

课程说明

- 圈子实现点赞、喜欢功能
- 圈子实现评论
- 圈子实现评论的点赞
- 小视频功能说明
- FastDFS环境搭建
- 小视频的功能实现

1、评论

点赞、喜欢、评论等均可理解为用户对动态的评论。

1.1、点赞

1.1.1、定义dubbo接口

```
1 package com.tanhua.dubbo.server.api;
2
3 import com.tanhua.dubbo.server.pojo.Publish;
4 import com.tanhua.dubbo.server.vo.PageInfo;
5
6 public interface QuanZiApi {
7
8     /**
9      * 发布动态
10     *
11     * @param publish
12     * @return
13     */
14     boolean savePublish(Publish publish);
15
16     /**
17     * 查询动态
18     *
19     * @return
20     */
21     PageInfo<Publish> queryPublishList(Long userId, Integer page, Integer
    pageSize);
22
23     /**
24     * 点赞
25     *
26     * @param userId
27     * @param publishId
28     * @return
29     */
30     boolean saveLikeComment(Long userId, String publishId);
31
32     /**
33     * 取消点赞、喜欢等
34     *
35     * @param userId
```

```

36     * @param publishId
37     * @return
38     */
39     boolean removeComment(Long userId, String publishId, Integer
commentType);
40
41     /**
42     * 喜欢
43     *
44     * @param userId
45     * @param publishId
46     * @return
47     */
48     boolean saveLoveComment(Long userId, String publishId);
49
50     /**
51     * 保存评论
52     *
53     * @param userId
54     * @param publishId
55     * @param type
56     * @param content
57     * @return
58     */
59     boolean saveComment(Long userId, String publishId, Integer type, String
content);
60
61     /**
62     * 查询评论数
63     *
64     * @param publishId
65     * @param type
66     * @return
67     */
68     Long queryCommentCount(String publishId, Integer type);
69
70 }
71

```

1.1.2、编写实现

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.alibaba.dubbo.config.annotation.Service;
4  import com.mongodb.client.result.DeleteResult;
5  import com.tanhua.dubbo.server.pojo.*;
6  import com.tanhua.dubbo.server.vo.PageInfo;
7  import org.apache.commons.lang3.StringUtils;
8  import org.bson.types.ObjectId;
9  import org.springframework.beans.factory.annotation.Autowired;
10 import org.springframework.data.domain.PageRequest;
11 import org.springframework.data.domain.Sort;
12 import org.springframework.data.mongodb.core.MongoTemplate;
13 import org.springframework.data.mongodb.core.query.Criteria;
14 import org.springframework.data.mongodb.core.query.Query;
15
16 import java.util.ArrayList;

```

```

17 import java.util.Date;
18 import java.util.List;
19
20 @Service(version = "1.0.0")
21 public class QuanZiApiImpl implements QuanZiApi {
22
23     @Autowired
24     private MongoTemplate mongoTemplate;
25
26     /**
27      * 点赞
28      *
29      * @param userId
30      * @param publishId
31      */
32     public boolean saveLikeComment(Long userId, String publishId) {
33         Query query = Query.query(Criteria
34             .where("publishId").is(new ObjectId(publishId))
35             .and("userId").is(userId)
36             .and("commentType").is(1));
37         long count = this.mongoTemplate.count(query, Comment.class);
38         if (count > 0) {
39             return false;
40         }
41         return this.saveComment(userId, publishId, 1, null);
42     }
43
44     /**
45      * 取消点赞、喜欢等
46      *
47      * @return
48      */
49     public boolean removeComment(Long userId, String publishId, Integer
commentType) {
50         Query query = Query.query(Criteria
51             .where("publishId").is(new ObjectId(publishId))
52             .and("userId").is(userId)
53             .and("commentType").is(commentType));
54         DeleteResult remove = this.mongoTemplate.remove(query,
Comment.class);
55         return remove.getDeletedCount() > 0;
56     }
57
58     /**
59      * 喜欢
60      *
61      * @param userId
62      * @param publishId
63      */
64     public boolean saveLoveComment(Long userId, String publishId) {
65         Query query = Query.query(Criteria
66             .where("publishId").is(new ObjectId(publishId))
67             .and("userId").is(userId)
68             .and("commentType").is(3));
69         long count = this.mongoTemplate.count(query, Comment.class);
70         if (count > 0) {
71             return false;
72         }

```

```

73         return this.saveComment(userId, publishId, 3, null);
74     }
75
76     /**
77      * 保存评论
78      *
79      * @param userId
80      * @param publishId
81      * @param type
82      * @return
83      */
84     public boolean saveComment(Long userId, String publishId, Integer
type, String content) {
85         try {
86             Comment comment = new Comment();
87             comment.setId(ObjectId.get());
88             comment.setUserId(userId);
89             comment.setContent(content);
90             comment.setPublishId(new ObjectId(publishId));
91             comment.setCommentType(type);
92             comment.setCreated(System.currentTimeMillis());
93             this.mongoTemplate.save(comment);
94             return true;
95         } catch (Exception e) {
96             e.printStackTrace();
97         }
98         return false;
99     }
100
101     @Override
102     public Long queryCommentCount(String publishId, Integer type) {
103         Query query =
Query.query(Criteria.where("publishId").is(publishId).and("commentType").i
s(type));
104         return this.mongoTemplate.count(query, Comment.class);
105     }
106 }
107

```

1.1.3、编写接口服务

MovementsController :

```

1  /**
2      * 点赞
3      *
4      * @param publishId
5      * @return
6      */
7      @GetMapping("/{id}/like")
8      public ResponseEntity<Long> likeComment(@PathVariable("id") String
publishId) {
9          try {
10             Long likeCount = this.movementsService.likeComment(publishId);
11             if (likeCount != null) {
12                 return ResponseEntity.ok(likeCount);
13             }

```

```

14         } catch (Exception e) {
15             e.printStackTrace();
16         }
17         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
18     }
19
20     /**
21      * 取消点赞
22      *
23      * @param publishId
24      * @return
25      */
26     @GetMapping("/{id}/dislike")
27     public ResponseEntity<Long> dislikeComment(@PathVariable("id") String
publishId) {
28         try {
29             Long likeCount =
this.movementsService.cancelLikeComment(publishId);
30             if (null != likeCount) {
31                 return ResponseEntity.ok(likeCount);
32             }
33         } catch (Exception e) {
34             e.printStackTrace();
35         }
36         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
37     }

```

1.1.4、编写服务实现

```

1  package com.tanhua.server.service;
2
3  import com.alibaba.dubbo.common.utils.CollectionUtils;
4  import com.alibaba.dubbo.config.annotation.Reference;
5  import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
6  import com.tanhua.dubbo.server.api.QuanZiApi;
7  import com.tanhua.dubbo.server.pojo.Publish;
8  import com.tanhua.dubbo.server.vo.PageInfo;
9  import com.tanhua.server.pojo.User;
10 import com.tanhua.server.pojo.UserInfo;
11 import com.tanhua.server.utils.RelativeDateFormat;
12 import com.tanhua.server.utils.UserThreadLocal;
13 import com.tanhua.server.vo.Movements;
14 import com.tanhua.server.vo.PageResult;
15 import com.tanhua.server.vo.PicUploadResult;
16 import org.apache.commons.lang3.StringUtils;
17 import org.springframework.beans.factory.annotation.Autowired;
18 import org.springframework.data.redis.core.RedisTemplate;
19 import org.springframework.stereotype.Service;
20 import org.springframework.web.multipart.MultipartFile;
21
22 import java.util.ArrayList;
23 import java.util.Date;
24 import java.util.List;
25
26 @Service

```

```

27 public class MovementsService {
28
29     @Reference(version = "1.0.0")
30     private QuanZiApi quanZiApi;
31
32     @Autowired
33     private PicUploadService picUploadService;
34
35     @Autowired
36     private UserInfoService userInfoService;
37
38     @Autowired
39     private RedisTemplate<String, String> redisTemplate;
40
41     /**
42      * 点赞
43      *
44      * @param publishId
45      * @return
46      */
47     public Long likeComment(String publishId) {
48         User user = UserThreadLocal.get();
49         boolean bool = this.quanZiApi.saveLikeComment(user.getId(),
publishId);
50         if (!bool) {
51             return null;
52         }
53
54         Long likeCount = 0L;
55
56         //保存点赞数到redis
57         String key = "QUANZI_COMMENT_LIKE_" + publishId;
58         if (!this.redisTemplate.hasKey(key)) {
59             Long count = this.quanZiApi.queryCommentCount(publishId, 1);
60             likeCount = count;
61             this.redisTemplate.opsForValue().set(key,
String.valueOf(likeCount));
62         } else {
63             likeCount = this.redisTemplate.opsForValue().increment(key);
64         }
65
66         //记录已点赞
67         String userKey = "QUANZI_COMMENT_LIKE_USER_" + user.getId() + "_" +
publishId;
68         this.redisTemplate.opsForValue().set(userKey, "1");
69
70         return likeCount;
71     }
72
73     /**
74      * 取消点赞
75      *
76      * @return
77      */
78     public Long cancelLikeComment(String publishId) {
79         User user = UserThreadLocal.get();
80         boolean bool = this.quanZiApi.removeComment(user.getId(),
publishId, 1);

```

