

课程说明

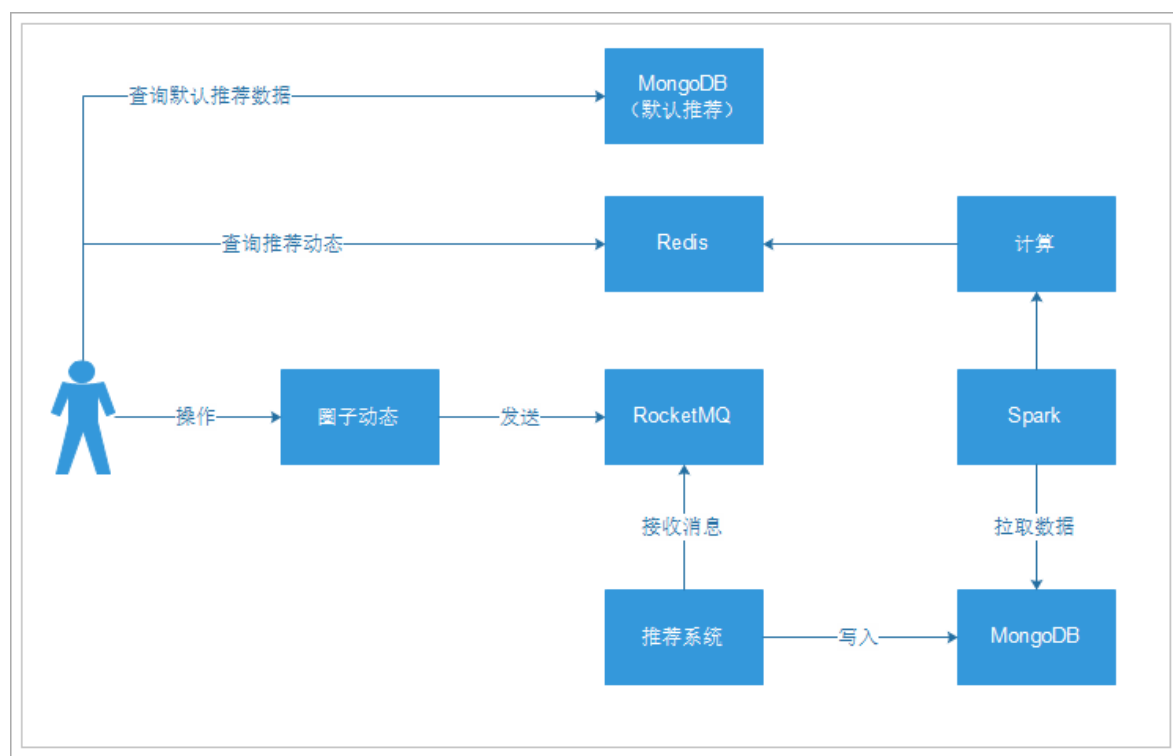
- 圈子推荐功能说明
- 圈子推荐功能流程
- 圈子推荐功能的实现
- 搭建推荐系统
- 使用Spark实现推荐逻辑
- 小视频推荐功能的实现

1、圈子推荐

1.1、功能说明

在圈子功能中，针对于用户发布的动态信息，系统可以根据用户的发布、浏览、点赞等操作，对动态信息做计算，然后对每个用户进行不同的推荐。

1.2、流程说明



流程说明：

- 用户对圈子的动态操作，如：发布、浏览、点赞、喜欢等，就会给RocketMQ进行发送消息；
- 推荐系统接收消息，并且处理消息数据，处理之后将结果数据写入到MongoDB中；
- Spark系统拉取数据，然后进行推荐计算；
- 计算之后的结果数据写入到Redis中，为每个用户都进行个性化推荐；
- 如果有用户没有数据的，查询MongoDB中的默认数据；

1.3、动态增加自增id

由于我们使用的推荐模型中，动态id需要是Long类型的，而我们之前使用的ObjectId类型的，所以需要增加Long类型的id。

1.3.1、修改Publish对象

```
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 import java.util.List;
11
12 /**
13  * 发表表，动态内容
14  */
15 @Data
16 @NoArgsConstructor
17 @AllArgsConstructor
18 @Document(collection = "quanzi_publish")
19 public class Publish implements java.io.Serializable {
20
21     private static final long serialVersionUID = 8732308321082804771L;
22
23     private ObjectId id; //主键id
24     private Long pid; //Long类型的id，用于推荐引擎使用
25     private Long userId;
26     private String text; //文字
27     private List<String> medias; //媒体数据，图片或小视频 url
28     private Integer seeType; // 谁可以看，1-公开，2-私密，3-部分可见，4-不给谁看
29     private List<Long> seeList; //部分可见的列表
30     private List<Long> notSeeList; //不给谁看的列表
31     private String longitude; //经度
32     private String latitude; //纬度
33     private String locationName; //位置名称
34     private Long created; //发布时间
35
36 }
37
38
```

1.3.2、修改发布逻辑

```
1 @Override
2     public String savePublish(Publish publish) {
3
4         // 校验
5         if (publish.getUserId() == null) {
6             return null;
7         }
8
9         try {
10             publish.setCreated(System.currentTimeMillis()); //设置创建时间
11
12         } catch (Exception e) {
13             // 处理异常
14         }
15     }
16
```

```

11         publish.setId(ObjectId.get()); //设置id
12         publish.setPid(this.idService.createId("publish",
publish.getId().toHexString()));
13
14         this.mongoTemplate.save(publish); //保存发布
15
16         Album album = new Album(); // 构建相册对象
17         album.setPublishId(publish.getId());
18         album.setCreated(System.currentTimeMillis());
19         album.setId(ObjectId.get());
20         this.mongoTemplate.save(album, "quanzi_album_" +
publish.getUserId());
21
22         //写入好友的时间线中
23         Criteria criteria =
Criteria.where("userId").is(publish.getUserId());
24         List<Users> users =
this.mongoTemplate.find(Query.query(criteria), Users.class);
25         for (Users user : users) {
26             TimeLine timeLine = new TimeLine();
27             timeLine.setId(ObjectId.get());
28             timeLine.setPublishId(publish.getId());
29             timeLine.setUserId(user.getUserId());
30             timeLine.setDate(System.currentTimeMillis());
31             this.mongoTemplate.save(timeLine, "quanzi_time_line_" +
user.getFriendId());
32         }
33
34         return publish.getId().toHexString();
35     } catch (Exception e) {
36         e.printStackTrace();
37         //TODO 出错的事务回滚
38     }
39
40     return null;
41 }

```

```

1 package com.tanhua.dubbo.server.service;
2
3 import org.apache.commons.lang3.StringUtils;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.data.redis.core.RedisTemplate;
6 import org.springframework.stereotype.Service;
7
8 //生成自增长的id, 原理: 使用redis的自增长值
9 @Service
10 public class IdService {
11
12     @Autowired
13     private RedisTemplate<String, String> redisTemplate;
14
15     public Long createId(String type, String strId) {
16         type = StringUtils.upperCase(type);
17
18         String idHashKey = "TANHUA_ID_HASH_" + type;
19         if (this.redisTemplate.opsForHash().hasKey(idHashKey, strId)) {

```

```

20         return
Long.valueOf(this.redisTemplate.opsForHash().get(idHashKey,
strId).toString());
21     }
22
23     String idkey = "TANHUA_ID_" + type;
24     Long id = this.redisTemplate.opsForValue().increment(idkey);
25
26     this.redisTemplate.opsForHash().put(idHashKey, strId,
id.toString());
27
28     return id;
29 }
30
31 }

```

itcast-tanhua-dubbo-service需要增加Redis依赖：

```

1 <dependency>
2     <groupId>org.springframework.boot</groupId>
3     <artifactId>spring-boot-starter-data-redis</artifactId>
4 </dependency>

```

1.4、动态计分规则

- 浏览 +1
- 点赞 +5
- 喜欢 +8
- 评论 + 10
- 文字长度：50以内1分，50~100之间2分，100以上3分
- 图片个数：每个图片一分

1.5、发送消息

1.5.1、QuanziMQService

itcast-tanhua-server增加依赖：

```

1 <!--RocketMQ相关-->
2 <dependency>
3     <groupId>org.apache.rocketmq</groupId>
4     <artifactId>rocketmq-spring-boot-starter</artifactId>
5     <version>2.0.0</version>
6 </dependency>
7 <dependency>
8     <groupId>org.apache.rocketmq</groupId>
9     <artifactId>rocketmq-client</artifactId>
10    <version>4.3.2</version>
11 </dependency>

```

配置文件：

```
1 #rocketmq相关配置
2 spring.rocketmq.nameServer=192.168.31.81:9876
3 spring.rocketmq.producer.group=tanhua
```

```
1 package com.tanhua.server.service;
2
3 import com.alibaba.dubbo.config.annotation.Reference;
4 import com.tanhua.dubbo.server.api.QuanZiApi;
5 import com.tanhua.dubbo.server.pojo.Publish;
6 import com.tanhua.server.pojo.User;
7 import com.tanhua.server.utils.UserThreadLocal;
8 import org.apache.rocketmq.spring.core.RocketMQTemplate;
9 import org.slf4j.Logger;
10 import org.slf4j.LoggerFactory;
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.stereotype.Service;
13
14 import java.util.Date;
15 import java.util.HashMap;
16 import java.util.Map;
17
18 @Service
19 public class QuanziMQService {
20
21     private static final Logger LOGGER =
22         LoggerFactory.getLogger(QuanziMQService.class);
23
24     @Autowired
25     private RocketMQTemplate rocketMQTemplate;
26
27     @Reference(version = "1.0.0")
28     private QuanZiApi quanZiApi;
29
30     /**
31      * 发布动态消息
32      *
33      * @param publishId
34      * @return
35      */
36     public boolean publishMsg(String publishId) {
37         return this.sendMsg(publishId, 1);
38     }
39
40     /**
41      * 浏览动态消息
42      *
43      * @param publishId
44      * @return
45      */
46     public boolean queryPublishMsg(String publishId) {
47         return this.sendMsg(publishId, 2);
48     }
49
50     /**
```

```

50     * 点赞动态消息
51     *
52     * @param publishId
53     * @return
54     */
55     public Boolean likePublishMsg(String publishId) {
56         return this.sendMsg(publishId, 3);
57     }
58
59     /**
60     * 取消点赞动态消息
61     *
62     * @param publishId
63     * @return
64     */
65     public Boolean disLikePublishMsg(String publishId) {
66         return this.sendMsg(publishId, 6);
67     }
68
69     /**
70     * 喜欢动态消息
71     *
72     * @param publishId
73     * @return
74     */
75     public Boolean lovePublishMsg(String publishId) {
76         return this.sendMsg(publishId, 4);
77     }
78
79     /**
80     * 取消喜欢动态消息
81     *
82     * @param publishId
83     * @return
84     */
85     public Boolean disLovePublishMsg(String publishId) {
86         return this.sendMsg(publishId, 7);
87     }
88
89     /**
90     * 评论动态消息
91     *
92     * @param publishId
93     * @return
94     */
95     public Boolean commentPublishMsg(String publishId) {
96         return this.sendMsg(publishId, 5);
97     }
98
99     /**
100    * 发送圈子操作相关的消息
101    *
102    * @param publishId
103    * @param type      1-发动态, 2-浏览动态, 3-点赞, 4-喜欢, 5-评论, 6-取消点
104    *                  赞, 7-取消喜欢
105    * @return
106    */
107    private Boolean sendMsg(String publishId, Integer type) {

