo.server.pojo.UserLocation	String

查看索引：

db.getCollection('user_location').getIndexes({})

0.014 sec.

Key	Value	Type
(1)	[3 elements]	Array
[0]	{ 4 fields }	Object
v	2	Int32
key	{ 1 field }	Object
_id	1	Int32
name	_id_	String
ns	tanhua.user_location	String
[1]	{ 5 fields }	Object
v	2	Int32
key	{ 1 field }	Object
location	2dsphere	String
name	location_index	String
ns	tanhua.user_location	String
2dsphereIndexVersion	3	Int32
[2]	{ 4 fields }	Object
v	2	Int32
key	{ 1 field }	Object
userId	1	Int32
name	userId	String
ns	tanhua.user_location	String