```

81         if (bool) {
82             String key = "QUANZI_COMMENT_LIKE_" + publishId;
83             //数量递减
84             Long likeCount =
this.redisTemplate.opsForValue().decrement(key);
85
86             //删除已点赞
87             String userKey = "QUANZI_COMMENT_LIKE_USER_" + user.getId() +
"_" + publishId;
88             this.redisTemplate.delete(userKey);
89
90             return likeCount;
91         }
92         return null;
93     }
94 }
95

```

1.1.5、修改查询动态点赞数

```

movements.setNickname(userInfo.getNickName());
movements.setTags(StringUtils.split(userInfo.getTags(), separatorChar: ','));
movements.setCommentCount(10); //TODO 评论数
movements.setDistance("1.2公里"); //TODO 距离

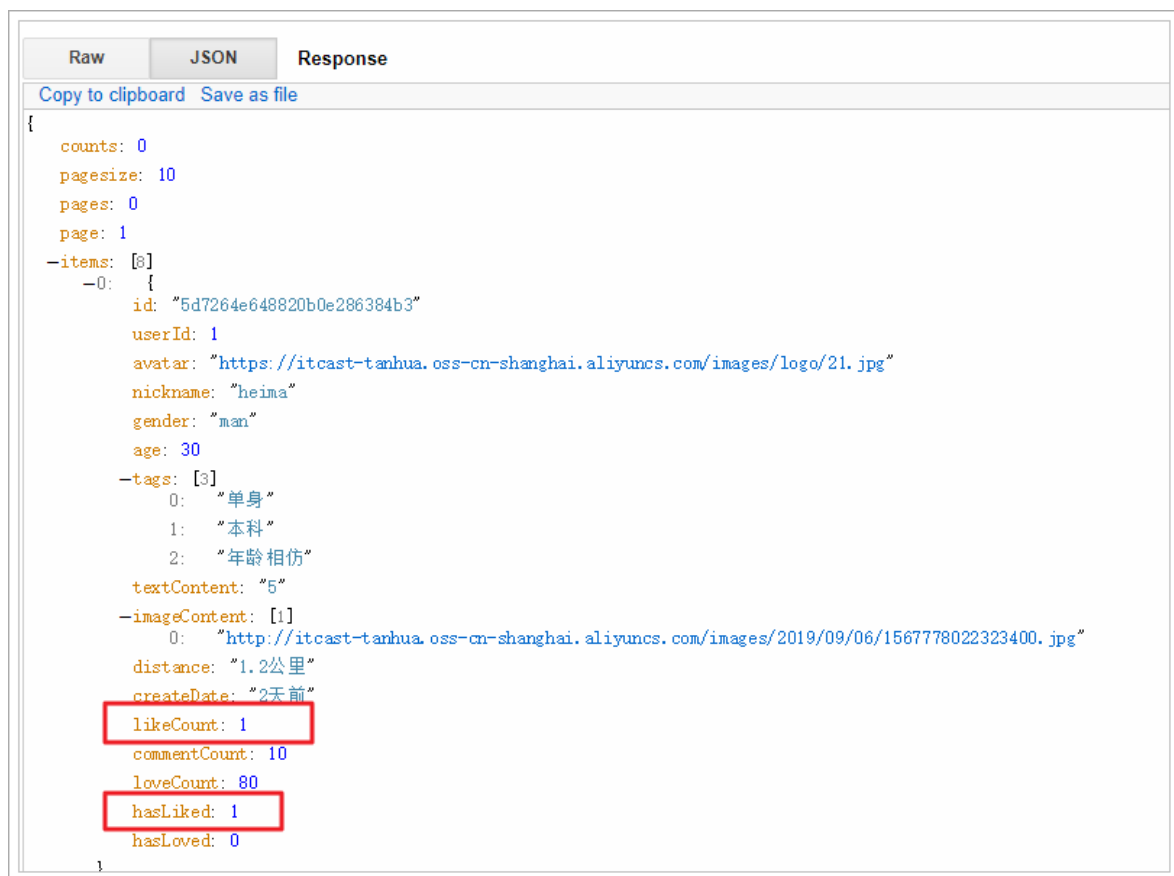
String userKey = "QUANZI_COMMENT_LIKE_USER_" + user.getId() + "_" + movements.getId();
movements.setHasLiked(this.redisTemplate.hasKey(userKey) ? 1 : 0); //是否点赞 (1是, 0否)

movements.setHasLoved(0); //TODO 是否喜欢 (1是, 0否)

String key = "QUANZI_COMMENT_LIKE_" + movements.getId();
String value = this.redisTemplate.opsForValue().get(key);
if (StringUtils.isEmpty(value)) {
    movements.setLikeCount(Integer.valueOf(value)); //点赞数
} else {
    movements.setLikeCount(0);
}

```

1.1.6、测试



1.2、喜欢

1.2.1、MovementsController

```
1  /**
2      * 喜欢
3      *
4      * @param publishId
5      * @return
6      */
7      @GetMapping("/{id}/love")
8      public ResponseEntity<Long> loveComment(@PathVariable("id") String
publishId) {
9          try {
10              Long loveCount = this.movementsService.loveComment(publishId);
11              if (null != loveCount) {
12                  return ResponseEntity.ok(loveCount);
13              }
14          } catch (Exception e) {
15              e.printStackTrace();
16          }
17          return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
18      }
19
20      /**
21      * 取消喜欢
22      *
23      * @param publishId
24      * @return
25      */
26      @GetMapping("/{id}/unlove")
```

```

27     public ResponseEntity<Long> disLoveComment(@PathVariable("id") String
publishId) {
28         try {
29             Long loveCount =
this.movementsService.cancelLoveComment(publishId);
30             if (null != loveCount) {
31                 return ResponseEntity.ok(loveCount);
32             }
33         } catch (Exception e) {
34             e.printStackTrace();
35         }
36         return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
37     }

```

1.2.2、MovementsService

```

1  /**
2      * 喜欢
3      *
4      * @param publishId
5      * @return
6      */
7  public Long loveComment(String publishId) {
8      User user = UserThreadLocal.get();
9      boolean bool = this.quanZiApi.saveLoveComment(user.getId(),
publishId);
10     if (!bool) {
11         return null;
12     }
13
14     Long loveCount = 0L;
15
16     //保存喜欢数到redis
17     String key = "QUANZI_COMMENT_LOVE_" + publishId;
18     if (!this.redisTemplate.hasKey(key)) {
19         Long count = this.quanZiApi.queryCommentCount(publishId, 3);
20         loveCount = count ;
21         this.redisTemplate.opsForValue().set(key,
String.valueOf(loveCount));
22     } else {
23         loveCount = this.redisTemplate.opsForValue().increment(key);
24     }
25
26     //记录已喜欢
27     String userKey = "QUANZI_COMMENT_LOVE_USER_" + user.getId() + "_" +
publishId;
28     this.redisTemplate.opsForValue().set(userKey, "1");
29
30     return loveCount;
31 }
32
33 /**
34     * 取消喜欢
35     *
36     * @return
37     */

```

```
38     public Long cancelLoveComment(String publishId) {
39         User user = UserThreadLocal.get();
40         boolean bool = this.quanZiApi.removeComment(user.getId(),
publishId, 3);
41         if (bool) {
42             String key = "QUANZI_COMMENT_LOVE_" + publishId;
43             //数量递减
44             Long loveCount =
this.redisTemplate.opsForValue().decrement(key);
45
46             //删除已点赞
47             String userKey = "QUANZI_COMMENT_LOVE_USER_" + user.getId() +
"_" + publishId;
48             this.redisTemplate.delete(userKey);
49
50             return loveCount;
51         }
52         return null;
53     }
```

1.2.3、测试



单身 本科 年龄相仿

1



距离1.2公里 42分钟前



1



10



1



heima 30

单身 本科 年龄相仿

test2



动态已喜欢

距离1.2公里 1小时前



0



10



1



heima 30

单身 本科 年龄相仿

test1

发布



交友



圈子



消息



视频



我的

free for personal use



1.3、查询单条动态

1.3.1、定义dubbo接口

QuanZiApi :

```
1  /**
2      * 根据id查询动态
3      *
4      * @param id
5      * @return
6      */
7  Publish queryPublishById(String id);
```

1.3.2、dubbo接口实现

QuanZiApiImpl :

```

1  @Override
2  public Publish queryPublishById(String id) {
3      return this.mongoTemplate.findById(new ObjectId(id), Publish.class);
4  }

```

1.3.3、定义服务接口

```

1  /**
2   * 查询单条动态信息
3   *
4   * @param publishId
5   * @return
6   */
7  @GetMapping("/{id}")
8  public ResponseEntity<Movements> queryById(@PathVariable("id") String
publishId) {
9      try {
10         Movements movements =
this.movementsService.queryById(publishId);
11         if(null != movements){
12             return ResponseEntity.ok(movements);
13         }
14     } catch (Exception e) {
15         e.printStackTrace();
16     }
17     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
18 }

```

1.3.4、服务实现

```

1  public Movements queryById(String publishId) {
2      Publish publish = this.quanZiApi.queryPublishById(publishId);
3      if (null == publish) {
4          return null;
5      }
6
7      Movements movements = new Movements();
8
9      movements.setId(publish.getId().toHexString());
10     movements.setImageContent(publish.getMedias().toArray(new String[]
{}));
11     movements.setTextContent(publish.getText());
12     movements.setUserId(publish.getUserId());
13     movements.setCreateDate(RelativeDateFormat.format(new
Date(publish.getCreated())));
14
15     UserInfo userInfo =
this.userInfoService.queryById(publish.getUserId());
16     if (null == userInfo) {
17         return null;
18     }
19     this.fillValueToMovements(movements, userInfo);
20
21     return movements;
22 }

```