```

```

107         try {
108             User user = UserThreadLocal.get();
109
110             Publish publish = this.quanZiApi.queryPublishById(publishId);
111
112             //构建消息
113             Map<String, Object> msg = new HashMap<>();
114             msg.put("userId", user.getId());
115             msg.put("date", System.currentTimeMillis());
116             msg.put("publishId", publishId);
117             msg.put("pid", publish.getPid());
118             msg.put("type", type);
119
120             this.rocketMQTemplate.convertAndSend("tanhua-quanzi", msg);
121         } catch (Exception e) {
122             LOGGER.error("发送消息失败! publishId = " + publishId + ", type
= " + type, e);
123             return false;
124         }
125
126         return true;
127     }
128 }
129

```

1.5.2、修改MovementsController

```

1  package com.tanhua.server.controller;
2
3  import com.tanhua.server.service.MovementsService;
4  import com.tanhua.server.service.QuanziMQService;
5  import com.tanhua.server.vo.Movements;
6  import com.tanhua.server.vo.PageResult;
7  import org.apache.commons.lang3.StringUtils;
8  import org.springframework.beans.factory.annotation.Autowired;
9  import org.springframework.http.HttpStatus;
10 import org.springframework.http.ResponseEntity;
11 import org.springframework.web.bind.annotation.*;
12 import org.springframework.web.multipart.MultipartFile;
13
14 @RestController
15 @RequestMapping("movements")
16 public class MovementsController {
17
18     @Autowired
19     private MovementsService movementsService;
20
21     @Autowired
22     private QuanziMQService quanziMQService;
23
24     /**
25      * 发送动态
26      *
27      * @param textContent
28      * @param location
29      * @param multipartFile
30      * @return

```

```

31     */
32     @PostMapping()
33     public ResponseEntity<Void> savePublish(@RequestParam("textContent")
34     String textContent,
35                                           @RequestParam("location")
36     String location,
37                                           @RequestParam("longitude")
38     String longitude,
39                                           @RequestParam("latitude")
40     String latitude,
41                                           @RequestParam(value =
42     "imageContent", required = false) MultipartFile[] multipartFile) {
43     try {
44         String publishId =
45         this.movementsService.savePublish(textContent, location, longitude,
46         latitude, multipartFile);
47         if (StringUtils.isEmpty(publishId)) {
48             // 发送消息
49             this.quanzimQService.publishMsg(publishId);
50             return ResponseEntity.ok(null);
51         }
52     } catch (Exception e) {
53         e.printStackTrace();
54     }
55     return
56     ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
57 }
58
59 /**
60  * 查询好友动态
61  *
62  * @param page
63  * @param pageSize
64  * @return
65  */
66 @GetMapping
67 public PageResult queryPublishList(@RequestParam(value = "page",
68 defaultvalue = "1") Integer page,
69                                     @RequestParam(value = "pagesize",
70 defaultvalue = "10") Integer pageSize) {
71     return this.movementsService.queryPublishList(page, pageSize,
72 false);
73 }
74
75 /**
76  * 查询推荐动态
77  *
78  * @param page
79  * @param pageSize
80  * @return
81  */
82 @GetMapping("recommend")
83 public PageResult queryRecommendPublishList(@RequestParam(value =
84 "page", defaultvalue = "1") Integer page,
85                                               @RequestParam(value =
86 "pagesize", defaultvalue = "10") Integer pageSize) {
87     return this.movementsService.queryPublishList(page, pageSize,
88 true);

```

```

75     }
76
77     /**
78     * 点赞
79     *
80     * @param publishId
81     * @return
82     */
83     @GetMapping("/{id}/like")
84     public ResponseEntity<Long> likeComment(@PathVariable("id") String
publishId) {
85         try {
86             Long likeCount = this.movementsService.likeComment(publishId);
87             if (likeCount != null) {
88
89                 //发送点赞消息
90                 this.quanzimQService.likePublishMsg(publishId);
91
92                 return ResponseEntity.ok(likeCount);
93             }
94         } catch (Exception e) {
95             e.printStackTrace();
96         }
97         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
98     }
99
100    /**
101    * 取消点赞
102    *
103    * @param publishId
104    * @return
105    */
106    @GetMapping("/{id}/dislike")
107    public ResponseEntity<Long> disLikeComment(@PathVariable("id") String
publishId) {
108        try {
109            Long likeCount =
this.movementsService.cancelLikeComment(publishId);
110            if (null != likeCount) {
111
112                //发送取消点赞消息
113                this.quanzimQService.disLikePublishMsg(publishId);
114
115                return ResponseEntity.ok(likeCount);
116            }
117        } catch (Exception e) {
118            e.printStackTrace();
119        }
120        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
121    }
122
123    /**
124    * 喜欢
125    *
126    * @param publishId
127    * @return

```

```

128     */
129     @GetMapping("/{id}/love")
130     public ResponseEntity<Long> loveComment(@PathVariable("id") String
publishId) {
131         try {
132             Long loveCount = this.movementsService.loveComment(publishId);
133             if (null != loveCount) {
134
135                 //发送喜欢消息
136                 this.quanzimQService.lovePublishMsg(publishId);
137
138                 return ResponseEntity.ok(loveCount);
139             }
140         } catch (Exception e) {
141             e.printStackTrace();
142         }
143         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
144     }
145
146     /**
147      * 取消喜欢
148      *
149      * @param publishId
150      * @return
151      */
152     @GetMapping("/{id}/unlove")
153     public ResponseEntity<Long> disLoveComment(@PathVariable("id") String
publishId) {
154         try {
155             Long loveCount =
this.movementsService.cancelLoveComment(publishId);
156             if (null != loveCount) {
157
158                 //发送取消喜欢消息
159                 this.quanzimQService.disLovePublishMsg(publishId);
160
161                 return ResponseEntity.ok(loveCount);
162             }
163         } catch (Exception e) {
164             e.printStackTrace();
165         }
166         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
167     }
168
169     /**
170      * 查询单条动态信息
171      *
172      * @param publishId
173      * @return
174      */
175     @GetMapping("/{id}")
176     public ResponseEntity<Movements> queryById(@PathVariable("id") String
publishId) {
177         try {
178             Movements movements =
this.movementsService.queryById(publishId);

```

```

179         if (null != movements) {
180
181             //发送消息
182             this.quanziMQService.queryPublishMsg(publishId);
183
184             return ResponseEntity.ok(movements);
185         }
186     } catch (Exception e) {
187         e.printStackTrace();
188     }
189     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
190 }
191
192 }
193

```

CommentsController :

```

1  /**
2   * 发表评论
3   *
4   * @param param
5   * @return
6   */
7  @PostMapping
8  public ResponseEntity<Void> saveComments(@RequestBody Map<String,
String> param) {
9      try {
10         String publishId = param.get("movementId");
11         String content = param.get("comment");
12         Boolean bool = this.commentsService.saveComments(publishId,
content);
13         if (bool) {
14
15             //发送消息
16             this.quanziMQService.sendCommentPublishMsg(publishId);
17
18             return ResponseEntity.ok(null);
19         }
20     } catch (Exception e) {
21         e.printStackTrace();
22     }
23     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
24 }

```

1.6、接收消息

1.6.1、创建itcast-tanhua-recommend工程

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```
4         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
5     <parent>
6         <artifactId>itcast-tanhua</artifactId>
7         <groupId>cn.itcast.tanhua</groupId>
8         <version>1.0-SNAPSHOT</version>
9     </parent>
10    <modelVersion>4.0.0</modelVersion>
11
12    <artifactId>itcast-tanhua-recommend</artifactId>
13
14    <dependencies>
15        <dependency>
16            <groupId>org.springframework.boot</groupId>
17            <artifactId>spring-boot-starter-web</artifactId>
18        </dependency>
19        <dependency>
20            <groupId>org.springframework.boot</groupId>
21            <artifactId>spring-boot-starter-test</artifactId>
22            <scope>test</scope>
23        </dependency>
24        <dependency>
25            <groupId>org.springframework.boot</groupId>
26            <artifactId>spring-boot-starter-data-redis</artifactId>
27        </dependency>
28        <dependency>
29            <groupId>org.springframework.boot</groupId>
30            <artifactId>spring-boot-starter-data-mongodb</artifactId>
31        </dependency>
32        <!--其他工具包依赖-->
33        <dependency>
34            <groupId>org.apache.commons</groupId>
35            <artifactId>commons-lang3</artifactId>
36        </dependency>
37        <dependency>
38            <groupId>commons-io</groupId>
39            <artifactId>commons-io</artifactId>
40            <version>2.6</version>
41        </dependency>
42        <!--RocketMQ相关-->
43        <dependency>
44            <groupId>org.apache.rocketmq</groupId>
45            <artifactId>rocketmq-spring-boot-starter</artifactId>
46            <version>2.0.0</version>
47        </dependency>
48        <dependency>
49            <groupId>org.apache.rocketmq</groupId>
50            <artifactId>rocketmq-client</artifactId>
51            <version>4.3.2</version>
52        </dependency>
53        <dependency>
54            <groupId>org.projectlombok</groupId>
55            <artifactId>lombok</artifactId>
56        </dependency>
57        <dependency>
58            <groupId>joda-time</groupId>
59            <artifactId>joda-time</artifactId>
60        </dependency>
```