```

23     }
24
25     private void fillValueToMovements(Movements movements, UserInfo
userInfo) {
26         movements.setAge(userInfo.getAge());
27         movements.setAvatar(userInfo.getLogo());
28         movements.setGender(userInfo.getSex().name().toLowerCase());
29         movements.setNickname(userInfo.getNickName());
30         movements.setTags(StringUtils.split(userInfo.getTags(), ','));
31         movements.setCommentCount(10); //TODO 评论数
32         movements.setDistance("1.2公里"); //TODO 距离
33
34         String userKey = "QUANZI_COMMENT_LIKE_USER_" + userInfo.getUserId()
+ "_" + movements.getId();
35         movements.setHasLiked(this.redisTemplate.hasKey(userKey) ? 1 : 0);
//是否点赞 (1是, 0否)
36
37         String key = "QUANZI_COMMENT_LIKE_" + movements.getId();
38         String value = this.redisTemplate.opsForValue().get(key);
39         if (StringUtils.isNotEmpty(value)) {
40             movements.setLikeCount(Integer.valueOf(value)); //点赞数
41         } else {
42             movements.setLikeCount(0);
43         }
44
45         String userLoveKey = "QUANZI_COMMENT_LOVE_USER_" +
userInfo.getUserId() + "_" + movements.getId();
46         movements.setHasLoved(this.redisTemplate.hasKey(userLoveKey) ? 1 :
0); //是否喜欢 (1是, 0否)
47
48         key = "QUANZI_COMMENT_LOVE_" + movements.getId();
49         value = this.redisTemplate.opsForValue().get(key);
50         if (StringUtils.isNotEmpty(value)) {
51             movements.setLoveCount(Integer.valueOf(value)); //喜欢数
52         } else {
53             movements.setLoveCount(0);
54         }
55     }

```

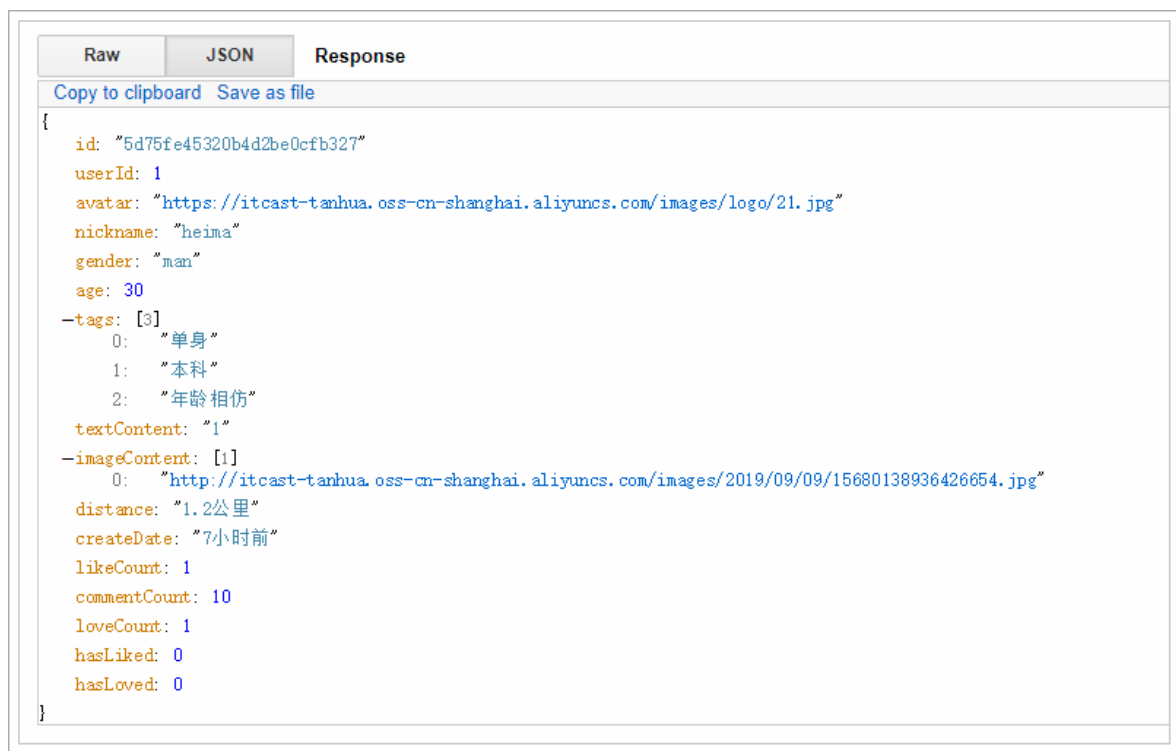
1.3.5、测试

▶

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

RawFormHeaders

Authorization: eyJhbGciOiJIUzI1NiJ9.eyJtb2JpbGU0iIzNTgwMDgwNzk0NCIsImkiIjoyfQ. o2D3ob0Cg0R0QWwBubyrEffdKoacwtSLe4S_tyhM7CU



1.4、单条动态评论

功能包括：查询评论列表，评论点赞、取消点赞。

1.4.1、定义dubbo接口

```
1  /**
2   * 查询评论
3   *
4   * @return
5   */
6  PageInfo<Comment> queryCommentList(String publishId, Integer page,
    Integer pageSize);
```

1.4.2、编写实现

```
1  @Override
2  public PageInfo<Comment> queryCommentList(String publishId, Integer
    page, Integer pageSize) {
3      PageRequest pageRequest = PageRequest.of(page - 1, pageSize,
        Sort.by(Sort.Order.asc("created")));
4      Query query = new Query(Criteria
5          .where("publishId").is(new ObjectId(publishId))
6          .and("commentType").is(2)).with(pageRequest);
7
8      //查询时间线表
9      List<Comment> timeLineList = this.mongoTemplate.find(query,
        Comment.class);
10
11      PageInfo<Comment> pageInfo = new PageInfo<>();
12      pageInfo.setPageNum(page);
13      pageInfo.setPageSize(pageSize);
14      pageInfo.setRecords(timeLineList);
15      pageInfo.setTotal(0); //不提供总数
16      return pageInfo;
```

1.4.3、服务接口实现

```

1  package com.tanhua.server.vo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;
6
7  /**
8   * 评论
9   */
10 @Data
11 @NoArgsConstructor
12 @AllArgsConstructor
13 public class Comments {
14
15     private String id; //评论id
16     private String avatar; //头像
17     private String nickname; //昵称
18     private String content; //评论
19     private String createDate; //评论时间: 08:27
20     private Integer likeCount; //点赞数
21     private Integer hasLiked; //是否点赞 (1是, 0否)
22
23 }
24

```

```

1  package com.tanhua.server.controller;
2
3  import com.tanhua.server.service.CommentsService;
4  import com.tanhua.server.service.MovementsService;
5  import com.tanhua.server.vo.Comments;
6  import com.tanhua.server.vo.PageResult;
7  import org.springframework.beans.factory.annotation.Autowired;
8  import org.springframework.http.HttpStatus;
9  import org.springframework.http.ResponseEntity;
10 import org.springframework.web.bind.annotation.*;
11
12 import java.util.Map;
13
14 @RestController
15 @RequestMapping("comments")
16 public class CommentsController {
17
18     @Autowired
19     private CommentsService commentsService;
20
21     @Autowired
22     private MovementsService movementsService;
23
24     /**
25      * 查询评论列表
26      *
27      * @param publishId

```

```

28     * @param page
29     * @param pageSize
30     * @return
31     */
32     @GetMapping
33     public ResponseEntity<PageResult>
queryCommentsList(@RequestParam("movementId") String publishId,
34
35     @RequestParam(value = "page", defaultValue = "1") Integer page,
36
37     @RequestParam(value = "pageSize", defaultValue = "10") Integer pageSize)
{
38     try {
39         PageResult pageResult =
this.commentsService.queryCommentsList(publishId, page, pageSize);
40         return ResponseEntity.ok(pageResult);
41     } catch (Exception e) {
42         e.printStackTrace();
43     }
44     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
45
46     /**
47     * 保存评论
48     */
49     @PostMapping
50     public ResponseEntity<Void> saveComments(@RequestBody
Map<String,String> param) {
51     try {
52         String publishId = param.get("movementId");
53         String content = param.get("comment");
54         Boolean result = this.commentsService.saveComments(publishId,
content);
55         if (result) {
56             return ResponseEntity.ok(null);
57         }
58     } catch (Exception e) {
59         e.printStackTrace();
60     }
61     return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
62
63     /**
64     * 点赞
65     *
66     * @param publishId
67     * @return
68     */
69     @GetMapping("/{id}/like")
70     public ResponseEntity<Long> likeComment(@PathVariable("id") String
publishId) {
71     try {
72         Long likeCount = this.movementsService.likeComment(publishId);
73         if (likeCount != null) {
74             return ResponseEntity.ok(likeCount);
75

```

```

76         }
77     } catch (Exception e) {
78         e.printStackTrace();
79     }
80     return
ResponseBody.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
81 }
82
83 /**
84  * 取消点赞
85  *
86  * @param publishId
87  * @return
88  */
89 @GetMapping("/{id}/dislike")
90 public ResponseEntity<Long> disLikeComment(@PathVariable("id") String
publishId) {
91     try {
92         Long likeCount =
this.movementsService.cancelLikeComment(publishId);
93         if (null != likeCount) {
94             return ResponseEntity.ok(likeCount);
95         }
96     } catch (Exception e) {
97         e.printStackTrace();
98     }
99     return
ResponseBody.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
100 }
101
102 }
103

```

```

1  package com.tanhua.server.service;
2
3  import com.alibaba.dubbo.config.annotation.Reference;
4  import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;
5  import com.tanhua.dubbo.server.api.QuanZiApi;
6  import com.tanhua.dubbo.server.pojo.Comment;
7  import com.tanhua.dubbo.server.vo.PageInfo;
8  import com.tanhua.server.pojo.User;
9  import com.tanhua.server.pojo.UserInfo;
10 import com.tanhua.server.utils.UserThreadLocal;
11 import com.tanhua.server.vo.Comments;
12 import com.tanhua.server.vo.PageResult;
13 import org.apache.commons.lang3.StringUtils;
14 import org.joda.time.DateTime;
15 import org.springframework.beans.factory.annotation.Autowired;
16 import org.springframework.data.redis.core.RedisTemplate;
17 import org.springframework.stereotype.Service;
18
19 import java.util.ArrayList;
20 import java.util.List;
21
22 @Service
23 public class CommentsService {
24

```

```
25     @Reference(version = "1.0.0")
26     private QuanZiApi quanZiApi;
27
28     @Autowired
29     private UserInfoService userInfoService;
30
31     @Autowired
32     private RedisTemplate<String, String> redisTemplate;
33
34     public PageResult queryCommentsList(String publishId, Integer page,
35     Integer pagesize) {
36
37         User user = UserThreadLocal.get();
38
39         PageInfo<Comment> pageInfo =
40     this.quanZiApi.queryCommentList(publishId, page, pagesize);
41
42         List<Comment> records = pageInfo.getRecords();
43
44         if (records.isEmpty()) {
45             PageResult pageResult = new PageResult();
46             pageResult.setPage(page);
47             pageResult.setPagesize(pagesize);
48             pageResult.setPages(0);
49             pageResult.setCounts(0);
50             return pageResult;
51         }
52
53         List<Long> userIds = new ArrayList<>();
54         for (Comment comment : records) {
55             if (!userIds.contains(comment.getUserId())) {
56                 userIds.add(comment.getUserId());
57             }
58         }
59
60         QueryWrapper<UserInfo> querywrapper = new QueryWrapper<>();
61         querywrapper.in("user_id", userIds);
62         List<UserInfo> userInfos =
63     this.userInfoService.queryList(querywrapper);
64
65         List<Comments> result = new ArrayList<>();
66         for (Comment record : records) {
67             Comments comments = new Comments();
68             comments.setContent(record.getContent());
69             comments.setCreateDate(new
70     DateTime(record.getCreated()).toString("yyyy年MM月dd日 HH:mm"));
71             comments.setId(record.getId().toHexString());
72
73             for (UserInfo userInfo : userInfos) {
74                 if (record.getUserId().longValue() ==
75     userInfo.getUserId().longValue()) {
76                     comments.setAvatar(userInfo.getLogo());
77                     comments.setNickname(userInfo.getNickname());
78                     break;
79                 }
80             }
81         }
82     }
```