```

61     </dependencies>
62
63 </project>

```

1.6.2、配置文件

application.properties

```

1  spring.application.name = itcast-rocketmq
2  server.port = 18082
3
4  # rocketmq相关配置
5  spring.rocketmq.nameServer=192.168.31.81:9876
6  spring.rocketmq.producer.group=tanhua
7
8  # Redis相关配置
9  spring.redis.jedis.pool.max-wait = 5000ms
10 spring.redis.jedis.pool.max-idle = 100
11 spring.redis.jedis.pool.min-idle = 10
12 spring.redis.timeout = 10s
13 spring.redis.cluster.nodes =
14   192.168.31.81:6379,192.168.31.81:6380,192.168.31.81:6381
15 spring.redis.cluster.max-redirects=5
16
17 # mongodb相关配置
18 spring.data.mongodb.uri=mongodb://192.168.31.81:27017/tanhua

```

log4j.properties

```

1  log4j.rootLogger=DEBUG,A1
2
3  log4j.appender.A1=org.apache.log4j.ConsoleAppender
4  log4j.appender.A1.layout=org.apache.log4j.PatternLayout
5  log4j.appender.A1.layout.ConversionPattern=[%t] [%c]-[%p] %m%n

```

1.6.3、RecommendQuanZi

存储到MongoDB的中的实体结构。

```

1  package com.tanhua.recommend.pojo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6  import org.bson.types.ObjectId;
7
8  @Data
9  @NoArgsConstructor
10 @AllArgsConstructor
11 public class RecommendQuanZi {
12
13     private ObjectId id;
14     private Long userId; // 用户id
15     private Long publishId; // 动态id, 需要转化为Long类型
16     private Double score; // 得分
17     private Long date; // 时间戳

```

```
18 | }
19 |
```

1.6.4、QuanZiMsgConsumer

```
1 package com.tanhua.recommend.msg;
2
3 import com.fasterxml.jackson.databind.JsonNode;
4 import com.fasterxml.jackson.databind.ObjectMapper;
5 import com.tanhua.recommend.pojo.Publish;
6 import com.tanhua.recommend.pojo.RecommendQuanZi;
7 import org.apache.commons.lang3.StringUtils;
8 import org.apache.rocketmq.spring.annotation.RocketMQMessageListener;
9 import org.apache.rocketmq.spring.core.RocketMQListener;
10 import org.bson.types.ObjectId;
11 import org.joda.time.DateTime;
12 import org.slf4j.Logger;
13 import org.slf4j.LoggerFactory;
14 import org.springframework.beans.factory.annotation.Autowired;
15 import org.springframework.data.mongodb.core.MongoTemplate;
16 import org.springframework.stereotype.Component;
17 import org.springframework.util.CollectionUtils;
18
19 @Component
20 @RocketMQMessageListener(topic = "tanhua-quanzi",
21     consumerGroup = "tanhua-quanzi-consumer")
22 public class QuanZiMsgConsumer implements RocketMQListener<String> {
23
24     private static final ObjectMapper MAPPER = new ObjectMapper();
25
26     private static final Logger LOGGER =
27         LoggerFactory.getLogger(QuanZiMsgConsumer.class);
28
29     @Autowired
30     private MongoTemplate mongoTemplate;
31
32     @Override
33     public void onMessage(String msg) {
34         try {
35             JsonNode jsonNode = MAPPER.readTree(msg);
36
37             Long userId = jsonNode.get("userId").asLong();
38             Long pid = jsonNode.get("pid").asLong();
39             String publishId = jsonNode.get("publishId").asText();
40             Integer type = jsonNode.get("type").asInt();
41
42             //1-发动态, 2-浏览动态, 3-点赞, 4-喜欢, 5-评论, 6-取消点赞, 7-取消喜
43             欢
44
45             RecommendQuanZi recommendQuanZi = new RecommendQuanZi();
46             recommendQuanZi.setUserId(userId);
47             recommendQuanZi.setId(ObjectId.get());
48             recommendQuanZi.setDate(System.currentTimeMillis());
49             recommendQuanZi.setPublishId(pid);
50
51             switch (type) {
52                 case 1: {
53                     int score = 0;
```

```

51         Publish publish = this.mongoTemplate.findById(new
Objectid(publishId), Publish.class);
52         if (StringUtils.length(publish.getText()) < 50) {
53             score += 1;
54         } else if (StringUtils.length(publish.getText()) <
100) {
55             score += 2;
56         } else if (StringUtils.length(publish.getText()) >=
100) {
57             score += 3;
58         }
59
60         if (!CollectionUtils.isEmpty(publish.getMedias())) {
61             score += publish.getMedias().size();
62         }
63
64         recommendQuanZi.setScore(Double.valueOf(score));
65
66         break;
67     }
68     case 2: {
69         recommendQuanZi.setScore(1d);
70         break;
71     }
72     case 3: {
73         recommendQuanZi.setScore(5d);
74         break;
75     }
76     case 4: {
77         recommendQuanZi.setScore(8d);
78         break;
79     }
80     case 5: {
81         recommendQuanZi.setScore(10d);
82         break;
83     }
84     case 6: {
85         recommendQuanZi.setScore(-5d);
86         break;
87     }
88     case 7: {
89         recommendQuanZi.setScore(-8d);
90         break;
91     }
92     default: {
93         recommendQuanZi.setScore(0d);
94         break;
95     }
96 }
97
98     String collectionName = "recommend_quanzi_" + new
DateTime().toString("yyyyMMdd");
99     this.mongoTemplate.save(recommendQuanZi, collectionName);
100
101     } catch (Exception e) {
102         LOGGER.error("处理消息失败~" + msg, e);
103     }
104 }

```

105 }
106

1.7、测试

1.7.1、发布动态

http://127.0.0.1/movements

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

Raw Form Headers

Add new header

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUiaXNzYwMjAyNjg2OCIsImkiOiJoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
---------------	--

Raw Form Files (4) Payload

Add new value Values from here will be URL encoded!

textContent	今天去吃大餐了
location	上海
longitude	11.11
latitude	22.22

multipart/form-data Set "Content-Type" header to overwrite this value.

发布4张图片：

Raw Form Files (4) Payload

Add new file field

选择文件	avatar_1.png	imageContent	x	avatar_1.png (198.9 KB)
选择文件	group_6.png	imageContent	x	group_6.png (79.8 KB)
选择文件	group_5.png	imageContent	x	group_5.png (84.8 KB)
选择文件	group_4.png	imageContent	x	group_4.png (89.2 KB)

multipart/form-data Set "Content-Type" header to overwrite this value.

数据：

(9) ObjectId("5d932c42320b4d0840e405f2")	{ 11 fields }	Object
_id	ObjectId("5d932c42320b4d0840e405f2")	ObjectId
pid	2	Int64
userId	1	Int64
text	今天去吃大餐了	String
medias	[4 elements]	Array
seeType	1	Int32
longitude	11.11	String
latitude	22.22	String
locationName	上海	String
created	1569926210997	Int64
_class	com.tanhua.dubbo.server.pojo.Publish	String

消息处理：

db.getCollection('recommend_quanzi_20191001').find({})

recommend_quanzi_20191001 0.007 sec.

Key	Value	Type
(1) ObjectId("5d932bf0320b4d1c042beebe")	{ 6 fields }	Object
(2) ObjectId("5d932c43320b4d1c042beebe")	{ 6 fields }	Object
_id	ObjectId("5d932c43320b4d1c042beebe")	ObjectId
userId	1	Int64
publishId	2	Int64
score	5.0	Double
date	1569926211228	Int64
_class	com.tanhua.recommend.pojo.RecommendQuanZi	String

1.7.2、浏览动态

http://127.0.0.1/movements/5d932c42320b4d0840e405f2

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

Authorization

eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUiOiIxNzYwMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY

消息处理：

(3) ObjectId("5d932e07320b4d1c042beebe")

{ 6 fields }

Object

_id	ObjectId("5d932e07320b4d1c042beebe")	ObjectId
userId	1	Int64
publishId	2	Int64
score	1.0	Double
date	1569926664676	Int64
_class	com.tanhua.recommend.pojo.RecommendQuanZi	String

1.7.3、点赞

http://127.0.0.1/movements/5d932c42320b4d0840e405f2/like

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

Authorization

eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUiOiIxNzYwMjAyNjg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY

消息处理：

(4) ObjectId("5d933156320b4d1c042beebe")

{ 6 fields }

Object

_id	ObjectId("5d933156320b4d1c042beebe")	ObjectId
userId	1	Int64
publishId	2	Int64
score	5.0	Double
date	1569927510939	Int64
_class	com.tanhua.recommend.pojo.RecommendQuanZi	String

1.7.4、取消点赞

http://127.0.0.1/movements/5d932c42320b4d0840e405f2/dislike

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUlOiNzYmMjYyNyJg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
---------------	---

消息处理：

(5) ObjectId("5d93319d320b4d1c042beec0")	{ 6 fields }	Object
_id	ObjectId("5d93319d320b4d1c042beec0")	ObjectId
userId	1	Int64
publishId	2	Int64
score	-5.0	Double
date	1569927581982	Int64
_class	com.tanhua.recommend.pojo.RecommendQuanZi	String

1.7.5、喜欢

http://127.0.0.1/movements/5d932c42320b4d0840e405f2/love

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUlOiNzYmMjYyNyJg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
---------------	---

消息处理：

(6) ObjectId("5d933643320b4d1c042beec1")	{ 6 fields }	Object
_id	ObjectId("5d933643320b4d1c042beec1")	ObjectId
userId	1	Int64
publishId	2	Int64
score	8.0	Double
date	1569928771354	Int64
_class	com.tanhua.recommend.pojo.RecommendQuanZi	String

1.7.6、取消喜欢

http://127.0.0.1/movements/5d932c42320b4d0840e405f2/unlove

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

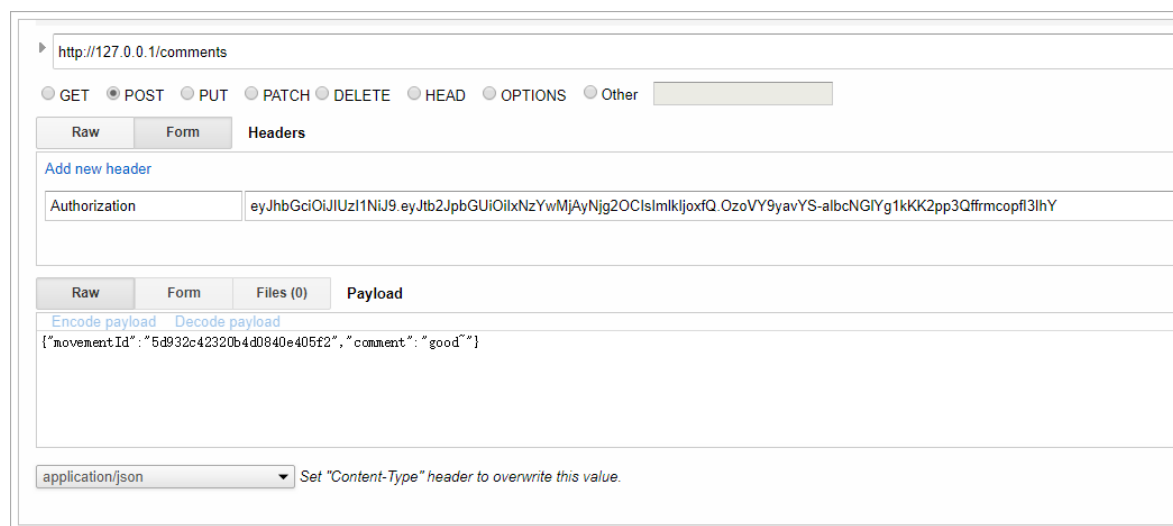
Add new header

Authorization	eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGUlOiNzYmMjYyNyJg2OCIsImkljoxfQ.OzoVY9yavYS-albcNGIYg1kKK2pp3Qffrmcopfl3lhY
---------------	---

消息处理：

(7) ObjectId("5d933690320b4d1c042beec2")	{ 6 fields }	Object
_id	ObjectId("5d933690320b4d1c042beec2")	ObjectId
userId	1	Int64
publishId	2	Int64
score	-8.0	Double
date	1569928848818	Int64
_class	com.tanhua.recommend.pojo.RecommendQuanZi	String

1.7.7、评论



http://127.0.0.1/comments

GET POST PUT PATCH DELETE HEAD OPTIONS Other

Raw Form Headers

Add new header

Header	Value
Authorization	eyJhbGciOiJIUzI1NiIsInR5cGU6IjY9ayYS-albcNGIYg1kKK2pp3Qffirmcopfl3lhY

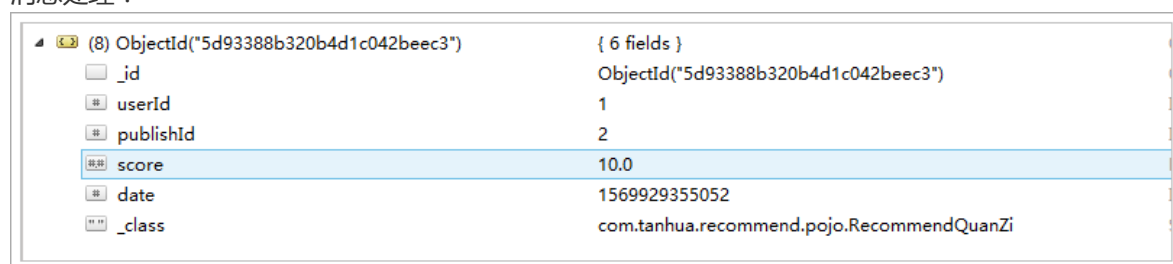
Raw Form Files (0) Payload

Encode payload Decode payload

```
{"movementId": "5d932c42320b4d0840e405f2", "comment": "good~"}
```

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

消息处理：



Field	Value
Object Id ("5d93388b320b4d1c042beec3")	{ 6 fields }
_id	Object Id ("5d93388b320b4d1c042beec3")
userId	1
publishId	2
score	10.0
date	1569929355052
_class	com.tanhua.recommend.pojo.RecommendQuanZi

2、推荐系统

构造数据：

```
1 @Test
2 public void testSavePublish() {
3     for (int i = 0; i < 100; i++) {
4         Publish publish = new Publish();
5         publish.setUserId(RandomUtils.nextLong(1,10));
6         publish.setPid(Long.valueOf(1000 + i));
7         publish.setLocationName("上海市");
8         publish.setSeeType(1);
9         publish.setText("今天天气不错~ " + i);
10        publish.setMedias(Arrays.asList("http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/10/01/15699262101875720.png",
11            "http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/10/01/15699262107836768.png",
12            "http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/10/01/15699262108584571.png",
13            "http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/10/01/15699262109221190.png"));
14        String publishId = this.quanZiApi.savePublish(publish);
15        System.out.println(publishId);
16    }
17 }
18
19 @Test
20 public void testSaveRecommend(){
21     for (int i = 0; i < 1000; i++) {
22         RecommendQuanZi recommendQuanZi = new RecommendQuanZi();
```

```

23         recommendQuanZi.setDate(System.currentTimeMillis());
24         recommendQuanZi.setId(ObjectId.get());
25
26         recommendQuanZi.setScore(Double.valueOf(RandomUtils.nextInt(0,15)));
27         recommendQuanZi.setUserId(RandomUtils.nextLong(1,10));
28         recommendQuanZi.setPublishId(RandomUtils.nextLong(1000,1099));
29
30         this.mongoTemplate.save(recommendQuanZi,
31             "recommend_quanzi_20191001");

```

2.1、搭建系统

搭建系统itcast-tanhua-spark用于实现推荐功能。

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0"
3          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
5          http://maven.apache.org/xsd/maven-4.0.0.xsd">
6      <parent>
7          <artifactId>itcast-tanhua</artifactId>
8          <groupId>cn.itcast.tanhua</groupId>
9          <version>1.0-SNAPSHOT</version>
10     </parent>
11     <modelVersion>4.0.0</modelVersion>
12     <packaging>jar</packaging>
13
14     <artifactId>itcast-tanhua-spark</artifactId>
15
16     <dependencies>
17         <dependency>
18             <groupId>org.apache.spark</groupId>
19             <artifactId>spark-mllib_2.12</artifactId>
20             <version>2.4.3</version>
21             <!--<scope>runtime</scope>-->
22
23             <exclusions>
24                 <exclusion>
25                     <groupId>javax.servlet</groupId>
26                     <artifactId>javax.servlet-api</artifactId>
27                 </exclusion>
28             </exclusions>
29         </dependency>
30         <dependency>
31             <groupId>org.mongodb.spark</groupId>
32             <artifactId>mongo-spark-connector_2.12</artifactId>
33             <version>2.4.1</version>
34         </dependency>
35         <dependency>
36             <groupId>javax.servlet</groupId>
37             <artifactId>javax.servlet-api</artifactId>
38             <version>3.1.0</version>
39             <!--<scope>provided</scope>-->
40         </dependency>
41     </dependencies>

```

```

41         <groupId>org.projectlombok</groupId>
42         <artifactId>lombok</artifactId>
43     </dependency>
44     <dependency>
45         <groupId>redis.clients</groupId>
46         <artifactId>jedis</artifactId>
47         <version>3.1.0</version>
48     </dependency>
49 </dependencies>
50
51 </project>

```

2.2、配置文件app.properties

```

1 spark.mongodb.input.uri =
  mongodb://192.168.31.81:27017/tanhua.recommend_quanzi_20191001?
  readPreference=primaryPreferred
2
3 redis.cluster.nodes =
  192.168.31.81:6379,192.168.31.81:6380,192.168.31.81:6381

```

2.3、MLlibRecommend

使用Spark MLlib进行推荐。

```

1 package com.tanhua.spark.mongo;
2
3 import org.apache.spark.api.java.JavaPairRDD;
4 import org.apache.spark.api.java.JavaRDD;
5 import org.apache.spark.mllib.recommendation.ALS;
6 import org.apache.spark.mllib.recommendation.MatrixFactorizationModel;
7 import org.apache.spark.mllib.recommendation.Rating;
8 import scala.Tuple2;
9
10 public class MLlibRecommend {
11
12     public MatrixFactorizationModel bestModel(JavaPairRDD<Long, Rating>
ratings){
13         //统计有用户数量和动态数量以及用户对动态的评分数目
14         Long numRatings = ratings.count();
15         Long numUsers = ratings.map(v1 ->
(v1._2()).user()).distinct().count();
16         Long numMovies = ratings.map(v1 ->
(v1._2()).product()).distinct().count();
17         System.out.println("用户: " + numUsers + "动态: " + numMovies + "评
论: " + numRatings);
18
19         //将样本评分表以key值切分成3个部分，分别用于训练（60%，并加入用户评分），校验
(20%)，and 测试（20%）
20         //该数据在计算过程中要多次应用到，所以cache到内存
21
22         Integer numPartitions = 4; // 分区数
23         // 训练集
24         JavaRDD<Rating> training = ratings
25             .filter(v -> v._1() < 6)
26             .values()

```

```

27         .repartition(numPartitions)
28         .cache();
29
30     // 校验集
31     JavaRDD<Rating> validation = ratings
32         .filter(v -> v._1() >= 6 && v._1() < 8)
33         .values()
34         .repartition(numPartitions).cache();
35
36     // 测试集
37     JavaRDD<Rating> test = ratings
38         .filter(v -> v._1() >= 8)
39         .values()
40         .cache();
41
42     Long numTraining = training.count();
43     Long numValidation = validation.count();
44     Long numTest = test.count();
45     System.out.println("训练集: " + numTraining + " 校验集: " +
numValidation + " 测试集: " + numTest);
46
47     //训练不同参数下的模型，并在校验集中验证，获取最佳参数下的模
48     int[] ranks = new int[]{10, 11, 12};
49     //     double[] lambdas = new double[]{0.01, 0.03, 0.1, 0.3, 1, 3};
50     double[] lambdas = new double[]{0.01};
51     //     int[] numIters = new int[]{8, 9, 10, 11, 12, 13, 14, 15};
52     int[] numIters = new int[]{8, 9, 10};
53
54     MatrixFactorizationModel bestModel = null;
55     double bestValidationRmse = Double.MAX_VALUE;
56     int bestRank = 0;
57     double bestLambda = -0.01;
58     int bestNumIter = 0;
59
60     for (int rank : ranks) {
61         for (int numIter : numIters) {
62             for (double lambda : lambdas) {
63                 MatrixFactorizationModel model =
ALS.train(training.rdd(), rank, numIter, lambda);
64                 double validationRmse = computeRmse(model, validation,
numValidation);
65                 System.out.println("RMSE(校验集) = " + validationRmse +
", rank = " + rank + ", lambda = " + lambda + ", numIter = " + numIter);
66
67                 if (validationRmse < bestValidationRmse) {
68                     bestModel = model;
69                     bestValidationRmse = validationRmse;
70                     bestRank = rank;
71                     bestLambda = lambda;
72                     bestNumIter = numIter;
73                 }
74             }
75         }
76     }
77
78
79
80     double testRmse = computeRmse(bestModel, test, numTest);