```

78         String key = "QUANZI_COMMENT_LIKE_" + comments.getId();
79         String value = this.redisTemplate.opsForValue().get(key);
80         if (StringUtils.isEmpty(value)) {
81             comments.setLikeCount(Integer.valueOf(value)); //点赞数
82         } else {
83             comments.setLikeCount(0);
84         }
85
86         String userKey = "QUANZI_COMMENT_LIKE_USER_" + user.getId() +
87         "_" + comments.getId();
88         comments.setHasLiked(this.redisTemplate.hasKey(userKey) ? 1 :
89         0); //是否点赞 (1是, 0否)
90
91         result.add(comments);
92     }
93
94     PageResult pageResult = new PageResult();
95     pageResult.setItems(result);
96     pageResult.setPage(page);
97     pageResult.setPagesize(pagesize);
98     pageResult.setPages(0);
99     pageResult.setCounts(0);
100
101     return pageResult;
102 }
103
104 /**
105  * 保存评论
106  *
107  * @param publishId
108  * @param content
109  * @return
110  */
111 public Boolean saveComments(String publishId, String content) {
112     User user = UserThreadLocal.get();
113     return this.quanZiApi.saveComment(user.getId(), publishId, 2,
114     content);
115 }
116

```

1.4.4、测试

8:45



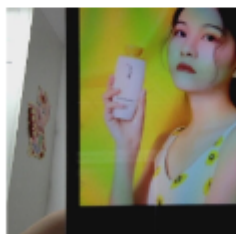
动态评论



heima 30

单身 本科 年龄相仿

test1



最新评论 0



heima_2

good~

👍 0

2019年09月11日 21:40



heima_2

test2

👍 0

2019年09月11日 21:40

发布评论

发送

free for personal use



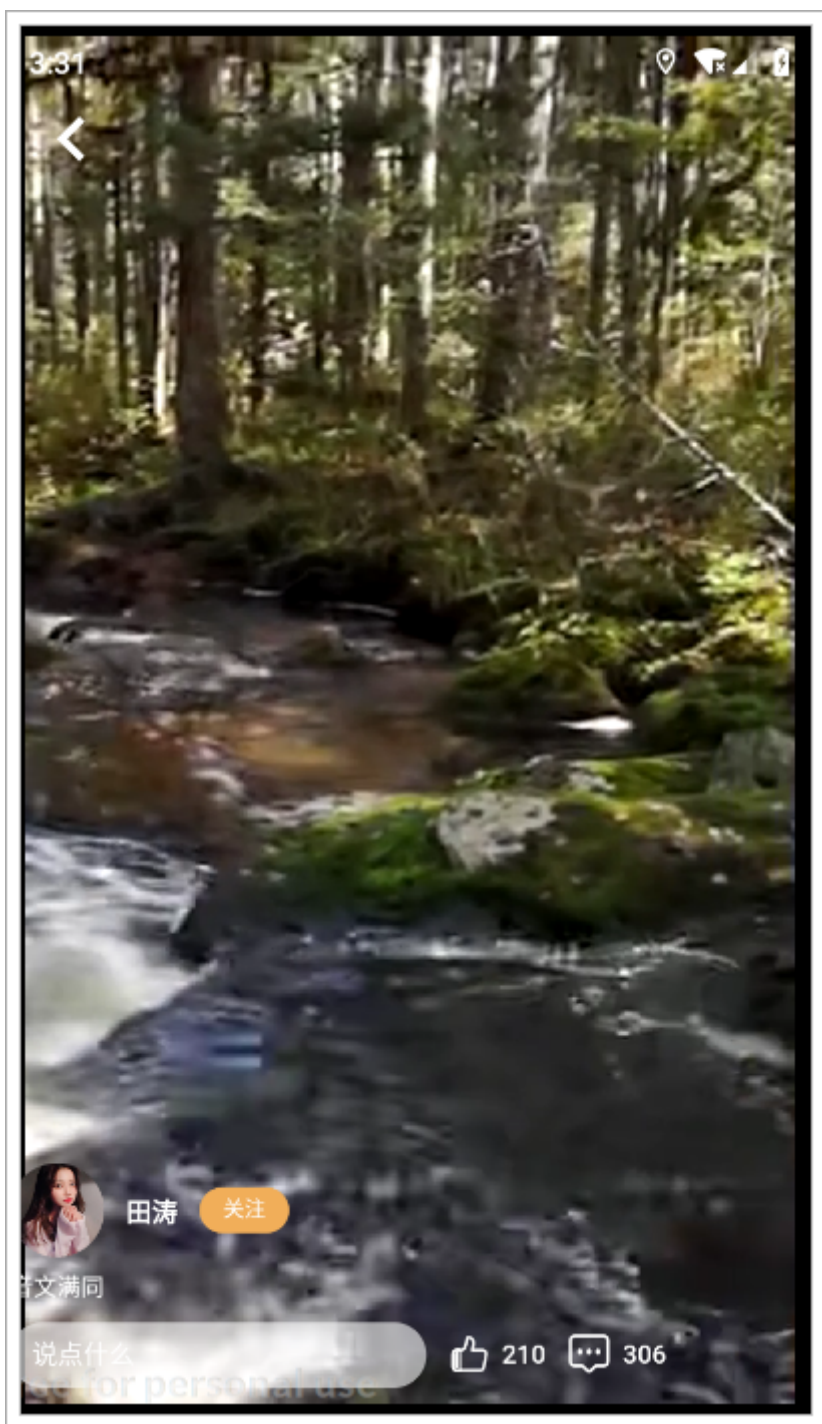
2、小视频功能说明

小视频功能类似于抖音、快手小视频的应用，用户可以上传小视频进行分享，也可以浏览查看别人分享的视频，并且可以对视频评论和点赞操作。

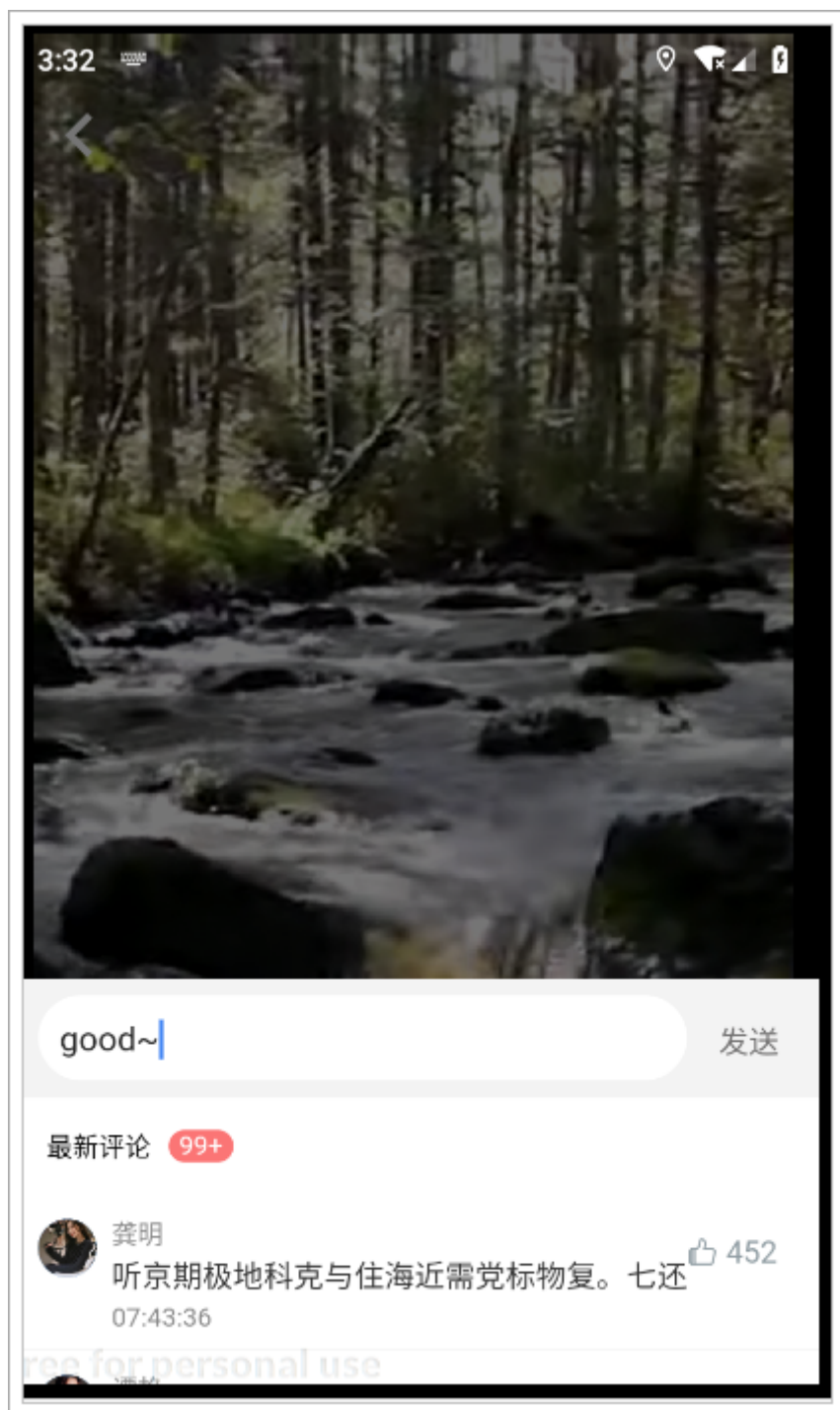
效果：



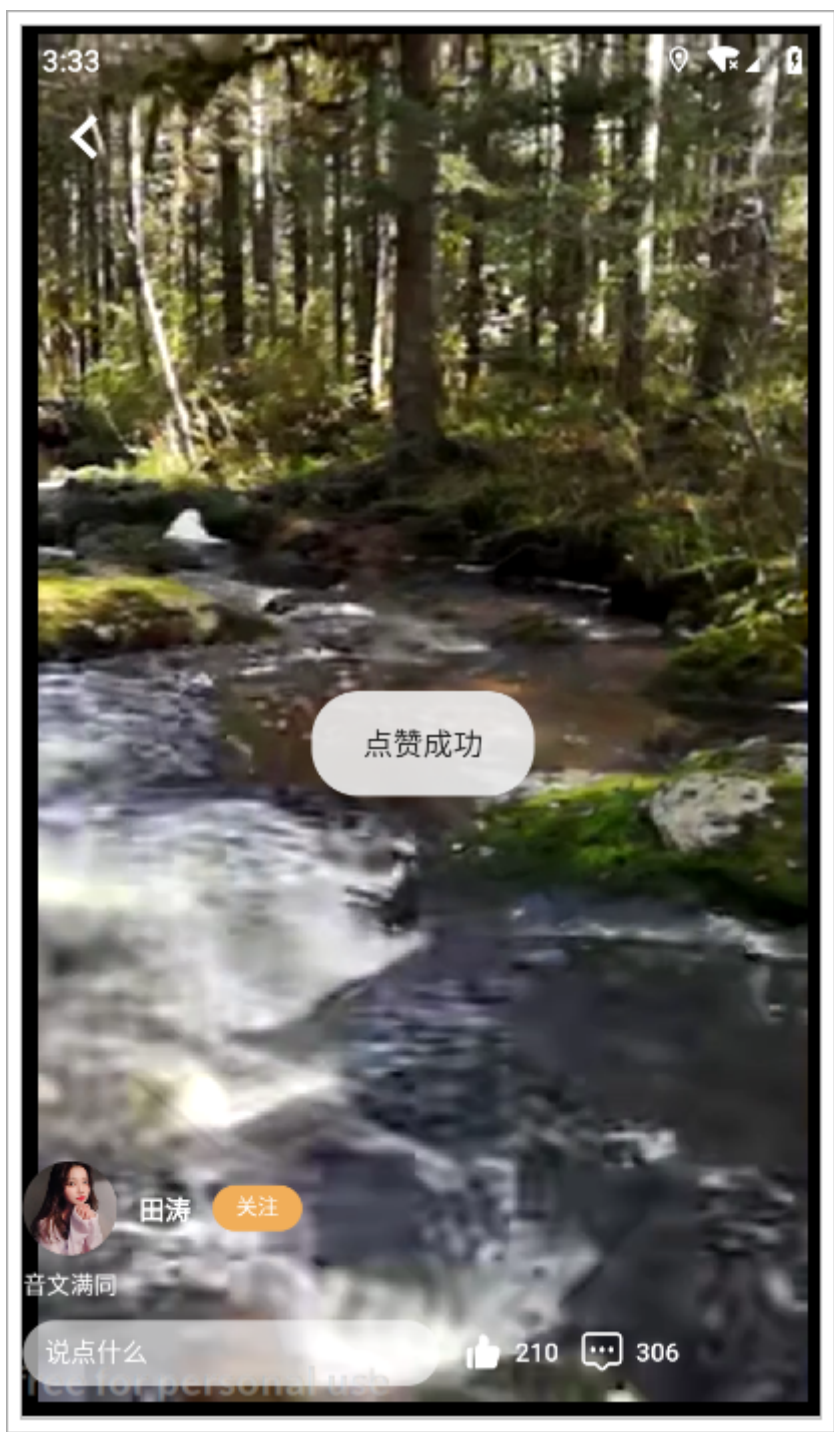
查看详情：



评论：



点赞：



3、技术方案

对于小视频的功能的开发，核心点就是：存储 + 推荐 + 加载速度。

- 对于存储而言，小视频的存储量以及容量都是非常巨大的
 - 所以我们选择自己搭建分布式存储系统 FastDFS进行存储
- 对于推荐算法，我们将采用多种权重的计算方式进行计算
- 对于加载速度，除了提升服务器带宽外可以通过CDN的方式进行加速，当然了这需要额外购买CDN服务

4、FastDFS环境搭建

4.1、搭建服务

我们使用docker进行搭建。

```

1  #拉取镜像
2  docker pull delron/fastdfs
3
4  #创建tracker容器
5  docker run -d --network=host --name tracker -v /var/fdfs/tracker:/var/fdfs
   delron/fastdfs tracker
6
7  #创建storage容器
8  docker run -d --network=host --name storage -e
   TRACKER_SERVER=192.168.31.81:22122 -v /var/fdfs/storage:/var/fdfs -e
   GROUP_NAME=group1 delron/fastdfs storage
9
10 #进入storage容器，到storage的配置文件中配置http访问的端口，配置文件在/etc/fdfs目录下的
    storage.conf。
11 docker exec -it storage /bin/bash
12
13 #默认的http端口为8888，可以修改也可以配置
14 # the port of the web server on this storage server
15 http.server_port=8888
16
17 #配置nginx，在/usr/local/nginx目录下，修改nginx.conf文件
18 #默认配置如下：
19
20     server {
21         listen      8888;
22         server_name localhost;
23         location ~ /group[0-9]/ {
24             ngx_fastdfs_module;
25         }
26         error_page   500 502 503 504 /50x.html;
27         location = /50x.html {
28             root html;
29         }
30     }
31
32 #默认的存储路径为/var/fdfs/data
33

```

```

[root@itcast data]# ls
00 0C 18 24 30 3C 48 54 60 6C 78 84 90 9C A8 B4 C0 CC D8 E4 F0 FC
01 0D 19 25 31 3D 49 55 61 6D 79 85 91 9D A9 B5 C1 CD D9 E5 F1 FD
02 0E 1A 26 32 3E 4A 56 62 6E 7A 86 92 9E AA B6 C2 CE DA E6 F2 FE
03 0F 1B 27 33 3F 4B 57 63 6F 7B 87 93 9F AB B7 C3 CF DB E7 F3 FF
04 10 1C 28 34 40 4C 58 64 70 7C 88 94 A0 AC B8 C4 D0 DC E8 F4 fdfs
05 11 1D 29 35 41 4D 59 65 71 7D 89 95 A1 AD B9 C5 D1 DD E9 F5 stora
06 12 1E 2A 36 42 4E 5A 66 72 7E 8A 96 A2 AE BA C6 D2 DE EA F6 sync
07 13 1F 2B 37 43 4F 5B 67 73 7F 8B 97 A3 AF BB C7 D3 DF EB F7
08 14 20 2C 38 44 50 5C 68 74 80 8C 98 A4 B0 BC C8 D4 E0 EC F8
09 15 21 2D 39 45 51 5D 69 75 81 8D 99 A5 B1 BD C9 D5 E1 ED F9
0A 16 22 2E 3A 46 52 5E 6A 76 82 8E 9A A6 B2 BE CA D6 E2 EE FA
0B 17 23 2F 3B 47 53 5F 6B 77 83 8F 9B A7 B3 BF CB D7 E3 EF FB

```

4.2、java client

导入依赖：