```

```

81         System.out.println("测试数据集在 最佳训练模型 rank = " + bestRank + ",
lambda = " + bestLambda + ", numIter = " + bestNumIter + ", RMSE = " +
testRmse);
82
83         // 计算均值
84         Double meanRating = training.union(validation).mapToDouble(v ->
v.rating()).mean();
85
86         // 计算标准误差值
87         Double baselineRmse = Math.sqrt(test.map(v -> (meanRating -
v.rating()) * (meanRating - v.rating())).reduce((v1, v2) -> (v1 + v2) /
numTest));
88
89         // 计算准确率提升了多少
90         double improvement = (baselineRmse - testRmse) / baselineRmse *
100;
91
92         System.out.println("最佳训练模型的准确率提升了: " +
String.format("%.2f", improvement) + "%.");
93
94
95         // 构建最佳训练模型
96         bestModel = ALS.train(ratings.values().rdd(), bestRank,
bestNumIter, bestLambda);
97
98         return bestModel;
99     }
100
101     /**
102      * 校验集预测数据和实际数据之间的均方根误差
103      */
104     public Double computeRmse(MatrixFactorizationModel model,
JavaRDD<Rating> data, Long n) {
105         // 进行预测
106         JavaRDD<Rating> predictions = model.predict(data.mapToPair(v ->
new Tuple2<>(v.user(), v.product())));
107
108         JavaRDD<Tuple2<Double, Double>> predictionsAndRatings =
predictions
109             .mapToPair(v -> new Tuple2<>(new Tuple2<>(v.user(),
v.product()), v.rating()))
110             .join(data.mapToPair(v -> new Tuple2<>(new Tuple2<>
(v.user(), v.product()), v.rating()))).values();
111
112         Double reduce = predictionsAndRatings.map(v -> (v._1 - v._2) *
(v._1 - v._2))
113             .reduce((v1, v2) -> (v1 + v2) / n);
114         //正平方根
115         return Math.sqrt(reduce);
116     }
117 }
118

```

2.4、SparkMongoDB

实现读取MongoDB，并且使用MLlib的推荐模型进行推荐，最终将推荐结果数据写入到Redis中。

```

1 package com.tanhua.spark.mongo;
2
3 import com.mongodb.spark.MongoSpark;
4 import com.mongodb.spark.rdd.api.java.JavaMongoRDD;
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.bson.Document;
13 import redis.clients.jedis.HostAndPort;
14 import redis.clients.jedis.JedisCluster;
15 import scala.Tuple2;
16
17 import java.io.InputStream;
18 import java.util.*;
19
20 public class SparkMongoDB {
21
22     public static void main(String[] args) throws Exception{
23
24         //读取外部的配置文件
25         InputStream inputStream =
SparkMongoDB.class.getClassLoader().getResourceAsStream("app.properties");
26         Properties properties = new Properties();
27         properties.load(inputStream);
28
29         //构建Spark配置
30         SparkConf sparkConf = new SparkConf()
31             .setAppName("SparkMongoDB")
32             .setMaster("local[*]")
33             .set("spark.mongodb.input.uri",
properties.getProperty("spark.mongodb.input.uri"));
34
35         //构建Spark上下文
36         JavaSparkContext jsc = new JavaSparkContext(sparkConf);
37
38         //加载MongoDB中的数据
39         JavaMongoRDD<Document> rdd = MongoSpark.load(jsc);
40
41         //      rdd.foreach(document -> System.out.println(document.toJson()));
42
43         //在数据中有同一个用户对不同的动态进行评价，需要进行合并操作
44         JavaRDD<Document> values = rdd.mapToPair(document -> {
45             Integer user = document.getLong("userId").intValue();
46             Integer product = document.getLong("publishId").intValue();
47             return new Tuple2<>(user + "_" + product, document);
48         }).reduceByKey((v1, v2) -> {
49             Double score = v1.getDouble("score") + v2.getDouble("score");
50             v1.put("score", score);
51             return v1;
52         }).values();
53
54         //得到数据中的用户id集合
55         List<Long> userIdList = rdd.map(v1 ->
v1.getLong("userId")).distinct().collect();

```

```

56
57 //      values.foreach(document ->
58 System.out.println(document.toJson()));
59
60 //按照日期对10进行取模作为key，Rating对象作为value，获取到数据用于后续的数据
61 处理
62 JavaPairRDD<Long, Rating> ratings = values.mapToPair(document -> {
63     Integer user = document.getLong("userId").intValue();
64     Integer product = document.getLong("publishId").intValue();
65     Double score = document.getDouble("score");
66     Long date = document.getLong("date");
67     Rating rating = new Rating(user, product, score);
68     return new Tuple2<>(date % 10, rating);
69 });
70
71 //通过MLlib模型进行推荐，获取到最优的推荐模型
72 MLlibRecommend mllibRecommend = new MLlibRecommend();
73 MatrixFactorizationModel bestModel =
74 mllibRecommend.bestModel(ratings);
75
76 //构建Redis环境
77 String redisClusterNodes =
78 properties.getProperty("redis.cluster.nodes");
79 String[] redisNodes = redisClusterNodes.split(",");
80 Set<HostAndPort> nodes = new HashSet<>();
81 for (String redisNode : redisNodes) {
82     String[] hostAndPorts = redisNode.split(":");
83     nodes.add(new HostAndPort(hostAndPorts[0],
84 Integer.valueOf(hostAndPorts[1])));
85 }
86 JedisCluster jedisCluster = new JedisCluster(nodes);
87
88 //分别对每一个用户进行推荐，推荐20个动态信息
89 for (Long userId : userIdList) {
90     Rating[] recommendProducts =
91 bestModel.recommendProducts(userId.intValue(), 20);
92
93     List<Integer> products = new ArrayList<>();
94     for (Rating rating : recommendProducts) {
95         products.add(rating.product());
96     }
97
98     //存储到redis
99 String key = "QUANZI_PUBLISH_RECOMMEND_" + userId;
100 System.out.println(key);
101 jedisCluster.set(key, StringUtils.join(products, ','));
102 }
103
104 //关闭连接
105 jedisCluster.close();
106
107 }
108 }
109

```

2.5、测试

```
192.168.31.81:6380> get QUANZI_PUBLISH_RECOMMEND_2
"1003,1012,1062,1079,1077,1081,1053,1097,1018,1085,1048,1026,1020,1042,1049,1005,1094,1022,1074,1035"
192.168.31.81:6380> get QUANZI_PUBLISH_RECOMMEND_1
-> Redirected to slot [4156] located at 192.168.31.81:6379
"1004,1050,1035,1003,1027,1016,1083,1041,1092,1037,1012,1030,1059,1061,1006,1009,1038,1043,1013,1090"
192.168.31.81:6379> get QUANZI_PUBLISH_RECOMMEND_3
-> Redirected to slot [12414] located at 192.168.31.81:6381
"1004,1030,1089,1043,1073,1002,1085,1022,1016,1023,1071,1098,1058,1037,1069,1034,1012,1083,1077,1060"
192.168.31.81:6381>
```