```

1 <dependency>
2   <groupId>com.github.tobato</groupId>
3   <artifactId>fastdfs-client</artifactId>
4   <version>1.26.7</version>
5   <exclusions>
6     <exclusion>
7       <groupId>ch.qos.logback</groupId>
8       <artifactId>logback-classic</artifactId>
9     </exclusion>
10  </exclusions>
11 </dependency>

```

4.2.1、application.properties

```

1 # =====
2 # 分布式文件系统FDFS配置
3 # =====
4 fdfs.so-timeout = 1501
5 fdfs.connect-timeout = 601
6 #缩略图生成参数
7 fdfs.thumb-image.width= 150
8 fdfs.thumb-image.height= 150
9 #TrackerList参数,支持多个
10 fdfs.tracker-list=192.168.31.81:22122

```

4.2.2、测试

```

1 package com.tanhua.server;
2
3 import com.github.tobato.fastdfs.domain.fdfs.StorePath;
4 import com.github.tobato.fastdfs.service.FastFileStorageClient;
5 import org.apache.commons.io.FileUtils;
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.SpringRunner;
11
12 import java.io.File;
13 import java.io.IOException;
14
15 @RunWith(SpringRunner.class)
16 @SpringBootTest
17 public class TestFastDFS {
18
19     @Autowired
20     protected FastFileStorageClient storageClient;
21
22     @Test
23     public void testUpload(){
24         String path = "C:\\Users\\zhijun\\Desktop\\pics\\1.jpg";
25         File file = new File(path);
26
27         try {

```

```

28         StorePath storePath =
this.storageClient.uploadFile(FileUtils.openInputStream(file),
file.length(), "jpg", null);
29
30         System.out.println(storePath); //StorePath [group=group1,
path=M00/00/00/wKgfUV2GJSuAOud_AAHnjh7KpOc1.1.jpg]
31
32         System.out.println(storePath.getFullPath()); //group1/M00/00/00/wKgfUV2GJSu
AOud_AAHnjh7KpOc1.1.jpg
33     } catch (IOException e) {
34         e.printStackTrace();
35     }
36 }
37

```

5、发布小视频

5.1、编写pojo

在dubbo接口工程中编写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  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 userId;
21      private String text; //文字
22      private String picUrl; //视频封面文件
23      private String videoUrl; //视频文件
24      private Long created; //创建时间
25      private Integer seeType; // 谁可以看, 1-公开, 2-私密, 3-部分可见, 4-不给谁看
26      private List<Long> seeList; //部分可见的列表
27      private List<Long> notSeeList; //不给谁看的列表
28      private String longitude; //经度
29      private String latitude; //纬度
30      private String locationName; //位置名称
31  }
32

```

5.2、定义接口

```
1 package com.tanhua.dubbo.server.api;
2
3 import com.tanhua.dubbo.server.pojo.Video;
4
5 public interface VideoApi {
6
7     /**
8      * 保存小视频
9      *
10     * @param video
11     * @return
12     */
13     Boolean saveVideo(Video video);
14
15 }
16
```

5.3、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.Video;
5 import org.bson.types.ObjectId;
6 import org.springframework.beans.factory.annotation.Autowired;
7 import org.springframework.data.mongodb.core.MongoTemplate;
8
9 @Service(version = "1.0.0")
10 public class VideoApiImpl implements VideoApi {
11
12     @Autowired
13     private MongoTemplate mongoTemplate;
14
15     @Override
16     public Boolean saveVideo(Video video) {
17         if(video.getUserId() == null){
18             return false;
19         }
20
21         video.setId(ObjectId.get());
22         video.setCreated(System.currentTimeMillis());
23
24         this.mongoTemplate.save(video);
25         return true;
26     }
27 }
28
```

5.4、接口服务

接口路径： **POST** /smallVideos

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

请求参数

Headers：

参数名称	参数值	是否必须	示例	备注
Content-Type	multipart/form-data	是		

Body:

参数名称	参数类型	是否必须	示例	备注
videoThumbnail	📁 文件	是		视频封面文件
videoFile	📁 文件	是		视频文件

返回数据

5.4.1、VideoController

```
1 package com.tanhua.server.controller;
2
3 import com.tanhua.server.service.VideoService;
4 import org.springframework.beans.factory.annotation.Autowired;
5 import org.springframework.http.HttpStatus;
6 import org.springframework.http.ResponseEntity;
7 import org.springframework.web.bind.annotation.PostMapping;
8 import org.springframework.web.bind.annotation.RequestMapping;
9 import org.springframework.web.bind.annotation.RequestParam;
10 import org.springframework.web.bind.annotation.RestController;
11 import org.springframework.web.multipart.MultipartFile;
12
13 @RestController
14 @RequestMapping("smallVideos")
15 public class VideoController {
16
17     @Autowired
18     private VideoService videoService;
19
20     @PostMapping
21     public ResponseEntity<Void> saveVideo(@RequestParam(value =
22 "videoThumbnail", required = false) MultipartFile picFile,
23                                         @RequestParam(value =
24 "videoFile", required = false) MultipartFile videoFile) {
25         try {
26             boolean bool = this.videoService.saveVideo(picFile, videoFile);
27             if(bool){
28                 return ResponseEntity.ok(null);
29             }
30         } catch (Exception e) {
31             e.printStackTrace();
32         }
33     }
34 }
```

```

31         return
32     ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
33 }
34

```

5.4.2、VideoService

```

1  package com.tanhua.server.service;
2
3  import com.alibaba.dubbo.config.annotation.Reference;
4  import com.github.tobato.fastdfs.domain.conn.FdfswebServer;
5  import com.github.tobato.fastdfs.domain.fdfs.StorePath;
6  import com.github.tobato.fastdfs.service.FastFileStorageClient;
7  import com.tanhua.dubbo.server.api.VideoApi;
8  import com.tanhua.dubbo.server.pojo.Video;
9  import com.tanhua.server.pojo.User;
10 import com.tanhua.server.utils.UserThreadLocal;
11 import com.tanhua.server.vo.PicUploadResult;
12 import org.apache.commons.lang3.StringUtils;
13 import org.springframework.beans.factory.annotation.Autowired;
14 import org.springframework.stereotype.Service;
15 import org.springframework.web.multipart.MultipartFile;
16
17 @Service
18 public class VideoService {
19
20     @Autowired
21     private PicUploadService picUploadService;
22
23     @Autowired
24     protected FastFileStorageClient storageClient;
25
26     @Autowired
27     private FdfswebServer fdfswebServer;
28
29     @Reference(version = "1.0.0")
30     private VideoApi videoApi;
31
32     public Boolean saveVideo(MultipartFile picFile, MultipartFile
33 videoFile) {
34         User user = UserThreadLocal.get();
35         Video video = new Video();
36         video.setUserId(user.getId());
37         video.setSeeType(1);
38         try {
39             //上传封面图片
40             PicUploadResult picUploadResult =
41 this.picUploadService.upload(picFile);
42             video.setPicUrl(picUploadResult.getName()); //图片路径
43
44             //上传视频
45             StorePath storePath =
46 storageClient.uploadFile(videoFile.getInputStream(),
47                             videoFile.getSize(),
48                             StringUtils.substringAfter(videoFile.getOriginalFilename(), "."),

```

```

46         null);
47         video.setVideoUrl(fdfswebServer.getWebServerUrl() + "/" +
storePath.getFullPath());
48
49         this.videoApi.saveVideo(video);
50
51         return true;
52     } catch (Exception e) {
53         e.printStackTrace();
54     }
55
56     return false;
57 }
58 }
59

```

5.4.3、测试

如果上传视频，会导致异常，是因为请求太大的缘故：

```

java.io.IOException: Stream closed
    at org.apache.catalina.connector.InputBuffer.read(InputBuffer.java:338) ~[tomcat-embed-core-
    at org.apache.catalina.connector.CoyoteInputStream.read(CoyoteInputStream.java:132) ~[tomcat-
    at org.apache.catalina.connector.CoyoteInputStream.read(CoyoteInputStream.java:110) ~[tomcat-
    at org.apache.commons.io.IOUtils.copyLarge(IOUtils.java:2314) ~[commons-io-2.6.jar:2.6]
    at org.apache.commons.io.IOUtils.copy(IOUtils.java:2270) ~[commons-io-2.6.jar:2.6]
    at org.apache.commons.io.IOUtils.copyLarge(IOUtils.java:2291) ~[commons-io-2.6.jar:2.6]
    at org.apache.commons.io.IOUtils.copy(IOUtils.java:2246) ~[commons-io-2.6.jar:2.6]
    at org.apache.commons.io.IOUtils.toByteArray(IOUtils.java:765) ~[commons-io-2.6.jar:2.6]
    at com.tanhua.server.interceptor.MyServletRequestWrapper.<init>(MyServletRequestWrapper.java
    at com.tanhua.server.interceptor.RequestReplaceFilter.doFilterInternal(RequestReplaceFilter.
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.j
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)
    at org.apache.catalina.core.ApplicationFilterChain.internalDoFilter(ApplicationFilterChain.j
    at org.apache.catalina.core.ApplicationFilterChain.doFilter(ApplicationFilterChain.java:166)

```

解决：application.properties

```

1 spring.servlet.multipart.max-file-size=30MB
2 spring.servlet.multipart.max-request-size=30MB

```

问题解决：

(2) ObjectId("5d862da345d9871d0c188319")	
id	ObjectId("5d862da345d9871d0c188319")
userId	100
picUrl	http://tcast-tanhua.oss-cn-shanghai.aliyuncs.com/images/2019/09/21/15690745484922678.png
videoUrl	http://192.168.31.81:8888/group1/M00/00/00/wKgFUV2GLYGATvIjACYJuhDRF3g165.mp4
created	1569074595143
seeType	1
_class	com.tanhua.dubbo.server.pojo.Video

6、小视频列表

6.1、定义dubbo服务

```

1 package com.tanhua.dubbo.server.api;
2
3 import com.tanhua.dubbo.server.pojo.Video;
4 import com.tanhua.dubbo.server.vo.PageInfo;
5
6 public interface VideoApi {
7

```

```

8      /**
9       * 保存小视频
10      *
11      * @param video
12      * @return
13      */
14      Boolean saveVideo(Video video);
15
16      /**
17      * 分页查询小视频列表，按照时间倒序排序
18      *
19      * @param page
20      * @param pageSize
21      * @return
22      */
23      PageInfo<Video> queryVideoList(Integer page, Integer pageSize);
24
25  }
26

```

6.2、实现dubbo服务

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.alibaba.dubbo.config.annotation.Service;
4  import com.tanhua.dubbo.server.pojo.Video;
5  import com.tanhua.dubbo.server.vo.PageInfo;
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.Query;
13
14 import java.util.List;
15
16 @Service(version = "1.0.0")
17 public class VideoApiImpl implements VideoApi {
18
19     @Autowired
20     private MongoTemplate mongoTemplate;
21
22     @Override
23     public Boolean saveVideo(Video video) {
24         if (video.getUserId() == null) {
25             return false;
26         }
27
28         video.setId(ObjectId.get());
29         video.setCreated(System.currentTimeMillis());
30
31         this.mongoTemplate.save(video);
32         return null;
33     }
34
35     @Override

```

```

36     public PageInfo<Video> queryVideoList(Integer page, Integer pageSize) {
37         Pageable pageable = PageRequest.of(page - 1, pageSize,
Sort.by(Sort.Order.desc("created"))));
38         Query query = new Query().with(pageable);
39         List<Video> videos = this.mongoTemplate.find(query, Video.class);
40         PageInfo<Video> pageInfo = new PageInfo<>();
41         pageInfo.setRecords(videos);
42         pageInfo.setPageNum(page);
43         pageInfo.setPageSize(pageSize);
44         pageInfo.setTotal(0); //不提供总数
45         return pageInfo;
46     }
47 }
48

```

6.3、定义VideoVo

```

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 VideoVo {
11
12     private String id;
13     private Long userId;
14     private String avatar; //头像
15     private String nickname; //昵称
16     private String cover; //封面
17     private String videoUrl; //视频URL
18     private String signature; //签名
19     private Integer likeCount; //点赞数量
20     private Integer hasLiked; //是否已赞（1是，0否）
21     private Integer hasFocus; //是否关注（1是，0否）
22     private Integer commentCount; //评论数量
23 }
24

```

6.4、VideoController

```

1  @RestController
2  @RequestMapping("smallVideos")
3  public class VideoController {
4      /**
5       * 查询小视频列表
6       *
7       * @param page
8       * @param pageSize
9       * @return
10      */
11      @GetMapping

```

```

12     public ResponseEntity<PageResult> queryVideoList(@RequestParam(value =
    "page", defaultValue = "1") Integer page,
13                                                         @RequestParam(value =
    "pagesize", defaultValue = "10") Integer pageSize) {
14         try {
15             if (page <= 0) {
16                 page = 1;
17             }
18             PageResult pageResult = this.videoService.queryVideoList(page,
    pageSize);
19             if (null != pageResult) {
20                 return ResponseEntity.ok(pageResult);
21             }
22         } catch (Exception e) {
23             e.printStackTrace();
24         }
25         return
    ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
26     }
27 }

```

6.5、VideoService

```

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 = this.videoApi.queryVideoList(page,
    pageSize);
12     List<Video> records = pageInfo.getRecords();
13     List<VideoVo> videoVoList = new ArrayList<>();
14     List<Long> userIds = new ArrayList<>();
15     for (Video record : records) {
16         VideoVo videoVo = new VideoVo();
17
18         videoVo.setUserId(record.getUserId());
19         videoVo.setCover(record.getPicUrl());
20         videoVo.setVideoUrl(record.getVideoUrl());
21         videoVo.setId(record.getId().toHexString());
22         videoVo.setSignature("我就是我~");
23
24         Long commentCount =
    this.quanZiApi.queryCommentCount(videoVo.getId(), 2);
25         videoVo.setCommentCount(commentCount == null ? 0 :
    commentCount.intValue()); // 评论数
26
27         String followUserKey = "VIDEO_FOLLOW_USER_" + user.getId() +
    "_" + videoVo.getUserId();
28         videoVo.setHasFocus(this.redisTemplate.hasKey(followUserKey) ?
    1 : 0); //是否关注
29

```

```

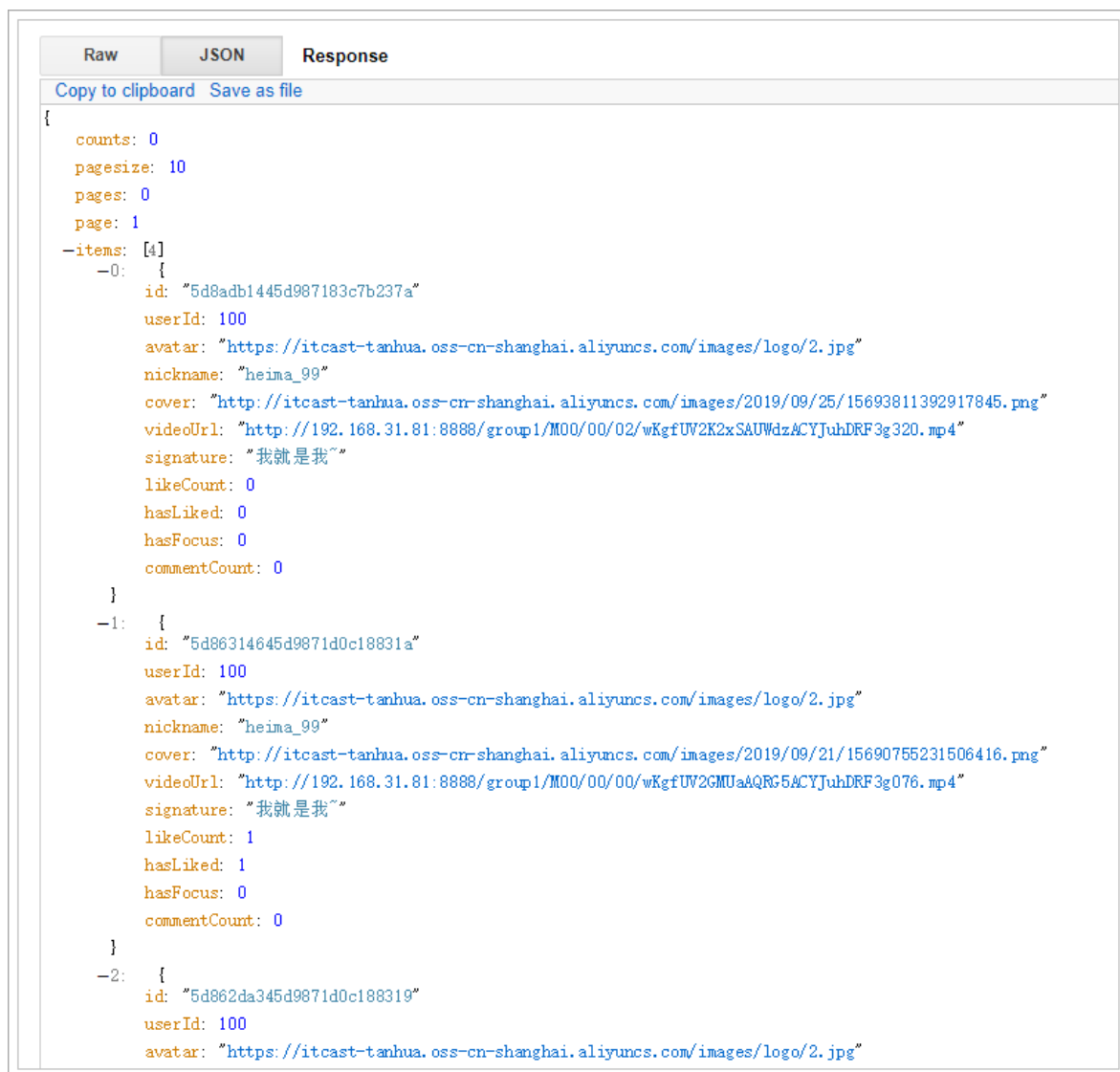
30         String userKey = "QUANZI_COMMENT_LIKE_USER_" + user.getId() +
        "_" + videoVo.getId();
31         videoVo.setHasLiked(this.redisTemplate.hasKey(userKey) ? 1 :
        0); //是否点赞 (1是, 0否)
32
33         String key = "QUANZI_COMMENT_LIKE_" + videoVo.getId();
34         String value = this.redisTemplate.opsForValue().get(key);
35         if (StringUtils.isEmpty(value)) {
36             videoVo.setLikeCount(Integer.valueOf(value)); //点赞数
37         } else {
38             videoVo.setLikeCount(0);
39         }
40
41         if (!userIds.contains(record.getUserId())) {
42             userIds.add(record.getUserId());
43         }
44
45         videoVoList.add(videoVo);
46     }
47
48     QueryWrapper<UserInfo> queryWrapper = new QueryWrapper();
49     queryWrapper.in("user_id", userIds);
50     List<UserInfo> userInfos =
        this.userService.queryList(queryWrapper);
51     for (VideoVo videoVo : videoVoList) {
52         for (UserInfo userInfo : userInfos) {
53             if (videoVo.getUserId().longValue() ==
        userInfo.getUserId().longValue()) {
54
55                 videoVo.setNickname(userInfo.getNickName());
56                 videoVo.setAvatar(userInfo.getLogo());
57
58                 break;
59             }
60         }
61     }
62
63     pageResult.setItems(videoVoList);
64     return pageResult;
65 }