可以看到已经有推荐的数据写入到了Redis中。

3、修改查询逻辑

之前是通过MongoDB直接查询，而现在需要先从Redis进行命中，如果未命中则需要进行MongoDB查询。

修改server工程中的MovementsService类型：

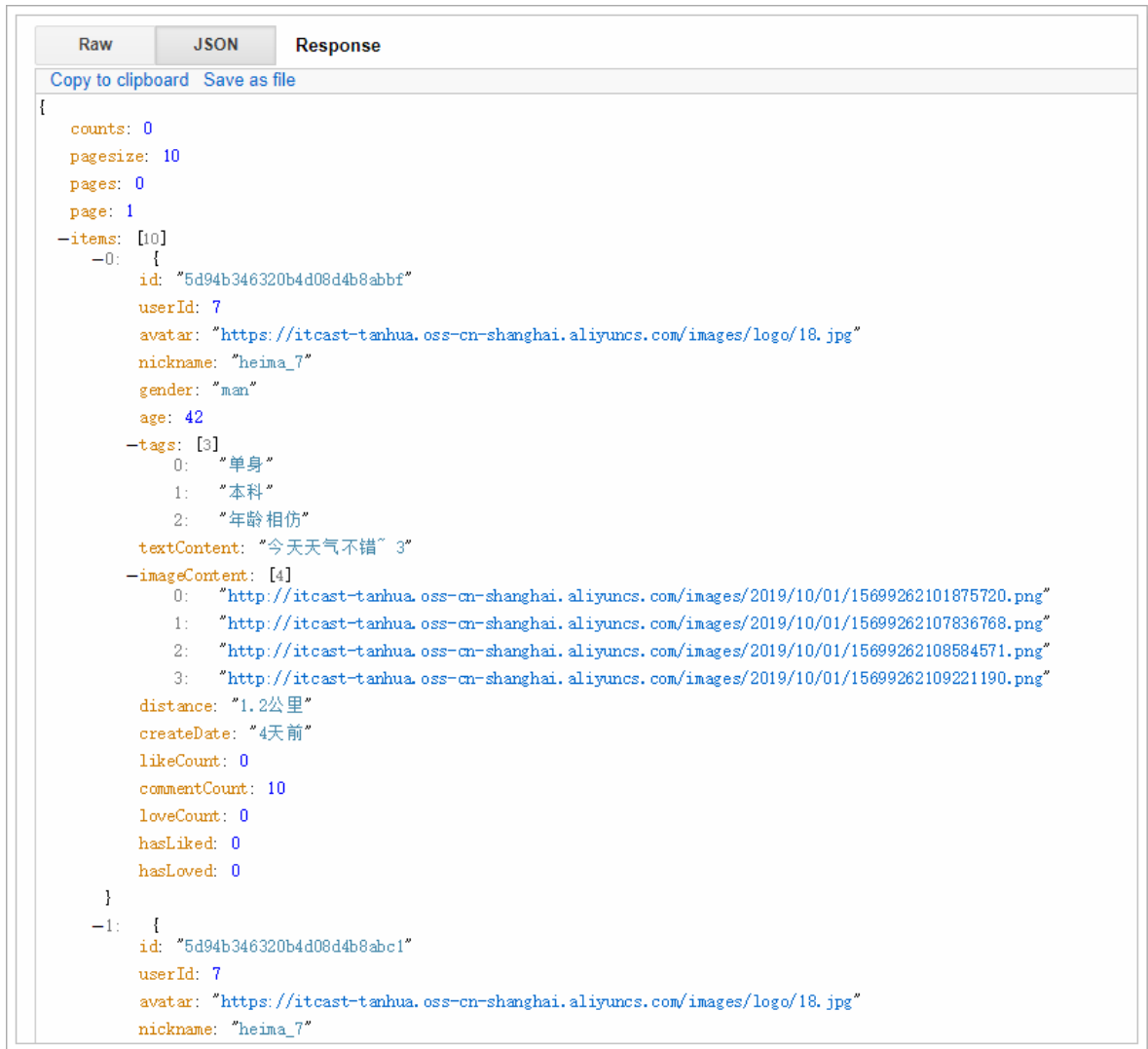
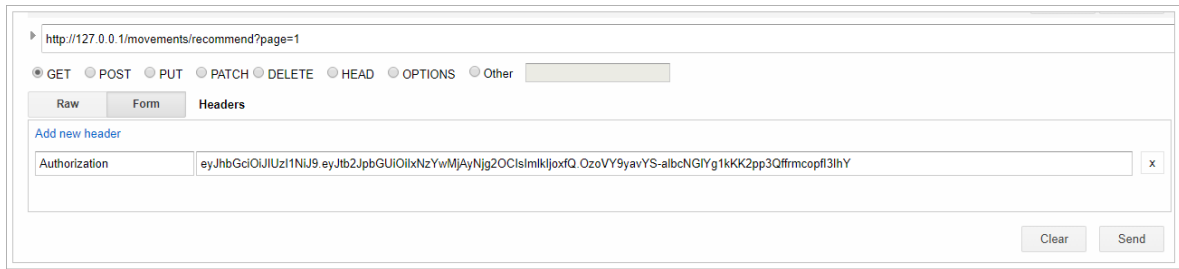
```
1  /**
2      * 查询动态
3      *
4      * @param page
5      * @param pageSize
6      * @return
7      */
8  public PageResult queryPublishList(Integer page, Integer pageSize,
9  boolean isRecommend) {
10      PageResult pageResult = new PageResult();
11      //获取当前的登录信息
12      User user = UserThreadLocal.get();
13
14      PageInfo<Publish> pageInfo = null;
15      if (isRecommend) { //推荐动态逻辑处理
16          // 查询Redis
17          String value =
18  this.redisTemplate.opsForValue().get("QUANZI_PUBLISH_RECOMMEND_" +
19  user.getId());
20          if (StringUtils.isEmpty(value)) {
21              String[] pids = StringUtils.split(value, ',');
22              int startIndex = (page - 1) * pageSize;
23              if(startIndex < pids.length){
24                  int endIndex = startIndex + pageSize - 1;
25                  if (endIndex >= pids.length) {
26                      endIndex = pids.length - 1;
27                  }
28
29                  List<Long> pidList = new ArrayList<>();
30                  for (int i = startIndex; i <= endIndex; i++) {
31                      pidList.add(Long.valueOf(pids[i]));
32                  }
33
34                  List<Publish> publishList =
35  this.quanZiApi.queryPublishByPids(pidList);
36                  pageInfo = new PageInfo<>();
37                  pageInfo.setRecords(publishList);
38              }
39          }
40      }
```

```

38
39         if (null == pageInfo) {
40             Long userId = isRecommend ? null : user.getId();
41             pageInfo = this.quanZiApi.queryPublishList(userId, page,
pageSize);
42         }
43
44         pageResult.setPageSize(pageSize);
45         pageResult.setPage(page);
46         pageResult.setCounts(0);
47         pageResult.setPages(0);
48
49         List<Publish> records = pageInfo.getRecords();
50
51         if (CollectionUtils.isEmpty(records)) {
52             //没有动态信息
53             return pageResult;
54         }
55
56         List<Movements> movementsList = new ArrayList<>();
57         for (Publish record : records) {
58             Movements movements = new Movements();
59
60             movements.setId(record.getId().toHexString());
61             movements.setImageContent(record.getMedias().toArray(new
String[]{}));
62             movements.setTextContent(record.getText());
63             movements.setUserId(record.getUserId());
64             movements.setCreateDate(RelativeDateFormat.format(new
Date(record.getCreated())));
65
66             movementsList.add(movements);
67         }
68
69         List<Long> userIds = new ArrayList<>();
70         for (Movements movements : movementsList) {
71             if (!userIds.contains(movements.getUserId())) {
72                 userIds.add(movements.getUserId());
73             }
74         }
75
76
77         QueryWrapper<UserInfo> queryWrapper = new QueryWrapper<>();
78         queryWrapper.in("user_id", userIds);
79         List<UserInfo> userInfos =
this.userInfoService.queryList(queryWrapper);
80         for (Movements movements : movementsList) {
81             for (UserInfo userInfo : userInfos) {
82                 if (movements.getUserId().longValue() ==
userInfo.getUserId().longValue()) {
83                     this.fillValueToMovements(movements, userInfo);
84                     break;
85                 }
86             }
87         }
88
89         pageResult.setItems(movementsList);
90         return pageResult;

```

测试：



可以看到，已经查询到了动态数据。

4、小视频推荐

小视频的推荐和动态推荐的实现逻辑非常的类似。

4.1、增加自增id

```

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.List;
10
11 @Data
12 @NoArgsConstructor
13 @AllArgsConstructor
14 @Document(collection = "video")
15 public class Video implements java.io.Serializable {
16
17     private static final long serialVersionUID = -3136732836884933873L;
18
19     private ObjectId id; //主键id
20     private Long vid;
21     private Long userId;
22     private String text; //文字
23     private String picUrl; //视频封面文件
24     private String videoUrl; //视频文件
25     private Long created; //创建时间
26     private Integer seeType; // 谁可以看, 1-公开, 2-私密, 3-部分可见, 4-不给谁看
27     private List<Long> seeList; //部分可见的列表
28     private List<Long> notSeeList; //不给谁看的列表
29     private String longitude; //经度
30     private String latitude; //纬度
31     private String locationName; //位置名称
32 }
33

```

修改VideoApiImpl逻辑：

```

1 @Override
2 public Boolean saveVideo(Video video) {
3     if (video.getUserId() == null) {
4         return false;
5     }
6
7     video.setId(ObjectId.get());
8     video.setCreated(System.currentTimeMillis());
9
10    //生成vid
11    video.setVid(this.idService.createId("video",
12    video.getId().toHexString()));
13
14    this.mongoTemplate.save(video);
15    return true;
16 }

```

4.2、动态计分规则

- 发布+2
- 点赞 +5
- 评论 + 10

4.3、发送消息

4.3.1、VideoMQService

```
1 package com.tanhua.server.service;
2
3 import com.alibaba.dubbo.config.annotation.Reference;
4 import com.tanhua.dubbo.server.api.QuanZiApi;
5 import com.tanhua.dubbo.server.api.VideoApi;
6 import com.tanhua.dubbo.server.pojo.Publish;
7 import com.tanhua.dubbo.server.pojo.Video;
8 import com.tanhua.server.pojo.User;
9 import com.tanhua.server.utils.UserThreadLocal;
10 import org.apache.rocketmq.spring.core.RocketMQTemplate;
11 import org.slf4j.Logger;
12 import org.slf4j.LoggerFactory;
13 import org.springframework.beans.factory.annotation.Autowired;
14 import org.springframework.stereotype.Service;
15
16 import java.util.HashMap;
17 import java.util.Map;
18
19 @Service
20 public class VideoMQService {
21
22     private static final Logger LOGGER =
23     LoggerFactory.getLogger(VideoMQService.class);
24
25     @Autowired
26     private RocketMQTemplate rocketMQTemplate;
27
28     @Reference(version = "1.0.0")
29     private VideoApi videoApi;
30
31     /**
32      * 发布小视频消息
33      *
34      * @return
35      */
36     public Boolean videoMsg(String videoId) {
37         return this.sendMsg(videoId, 1);
38     }
39
40     /**
41      * 点赞小视频
42      *
43      * @return
44      */
45     public Boolean likeVideoMsg(String videoId) {
46         return this.sendMsg(videoId, 2);
47     }
48
49     /**
50      * 取消点赞小视频
51      *
52      * @return
53      */
54     public Boolean disLikeVideoMsg(String videoId) {
55         return this.sendMsg(videoId, 3);
56     }
57
58     /**
```

```

58     * 评论小视频
59     *
60     * @return
61     */
62     public Boolean commentVideoMsg(String videoId) {
63         return this.sendMsg(videoId, 4);
64     }
65
66     /**
67     * 发送小视频操作相关的消息
68     *
69     * @param videoId
70     * @param type      1-发动态, 2-点赞, 3-取消点赞, 4-评论
71     * @return
72     */
73     private Boolean sendMsg(String videoId, Integer type) {
74         try {
75             User user = UserThreadLocal.get();
76
77             Video video = this.videoApi.queryVideoById(videoId);
78
79             //构建消息
80             Map<String, Object> msg = new HashMap<>();
81             msg.put("userId", user.getId());
82             msg.put("date", System.currentTimeMillis());
83             msg.put("videoId", videoId);
84             msg.put("vid", video.getVid());
85             msg.put("type", type);
86
87             this.rocketMQTemplate.convertAndSend("tanhua-video", msg);
88         } catch (Exception e) {
89             LOGGER.error("发送消息失败! videoId = " + videoId + ", type = " +
type, e);
90             return false;
91         }
92
93         return true;
94     }
95 }
96

```

4.3.2、VideoController

```

1  package com.tanhua.server.controller;
2
3  import com.tanhua.server.service.MovementsService;
4  import com.tanhua.server.service.VideoMQService;
5  import com.tanhua.server.service.VideoService;
6  import com.tanhua.server.vo.PageResult;
7  import org.apache.commons.lang3.StringUtils;
8  import org.springframework.beans.factory.annotation.Autowired;
9  import org.springframework.http.HttpStatus;
10 import org.springframework.http.ResponseEntity;
11 import org.springframework.web.bind.annotation.*;
12 import org.springframework.web.multipart.MultipartFile;
13
14 import java.util.Map;

```

```

15
16 @RestController
17 @RequestMapping("smallVideos")
18 public class VideoController {
19
20     @Autowired
21     private VideoService videoService;
22
23     @Autowired
24     private MovementsService movementsService;
25
26     @Autowired
27     private CommentsController commentsController;
28
29     @Autowired
30     private VideoMQService videoMQService;
31
32     /**
33      * 发布小视频
34      *
35      * @param picFile
36      * @param videoFile
37      * @return
38      */
39     @PostMapping
40     public ResponseEntity<Void> saveVideo(@RequestParam(value =
41 "videoThumbnail", required = false) MultipartFile picFile,
42                                     @RequestParam(value =
43 "videoFile", required = false) MultipartFile videoFile) {
44         try {
45             String id = this.videoService.saveVideo(picFile, videoFile);
46             if (StringUtils.isEmpty(id)) {
47                 //发送消息
48                 this.videoMQService.videoMsg(id);
49
50                 return ResponseEntity.ok(null);
51             }
52         } catch (Exception e) {
53             e.printStackTrace();
54         }
55         return
56         ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
57     }
58
59     /**
60      * 查询小视频列表
61      *
62      * @param page
63      * @param pageSize
64      * @return
65      */
66     @GetMapping
67     public ResponseEntity<PageResult> queryVideoList(@RequestParam(value =
68 "page", defaultValue = "1") Integer page,
69                                     @RequestParam(value =
70 "pagesize", defaultValue = "10") Integer pageSize) {
71         try {
72             if (page <= 0) {

```

```

68         page = 1;
69     }
70     PageResult pageResult = this.videoService.queryVideoList(page,
pageSize);
71     if (null != pageResult) {
72         return ResponseEntity.ok(pageResult);
73     }
74     } catch (Exception e) {
75         e.printStackTrace();
76     }
77     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
78 }
79
80 /**
81  * 视频点赞
82  *
83  * @param videoId 视频id
84  * @return
85  */
86 @PostMapping("/{id}/like")
87 public ResponseEntity<Long> likeComment(@PathVariable("id") String
videoId) {
88     try {
89         Long likeCount = this.movementsService.likeComment(videoId);
90         if (likeCount != null) {
91             this.videoMQService.likeVideoMsg(videoId);
92             return ResponseEntity.ok(likeCount);
93         }
94     } catch (Exception e) {
95         e.printStackTrace();
96     }
97     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
98 }
99
100 /**
101  * 取消点赞
102  *
103  * @param videoId
104  * @return
105  */
106 @PostMapping("/{id}/dislike")
107 public ResponseEntity<Long> disLikeComment(@PathVariable("id") String
videoId) {
108     try {
109         Long likeCount =
this.movementsService.cancelLikeComment(videoId);
110         if (null != likeCount) {
111             this.videoMQService.disLikeVideoMsg(videoId);
112             return ResponseEntity.ok(likeCount);
113         }
114     } catch (Exception e) {
115         e.printStackTrace();
116     }
117     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
118 }

```

```

119
120     /**
121     * 评论点赞
122     *
123     * @param publishId
124     * @return
125     */
126     @PostMapping("/comments/{id}/like")
127     public ResponseEntity<Long> commentsLikeComment(@PathVariable("id")
String publishId) {
128         try {
129             Long likeCount = this.movementsService.likeComment(publishId);
130             if (likeCount != null) {
131                 return ResponseEntity.ok(likeCount);
132             }
133         } catch (Exception e) {
134             e.printStackTrace();
135         }
136         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
137     }
138
139     /**
140     * 评论取消点赞
141     *
142     * @param publishId
143     * @return
144     */
145     @PostMapping("/comments/{id}/dislike")
146     public ResponseEntity<Long> disCommentsLikeComment(@PathVariable("id")
String publishId) {
147         try {
148             Long likeCount =
this.movementsService.cancelLikeComment(publishId);
149             if (null != likeCount) {
150                 return ResponseEntity.ok(likeCount);
151             }
152         } catch (Exception e) {
153             e.printStackTrace();
154         }
155         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
156     }
157
158     /**
159     * 提交评论
160     *
161     * @param param
162     * @param videoId
163     * @return
164     */
165     @PostMapping("/{id}/comments")
166     public ResponseEntity<Void> saveComments(@RequestBody Map<String,
String> param,
167                                             @PathVariable("id") String
videoId) {
168         param.put("movementId", videoId);

```

```

169         ResponseEntity<Void> entity =
this.commentsController.saveComments(param);
170
171         if (entity.getStatusCode().is2xxSuccessful()) {
172             //发送消息
173             this.videoMQService.commentVideoMsg(videoId);
174         }
175
176         return entity;
177     }
178
179     /**
180     * 评论列表
181     */
182     @GetMapping("/{id}/comments")
183     public ResponseEntity<PageResult>
queryCommentsList(@PathVariable("id") String videoId,
184
185     @RequestParam(value = "page", defaultValue = "1") Integer page,
186
187     @RequestParam(value = "pagesize", defaultValue = "10") Integer pagesize)
188     {
189         return this.commentsController.queryCommentsList(videoId, page,
190         pagesize);
191     }
192
193     /**
194     * 视频用户关注
195     */
196     @PostMapping("/{id}/userFocus")
197     public ResponseEntity<Void> saveUserFocusComments(@PathVariable("id")
Long userId) {
198         try {
199             Boolean bool = this.videoService.followUser(userId);
200             if (bool) {
201                 return ResponseEntity.ok(null);
202             }
203         } catch (Exception e) {
204             e.printStackTrace();
205         }
206         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
207     }
208
209     /**
210     * 视频用户关注
211     */
212     @PostMapping("/{id}/userUnFocus")
213     public ResponseEntity<Void>
saveUserUnFocusComments(@PathVariable("id") Long userId) {
214         try {
215             Boolean bool = this.videoService.disFollowUser(userId);
216             if (bool) {
217                 return ResponseEntity.ok(null);
218             }
219         } catch (Exception e) {
220             e.printStackTrace();
221         }
222     }

```

```

218         return
219         ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
220     }
221 }

```

4.4、接收消息

4.4.1、RecommendVideo

```

1  package com.tanhua.recommend.pojo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6  import org.bson.types.ObjectId;
7
8  @Data
9  @NoArgsConstructor
10 @AllArgsConstructor
11 public class RecommendVideo {
12
13     private ObjectId id;
14     private Long userId; // 用户id
15     private Long videoId; // 视频id, 需要转化为Long类型
16     private Double score; // 得分
17     private Long date; // 时间戳
18 }
19

```

4.4.2、VideoMsgConsumer

```

1  package com.tanhua.recommend.msg;
2
3  import com.fasterxml.jackson.databind.JsonNode;
4  import com.fasterxml.jackson.databind.ObjectMapper;
5  import com.tanhua.recommend.pojo.Publish;
6  import com.tanhua.recommend.pojo.RecommendQuanZi;
7  import com.tanhua.recommend.pojo.RecommendVideo;
8  import org.apache.commons.lang3.StringUtils;
9  import org.apache.rocketmq.spring.annotation.RocketMQMessageListener;
10 import org.apache.rocketmq.spring.core.RocketMQListener;
11 import org.bson.types.ObjectId;
12 import org.joda.time.DateTime;
13 import org.slf4j.Logger;
14 import org.slf4j.LoggerFactory;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.data.mongodb.core.MongoTemplate;
17 import org.springframework.stereotype.Component;
18 import org.springframework.util.CollectionUtils;
19
20 @Component
21 @RocketMQMessageListener(topic = "tanhua-video",
22     consumerGroup = "tanhua-video-consumer")
23 public class VideoMsgConsumer implements RocketMQListener<String> {
24

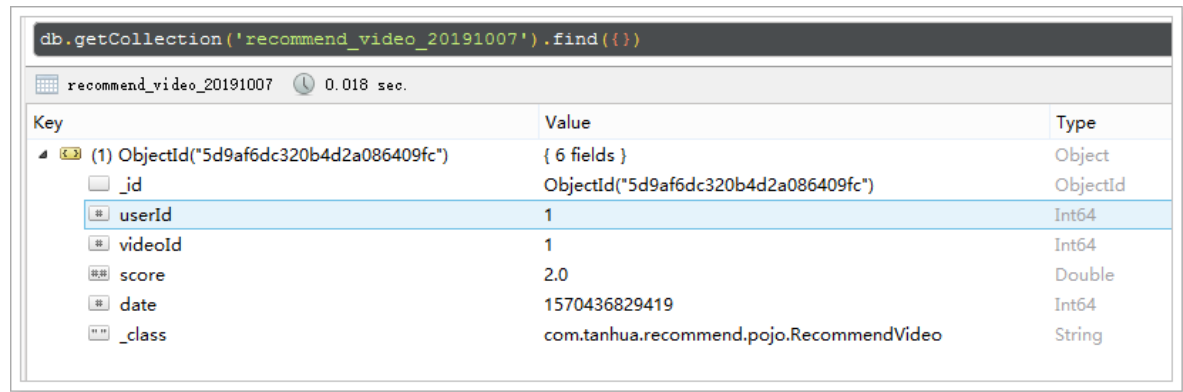
```

```

25     private static final ObjectMapper MAPPER = new ObjectMapper();
26
27     private static final Logger LOGGER =
28     LoggerFactory.getLogger(VideoMsgConsumer.class);
29
30     @Autowired
31     private MongoTemplate mongoTemplate;
32
33     @Override
34     public void onMessage(String msg) {
35         try {
36             JsonNode jsonNode = MAPPER.readTree(msg);
37
38             Long userId = jsonNode.get("userId").asLong();
39             Long vid = jsonNode.get("vid").asLong();
40             String videoId = jsonNode.get("videoId").asText();
41             Integer type = jsonNode.get("type").asInt();
42
43             //1-发动态, 2-点赞, 3-取消点赞, 4-评论
44             RecommendVideo recommendVideo = new RecommendVideo();
45             recommendVideo.setUserId(userId);
46             recommendVideo.setId(ObjectId.get());
47             recommendVideo.setDate(System.currentTimeMillis());
48             recommendVideo.setVideoId(vid);
49
50             switch (type) {
51                 case 1: {
52                     recommendVideo.setScore(2d);
53                     break;
54                 }
55                 case 2: {
56                     recommendVideo.setScore(5d);
57                     break;
58                 }
59                 case 3: {
60                     recommendVideo.setScore(-5d);
61                     break;
62                 }
63                 case 4: {
64                     recommendVideo.setScore(10d);
65                     break;
66                 }
67                 default: {
68                     recommendVideo.setScore(0d);
69                     break;
70                 }
71             }
72
73             String collectionName = "recommend_video_" + new
74             DateTime().toString("yyyyMMdd");
75             this.mongoTemplate.save(recommendVideo, collectionName);
76
77         } catch (Exception e) {
78             LOGGER.error("处理小视频消息失败~" + msg, e);
79         }
80     }

```

4.4.3、测试



Key	Value	Type
(1) ObjectId("5d9af6dc320b4d2a086409fc")	{ 6 fields }	Object
_id	ObjectId("5d9af6dc320b4d2a086409fc")	ObjectId
userId	1	Int64
videoId	1	Int64
score	2.0	Double
date	1570436829419	Int64
_class	com.tanhua.recommend.pojo.RecommendVideo	String