```

6.6、测试

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

- URL:** `http://172.16.55.153/smallVideos`
- Method:** `GET` (selected from a dropdown menu including POST, PUT, PATCH, DELETE, HEAD, OPTIONS, and Other).
- Headers Tab:**
 - Add new header:** A link to add a new header.
 - Authorization:** A header field with a value: `eyJhbGciOiJIUzI1NiU9.eyJjb2JpbGUuOiixNTgwMDgwNzk4OClzImkiOiJjoxMDB9.LG78GYdu4zl36rXJlyRkR1jrmjo1UutqdvpsNnhZsKs`.



7、视频点赞

点赞逻辑与之前的圈子点赞一样，所以实现也是一样的。

mock地址：<https://mock.bxuegu.com/project/164/interface/api/77987>



VideoController :

```
1  /**
2   * 视频点赞
3   *
4   * @param videoId 视频id
5   * @return
6   */
7  @PostMapping("/{id}/like")
8  public ResponseEntity<Long> likeComment(@PathVariable("id") String
videoId) {
9      return this.movementsController.likeComment(videoId);
10 }
11
12 /**
13 * 取消点赞
14 *
15 * @param videoId
16 * @return
17 */
18 @PostMapping("/{id}/dislike")
19 public ResponseEntity<Long> disLikeComment(@PathVariable("id") String
videoId) {
20     return this.movementsController.disLikeComment(videoId);
21 }
```

8、小视频评论

小视频的评论与圈子的评论逻辑类似，所以也可以使用同一套逻辑进行实现。

8.1、评论列表

mock地址：<https://mock.boxuegu.com/project/164/interface/api/72177>

```
1  /**
2   * 评论列表
3   */
4  @GetMapping("/{id}/comments")
5  public ResponseEntity<PageResult> queryCommentsList(@PathVariable("id")
String videoId,
6                                                         @RequestParam(value =
"page", defaultValue = "1") Integer page,
7                                                         @RequestParam(value =
"pagesize", defaultValue = "10") Integer pagesize) {
8      return this.commentsController.queryCommentsList(videoId, page,
pagesize);
9  }
```

8.2、发布评论

mock地址：<https://mock.boxuegu.com/project/164/interface/api/72919>

```

1  /**
2   * 提交评论
3   *
4   * @param param
5   * @param videoId
6   * @return
7   */
8  @PostMapping("/{id}/comments")
9  public ResponseEntity<Void> saveComments(@RequestBody Map<String,
10 String> param,
11                                           @PathVariable("id") String
12 videoId) {
13     param.put("movementId", videoId);
14     return this.commentsController.saveComments(param);
15 }

```

8.3、评论点赞

mock地址：<https://mock.boxuegu.com/project/164/interface/api/75306>

```

1  /**
2   * 评论点赞
3   *
4   * @param publishId
5   * @return
6   */
7  @PostMapping("/comments/{id}/like")
8  public ResponseEntity<Long> commentsLikeComment(@PathVariable("id")
9 String publishId) {
10     return this.movementsController.likeComment(publishId);
11 }
12
13 /**
14 * 评论取消点赞
15 *
16 * @param publishId
17 * @return
18 */
19 @PostMapping("/comments/{id}/dislike")
20 public ResponseEntity<Long> disCommentsLikeComment(@PathVariable("id")
21 String publishId) {
22     return this.movementsController.disLikeComment(publishId);
23 }

```

8.4、关注用户

关注用户是关注小视频发布的作者，这样我们后面计算推荐时，关注的用户将权重更重一些。

8.4.1、定义dubbo服务

```

1  package com.tanhua.dubbo.server.pojo;
2
3  import lombok.AllArgsConstructor;
4  import lombok.Data;
5  import lombok.NoArgsConstructor;

```

```

6  import org.bson.types.ObjectId;
7  import org.springframework.data.mongodb.core.mapping.Document;
8
9  @Data
10 @NoArgsConstructor
11 @AllArgsConstructor
12 @Document(collection = "follow_user")
13 public class FollowUser implements java.io.Serializable{
14
15     private static final long serialVersionUID = 3148619072405056052L;
16
17     private ObjectId id; //主键id
18     private Long userId; //用户id
19     private Long followUserId; //关注的用户id
20     private Long created; //关注时间
21 }
22

```

```

1  package com.tanhua.dubbo.server.api;
2
3  import com.tanhua.dubbo.server.pojo.Video;
4  import com.tanhua.dubbo.server.vo.PageInfo;
5
6  public interface VideoApi {
7
8      /**
9       * 保存小视频
10      *
11      * @param video
12      * @return
13      */
14     Boolean saveVideo(Video video);
15
16     /**
17      * 分页查询小视频列表，按照时间倒序排序
18      *
19      * @param page
20      * @param pageSize
21      * @return
22      */
23     PageInfo<Video> queryVideoList(Integer page, Integer pageSize);
24
25     /**
26      * 关注用户
27      *
28      * @param userId
29      * @param followUserId
30      * @return
31      */
32     Boolean followUser(Long userId, Long followUserId);
33
34     /**
35      * 取消关注用户
36      *
37      * @param userId
38      * @param followUserId
39      * @return

```

```

40     */
41     Boolean disFollowUser(Long userId, Long followUserId);
42
43 }
44

```

接口实现：

```

1  //VideoApiImpl:
2  @Override
3      public Boolean followUser(Long userId, Long followUserId) {
4      try {
5          FollowUser followUser = new FollowUser();
6          followUser.setId(ObjectId.get());
7          followUser.setUserId(userId);
8          followUser.setFollowUserId(followUserId);
9          followUser.setCreated(System.currentTimeMillis());
10         this.mongoTemplate.save(followUser);
11         return true;
12     } catch (Exception e) {
13         e.printStackTrace();
14     }
15     return false;
16 }
17
18 @Override
19 public Boolean disFollowUser(Long userId, Long followUserId) {
20     Query query =
21     Query.query(Criteria.where("userId").is(userId).and("followUserId").is(followUserId));
22     DeleteResult deleteResult = this.mongoTemplate.remove(query,
23     FollowUser.class);
24     return deleteResult.getDeletedCount() > 0;
25 }

```

8.4.2、服务实现

```

1  //VideoController:
2  /**
3   * 视频用户关注
4   */
5  @PostMapping("/{id}/userFocus")
6  public ResponseEntity<Void> saveUserFocusComments(@PathVariable("id")
7  Long userId) {
8      try {
9          Boolean bool = this.videoService.followUser(userId);
10         if (bool) {
11             return ResponseEntity.ok(null);
12         }
13     } catch (Exception e) {
14         e.printStackTrace();
15     }
16     return
17     ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
18 }

```

```

18     /**
19      * 取消视频用户关注
20      */
21     @PostMapping("/{id}/userUnFocus")
22     public ResponseEntity<Void> saveUserUnFocusComments(@PathVariable("id")
23 Long userId) {
24         try {
25             Boolean bool = this.videoService.disFollowUser(userId);
26             if (bool) {
27                 return ResponseEntity.ok(null);
28             }
29         } catch (Exception e) {
30             e.printStackTrace();
31         }
32         return
33         ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
34     }

```

```

1 //VideoService:
2 /**
3  * 关注用户
4  *
5  * @param userId
6  * @return
7  */
8 public Boolean followUser(Long userId) {
9     User user = UserThreadLocal.get();
10    this.videoApi.followUser(user.getId(), userId);
11
12    //记录已关注
13    String followUserKey = "VIDEO_FOLLOW_USER_" + user.getId() + "_" +
14    userId;
15    this.redisTemplate.opsForValue().set(followUserKey, "1");
16
17    return true;
18 }
19 /**
20  * 取消关注
21  *
22  * @param userId
23  * @return
24  */
25 public Boolean disFollowUser(Long userId) {
26     User user = UserThreadLocal.get();
27     this.videoApi.disFollowUser(user.getId(), userId);
28
29     String followUserKey = "VIDEO_FOLLOW_USER_" + user.getId() + "_" +
30     userId;
31     this.redisTemplate.delete(followUserKey);
32
33     return true;
34 }

```