4.5、构造数据

```
1 package com.tanhua.dubbo.server.api;
2
3 import com.tanhua.dubbo.server.pojo.Video;
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.data.mongodb.core.MongoTemplate;
11 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
12
13 @RunWith(SpringJUnit4ClassRunner.class)
14 @SpringBootTest
15 public class TestVideoApi {
16
17     @Autowired
18     private VideoApi videoApi;
19
20     @Autowired
21     private MongoTemplate mongoTemplate;
22
23     @Test
24     public void testSaveVideo() {
25         for (int i = 0; i < 100; i++) {
26             Video video = new Video();
27             video.setId(ObjectId.get());
28             video.setUserId(RandomUtils.nextLong(1, 10));
29             video.setVid(Long.valueOf(1000 + i));
30             video.setLocationName("上海市");
31             video.setSeeType(1);
32             video.setPicUrl("http://itcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/10/07/15704367549907417.png");
33
34             video.setVideoUrl("http://192.168.31.81:8888/group1/M00/00/02/wkgfUV2a9pmAQgILACYJuhDRF3g362.mp4");
35             video.setCreated(System.currentTimeMillis());
36             String videoId = this.videoApi.saveVideo(video);
37             System.out.println(videoId);
38         }
39     }
40 }
```

```

38     }
39
40     @Test
41     public void testSaveRecommend(){
42         for (int i = 0; i < 1000; i++) {
43             RecommendVideo recommendQuanZi = new RecommendVideo();
44             recommendQuanZi.setDate(System.currentTimeMillis());
45             recommendQuanZi.setId(ObjectId.get());
46
47             recommendQuanZi.setScore(Double.valueOf(RandomUtils.nextInt(0,15)));
48             recommendQuanZi.setUserId(RandomUtils.nextLong(1,10));
49             recommendQuanZi.setVideoId(RandomUtils.nextLong(1000,1099));
50
51             this.mongoTemplate.save(recommendQuanZi,
52 "recommend_video_20191007");
53         }
54     }
55 }

```

4.6、实现推荐

```

1  package com.tanhua.spark.mongo;
2
3  import com.mongodb.spark.MongoSpark;
4  import com.mongodb.spark.rdd.api.java.JavaMongoRDD;
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.bson.Document;
13 import redis.clients.jedis.HostAndPort;
14 import redis.clients.jedis.JedisCluster;
15 import scala.Tuple2;
16
17 import java.io.InputStream;
18 import java.util.*;
19
20 public class SparkVideoMongoDB {
21
22     public static void main(String[] args) throws Exception{
23
24         //读取外部的配置文件
25         InputStream inputStream =
26 SparkVideoMongoDB.class.getClassLoader().getResourceAsStream("app.properties");
27
28         Properties properties = new Properties();
29         properties.load(inputStream);
30
31         //构建Spark配置
32         SparkConf sparkConf = new SparkConf()
33             .setAppName("SparkVideoMongoDB")
34             .setMaster("local[*]")

```

```

33         .set("spark.mongodb.input.uri",
properties.getProperty("spark.video.mongodb.input.uri"));
34
35     //构建Spark上下文
36     JavaSparkContext jsc = new JavaSparkContext(sparkConf);
37
38     //加载MongoDB中的数据
39     JavaMongoRDD<Document> rdd = MongoSpark.load(jsc);
40
41     //      rdd.foreach(document -> System.out.println(document.toJson()));
42
43     //在数据中有同一个用户对不同的小视频进行评价，需要进行合并操作
44     JavaRDD<Document> values = rdd.mapToPair(document -> {
45         Integer user = document.getLong("userId").intValue();
46         Integer product = document.getLong("videoId").intValue();
47         return new Tuple2<>(user + "_" + product, document);
48     }).reduceByKey((v1, v2) -> {
49         Double score = v1.getDouble("score") + v2.getDouble("score");
50         v1.put("score", score);
51         return v1;
52     }).values();
53
54     //得到数据中的用户id集合
55     List<Long> userIdList = rdd.map(v1 ->
v1.getLong("userId")).distinct().collect();
56
57     //      values.foreach(document ->
System.out.println(document.toJson()));
58
59     //按照日期对10进行取模作为key，Rating对象作为value，获取到数据用于后续的数据
处理
60     JavaPairRDD<Long, Rating> ratings = values.mapToPair(document -> {
61         Integer user = document.getLong("userId").intValue();
62         Integer product = document.getLong("videoId").intValue();
63         Double score = document.getDouble("score");
64         Long date = document.getLong("date");
65         Rating rating = new Rating(user, product, score);
66         return new Tuple2<>(date % 10, rating);
67     });
68
69     //通过MLlib模型进行推荐，获取到最优的推荐模型
70     MLlibRecommend mllibRecommend = new MLlibRecommend();
71     MatrixFactorizationModel bestModel =
mllibRecommend.bestModel(ratings);
72
73     //构建Redis环境
74     String redisClusterNodes =
properties.getProperty("redis.cluster.nodes");
75     String[] redisNodes = redisClusterNodes.split(",");
76     Set<HostAndPort> nodes = new HashSet<>();
77     for (String redisNode : redisNodes) {
78         String[] hostAndPorts = redisNode.split(":");
79         nodes.add(new HostAndPort(hostAndPorts[0],
Integer.valueOf(hostAndPorts[1])));
80     }
81     JedisCluster jedisCluster = new JedisCluster(nodes);
82
83     //分别对每一个用户进行推荐，推荐20个小视频信息

```

```

84         for (Long userId : userIdList) {
85             Rating[] recommendProducts =
bestModel.recommendProducts(userId.intValue(), 20);
86
87             List<Integer> products = new ArrayList<>();
88             for (Rating rating : recommendProducts) {
89                 products.add(rating.product());
90             }
91
92             //存储到redis
93             String key = "QUANZI_VIDEO_RECOMMEND_" + userId;
94             jedisCluster.set(key, StringUtils.join(products, ','));
95         }
96
97         //关闭连接
98         jedisCluster.close();
99         jedisCluster.close();
100
101     }
102 }
103

```

测试：

```

192.168.31.81:6379> get QUANZI_VIDEO_RECOMMEND_1
"1046,1083,1006,1053,1016,1064,1072,1067,1025,1089,1084,1063,1091,1038,1013,1057,1042,1095,1035,1041"
192.168.31.81:6379> get QUANZI_VIDEO_RECOMMEND_2
-> Redirected to slot [13937] located at 192.168.31.81:6381
"1063,1013,1041,1047,1081,1044,1050,1079,1040,1018,1077,1061,1009,1095,1045,1058,1066,1053,1012,1033"
192.168.31.81:6381> get QUANZI_VIDEO_RECOMMEND_3
-> Redirected to slot [9808] located at 192.168.31.81:6380
"1075,1052,1091,1092,1076,1079,1037,1016,1004,1009,1042,1063,1045,1059,1001,1032,1017,1086,1014,1094"
192.168.31.81:6380>

```

4.7、修改查询逻辑

修改VideoService的实现：

```

1  public PageResult queryVideoList(Integer page, Integer pageSize) {
2
3      User user = UserThreadLocal.get();
4
5      PageResult pageResult = new PageResult();
6      pageResult.setPage(page);
7      pageResult.setPageSize(pageSize);
8      pageResult.setPages(0);
9      pageResult.setCounts(0);
10
11      PageInfo<Video> pageInfo = null;
12
13      //先从Redis进行命中，如果命中则返回推荐列表，如果未命中查询默认列表
14      String redisValue =
this.redisTemplate.opsForValue().get("QUANZI_VIDEO_RECOMMEND_" +
user.getId());
15      if (StringUtils.isEmpty(redisValue)) {
16          String[] pids = StringUtils.split(redisValue, ',');
17          int startIndex = (page - 1) * pageSize;
18          if (startIndex < pids.length) {
19              int endIndex = startIndex + pageSize - 1;
20              if (endIndex >= pids.length) {

```

```

21         endIndex = pids.length - 1;
22     }
23
24     List<Long> vidList = new ArrayList<>();
25     for (int i = startIndex; i <= endIndex; i++) {
26         vidList.add(Long.valueOf(pids[i]));
27     }
28
29     List<Video> videoList =
this.videoApi.queryVideoListByPids(vidList);
30     pageInfo = new PageInfo<>();
31     pageInfo.setRecords(videoList);
32 }
33 }
34
35 if(null == pageInfo){
36     pageInfo = this.videoApi.queryVideoList(page, pageSize);
37 }
38
39 List<Video> records = pageInfo.getRecords();
40 List<VideoVo> videoVoList = new ArrayList<>();
41 List<Long> userIds = new ArrayList<>();
42 for (Video record : records) {
43     VideoVo videoVo = new VideoVo();
44
45     videoVo.setUserId(record.getUserId());
46     videoVo.setCover(record.getPicUrl());
47     videoVo.setVideoUrl(record.getVideoUrl());
48     videoVo.setId(record.getId().toHexString());
49     videoVo.setSignature("我就是我~");
50
51     Long commentCount =
this.quanZiApi.queryCommentCount(videoVo.getId(), 2);
52     videoVo.setCommentCount(commentCount == null ? 0 :
commentCount.intValue()); // 评论数
53
54     String followUserKey = "VIDEO_FOLLOW_USER_" + user.getId() +
"_" + videoVo.getUserId();
55     videoVo.setHasFocus(this.redisTemplate.hasKey(followUserKey) ?
1 : 0); //是否关注
56
57     String userKey = "QUANZI_COMMENT_LIKE_USER_" + user.getId() +
"_" + videoVo.getId();
58     videoVo.setHasLiked(this.redisTemplate.hasKey(userKey) ? 1 :
0); //是否点赞 (1是, 0否)
59
60     String key = "QUANZI_COMMENT_LIKE_" + videoVo.getId();
61     String value = this.redisTemplate.opsForValue().get(key);
62     if (StringUtils.isEmpty(value)) {
63         videoVo.setLikeCount(Integer.valueOf(value)); //点赞数
64     } else {
65         videoVo.setLikeCount(0);
66     }
67
68     if (!userIds.contains(record.getUserId())) {
69         userIds.add(record.getUserId());
70     }
71

```

```
72         videoVoList.add(videoVo);
73     }
74
75     QueryWrapper<UserInfo> queryWrapper = new QueryWrapper();
76     queryWrapper.in("user_id", userIds);
77     List<UserInfo> userInfos =
78     this.userInfoService.queryList(queryWrapper);
79     for (VideoVo videoVo : videoVoList) {
80         for (UserInfo userInfo : userInfos) {
81             if (videoVo.getUserId().longValue() ==
82             userInfo.getUserId().longValue()) {
83
84                 videoVo.setNickname(userInfo.getNickName());
85                 videoVo.setAvatar(userInfo.getLogo());
86
87                 break;
88             }
89         }
90     }
91
92     pageResult.setItems(videoVoList);
93     return pageResult;
94 }
```